Tai-hoon Kim Haeng-kon Kim
Muhammad Khurram Khan Akingbehin Kiumi
Wai-chi Fang Dominik Ślęzak (Eds.)

Communications in Computer and Information Science 117

Advances in Software Engineering

International Conference, ASEA 2010
Held as Part of the Future Generation
Information Technology Conference, FGIT 2010
Jeju Island, Korea, December 2010, Proceedings

Springer

Communications
in Computer and Information Science 117

Tai-hoon Kim Haeng-kon Kim
Muhammad Khurram Khan Akingbehin Kiumi
Wai-chi Fang Dominik Ślęzak (Eds.)

Advances in Software Engineering

International Conference, ASEA 2010
Held as Part of the Future Generation
Information Technology Conference, FGIT 2010
Jeju Island, Korea, December 13-15, 2010
Proceedings

Springer

Volume Editors

Tai-hoon Kim
Hannam University, Daejeon, South Korea
E-mail: taihoonn@hnu.kr

Haeng-kon Kim
Catholic University of Daegu, South Korea
E-mail: hangkon@cu.ac.kr

Muhammad Khurram Khan
King Saud University, Riyadh, Saudi Arabia
E-mail: mkhurram@ksu.edu.sa

Akingbehin Kiumi
University of Michigan-Dearborn, USA
E-mail: kiumi@umich.edu

Wai-chi Fang
National Chiao Tung University, Hsinchu, Taiwan
E-mail: wfang@mail.nctu.edu.tw

Dominik Ślęzak
University of Warsaw & Infobright, Warsaw, Poland
E-mail: dominik.slezak@infobright.com

Library of Congress Control Number: 2010940172

CR Subject Classification (1998): D.2, H.3.3, I.2, H.4, J.1, D.2.1

ISSN	1865-0929
ISBN-10	3-642-17577-5 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-17577-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

Welcome to the Proceedings of the 2010 International Conference on Advanced Software Engineering and Its Applications (ASEA 2010) – one of the partnering events of the Second International Mega-Conference on Future Generation Information Technology (FGIT 2010).

ASEA brings together researchers from academia and industry as well as practitioners to share ideas, problems and solutions relating to the multifaceted aspects of software engineering, including its links to computational sciences, mathematics and information technology.

In total, 1,630 papers were submitted to FGIT 2010 from 30 countries, which includes 175 papers submitted to ASEA 2010. The submitted papers went through a rigorous reviewing process: 395 of the 1,630 papers were accepted for FGIT 2010, while 40 papers were accepted for ASEA 2010. Of the 640 papers were selected for the special FGIT 2010 volume published by Springer in the LNCS series. 32 papers are published in this volume, and 2 papers were withdrawn due to technical reasons.

We would like to acknowledge the great effort of the ASEA 2010 International Advisory Board and members of the International Program Committee, as well as all the organizations and individuals who supported the idea of publishing this volume of proceedings, including SERSC and Springer. Also, the success of the conference would not have been possible without the huge support from our sponsors and the work of the Chairs and Organizing Committee.

We are grateful to the following keynote speakers who kindly accepted our invitation: Hojjat Adeli (Ohio State University), Ruay-Shiung Chang (National Dong Hwa University), and Andrzej Skowron (University of Warsaw). We would also like to thank all plenary and tutorial speakers for their valuable contributions.

We would like to express our greatest gratitude to the authors and reviewers of all paper submissions, as well as to all attendees, for their input and participation.

Last but not least, we give special thanks to Rosslin John Robles and Maricel Balitanas. These graduate school students of Hannam University contributed to the editing process of this volume with great passion.

December 2010

Tai-hoon Kim
Haeng-kon Kim
Muhammad Khurram Khan
Akingbehin Kiumi
Wai-chi Fang
Dominik Ślęzak

Organization

Organizing Committee

General Chair	Haeng-kon Kim (Catholic University of Daegu, Korea)
Program Co-chairs	Muhammad Khurram Khan (King Saud University, Saudi Arabia) Tai-hoon Kim (Hannam University, Korea) Akingbehin Kiumi (University of Michigan-Dearborn, USA)
Publicity Co-chairs	Tao Jiang (Huazhong University of Science and Technology, China) Silvia Abrahao (Valencia University of Technology, Spain) June Verner (University of New South Wales, Australia)
Publication Chair	Yong-ik Yoon (Sookmyung Women's University, Korea)

International Advisory Board

Jose Luis Arciniegas Herrera (Universidad del Cauca, Colombia)
Aboul Ella Hassanien (Cairo University, Egypt)
Byeong-Ho Kang (University of Tasmania, Australia)
Tien N. Nguyen (Iowa State University, USA)

Program Committee

A. Hamou-Lhadj	Jiro Tanaka	Morshed Chowdhury
Ami Marowka	Jonathan Lee	Olga Ormandjieva
Carmine Gravino	Jongmoon Baik	P.R. Srivastava
Chia-Chu Chiang	Jose L. Arciniegas	Rattikorn Hewett
Chima Adiele	Joseph Balikuddembe	Ricardo Campos
Dinesh Verma	Karel Richta	Rita Francese
Doo-Hwan Bae	Kendra Cooper	Robert Glass
Emilia Mendes	Kin Fun Li	Rocco Oliveto
Fausto Fasano	Kurt Wallnau	Rudolf Ferenc
Giuseppe Scanniello	Laurence Duchien	Shawkat Ali
Gongzhu Hu	Lerina Aversano	Silvia Abrahao
Harvey Siy	Luigi Buglione	Tokuro Matsuo
Hironori Washizaki	Maria Bielikova	Vincenzo Deufemia
Hyeon Soo Kim	Maria Tortorella	Wuwei Shen
Jennifer Pérez Benedí	Mokhtar Beldjehem	Yijun Yu

Table of Contents

Effective Web and Desktop Retrieval with Enhanced Semantic Spaces

Amjad M. Daoud

American University of the Middle East, Kuwait

Abstract. We describe the design and implementation of the NETBOOK prototype system for collecting, structuring and efficiently creating semantic vectors for concepts, noun phrases, and documents from a corpus of free full text ebooks available on the World Wide Web. Automatic generation of concept maps from correlated index terms and extracted noun phrases are used to build a powerful conceptual index of individual pages. To ensure scalabilty of our system, dimension reduction is performed using Random Projection [13]. Furthermore, we present a complete evaluation of the relative effectiveness of the NETBOOK system versus the Google Desktop [8].

Keywords: Semantic Vectors, NETBOOK, Dimension Reduction, Retrieval Effectiveness.

1 Introduction

The problem of storing, managing, and accessing information is a classic problem in human society, where the ultimate goal is constructing information retrieval systems that can understand, in a non-trivial sense, texts as humans. So, enhancing conventional search techniques with semantic understanding has taken on an even greater significance and true progress would be far-reaching.

For the last four decades, researchers have explored the statistical, lexical, and semantic characteristics of various collections of messages, bibliographic citations, HTML and XML pages, TREC collections, and other types of documents. They developed automatic indexing techniques [11], prepared lexical-relational thesauri [7], devised efficient storage structures and algorithms [5], and proposed and evaluated retrieval approaches [10].

Currently, novice information seekers depend on search engines such as GOOGLE and YAHOO to handle the explosion of information available on the world wide web, or browse through infinite maze of hyperlinks. Most internet search engines, given a search query, try to find web documents containing the terms listed in the query. Ranking algorithms such as the PageRank depends primarily on the connectivity of pages it indexes. However, the search process treats web pages as bag of unrelated words and words are treated purely as semanticless identifiers. These same search engines are unable to distinguish different word senses, not mentioning documents about different topical domains. Frequently, users must manually scan through a long list of interleaved results in order to zoom in onto the desired ones.

In section 2, we describe the design and implementation of the NETBOOK system and survey related literature on context vector models for representing concepts with vectors in a high dimensional vector space. In section 4, we evaluate the NETBOOK system versus the GOOGLE Desktop system. In section 5, we summarize our research findings.

2 The NETBOOK System

2.1 Early Work

In earlier versions of the NETBOOK system, we used information that is readily available in chapter titles and index terms to build simple Concept Maps [9] [3] [2]. Non-frequent single terms and noun phrases found in titles can be extracted and filtered to label concept nodes.

Afterwards, the index term topical discriminating power (i.e. on the topic of the page it occurs in) can be computed using the term discriminating power on the page itself. A term is a good discriminator for a topic if most documents that contain that term are topically related and finding these that tend to occur in the context of a given topic is important to achieve higher precision.

Of course, a more elaborate scheme would be to rely on advanced NLP and AI to derive elaborate Concept Maps and automatically generate good ones efficiently for large collections. Although relationship links in concept maps are perceived to be essential to semantic understanding, a recent study concluded the sufficiency of concept labels to guide retrieval and refine queries [9].

One way to solve the problem is to figure out which one of the different word senses or the *topic* that a document assumes. For instance, let's say the term "Jordan" carries two senses, the basketball player and the name of a country. A short document or a page containing "Jordan" will use the term to denote either the basketball player or the country, but probably not both. A GOOGLE query on the topic "jordan basketball" will never yield anything about basketball in Jordan. Modifying the query to "basketball in jordan" would face the same fate as "in" is treated as a stopword.

2.2 The Context Vectors Model

Clearly, word sense discrimination and disambiguation [14] is required to effectively navigate the web. The core principle here is that the weighted sum of vectors for words found in a particular region of text (called context vectors) can be clustered, and the centroids of these clusters can be treated as word-senses: occurrences of an ambiguous word can then be mapped to one of these word-senses, with a confidence or probability derived from the similarity between the context vector for this occurrence and the nearest centroids.

However, these early approaches often yielded high dimensional matrices that required factorization that did not scale well for large collection such as the web. [6] applied dimension reduction to a term-document matrix, in the hope of creating a more semantically aware search engine (e.g., a search engine that can locate documents based on synonyms and related terms as well as matching keywords). Such systems would find documents that talk about H1N1 even if the query was simply "swan flu".

Using a powerful ontology of a few seed concepts and their relationships can help to iteratively infer analogous relationships for other similar words that are nearby in the vector space that denote potential cognitive significance. Moreover, given that we can compute *context vectors* for regions of text, it is possible to detect disfluencies [13] when context vectors leap from one part of the space to a completely different part of the space. On the other hand, it would be easy to make improvements by extra rounds of training.

2.3 The Ebooks Collection

Recent advances in computer hardware and dramatic improvements in CPU speed and disk storage have enabled us to carry out evaluation experiments regarding the most *effective* methods for the storage and retrieval of pages of full text ebooks available online. The NETBOOK system represents on-going attempts to answer many of these questions with a very large collection of computer science, math, and physics resources. The collection of ebooks contains about 30000 unique ebooks with more than 18 million individual pages. Although, presenting the NETBOOK system in terms of vector models suggests a strongly geometric account, it is worth noting that probabilistic interpretations could be applied rather than geometric insights. [12] points out that quantum mechanics is already a clearly extant framework that combines both probabilistic and geometric insights, coordinates of vectors being related to probability amplitudes and both are compatible ways of looking at related conceptual structures [13].

Let D be the ebooks collection consisting of n documents and T the dictionary consisting of m index terms used for content identification. In the rest of the paper, our unit of retrieval is a page in a book. A page is chosen rather than paragraphs to simplify mapping author index terms to retrievable and browsable pages. So D consists really of n pages. Also, the adoption of pages as units of retrieval allows us to compute the recall and precision of each search based on authors choice of index terms, and to automatically refine returned pages to accomplish better ranking [see section 4] and to construct better automatic relevance feedback.

Define an indexing function $I(D_i)$ that returns a subset of weighted index terms and their topics given the text of a document D_i, its table of contents that defines possible topics, and concordance that defines terms found in its index and pages they occur in:

$$I(D_i) = \{(t_{ij}, w_{ij}, \lambda_{ij}) | 1 \leq j \leq m; t_j \in T\}$$

where t_{ij} represents the assignment of term t_j to the document D_i, w_{ij} is a real number in the interval $[0, 1]$ which reflects the discriminating power (i.e. importance) of the term t_j for identifying the document D_i, λ_{ij} which reflects the discriminating power of the term t_j for identifying the *topic* of the document D_i chosen from the set of topics automatically generated from the documents table of contents. λ_{ij} is the average of the similarity of D_i to other documents discriminated by t_j.

The same indexing function can be applied to the text of a query to get a comparable query representative

$$I(Q) = \{(q_j, w_{qj}) | 1 \leq j \leq m; q_j \in T\}$$

An inverted list associated with the term t_j is the set of documents indexed by such a term and their topics

$$D_{t_j} = \{D_i | t_j \in I(D_i)\}$$

Given a query Q, the set P_Q of documents which *possibly* satisfy the query is given by

$$P(Q) = \bigcup_{q_j \in Q} D_{q_j}$$

The set P_Q represents the union of the inverted lists associated with each of the query terms, (i.e., the set of documents which share at least one term in common with the query).

The set $R(Q, h)$ of the $r = |R(Q, h)|$ documents which *best* satisfy the query is given by

$$R(Q, h) = \{D_i | S(D_i, Q) \geq h; D_i \in P(Q)\}$$

where S is a similarity function which returns a real number such that a high value implies a high degree of resemblance and h is a threshold value. A ranked output can be obtained by arranging the retrieved items in decreasing order of query-document similarity as measured by the S-values.

2.4 The Retrieval Process

A straightforward procedure to obtain best match documents is to match the query against each of the documents in the collection, compute similarities, sort the similarities into descending order and take the r highest rank documents. Obviously, it would require $O(n)$ computations, which is impractical for large collections. More practical approaches are the cluster approach and the inverted approach which we discuss next:

Cluster Approach. In this approach, the collection is preprocessed and partitioned into clusters, each cluster containing similar documents. The first stage of the retrieval process is to find those similar clusters that are most significantly correlated with the given query and then the query is matched against each document contained in all identified clusters. Clustering has been applied successfully to web search retrieval and when applied to few document (top ranked) snippets could be effective. However for large collections of millions of pages; preprocessing is quite prohibitive.

Inverted File Approach. If the inverted file is available, it can be used to build a sparser Initial Document Concepts Matrix as the number of documents that must be considered is reduced. Most of the commonly used similarity functions that have been used in IR systems (i.e. Vector Space Model (VSM)) involve the terms in common between the query and the document. Hence,

$$S(D_i, Q) \neq 0$$

iff the query and the document vectors have at least one common term. Thus, we have to process only the set $D_i \in P_Q$ of documents which appear at least once in the postings corresponding to the query terms, while all other documents can be discarded.

The weight $w_{t_{ij}}$ which reflects the presumed importance of term t_j for qualifying the content of document D_i is defined as

$$w_{t_{ij}} = (0.5 + 0.5F_{ij})/F_{max_i}$$

where F_{ij} represents the occurrence frequency of the term t_j in the document D_j normalized by F_{max_i}, the maximum occurrence frequency among the terms associated with D_i [1]. The effect of such normalization is that longer documents do not produce higher term weights than shorter ones.

The same indexing and weighting process is performed on the text of the query; producing a query representation consisting of a set of pairs (q_j, w_{q_j}) with w_{q_j} denoting the degree of importance of the term q_j. The weight attached to each query term is determined also by its IDF (Inverse Document Frequency). For each term j, it is computed as $IDF_j = \frac{\log n}{n_j}$ where n is the number of documents in the collection and n_j is the number of documents in which the term j appears: $n_j = |D_{t_j}|$.

Using this approach, query terms are assigned weights inversely proportional to their frequency of occurrence in the collection. Thus, infrequently occurring terms are assigned larger weights than terms which occur in many documents. Furthermore, the local occurrence of a term within a document, reflects its *importance*, and the total occurrence of the same term within the collection reflects its *discrimination power*. In the next section, we discuss how to enhance the retrieval process further by using random projections for dimensionality reduction.

3 Enhancing the Retrieval Process

3.1 Reducing Dimensions Using Random Projection

Reducing dimensions is one of the key features that is used to uncover the underlying concepts. Random Projection main hypothesis is that high dimensional vectors chosen at random are *nearly orthogonal* and much computationally cheaper to produce than methods such as Singular Value Decomposition [10]. Singular value decomposition is the algorithm used in latent semantic indexing [6]. For an $M \times N$ matrix A, the full singular value decomposition complexity is quadratic [4].

3.2 Scaling Context Vectors Model

Because Random Projection uses nearly orthogonal context vectors that are be created independently of one another. This is crucial for indexing large collections of individual pages and computations can be distributed over a large network of computers.

3.3 Incremental Updates

Related to distributed or parallelizable model creation is the consideration of incremental updates. It is easy to update a basic Random Projection model incrementally: each time we add a new document, we create an independent new Random Vector for it, and it does not matter if this is batched separately from previous additions. Therefore, incremental addition of new terms and documents to the context vector models does not require rebuilding them from scratch.

4 Effectiveness Results

In this section, we present a complete evaluation of the relative effectiveness of the NETBOOK system versus the GOOGLE Desktop search engine.

The standard formulation for recall and precision [11] were used; if A is the set of relevant documents for a given query, and B is the set of retrieved documents, then

$$\text{Recall} = \frac{|A \cap B|}{|A|}$$

$$\text{Precision} = \frac{|A \cap B|}{|B|}$$

where "$|x|$" denotes the number of documents in set x.

To validate our NETBOOK technical choices, we have collected the **top ten searches** [15] in the areas of computer science, math, and physics. Then, for each search, we identified relevant pages by manually checking the book indexes. If the retrieved page has a search term in the index of the book pointing at the retrieved page, the page was judged "definitely relevant"; and if the book has the search item on other pages of the same chapter, it was judged "likely relevant"; otherwise, it was judged "irrelevant". All judgments were averaged and fed into the SMART evaluation package [11] at two cutoff points: 3 (likely relevant) and 4 (definitely relevant). The rational behind this relevance judgments is that authors know their material best.

Table 1. Total Retrieved, Total Relevant-Retrieved, Precision, and Recall for the NETBOOK system and the GOOGLE Desktop, out of 1298 Manually Judged Relevant

System	Retrieved	Rel-ret	Precision	Recall
NETBOOK	1873	901	**0.4810**	0.6941
GOOGLE	3814	632	0.1657	0.4869

Table 2. Average Retrieved, Average Relevant-Retrieved, Precision, and Recall for the NETBOOK system and the GOOGLE Desktop for Queries of Length 3, out of 161 Relevant

System	Retrieved	Rel-ret	Precision	Recall
NETBOOK	12.67	7.34	**0.5783**	0.0456
GOOGLE	22.27	6.78	0.304	0.042

Table 3. Recall Values for the NETBOOK system and the GOOGLE Desktop

System	Exact	at 5 docs	at 10 docs	at 15 docs	at 30 docs
NETBOOK	0.3631	0.0035	0.0126	0.0271	**0.0672**
GOOGLE	0.5320	0.0055	0.0110	0.0194	0.0346

In Tables 1 and 2, we show the number of retrieved documents, and the number of relevant retrieved documents, then we compute the precision and recall for each system and for question lengths of three terms. We notice that NETBOOK retrieved more relevant items than GOOGLE Desktop.

In Table 3, we show recall values for the NETBOOK system and the GOOGLE Desktop. The exact recall is the recall for exactly the retrieved document set, averaged over all queries (num_rel_docs with rank less than or equal to num_wanted / num_rel_docs). Also, we show recall values after 5, 10, 15, and 30 documents have been retrieved. Generally, we notice that the NETBOOK system generally achieves better recall values.

In Table 4, we show precision values for the NETBOOK system and the GOOGLE Desktop. The exact precision is the precision for exactly the retrieved document set (i.e., after num_wanted documents were retrieved). Also, we show precision values after 5, 10, 15, and 30 documents have been retrieved. We notice that the NETBOOK system generally achieves higher precision.

4.1 SAS Tests

Relevant Retrieved Test. The Relevant Retrieved test revealed a statistically significant difference between the for the NETBOOK system and the GOOGLE Desktop at the p = 0.01 level. We tested the hypothesis that there is no difference in the number of relevant documents retrieved by both systems. Table 5 shows the LSMEAN results of this metric for for the NETBOOK system and the

Table 4. Precision Values for the NETBOOK system and the GOOGLE Desktop

System	Exact	at 5 docs	at 10 docs	at 15 docs	at 30 docs
GOOGLE	0.33	0.49	0.45	0.47	0.46
NETBOOK	0.31	0.34	0.35	0.39	0.42

Table 5. Means for RelevantRetrieved for the NETBOOK system and the GOOGLE desktop system

GLM Procedure Least Squares Means				
System	LSMEAN	$Pr > \|T\|\ H0 : LSMEAN(i) = LSMEAN(j)$		
			1 (NETBOOK)	2 (GOOGLE)
NETBOOK	15.56	1	-	0.0002
GOOGLE	8.46	2	0.0002	-

Table 6. LSMEAN Results of the Relevant Retrieved Metric for Different Query Lengths

Question Length	GOOGLE	NETBOOK
2	6.3	16.1
3	8.2	23.2
4	4.3	16.5
6	6.3	18.4

GOOGLE Desktop. There is a significant difference between both systems. We observe that the NETBOOK system retrieved more relevant documents than the GOOGLE desktop system.

Also there is evidence that the Relevant Retrieved metric is dependent on the query length. The LSMEAN results are tabulated in Table 6. Query length 3 seems generally best.

NumRetrieved (Number of Documents Retrieved) Test. The NumRetrieved test revealed a statistically significant difference between the for the NETBOOK system and the GOOGLE Desktop at the p = 0.01 level. We tested the hypothesis that there is no difference in the number of retrieved documents by different systems. Table 7 shows the LSMEAN results of this metric for the Two retrieval systems. There is a significant difference between the NETBOOK system and the GOOGLE Desktop, as the NETBOOK system retrieved more documents.

Also there is evidence that the Relevant Retrieved metric is dependent on the query length. The LSMEAN results are tabulated in Table 6. As expected, the NETBOOK system retrieved more documents than the GOOGLE Desktop. In all cases, longer queries retrieved more documents.

Table 7. Means for NumRetrieved for the NETBOOK system and the GOOGLE Desktop

GLM Procedure Least Squares Means				
System	LSMEAN	$Pr > \lvert T \rvert \ H0 : LSMEAN(i) = LSMEAN(j)$		
			1 (NETBOOK)	2 (GOOGLE DESKTOP)
GOOGLE	16.06	1	-	0.0467
NETBOOK	25.75	2	0.0467	-

Table 8. LSMEAN Results of the NumRetrieved Metric for Different Query Lengths

Question Length	GOOGLE	NETBOOK
2	9.85	21.51
3	13.02	27.36
4	15.71	28.42
6	21.69	28.42

Table 9. Means for Average Precision for the NETBOOK system and the GOOGLE Desktop

GLM Procedure Least Squares Means				
System	LSMEAN	$Pr > \lvert T \rvert \ H0 : LSMEAN(i) = LSMEAN(j)$		
			1 (NETBOOK)	2 (GOOGLE DESKTOP)
GOOGLE	0.11	1	-	0.0303
NETBOOK	0.20	2	0.0303	-

Average Precision Test. Table 9 shows the LSMEAN results of the 11-point average precision test for the NETBOOK system and the GOOGLE desktop. There is a significant difference both systems at the 0.01 level.

5 Conclusions

We have described the design and implementation of the NETBOOK prototype system for collecting, structuring and using semantic information derived from full text ebooks available on the World Wide Web. Furthermore, we presented a complete evaluation of the relative effectiveness of the NETBOOK system versus the GOOGLE Desktop search system [8]. Our results shows clearly the increased effectiveness of our approach. We are planning to check the NETBOOK system against other available search engines such as Indri/Lemur in the near future.

References

1. Belkin, N.J., Croft, W.B.: Retrieval techniques. Annual Review of Information Science and Technology 22, 109–145 (1987)
2. Briggs, G., Shamma, D., Caas, A.J., Carff, R., Scargle, J., Novak, J.D.: Concept Maps Applied to Mars Exploration Public Outreach. In: Caas, A.J., Novak, J.D., Gonzlez, F. (eds.) Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping. Universidad Pblica de Navarra, Pamplona (2004)
3. Caas, A.J., Hill, G., Carff, R., Suri, N., Lott, J., Eskridge, T., Gmez, G., Arroyo, M., Carvajal, R.: CmapTools: A Knowledge Modeling and Sharing Environment. In: Caas, A.J., Novak, J.D., Gonzlez, F.M. (eds.) Concept Maps: Theory, Methodology, Technology, Proceedings of the First International Conference on Concept Mapping. Universidad Pblica de Navarra, Pamplona (2004)
4. Brand, M.: Incremental singular value decomposition of uncertain data with missing values. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 707–720. Springer, Heidelberg (2002)
5. Broder, A.Z.: On the resemblance and containment of documents. Compression and Complexity of Sequences (1997)
6. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
7. Edward, A.F.: Lexical relations: Enhancing effectiveness of IR systems. ACM SIGIR Forum 15(3), 5–36 (Winter 1980)
8. `http://desktop.google.com/`
9. Leake, D., Maguitman, A., Reichherzer, T., Caas, A., Carvalho, M., Arguedas, M., Brenes, S., Eskridge, T.: Aiding knowledge capture by searching for extensions of knowledge models. In: Proceedings of KCAP-2003. ACM Press, St. Augustine (2003)
10. Sahlgren, M.: The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. PhD thesis, Department of Linguistics, Stockholm University (2006)
11. Salton, G.: The SMART system 1961-1976: Experiments in dynamic document processing. In: Encyclopedia of Library and Information Science, pp. 1–36 (1980)
12. Van Rijsbergen, C.J.: The Geometry of Information Retrieval. Cambridge University Press, Cambridge (2004)
13. Widdows, D., Ferraro, K.: Semantic Vectors: A Scalable Open Source Package and Online Technology Management Application, Google Code (2009)
14. Hinrich Schutze, H.: Automatic word sense discrimination. Computational Linguistics 24(1), 97–124 (1998)
15. Spink, A., Wolfram, D., Jansen, B.J., Saracevic, T.: Searching the Web: The Public and their Queries. Journal of the American Society for Information Sciences and Technology 52(3), 226–234

Considering Patterns in Class Interactions Prediction

Nazri Kama[1,2], Tim French[2], and Mark Reynolds[2]

[1] Advanced Informatics School, Universiti Teknologi Malaysia, Malaysia
[2] Computer Science and Software Engineering, The University of Western Australia, Australia
{nazri,tim,mark}@csse.uwa.edu.au

Abstract. Impact analysis has been defined as an activity of assessing the potential consequences of making a set of changes to software artifacts. Several approaches have been developed including performing impact analysis on a reflected model of class interactions analysis using class interactions prediction. One of the important elements in developing the reflected model is a consideration of any design pattern that the software employs. In this paper we propose a new class interactions prediction approach that includes a basic pattern analysis i.e., Boundary-Controller-Entity (BCE) pattern in its prediction process. To demonstrate the importance of the pattern consideration in the prediction process, a comparison between the new approach (with pattern consideration) and two selected current approaches (without pattern consideration) were conducted. The contributions of the paper are two-fold: (1) a new class interactions prediction approach; and (2) evaluation results show the new approach gives better accuracy of class interactions prediction than the selected current approaches.

Keywords: impact analysis; requirement interactions; class interactions; pattern; traceability; requirement; class.

1 Introduction

A typical problem with software is that when changes happen to any part of the software, it may produce unintended, expensive or even disastrous effects [1,2]. Change impact analysis or impact analysis has been used to manage these problems [1-5]. According to Bohner [1], impact analysis is an activity of assessing the effect before making changes to the software. This activity is also known as a predictive impact analysis.

The predictive impact analysis is partitioned into two categories which are low level analysis and high level analysis. The low level analysis category focuses on predicting a change impact based on low level artifacts analysis e.g., class interactions analysis [2,3]. The class interactions analysis provides reasonable accurate results since the class represent final implementation of user requirements. However, despite giving good results, class interactions analysis faces several challenges: (1) it requires detailed understanding of the system and its implementation; and (2) the amount of information to be analyzed can be so overwhelming that it may lead the analysis results to error [6,7].

As an alternative, the high level analysis category has been introduced [6-10]. This category performs change impact prediction using high level artifacts analysis e.g., high level model [4-8]. One of the high level models is a class interaction prediction [7,8]. Results by performing the change impact prediction on the class interaction prediction are considered reflect to the low level analysis or actual class interactions analysis. Therefore, the accuracy of the class interaction prediction is critically important as it reflects the accuracy of the change impact prediction.

One of the techniques to develop the class interaction prediction is through reflection of significant object interactions in requirement artifacts [7,8]. The significant object refers to an object that has a traceability link with any class name in class artifacts. This technique is based on the precept that the interactions between significant objects in requirement artifacts reflect the actual class interactions in coding artifacts.

However, this technique faces a challenge when the software employs pattern or design pattern [9,10] in its implementation. This will make the reflection of the significant object interactions to class interactions is inaccurate as the design pattern class has no reflection with the significant object in requirement artifacts. For example in the Boundary-Controller-Entity (BCE) pattern [9], this pattern does not allow any interaction between Boundary and Entity classes. It creates a Controller class that acts as a mediator class to manage interactions between these classes. Since the Controller class is independently created to support the pattern implementation, this class has no reflection with any significant object in requirement artifacts. This situation leads to inaccurate reflection results.

To support the above challenge, we introduce a new class interactions prediction approach that includes pattern consideration in its prediction process. This approach composes two main steps which are reflecting the significant object interactions to predict actual class interactions and performing pattern analysis. As a preliminary work, we have selected the BCE pattern as the pattern to be considered in the prediction process.

This paper is laid out as follows: Section 2 briefly describes the related work. Then, Section 3 comprehensively explains the new class interactions prediction approach. Subsequently, Section 4 and Section 5 present the evaluation strategy and evaluation results. Thereafter, Section 6 discusses the results. Next, Section 7 presents the conclusion and future works.

2 Related Works

There has not been much work related to development of class interactions or class diagram from requirement artifacts. There are two categories of class interactions development which are requirement artifacts analysis and non-requirement artifacts analysis. For the requirement artifacts analysis, typically noun and noun phrase keyword analysis has been used to reflect the class name. Interactions between the keywords are then reflected to class interactions. Liang [11] proposes a use case goal analysis rather than a use case description analysis. Sharble and and Cohen introduce grammatical analysis [12]. There are two approaches for grammatical analysis which are data-driven which emphasis on information that the keyword possesses and responsibility-driven that focuses on services provided by the keyword. Premerlani [9] shows a structured approach using Unified Modeling Language (UML) object interactions diagram to build the object model.

For the non-requirement artifacts analysis, Bahrami [13] introduces "common class pattern" approach that is based on identification of various kinds of classes, of which a system will typically consist. Among the classes are physical classes, business classes, logical classes, application classes, computer classes and behavioral classes. Wirfs-Brock et al [14] show a Class-Responsibility-Collaborator (CRC) card that is used in brainstorming sessions. Each developer plays one or more cards where new classes are identified from the message passing between the developers.

3 A New Class Interactions Prediction Approach

The following Figure 1 shows the overall structure of the new approach.

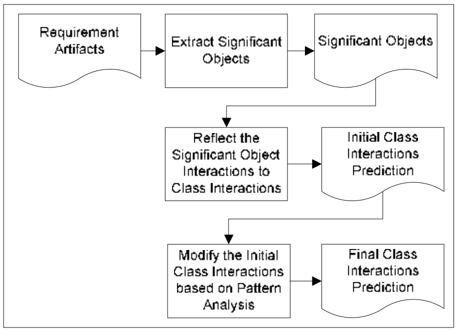

Requirement Artifacts
Extract Significant Objects
Significant Objects
Reflect the Significant Object Interactions to Class Interactions
Initial Class Interactions Prediction
Modify the Initial Class Interactions based on Pattern Analysis
Final Class Interactions Prediction

Fig. 1. Class Interactions Prediction Approach

Looking at the above figure, the new class interaction prediction approach consists of three main steps. The first step is to extract significant object from requirement artifacts. The extraction of the significant object is important as not all objects in requirement artifacts contributes to class interaction. As described earlier, the significant object refers to an object that only has traceability link with class name in class artifacts. The second step is to reflect the detected significant object and its interactions to develop an initial class interactions prediction. This reflection is based on the precept that the interactions between significant objects in requirement artifacts reflect the actual class interactions. Finally, the initial class interactions prediction is modified based on pattern analysis. The modification is based on the precept that design pattern class that exists in actual class interactions has no reflection from the significant object in requirement artifacts. A detailed explanation of each step is described in the following sub-sections:

3.1 Step 1: Extract Significant Object from Requirement Artifacts

This step focuses on extracting significant objects in requirement artifacts. As described earlier, the significant object is an object in requirement artifacts that has traceability link with any class name in class artifacts. We have developed a new traceability link detection technique that is based on similarity analysis concept between: (1) the object name in requirement artifacts and class name in class artifacts; and (2) the object name in requirement artifacts and class attribute name in class artifacts. If the analysis detects similarity in (1) or (2), a traceability link is considered exist and the object name is considered as the significant object.

There are two types of similarity analysis: (1) SA1: Similarity analysis between object name in requirement artifacts and class name; and (2) SA2: Similarity analysis between object name in requirement artifacts and class attribute name. In brief for SA1, there are three types of sub-analyses which are: (1) SA1.1: Similarity analysis of an object name that has three nouns with a class name; (2) SA1.2: Similarity analysis of an object name that has two nouns with a class name and; (3) SA1.3: Similarity analysis of an object name that has a noun with a class name. For the SA2, there are two types of sub-analyses which are: (1) SA2.1: Similarity analysis of an object name that has three nouns with a class attribute name; (2) SA2.2: Similarity analysis of an object name that has two nouns with a class attribute name. A detailed explanation of the similarity analysis concept can be found in [15].

3.2 Step 2: Reflect Significant Object Interactions to Class Interactions

This step focuses on reflecting the significant object interactions to class interactions. Prior to reflecting the significant object interactions, the interactions among the significant objects are detected. There are two situations where the significant object interactions can be detected.

The first situation is the detection of the significant objects interaction in a requirement description. Given an example of a requirement description (RD) "RD- The student registers any courses using their *StudentID*", this requirement description consists of three significant objects that are interacting which are *Student* object, *Course* object and *StudentID* object. According to the requirement description, we interpret interactions between the significant objects as *Student* object interacts with the *Course* object and the *Course* object interacts with the *StudentID* object.

The second situation is the detection of the significant objects interaction in two interacting requirements. Given an example of two interacting requirement descriptions "*RD1- the student must log in to the system using student card information*" and "*RD2- the system registers any selected courses by the student*", the RD1 and RD2 consist of two significant objects which are *Student* and *Student Card* objects for the RD1 and *Courses* and *Student* objects for the RD2. These two requirement descriptions are interacting between them since the *Course* object in the RD2 can only add a new *Course* after the *Student Card* in the RD1 has been successfully verified. Based on this situation, the *Course* object has interaction with the *Student Card* object.

Based on the detected significant object interactions, the reflection process is then performed. The following Figure 2 is used to describe the reflection process.

SO Interactions

	SO1	SO2	SO3
SO1	▨		
SO2	√	▨	
SO3	√	√	▨

Traceability link

SO Name	Class Name
SO1	CL1
SO2	CL2
SO3	CL3

Class Interactions Prediction

	CL1	CL2	CL3
CL1	▨		
CL2	√	▨	
CL3	√	√	▨

Fig. 2. Example of Reflecting from Significant Object Interactions to Class Interactions

Looking at the above figure, there are three interacting significant objects (SO) which are SO1, SO2 and SO3. The interactions among these significant objects are: (1) SO1 interacts with SO2; (2) SO1 interacts with SO3; and (3) SO2 interacts with SO3. The traceability link between the significant objects and the class (CL) artifacts are: (1) SO1 has traceability link with CL1; (2) SO2 has traceability with CL2; and (3) SO3 has traceability with CL3. Given the interactions and traceability link data, the class interactions prediction is then developed.

3.3 Step 3: Modify the Initial Class Interactions Prediction Based on Pattern Analysis

This step concentrates on modifying the initial class interactions prediction based on Pattern analysis. According to Gamma [16], Pattern is defined as a tested solution structure for common occurring design problems. By having a tested solution, it indirectly assists software developer to solve some common design problems. For example, one of the design problems is the difficulty to maintain a software system because of interdependencies of all components. The interdependencies cause ripple effects when a change is made anywhere. A high coupling classes cause difficulty or almost impossible to reuse them as they depend on various classes. Furthermore, adding a new data often requires re-implementing of business logic classes which then requires maintenance in various places.

One of the Patterns that solve the above problem is a Boundary-Controller-Entity (BCE) pattern [16-18]. This pattern separates the application classes into three categories which are Boundary class, Controller class and Entity class. The Entity class is a class that possesses data or business rules that access to and updates data from/to database. For the Boundary class, it is a class that renders the content of the Entity class and presents the content to user. Finally the Controller class is responsible for translating the message from Boundary class and passing it to the Entity class. By separating these classes, any changes to the application can be easily managed. For instance if change happens to user interface or Boundary class, the business logic class or Entity class will not be affected as these two classes are separated by the Controller class. Similarly if change happens to the Entity class, the Boundary class will not be affected.

We introduce three sub-steps to modify the initial class interactions prediction. As described earlier, we have selected the BCE pattern as the preliminary work for pattern analysis. The modification steps are: (1) Step 3.1: Classify class into Pattern class types; (2) Step 3.2: Categorize class into use case and; (3) Step 3.3: Establish Controller class interactions. Details of each sub-step are described as follows:

3.3.1 Step 3.1: Classify Class into Pattern Class Types

The BCE pattern consists of three types of classes which are Boundary class, Controller class and Entity class. The classification is done by reviewing role of each class. There are three types of roles that a class has potential to become the Boundary class [18]. The roles are: (1) a class that communicate with human user of a software system. This class is also known as user interface class; (2) a class that communicates with other systems or external system. This class is also known as system interface class or; (3) a class that communicates with devices to detect external events. This class is also known as hardware interface class. For the Entity class, there are two types of roles which are [18]: (1) a class that stores information or; (2) a class that performs business logic or information processing. Finally, a class that manages interactions between the Boundary class and Entity class is considered as the Controller class.

3.3.2 Step 3.2: Categorize Class into Use Case

To categorize class into use case, each class is reviewed to identify which use case that the class is coming from. To identify the use case, the detected horizontal traceability links between class and significant object in requirement artifacts (from Step 1) are analyzed. The analysis focuses on tracing which requirement artifacts that the class belongs to. However, some significant objects may exist in different requirement descriptions or different use case. If a class is coming from different use cases, the class can be categorized in any of those use cases.

3.3.3 Step 3.3: Establish Controller Class Interactions

There are two types of Controller class interactions which are: (1) interactions between Controller class and Boundary class and; (2) interactions between Controller class and Boundary class. These types of interactions are developed based on use case specification. The reason of developing the Controller class interactions based on the use case specification are: (1) a Controller class is created per use case specification and; (2) a Controller class is used to coordinate use case implementation [15,18].

Since the Controller class interactions are developed based on use case, Boundary and Entity classes need to be categorized into use case. The categorization is needed because of the Controller class interaction with the Boundary and Entity classes will be developed based on use case. After classifying the Boundary and Entity classes, the new Controller class interactions are then established. There are two scenarios where the Controller class interactions can be established: (1) Scenario 1: Establish Controller class interactions among the classes in a same use case and; (2) Scenario 2: Establish Controller class interactions across different use cases (also called inter-use case interaction).

Scenario 1: Establish Controller class interactions in a same use case. This scenario explains that the two types of Controller class interactions are established among classes (Boundary and Entity) in a same use case. This scenario could happen when the interacting Boundary and Entity classes reflect to the interacting significant objects belong to a same use case. There are two steps to establish the Controller class interactions in this scenario. The first step is to eliminate interactions between Boundary class and Entity class. The elimination is based on the BCE pattern interaction rules [18]. According to the rules, the pattern does not allow any interaction between Boundary class and Entity class and if exists, the interaction need to be eliminated. The second step is to establish the Controller class interactions. The interactions are: (1) from Boundary class to a Controller class; and (2) from Controller class to Entity class.

Scenario 2: Establish Controller class interactions across different use cases. This scenario is motivated by a scenario when any interacting classes (Boundary and Entity) that reflect to the interacting significant objects in which any of the significant objects exist in different use cases (during the categorization of class into use case-see Step 3.2). This scenario indirectly shows the class which reflects to the significant object that exist in different use cases will have interaction with Controller class that belong to those use cases.

4 Evaluation Strategy

The evaluation aims to compare the accuracy of class interactions prediction produced by the new approach and selected current class interactions prediction approaches. There are four elements have been considered in the evaluation strategy which are case studies, evaluation process and evaluation metrics. The following sub-sections describe detailed implementation of each element.

4.1 Case Study

Five software projects were selected to evaluate the capability of the improved approach. These software projects were developed by several groups of final year post-graduate students of software engineering course at the Centre for Advanced Software Engineering, Universiti Teknologi Malaysia.

4.2 Evaluation Process

There are three steps in the evaluation process. The first step is to extract software artifacts documentations versions from each software project. There are two sets of the extracted documentations which are design phase and coding phase versions. The reason of extracting these versions is to validate the effectiveness of the approaches (current and proposed) to develop class interactions prediction with and without design pattern class involvement. We assume that the design phase version consists of minimal design pattern class involvement as most design pattern class has not been developed yet. For the coding phase version, maximal design pattern class involvement where most of the design pattern classes have been developed. Each version consists of three types of software artifacts which are requirement artifacts and coding artifacts.

The second step is to develop class interactions prediction for each software project using selected current approaches and the proposed approach. We have selected two current class interaction prediction approaches which are Grammatical Analysis (GA) [11] and Use-Case Goal (UCG) [12]. Both approaches use reflection of object interactions concept to develop class interactions prediction. The main difference between the current approaches and the proposed approach is that the new proposed approach includes Pattern analysis in its prediction process.

The third step is to compare the current class interactions prediction approaches results with the proposed class interactions prediction approach results. We employed our previous developed set of evaluation metrics [7,8] to evaluate the accuracy of class interactions prediction.

4.3 Evaluation Metrics

This study has employed the evaluation metric. Briefly, each generated class interactions prediction can be assessed according to four numbers: NP-NI (the number of pairs of classes correctly predicted to not interacting), P-NI (the number of pairs incorrectly predicted to interacting), NP-I (the number of classes incorrectly predicted to not interacting) and P-I (the number of classes correctly predicted to interacting). These numbers is then used to calculate a Kappa value [19], which reflects the accuracy or the prediction (0 is no better than random chance, 0.4-0.6 is moderate agreement, 0.6-0.8 is substantial agreement, and 0.8-1 is almost perfect agreement).

5 Evaluation Results

The evaluation results are divided into two groups which are: (1) class interactions prediction results produced by current approaches (GA and UCG approaches) and; (2) class interactions prediction results produced by the proposed approach.

Results Produced by the Current Approaches (without Pattern Analysis)
The following Table 1 and Table 2 show the prediction results produced by the GA and UCG approaches accordingly.

Table 1. Prediction Results Produced By the GA Approach

Attribute	Design Phase Version					Coding Phase Version				
	P1	P2	P3	P4	P5	P1	P2	P3	P4	P5
NP-NI	98	95	98	106	119	253	258	302	287	274
P-NI	7	6	4	6	5	102	135	112	105	143
NP-I	5	7	6	4	6	110	105	131	132	110
P-I	100	102	102	115	123	201	243	275	256	253
Corr (%)	93	94	96	95	96	66	64	71	71	64
Com (%)	95	94	94	97	95	65	70	68	66	70
Kappa (k)	0.9	0.89	0.92	0.93	0.93	0.4	0.42	0.47	0.45	0.42

Table 2. Prediction Results Produced By the UCG Approach

Attribute	Design Phase Version					Coding Phase Version				
	P1	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P5**
NP-NI	94	90	101	102	110	244	284	297	287	276
P-NI	6	7	7	10	8	110	132	123	105	127
NP-I	8	6	9	6	6	104	98	114	132	123
P-I	102	107	93	113	129	208	227	286	256	254
Corr (%)	94	94	93	92	94	65	63	70	71	67
Com (%)	93	95	91	95	96	67	70	72	66	67
Kappa (k)	0.89	0.9	0.86	0.88	0.91	0.41	0.42	0.49	0.45	0.42

Iteration 1-Design Phase Results (GA and UCG): Both approaches show: (a) the correctness and completeness values across software projects indicate a high prediction value where more than two-third of the actual class interactions were predicted and; (b) The kappa values show an almost perfect strength of agreement between the class interactions prediction and the actual class interactions.

Iteration 2-Coding Phase Results (GA and UCG): Both approaches show: (a) the correctness and completeness values across software projects indicate a low prediction value where less than two-third of the actual class interactions were predicted and; (b) the kappa values show the approaches produce a moderate strength of agreement between the class interactions prediction and the actual class interactions.

5.1 Results Produced by the Proposed Approach (with Pattern Analysis)

The following Table 3 shows the prediction results produced by the proposed approach.

Table 3. Prediction Results Produced by the Proposed Approach

Attributes	Design Phase Version					Coding Phase Version				
	P1	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P5**
NP-NI	98	95	98	106	119	276	303	338	335	344
P-NI	7	6	4	6	5	53	61	45	56	34
NP-I	5	7	6	4	6	68	62	52	33	46
P-I	100	102	102	115	123	269	315	385	356	356
Corr (%)	93	94	96	95	96	84	84	90	86	91
Com (%)	95	94	94	97	95	80	84	88	92	89
Kappa (k)	0.9	0.89	0.92	0.93	0.93	0.68	0.71	0.80	0.81	0.82

Iteration 1- Design Phase and Iteration 2- Coding Phase Results: Both iterations results show (a) the correctness and completeness values across software projects indicate a high prediction value where more than two-third of the actual class interactions were predicted and; (b) Iteration 1 results: the kappa values show the approach produce an almost perfect strength of agreement between the class interactions prediction and the actual class interactions. Iteration 2: the kappa values show a substantial and an almost perfect strength of agreement in between the class interactions prediction and the actual class interactions.

6 Analysis of Results

Prior to analyzing the results, we construct the following Table 4 in order to show the number of developed design pattern class and kappa value results across all software projects. We then divide the analysis into two sections.

Table 4. Number of Developed Design Pattern Class and Kappa Value

	Iteration 1- Design Phase					**Iteration 2- Coding Phase**				
	P1	**P2**	**P3**	**P4**	**P5**	**P1**	**P2**	**P3**	**P4**	**P5**
No of Developed Design Pattern Class	0	0	0	0	0	13	11	10	9	8
Kappa Value	0.89	0.9	0.86	0.88	0.91	0.4	0.42	0.49	0.45	0.42

6.1 Current Approaches Iteration 1 Results vs. New Approach Iteration 1 Results Analysis

Looking at Table 1, Table 2 and Table 3 results, all Kappa values indicate a high prediction value. Also, the results show there is no significant different between results produced by the selected current approaches and the new approach. Relating the accuracy of the prediction results to the number of design pattern class (see Table 3), we would like to explain that there is no clear indication that the number of design pattern class affects the accuracy of the prediction results. This is because of there is no design pattern class involvement in the prediction. Therefore, we make an argument that all approaches manage to produce high accuracy of prediction results when there is no design pattern class exist in the actual class interactions.

6.2 Current Approaches Iteration 2 Results vs. New Approach Iteration 2 Results Analysis

Looking at Table 1, Table 2 and Table 3 results, both Kappa values for the current approaches indicate a low prediction value whereby the new approach indicates a high prediction value. Relating the accuracy of the prediction results to the number of design pattern class (see Table 3), we would like to explain that there is a clear indication that the number of design pattern class in the actual class interactions affects the accuracy of the prediction results. This can be seen by looking at the current approaches results. The results show that the accuracy of the prediction results (Kappa value) is lower when design pattern class exists in the actual class interactions.

Therefore, we make two arguments from this iteration results. The first argument is that the current approaches are not able to produce high accuracy of prediction results when there is design pattern class exists in the actual class interactions. The second argument is that the new approach gives high accuracy of prediction results than the current approaches when design pattern class exists in the actual class interactions. This argument indirectly supports the importance of pattern consideration in the class interactions prediction process.

7 Conclusion and Future Work

We have proposed a new class interactions prediction approach that composes two main steps. The first step develops an initial class interactions prediction through reflection of significant object interactions in requirement artifacts via traceability analysis. The second step improves the initial class interactions prediction through Pattern analysis. The difference with the proposed approach and current approaches is that the proposed approach includes the Pattern analysis in its prediction process.

Besides introducing the new approach, we have compared the capability of the new approach to predict class interactions with two selected current approaches. The evaluation results reveal that that the new approach gives better accuracy of prediction results than the selected current approaches. This evaluation results also indirectly show the importance of Pattern consideration in class interactions prediction process.

However, current application of the new approach is restricted to a software application that employs Boundary-Controller-Entity pattern. The extension to other patterns will be investigated by future work. Furthermore, a demonstration on performing predictive impact analysis using the new class interaction prediction will be included as well.

References

1. Bohner, S., Arnold, R.: Impact Analysis - Towards a Framework for Comparison. In: Proceeding of the IEEE International Conference on Software Maintenance, pp. 292–301. IEEE Press, Washington (1993)
2. Breech, B., Danalis, A., Shindo, S., Pollock, L.: Online Impact Analysis via Dynamic Compilation Technology. In: Proceeding of the 20th IEEE International Conference on Software Maintanence, pp. 453–457. IEEE Press, Illinois (2004)
3. Law, J., Rothermel, G.: Incremental Dynamic Impact Analysis for Evolving Software Systems. In: Proceeding of the 14th International Symposium on Software Reliability Engineering, p. 430. IEEE Press, Colorado (2003)
4. Li, Y., Li, J., Yang, Y., Mingshu, L.: Requirement-centric Traceability for Change Impact Analysis: A Case Study. In: Wang, Q., Pfahl, D., Raffo, D.M. (eds.) ICSP 2008. LNCS, vol. 5007, pp. 100–111. Springer, Heidelberg (2008)
5. Hassine, J., Rilling, J., Hewitt, J., Dssouli, R.: Change Impact Analysis for Requirement Evolution using Use Case Maps. In: Proceeding of the 8th International Workshop on Principles of Software Evolution, pp. 81–90. IEEE Press, Washington (2005)
6. Shiri, M., Hassine, J., Rilling, J.: A Requirement Level Modification Analysis Support Framework. In: Proceeding of the 3rd International IEEE Workshop on Software Evolvability, pp. 67–74. IEEE Press, Paris (2007)
7. Kama, N., French, T., Reynolds, M.: Predicting Class Interaction from Requirement Interaction. In: Proceeding of the 13th IASTED International Conference on Software Engineering and Application, pp. 30–37. ACTA Press, Cambridge (2009)

8. Kama, N., French, T., Reynolds, M.: Predicting Class Interaction from Requirement Interaction: Evaluating a New Filtration Approach. In: Proceeding of the IASTED International Conference on Software Engineering. ACTA Press, Innsbruck (2010)
9. Premerlani, J., Rumbaugh, J., Eddy, M., Lorensen, W.: Object-Oriented Modeling and Design. Prentice-Hall, Englewood Cliffs (1991)
10. Fowler, M.: Analysis Patterns – Reusable Object Models. Addison Wesley, Reading (2002)
11. Liang, Y.: From Use Cases to Classes: A Way of Building Object Model with UML. Journal of Information and Software Technology 45, 83–93 (2002)
12. Sharble, R.C., Cohen, S.S.: The Object-oriented Brewery: A Comparison of Two Object-oriented Development Methods. Software Engineering Notes 18, 60–73 (1993)
13. Bahrami, A.: Object Oriented Systems Development. McGraw-Hill, New York (1999)
14. Wirfs-Brock, R., Wilkerson, B., Wiener, L.: Designing Object-Oriented Software. Prentice Hall, Englewood Cliffs (1990)
15. Kama, N., French, T., Reynolds, M.: Considering Pattern in Class Interactions Prediction. Draft of Technical report, Computer Science and Software Engineering, UWA (2010), http://people.csse.uwa.edu.au/nazri/Techreport/ASEATechReport.pdf
16. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, Reading (2000)
17. Liyan, C.: Application Research of Using Design Pattern to Improve Layered Architecture. In: Proceeding of the International Conference on Control, Automation and Systems Engineering, pp. 303–306. IEEE Press, China (2009)
18. Jacobson, I., Booch, G., Rumbaugh, J.: The Unified Software Development Process. Addison-Wesley, Reading (1999)
19. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Journal of Educational and Psychological Measurement 20, 37–46 (1960)

Design of an Unattended Monitoring System Using Context-Aware Technologies Based on 3 Screen

Seoksoo Kim

Dept. of Multimedia, Hannam Univ., 133 Ojeong-dong, Daedeok-gu, Daejeon-city, Korea
sskim0123@naver.com

Abstract. More recently, the importance of a multimedia-based monitoring system, keeps growing for the purpose of preventing crimes and, thus, various systems are being released. Nevertheless, most systems are lacking an independent feature of detection but simply saves a video or image. In order to overcome problems, therefore, this study has developed an unattended monitoring system based on a 3 Screen service environment by combining image tracking and analysis technologies, which can extract a human from images, with context-aware technologies using behavior patterns of people tracked and data of a surrounding environment, fully utilizing the wireless network infrastructure that has been increasingly available.

Keywords: Context-Aware, Unattended Monitoring, 3 Screen, Monitoring System.

1 Introduction

After the release of iPhone, consumers are turning their eyes upon smart phones and companies expect introduction of smart phones can increase productivity, which allowed the smart phone market to experience an explosive growth during the past a few month. The manufacturers and service providers of mobile phones are making increasing efforts to offer more smart phone products.

Also, expression of Augmented Reality(AR) through a smart phone requires context-aware technology. In turn, the technology calls for accurate provision of product information after tracking a product and accurate understanding of the information by participants based on a smooth flow of product information [1,2].

Meanwhile, 3 Screen is a concept first held by AT&T which is a service environment that users can enjoy contents anytime and anywhere by connecting a TV, PC, or mobile with the Internet [3]. The technology constantly provides services no matter where the user is or what device he uses. And the contents do not need to be converted according to devices, for the technology can be applied to all formats.

Apple is one of the companies with the powerful potential of leading the 3 Screen market. With the release of iPhone, Apple has caused the smart phone craze to sweep the world and now holds a tremendous amount of contents available through 'App Store'. The company also operates 'iTunes' which offers music, games, movies, etc. and 'Mobile Me', allowing users to share pictures or files through synchronization of iPod Touch, iPhone, and a computer.

At present, Apple is planning to offer Apple TV for 3 Screen services. Although Apple has started from the hardware business, the company has proved its strong power in software also and has great potential of leading the 3 Screen service market.

Moreover, besides Apple, Microsoft views 3 Screen as one of 10 trends in 2010, recognizing the potential growth, and many IT companies desire to participate in the market. Along with the rising demand for 3 Screen services, therefore, this study aims to provide a solution environment which can ensure personal security and convenient services through an unattended monitoring system based on a platform of a company which has greater potential of leading 3 Screen service market such as Apple.

Fig. 1. 3 screen service provided by apple

Recently, the crime rates in the world as well as Korea show no sign of decline but only keep increasing. Thus, numerous security businesses are showing up and Table 1 depicts the structure of the industry [4].

Table 1. Structure of the security industry

	Examples	**Concepts**
Security Services	unattended electronic security system	Service of dispatching security agents when an alarm goes off
	Manned security system	Service of providing bodyguards or security agents staying in facilities
Security Equipment Manufacturing	Video security device	Manufacturing of video equipment for surveillance such as CCTV, DVR, or monitors
	Admission control	Smart card, biometrics, etc.
	Alarm	Various alarms and sensors
	Fire prevention	Fire alarms and sprinklers

Of various crimes, this study targets prevention of crimes related with properties. In addition, the system developed in this study is focused on personal security, which can be customized according to the needs of a user, not place security, which considers only a specific area such as a parking lot, an apartment, or a road.

2 Analysis of a Moving Object and Context Awareness

2.1 Calculation of Directions by Prediction of Locations

Image frames are input almost at regular intervals, so a travel distance between frames becomes a relative velocity. Thus, a distance to be traveled can be predicted by calculating a velocity and acceleration up to the present point [5.6]. For example, Figure 2 shows a specific object on the X-axis is moving to , , and in sequence while is a next point to be reached.

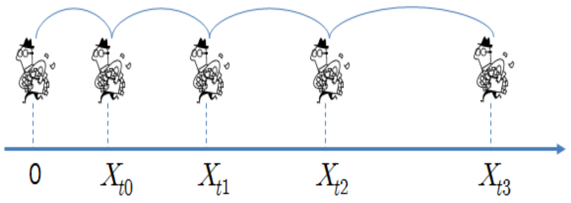


Fig. 2. Predicted locations of moving object

If the object is currently located at, Formula(1) for a travel distance between and is '-', which is a relative velocity. In the same manner, the velocity required for traveling from to can be calculated by Formula(2). Next, we can predict the location of by calculating the variation rate of the speed at . And Formula(3) can be obtained, for acceleration is calculated using a difference of velocities. Also, as we already know the velocity and acceleration at , the location of can be predicted as in Formula(4). Here, the direction is determined according to whether or not the calculated value is positive.

$$\text{Velocity at } X_{t1} = X_{t1} - X_{t0} \tag{1}$$

$$\text{Velocity at } X_{t2} = X_{t2} - X_{t1} \tag{2}$$

$$\text{Acceleration at } X_{t2} = \text{Velocity at } X_{t2} - \text{Velocity at } X_{t1} = X_{t2} - 2X_{t1} + X_{t0} \tag{3}$$

$$\text{Predicted location of } X_{t3} = \text{Velocity at } X_{t2} + \text{Acceleration at } X_{t2} = 2X_{t2} - 3X_{t1} + X_{t0} \tag{4}$$

2.2 Calculation of Color Values

Detection of a moving object through camera images, attempting to accept the real world as it is, causes a lot of variables and errors. Hence, in addition to movements, supplementary calculation should be done by using other features such as colors and shapes.

This study uses colors of an object in order for better application in case there is rotation or changes of an object [7.8]. While an object is detected by colors, ratios of colors are calculated for more precise tracking. As depicted in Formula(5), the RGB color model is used for color analysis. This is because the model is the most common way to analyze colors, and the color model is specifications of 3-D coordinates, expressing each color as a single point.

$$r = \frac{R}{(R+G+B)} \quad g = \frac{G}{(R+G+B)} \quad b = \frac{B}{(R+G+B)} \tag{5}$$

The almost limitless number of colors in the RGB model are divided into specific areas to ensure more effective extraction for detection of a moving object. Also, the RGB model is converted into the HSI color model, which is less sensitive to illumination, so that it can be strong against sudden changes of light.

The HSI model is composed of hue, saturation, and intensity, so it can further divide brightness. Formula(6) shows how to convert the RGB color model into the HSI model.

$$i = \frac{1}{3}(r+g+b) \quad s = 1 - \frac{3}{(r+g+b)}[\min(r,g,b)]$$
$$h = \cos^{-1}\left[\frac{\frac{1}{2}[(r-g)+(r-b)]}{\sqrt{(r-g)^2+(r-b)(g-b)}}\right] \tag{6}$$

2.3 Context-Aware Algorithm

Context awareness refers to offering of proper information or services to a user by detecting a change of situations or a system changing its conditions by itself. To make possible context awareness, a situation should be reasoned by using collected context data.

The method of reasoning is largely divided into Rule-Based Reasoning(RBR)[9] and Case-Based Reasoning(CBR)[10]. RBR method decides conditions to be met and searches for rules satisfied by the designated conditions in order to select the best rules. CBR method solves a new problem by referring to similar cases in the past. This method is effective to select services appropriate for a given situation.

3 Design of Unattended Monitoring System Based on 3 Screen

The system developed in this study has six steps as depicted in Figure 3.

- **Step 1:** Test bed for system implementation and development tools

Carrying out this study requires a test bed for system implementation/execution and an environment for development tools. Thus, the test bed consists of an emulator(on a PC) based on the iPhone OS platform of Apple, an iPhone(a smart phone), and a camera used for capturing images from a PC. On the other hand, the

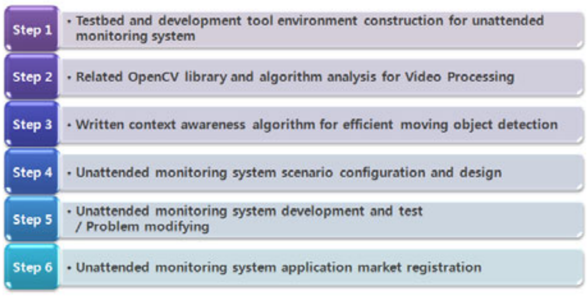
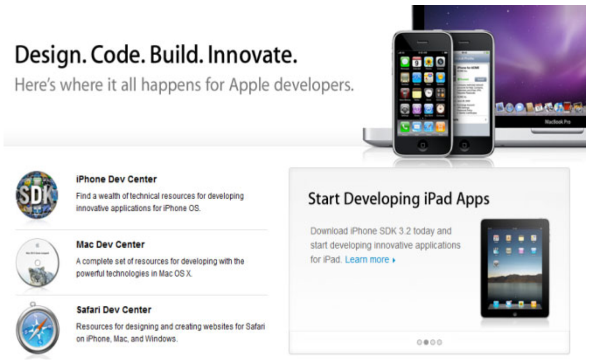

Step 1 · Testbed and development tool environment construction for unattended monitoring system
Step 2 · Related OpenCV library and algorithm analysis for Video Processing
Step 3 · Written context awareness algorithm for efficient moving object detection
Step 4 · Unattended monitoring system scenario configuration and design
Step 5 · Unattended monitoring system development and test / Problem modifying
Step 6 · Unattended monitoring system application market registration

Fig. 3. Six Steps of 3 Screen-Based Unattended Monitoring System


Design. Code. Build. Innovate.
Here's where it all happens for Apple developers.

iPhone Dev Center
Find a wealth of technical resources for developing innovative applications for iPhone OS.

Mac Dev Center
A complete set of resources for developing with the powerful technologies in Mac OS X.

Safari Dev Center
Resources for designing and creating websites for Safari on iPhone, Mac, and Windows.

Start Developing iPad Apps
Download iPhone SDK 3.2 today and start developing innovative applications for iPad. Learn more ▸

Fig. 4. SDK, Xcode, Supplied by Apple

development tools include Xcode depicted in Figure 4. The iPhone OS platform and Xcode are distributed through the home page of Apple and can be easily used.

- Step 2: Structure and design of the scenario of the unattended monitoring system
After establishing the development environment, the objective composition of the scenario and the overall structure of the system should be designed.

- Step 3: OpenCV library and algorithm analysis for image processing
OpenCV(Open Source Computer Vision) is a powerful image-processing library developed by Intel, which makes possible both a basic-level and a high-level image processing with a great number of algorithms. Thus, this study analyzes the library in order to find out the most effective algorithm for the unattended monitoring system and tracks a moving object.

- Step 4: Context-aware algorithm for effective detection of a moving object
A moving object is tracked and analyzed by using the selected algorithm, and a context-aware algorithm is written in order to deal with vulnerabilities that might arise from image processing. The context-aware algorithm is written by taking into account context information such as the location of a camera, surroundings, a movement pattern of an object, and so on.

- Step 5: Development/test of the unattended monitoring system and problem-solving
After all preparations are complete, development begins according to the system design. And after the development is done, complementary work should done in order to solve problems arising during tests.

- Step 6: Registration of the system in the application market
After the development and test of the system is complete, the system is registered in 'App Store', Apple's application market, and the system is further upgraded based on user evaluation.

4 Processing Flow the Unattended Monitoring System

'App Store' is a market used by a great number of consumers along with the recent craze for a smart phone in the world and the system developed in this study can be offered through the market, allowing many people to test the system. This can help to prevent national crimes as well.

The processing flow of the system is depicted in Figure 5. Of various methods of image processing technologies, this study uses a block-based method most widely used for effective movement tracking and correction. However, the method is quite weak in changes of sizes or pixels, rotation, noise, and so on.

Hence, a context-aware technology is used in this study, utilizing context data such as the location of a camera, surroundings, or a movement pattern of an object, in order to overcome such vulnerabilities.

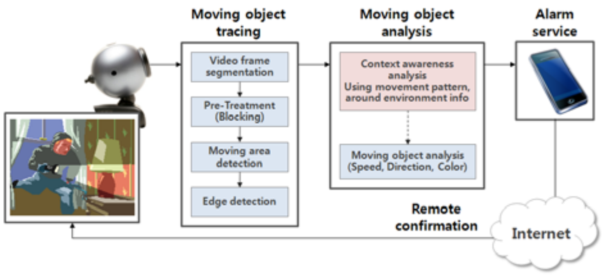


Fig. 5. The Overall Processing Flow of the Unattended Monitoring System

5 Conclusion

More recently, the importance of a multimedia-based monitoring system, keeps growing for the purpose of preventing crimes and, thus, various systems are being released. Nevertheless, most systems are lacking an independent feature for detection but simply saves a video or image.

This is ineffective because data should be reviewed in sequence when an intruder is detected and it takes a great amount of time for a manager to recognize an invasion. In addition, a lot of space is needed to save videos.

Therefore, in order to solve such problems, this study has developed a 6-step unattended monitoring system based on a 3 Screen service environment by combining image tracking and analysis technologies, which can extract a human from images, with context-aware technologies using behavior patterns of people tracked and data of a surrounding environment, fully utilizing the wireless network infrastructure that has been increasingly available.

Acknowledgement

This paper is supported by 2010 Korea Sanhak Foundation.

References

1. Mizoguchi, R., Ikeda, M.: Towards Ontology Engineering, Technical Report AI-TR-96-1, I.S.I.R., Osaka Universiy (1996)
2. Oh, J., Min, J., Hu, J., Hwang, B., Bohee, H.: Development of Video-Detection Integration Algorithm on Vehicle Tracking. The KSCE Journal of Civil Engineering 29(5D), 635–644 (2009)
3. Wall Street Journal, Phone Giants Roll Out ‘Three Screen’ Strategy (2008)
4. Kim, J.H.: Growth Strategy of Security Industry. SERI Business Note No. 4 (2009)
5. Broggi, A., Bertozzi, M., Fascioli, A., Sechi, M.: Shape-based pedestrian detection. In: Proc. IEEE Intelligent Vehicles Symposium 2000, pp. 215–220 (2000)
6. Peterfreund, N.: Robust Tracking of Position and Velocity With Kalman Snakes. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(6), 564–569 (1999)
7. Cheng, H.D., Sun, Y.: A hierarchical approach to color image segmentation using homogeneity. IEEE Transaction on Image Processing 9(12), 2071–2082 (2000)
8. Gonzales, R., Woods, R., Eddins, S.: Digital Image Processing Using MATLAB. Pearson Prentice Hall (2004)
9. Kim, J.P., Kim, M.H.: Rule based Inference Engine for System Context awareness. ICUIMC, pp. 29–37 (2007)
10. Chanchien, S.W., Lin, M.: Design and Implementation of a Casebased Reasoning System for Marketing Plans. Expert Systems with Applications 28, 43–53 (2005)

Requirements Elicitation Using Paper Prototype

Jaya Vijayan[1] and G. Raju[2]

[1] Lecturer, Department of computer science, Rajagiri College of social sciences, Kalamassery, Cochin
[2] Professor, Kannur University, Kannur, India

Abstract. Requirements engineering is both the hardest and critical part of software development since errors at this beginning stage propagate through the development process and are the hardest to repair later. This paper proposes an improved approach for requirements elicitation using paper prototype. The paper progresses through an assessment of the new approach using student projects developed for various organizations. The scope of implementation of paper prototype and its advantages are unveiled.

Keywords: Requirements Engineering, Elicitation, Paper prototype.

1 Introduction

Developing large & complex software application is very challenging, it is quite different from one-time programs where author and user are the same. Even though we have made significant progress in software development, we still face some challenges that we have experienced in the past. Some of the reasons for failure of projects are Schedule slippage, Cost over-runs, poor quality of software, poor maintainability. In this context the software engineering practices have great significance.

Requirements engineering is an important and fundamental aspect of software development. Regardless of the techniques used the basics still remains the same. Requirements engineering is described in 5 steps

- Requirement elicitation
- Requirement Analysis
- Requirement Specification
- Requirement Validation
- Requirements Management

The organization of the paper is as follows:

Section 2: Requirements elicitation & Gathering process.
Section 3: The Challenges in requirements elicitation & Analysis and also guidelines to do the elicitation activity.
Section 4: The paper prototype approach
Section 5: Explanation of the Case study.

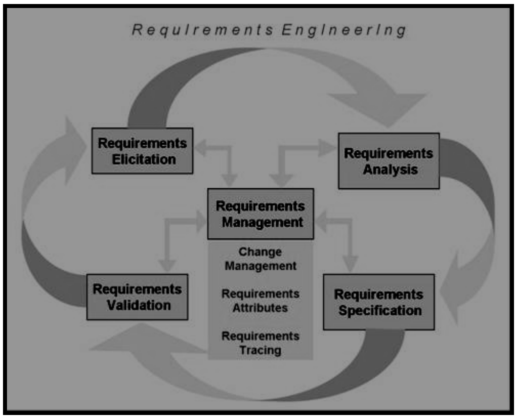


Fig. 1. The requirements engineering process

Section 6: Performance analysis of the proposed approach.
Session 7: Summary & Conclusion.

2 Requirements Elicitation and Gathering

The elicitation of requirements is perhaps the activity most often regarded as the first step in the RE process. One of the important goals of elicitation is to find out what problem needs to be solved [4].

The choice of elicitation technique depends on time & resources available to the requirements engineer, and of course the kind of information that needs to be elicited. We distinguish a number of classes of elicitation techniques as follows:

- Interviewing and questionnaires
- Requirements workshops
- Braining Storming and idea reduction
- Storyboards
- Use Cases
- Role Playing
- Prototyping

3 Challenges in Requirements Elicitation and Analysis

Requirements elicitation is both the hardest and most critical part of software development, since errors at this beginning stage propagate through the development process and are the hardest to repair later. Requirements elicitation is a difficult process in which one has to deal with ambiguity, informality, incompleteness and inconsistency, in which the “knowledge” of the requirements is not clear.

Problems in requirements elicitation

Errors in requirements elicitation are, overall, most serious in software development, and the hardest to repair. 70% of the systems errors are due to inadequate system specification [1]

Classification of elicitation problems

- **Problems of scope.** The boundary of the system is ill-defined, so that unnecessary design information may be given, or necessary design information left out.
- **Problems of understanding.** Users have incomplete understanding of their needs; analysts have poor knowledge of the problem domain; user and analyst speak different languages (literally or figuratively); "obvious" information may be omitted; different users may have conflicting needs or perceptions of their needs; requirements are often vaguely expressed, e.g., "user friendly" or "robust".
- **Problems of volatility.** Requirements evolve over time, either because of changing needs or because of changing perceptions by the stakeholders [1].

3.1 Requirements Engineering Some Guidelines

The analyst has to consider the following while doing the requirements gathering:

- Assess the business & technical feasibility for the new system.
- Identify the people who will help specify requirements
- Define the technical environment into which the system or product will be based
- Identify the domain constraints
- Define one or more requirements elicitation methods.
- Encourage participation from many people so that requirements are defined from different points of view.

4 Method Overview

Prototyping is one of the techniques used in requirements engineering. Instead of expensive Prototypes, ***a throwaway paper prototype*** method is suggested for requirements engineering..A paper prototype is a visual representation of what the System will look like. It can be hand drawn or created by using a graphics program. Usually paper prototype is used as part of the usability testing, where the user gets a feel of the User Interface.

The entire approach is divided into the following Steps:

- Domain Knowledge acquisition
- System understanding
- Requirements elicitation
- Prototype Validation
- Requirements Stabilization

4.1 Domain Knowledge Acquisition

As a prerequisite to the elicitation activity the analyst have to perform a domain study. He/She has to understand the basic terms and processes in the domain before moving on to details of the system.

4.2 System Understanding

The business analyst has to procure knowledge about the existing system through the manuals and users of the system. Once the system study is complete the analyst will get a fair idea about the system to be developed and can proceed for the elicitation and requirements gathering process.

4.3 Requirements Elicitation

After the initial meeting with the user and the preliminary investigation about the system the analyst gets a fair idea about the major requirements of the system. He/She has to accordingly build a throwaway paper prototype of the system. The paper prototype should concentrate on the main requirements and is to be presented before the users. The analyst with the aid of the prototype discusses with the user and gathers more requirements. The gathered requirements are recorded in a tabular format in the paper which will be used for tracing back in the next iterations. Details of the users, the requirement details, origin of the requirement and feedback details are recorded in the tabular format. The analyst gets the feedback from the users there itself. If the analyst comes across conflicting feedbacks on the requirements He/She has to discuss with the users who have given the conflicting views and has to come to a consensus.

4.4 Paper Prototype Validation

Once requirements have been gathered, it is categorized and organized into related subsets. The activity in this step is validation of prototype. The prototype is validated for omissions, ambiguity etc...

4.5 Requirements Stabilization

The paper prototype is changed according to the user feedback and clarifications. It is refined till the user is satisfied with the prototype and all the requirements are gathered. Once the prototype is finalized requirement are stabilized. The requirements specification is done after the stabilization process and the prototype can be discarded. Throwaway prototyping is suggested because once the requirements are gathered the prototype is thrown away and the rest of the phases are performed as usual so that the quality of the system is maintained.

Before finiliasing the requirements an approval from the users can be sought so that the amount of rework on requirements can be minimized. A sample paper prototype is given below.

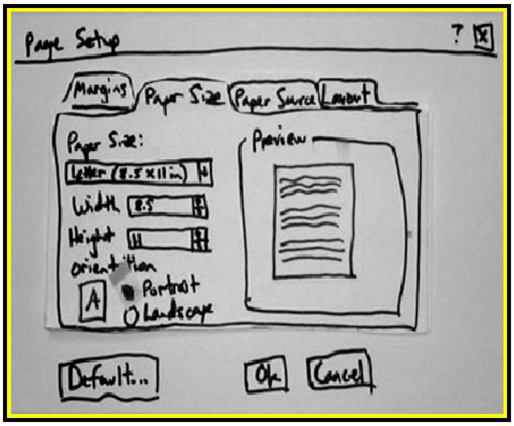


Fig. 2. A sample Paper prototype of the Page setup Dialog of Microsoft windows)

5 Case Study

In order to examine the merits of using paper prototype for requirements elicitation technique, several case studies were conducted through student projects. We have trained a group of Post graduate students in using the paper prototype approach and have made them to use the same in their projects. The data from one of the studies is presented in this paper in order to help analyze the usefulness of this technique.

"Safety and Compliances Software" Package called abcSAC Suite is an integrated software solution for helping organizations in USA, to stay in compliance with their government's safety regulations for specific categories and activities, along with helping them track various other desirable safety related issues The project consists 40 different modules. The modules selected for the project are Worker Safety, Exposure Management, Training Management. This package is developed for ABC Systems, Inc., Connecticut, USA. abcSAC suite is being developed as a windows application using Microsoft .Net technology, CSLA framework and Microsoft SQL Server 2005 technologies. It will be capable to reduce the risk factors in organizations by using this application to schedule trainings to its employees or other staff in proper time by analyzing existing database, on various safety and related issues. ABC Systems Inc. expects that after the full development of "Safety and Compliances Software" Package, they will be the big player in the USA Safety and Compliance Software domain.

a) Worker Safety Module

Typical applications fall into two categories named as OSHA Compliance Applications and Safety Program Applications. The first category involves applications designed to help meet the United States government's OSHA (Occupational Safety and Health Administration, U.S. Department of Labor) requirements and regulations, and the second category of applications are those that can help with instituting work place safety programs that help analyze and reduce future worker injuries and incidents.

b) Exposure Management

The purpose of the Exposure Management System is to provide a facility to help identify whether people are exposed to specific hazards and whether they have received appropriate training. The Exposure Management System works cooperatively with the Training Management System to accomplish its goals. Exposure Module consist the following processes.

1. Exposure Identification and Entry
2. Exposure Reviews and Reports

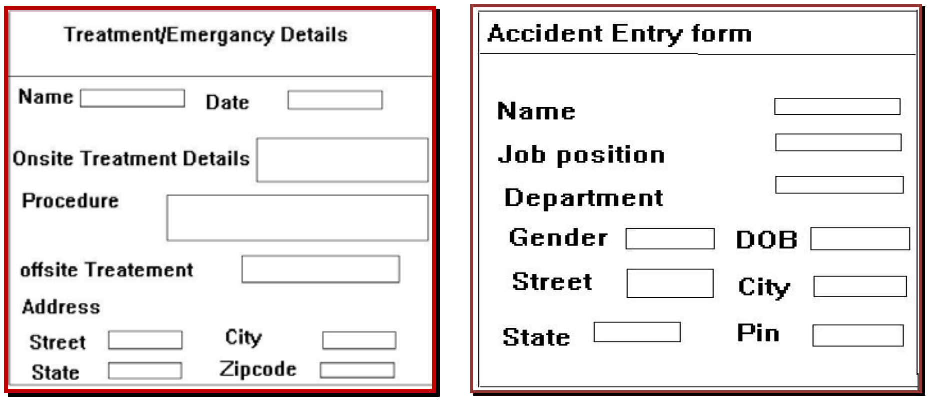
c) Training Management

The purpose of the Training Management System is to provide a facility to help identify whether people are exposed to specific hazards and whether they have received appropriate training. The Training Management System works cooperatively with the Exposure Management System to accomplish its goals. This means whenever a new type of accident occurs in the organization, they are getting the exposure to its various effects (when it happened, reasons for this accident, how long it lasted, whom it is affected etc….). That is, once an accident happens, the organization is getting familiar to the different consequences of it. And they are planning to prevent the next occurrence of the same accident again. So they will give proper training to the people. Training Management module consist the following processes.

a. Training
b. Training Report Generation

The Users who took part in the elicitation process were the technical staff of the organization.

The process for this project was completed in 1-3 iterations and the amount of change in requirements was less than 20%.Some of the sample paper prototypes used in this project are as follows



Treatment/Emergancy Details

Name Date

Onsite Treatment Details

Procedure

offsite Treatement

Address

Street City

State Zipcode

Accident Entry form

Name

Job position

Department

Gender DOB

Street City

State Pin

Fig. 3. Samples of the Paper prototype used for the elicitation activity of this Project

6 Performance Analysis of the Proposed Approach

We have conducted a study based on a group of students who have used paper prototype for requirements gathering. Feedbacks were gathered from students who have used the paper prototype for elicitation activity. A questionnaire was used for the assessment of the approach. The questionnaire addressed the openness of the users towards the approach, the number of iterations taken to complete the requirements gathering, the % of rework required at the design phase and also the effectiveness of the method.

Most of the respondents have done medium sized database projects. About 60% of users were partially open towards the use of this method for elicitation and 40% were open. Most of the respondents have completed their elicitation activity in 1-3 iterations. Only few have taken more than 3 iterations to complete their elicitation. The amount of rework suggested by the respondents was between 10-15% only. Majority of the users and respondents have strongly recommended this method for elicitation for small and medium sized projects.

6.1 Advantages

The participants are not caught up in the look of the system, Paper allows imagination to work. There is no technology barrier for the user; he/she can easily understand and use the paper prototype. A more user friendly way of requirements elicitation.

7 Conclusion

In this paper we have proposed a novel approach of using paper prototype for requirements elicitation. The paper focuses on the effectiveness of using Paper Prototype method for requirements elicitation and the major benefits of using the same. The paper prototype technique has been analyzed with the help of a group of student who has adopted this method for requirements Elicitation. We acknowledge that the data from the projects might not be sufficient to conduct the test, however, the feedback of persons who have used the method is an indicator that well-planned and meticulously used paper prototype will be an effective technique for elicitation for small and medium sized projects. The analysis of the approach has indicated that the paper prototype method for requirements elicitation is a suitable method for Small and medium sized projects. The method need to be tested for industry projects for further results.

Our further research is extended to find out the following

- Whether the technique can be applicable for all Categories of projects?
- Can it be effectively employed for small, medium and large projects?
- The Cost & time factor of the approach.

References

1. Rajagopal, P., Lee1, R., Ahlswede, T.: A New Approach for Software Requirements Elicitation. In: Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks, SNPD/SAWN 2005 (2005)
2. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: International Conference on Software Engineering, Proceedings of the Conference on the Future of Software Engineering
3. Jiang, L., Far, B.H., Eberlein, A.: Combining Requirements Engineering Techniques – Theory and Case Study. In: Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS 2005 (2005)
4. Herlea Damain, D.E.: Challenges in requirements engineering. Computer Science Technical Reports
5. Pressman, R.S.: Software engineering A practitioners approach. Mc-Graw Hill International Publication, New York
6. Sommerville, I., Sawyer, P.: Requirements Engineering – A good practice guide. John Wiley & Sons Ltd., Chichester
7. `http://www.goldpractices.com/practices/rto/index.php`
8. `http://dwmaster.raleigh.ibm.com/dwcontent/developerworks/library/library-html`

Quality-Driven Architecture Conformance[*]

Jose L. Arciniegas H.[1] and Juan C. Dueñas L.[2]

[1] Universidad del Cauca, Calle 5 # 4-70 Popayán, Colombia
jlarci@unicauca.edu.co
[2] Universidad Politécnica de Madrid, Ciudad Universitaria (s/n), CP: 28040, Madrid, Spain
jcduenas@dit.upm.es

Abstract. This paper describes the Quality-driven Architecture Conformance (QAC) discipline. Conformance is a particular type of assessment; in this case the architecture is compared with respect to a standard. Conformance process determines the degree of fulfillment of the architecture against a specific standard. QAC guarantees the compatibility and transferability of a system. The main contributions of this paper are: The definition of the QAC conceptual model, a workflow for the application of QAC and a summary of QAC techniques and tools. A case study has been executed in order to validate the QAC discipline.

Keywords: Software Architecture, Assessment, Conformance, Process, Quality.

1 Introduction

Conformance is and has been applied at implementation level. In that case, it is executed in order to measure the degree of fulfillment of the solution (implementation) against a standard, regulation or other specification. Conformance has a big incidence in the business area, because it gives confidence to users, consumers, manufactures, service providers and regulators. In some cases conformance is made obligatory by government regulations in order to check whether products, services, material, processes, systems and personnel measure up to the requirements of standards, regulations or other specifications [1].

We define architecture conformance as follows:

"Software architecture conformance is a type of assessment, where the architecture is compared with respect to a standard. Conformance process determines the degree of fulfillment of the architecture with respect to the standard".

So architecture conformance guarantees compatibility and transferability. QAC can be used as:

- a hold for the identification of non-functional requirements or system quality factors, described by quality models such as ISO-25010 [2],

[*] The authors wish to thank the Spanish company Telvent for their support in this work. The work has been partially granted by the ITEA MOSIS project, the Spanish Ministerio de Industria, Turismo y Comercio and REVVIS project of CYTED.

- a way to compare some specific architectural aspects of the system against reference architectures published by standardization bodies (understanding them in a broad sense) by means of architectural conformance checking,
- a means to externalize the detailed design, implementation and test of the assets or services in the architecture that are closer to the reference architectures for the intended architectural aspects,
- a way to adapt the evolution of the system architecture to the evolution of reference architectures for some aspects, or
- a vehicle to support the dynamic evolution of running systems after deployment and keep their architectures updated.

2 QAC Conceptual Model

QAC is a quality-driven discipline that proposes methodological guidelines for architecture conformance. In the same manner than Software Architecture Assessment, Architecture Conformance guarantees the quality of solutions with respect to other candidate architectures and allowing a rapid feedback. In addition, the conformance shows the coincidences and differences between the candidate architecture and the standard architecture. Usually, the standard has been proposed in order to guarantee a level of quality. The standards are agreements by international or national organizations, where the most usual ad-hoc concepts or practices are formally specified. Standards are a basis for comparison, a reference point against other things can be evaluated. They set the measure for all conformance process. In the telecommunication and computer science areas, some international organizations in charge of standardization processes are IEEE, ANSI, ISO, OMG, W3C, IETF, and others. Standards are continuously reviewed and sometimes updated in concordance with innovations.

In Fig. 1 we have outline the typical content of a standard. This schema is used only as general template for conformance processes. For example, some standards define only concepts, others defines process (practices), etc. A standard is a document where there are included *assertions* about a topic, and the assertions can be mandatory or optional. The assertions can be classified as: *basic* if affect only to a specific basic element, or can be *general* if they affect to more than one element. In the standard are defined the *basic elements* that should be included in the architecture. In addition are defined the *relationships* between them.

A standard is specified for a certain domain; therefore the *context* in this domain is clearly defined. If it is required, some concepts and their notation, conventions and external conditions are defined. The external conditions can contain rules or legal policies, physical constrains, etc. Usually, standard also defines some *practices*; they are product of practical experiences. Practices can be processes, guidelines, patterns or scenarios than have been proved by the industry in several real implementations.

For example, in the next paragraph from OSGi specification [3]:

"*In the OSGi Service Platform, bundles are the only entities for deploying Java-based applications. A bundle is comprised of Java classes and other resources which together can provide functions to end users and provide components called services to other bundles, called services. A bundle is deployed as a Java ARchive (JAR) file. JAR files are used to store applications and their resources in a standard ZIP-based file format.*"

We can identify:

- Assertions: "*bundles are the only entities for deploying Java-based applications*"
- Basic Elements: "*bundles*", "*classes*", "*resources*", "*services*", etc.
- Relationships: "*A bundle is comprised of Java classes and other resources*"
- Practices: "*JAR files are used to store applications and their resources in a standard ZIP-based file format*", the standard in this case uses a pattern, (standard format).
- Context: "*In the OSGi Service Platform,…bundles… provide functions to end users… called services*".

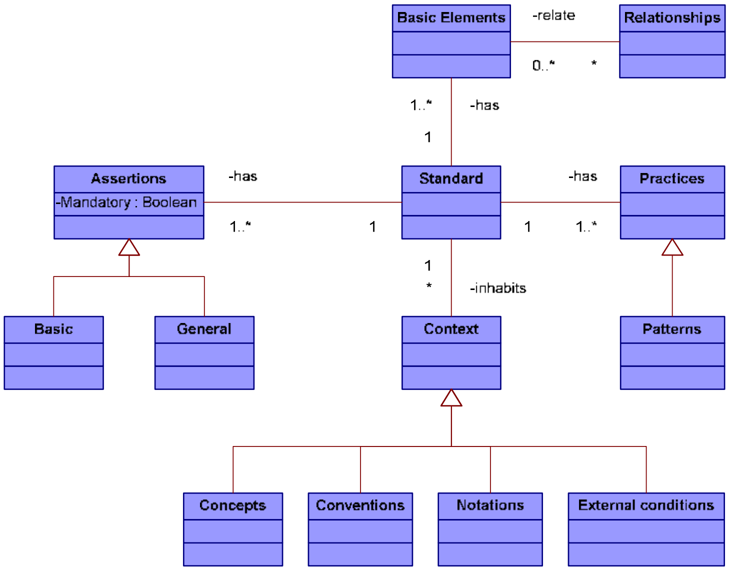


Fig. 1. Typical structure of a standard

QAC considers assets as basic element of comparison. Two types of assets should be treated independently, Significant Candidate Assets (SCA) and Significant Standard Assets (SSA). Basically, the differences between them are their origin: SCA from the candidate architecture and SSA from the standard architecture (see Fig. 2a). The conformance process needs SCA and SSA to compare and identify differences and coincidences, some methods and techniques can be used in order to achieve this objective (see section 4).

Conformance is a type of assessment (see Fig. 2b). During architecture conformance a special validation is performed (*compliance*), which has a simple objective, to locate *commonalities* or *differences*. Commonalities (SCA ∩ SSA) correspond to the set of assets that have been defined in the candidate architecture in the same way than in the standard. Obviously, uncommon assets are located in the differences. The differences take an interest value when the stakeholders take decisions. They are:

- Proposal for enhancement of SCA (SSA-SCA): as a product of the difference between SSA and SCA, new requirements are identified and some lacks can be located in the candidate architecture.
- Proposal for enhancement of the standard (SCA-SSA): as a product of the difference between SCA and SSA, some lacks may be located in the standard; it is a frequent case when the candidate architecture goes beyond the scope of the standard.

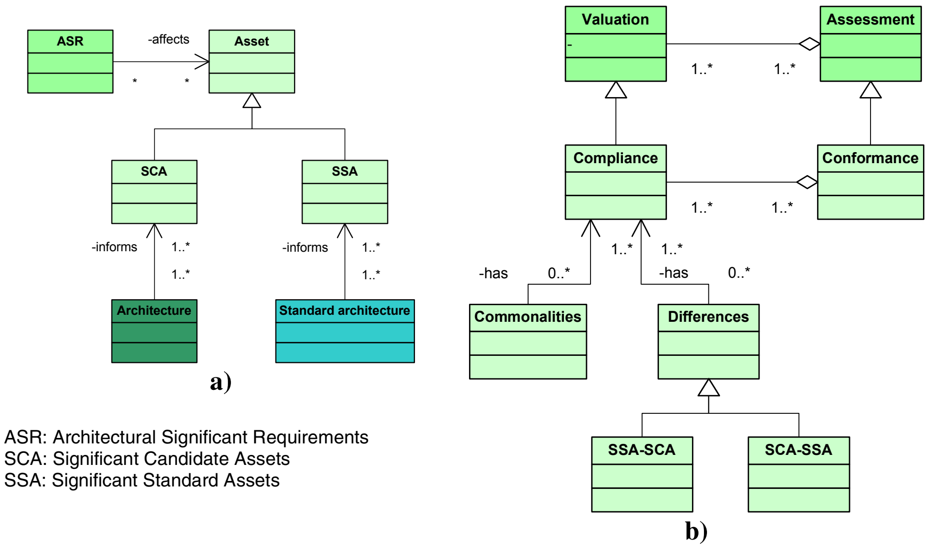


Fig. 2. Relationship between Assets, Conformance and Assessment

The main result of architecture conformance is reported by comparing some alternatives of solution (pros/contras) with respect to certain standards. QAC does not consider the complete architecture to be only one view concerned to specific SCA. Other secondary outcomes can be obtained from QAC, for example: a better understanding of the architecture, better communication with the stakeholders, detection of architecture limitation and risk and others.

3 QAC Workflow

The Fig. 3 shows the workflow of QAC. The following activities have been defined:

Preparation
Usually, a standard is a detailed and formal specification; some of them are voluminous documents. At least one member of the staff should have knowledge about one or more standards. Sometimes, there are several standards related with the same topic or quality attribute. So an expert or at least a person with the minimum knowledge about the standards is required. After that, during the preparation step, one or a small set of standards should be chosen in order to estimate scope, impact, cost, planning and duration.

Prioritizing requirements and filtering

These two phases should be performed for both the candidate architecture and the standard architecture. At the end, two prioritized list of ASR are obtained by conforming the SCA and SSA.

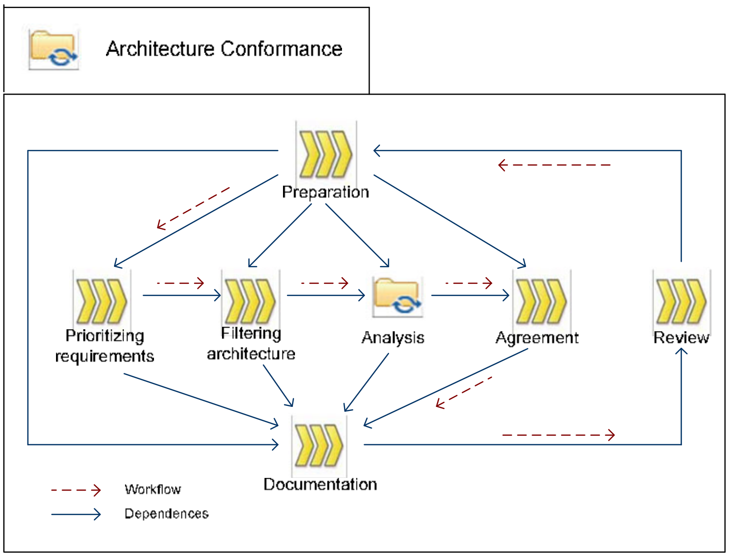


Fig. 3. QAC workflow

Analysis

Several types of conformance analysis could be performed (See Fig. 4). No order is established, because it depends of the objectives of the architecture conformance and the information defined in the standard. The conformance relates to the standard; that means nothing out of the standard can be assessed for conformance.

Context conformance, in this case only concepts, notations, conventions and external conditions are verified. This kind of conformance is often used for specific standard language, or in a process where is used a specific notation. The critical part is "concept conformance" because semantic conformance is required. Notation, conventions and external conditions can be analyzed using syntactic conformance. Context conformance has as result compliances with respect to concepts, notations, conventions and external conditions, but in addition, other important results are detected such as new concepts, inconsistencies and similarities can be found in the candidate architecture.

Architectural asset conformance, it is a validation where the assets are checked. Two issues should be validated; presence of the asset and verification of that asset is in conformance to the standard specification. Architectural asset conformance has as result: the list of assets in conformance, list of missed assets, and a list of assets that are in the architecture but their description is not in conformance. In addition new assets can be found in the candidate architecture, they have special interest because need a special justification.

Relationship conformance, at the same way the relationships among assets should be verified. In first place the presence of this relationship and for other way the consistency, type of relationship, navigability, visibility, multiplicity, etc. Relationship conformance has as result: the list of relationship in conformance, missed relationships, and inconsistency of some relationships. At the same as assets, new relationships can be defined.

Practice conformance. Standards define also processes, guidelines, patterns, etc. that can be used in the candidate architecture. In some cases are recommendations that should be analyzed in the specific context. For example ISO 9000 [1] and CMMI [4] specifications defined some practices that should be used during development in order to guarantee quality of the products. Practice conformance has as result the conformance or not of one specific practice.

Assertion conformance. It is very similar to requirement assessment, because assertions in practice are special requirements that must be supported in the architecture. Assertion conformance has as result the conformance or not of one specific assertion.

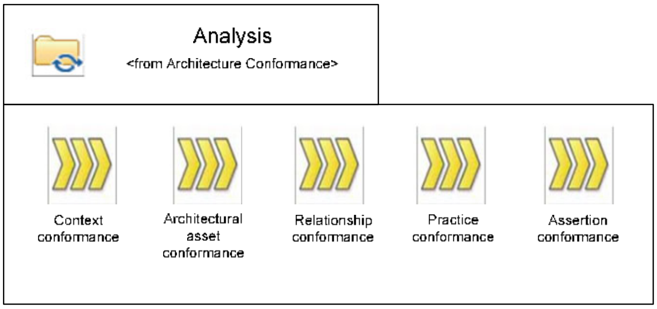


Fig. 4. Architecture conformance analysis

Agreement and documentation

During agreement and documentation the main results of QAC should be reflected, that is, the commonalities and differences. In addition, the concerns, trade-offs and decisions must also be included. The concerns obtained in QAC become relevant inputs to improve future architectures or standards.

Review

As results of concerns and possible trade-offs and decisions, the process can be reviewed several times if is required. The process of revision takes importance for the evolution of the software, in this case in two directions: evolution of the software architecture and evolution of the standards. Architecture conformance is a good mechanism to learn and enhance previous experiences.

4 QAC Methods and Techniques

Some methods and techniques from architecture assessment could be used for architecture conformance, for example ATAM [5], SARA [6] and BAPO [7].

However, a reduced set of methods have been proposed for analysis of architecture conformance. Each method can use one or more techniques, in some cases techniques are supported on tools (see Fig. 5).

Static architecture conformance is applied under the static architecture view. Two static techniques can be executed: *Semantic* verifies the real meaning of the architecture assets, i.e. if its description, operations, attributes, etc. are conform to standard. And *syntactic* verifies associations, dependences and generalizations among the architecture assets.

Dynamic architecture conformance is applied under the dynamic architecture view, where the behavior is checked. In order to reduce the complexity the behavior is proved on partial scenarios, i.e. interaction among a part of architecture assets.

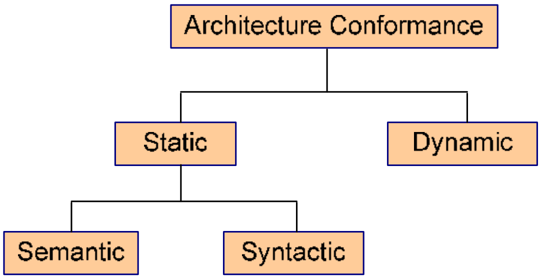


Fig. 5. Taxonomy of architecture conformance techniques

In the following related works, the closest methods and techniques for conformace analysis are proposed:

SACAM [8] proposes a process based on comparisons for business goals. It uses tactics, architecture styles and patterns as indicators to evaluate if a quality attribute is supported.

In ALMA [9] for modifiability analysis some techniques are used to detect equivalence classes and classification.

In SAA [10] a process of elicitation (a subjective top-down process of comparison of architectures) is defined. No techniques are proposed.

Emmerinch [11] presents a model to identify the issue of standard compliance. The identification is done from UML class diagram (static conformance). Practices, properties and policies are checked.

Sørumärd [12] identifies four strategies for development process conformance, they are based on: interviews, computer-support enactment, statistical process control and event-stream comparisons. In addition, Sørumärd proposes a model to measure the conformance process. It is based on deviation vector technique applied to resources and products obtained during development process.

Dae-Kyoo Kim [13] presents a method for conformance evaluation between UML model (class diagrams) and design patterns. Kim makes evaluation both in syntactic and semantic conformance.

Nguyen [14] presents a model to find the concordance and consistence of documents during development process by finding causal dependences between documents (document traceability). The techniques used are based on logics, for example: model checking, formal framework, hypertext, abstract dependence graph, static checking and so on.

Gnesi [15] contributes in dynamic conformance for UML statecharts. This work is based on mathematical basis, input/output transition systems. However, it was developed for testing conformance between specification and implementation.

Other works have been found in relation with conformance process. For example, ontology based algorithms that allow the search of common assets in an architecture [9], numerical and graph-based algorithms to reduce complexity, use cases to isolate parts of a system, comparison of abstract syntax tree of similar systems, measurement of similarities using metrics (internal or external as was defined in [2] to measure quality aspects) and so on. However, they are focused to the implementation against its specification in order to detect errors and inconsistencies.

5 Case Study

This case study analyzes the security aspects that should be covered in the Remote Management for Deployment of Services (RMDS), it is a distributed system managing the service during their deployment of a residential environment. In this case the final users dispose of a service gateway and a service platform, which allow receiving and using service from diverse providers. The service deployment can be performed by the system manager or the remote user. In consequence management of principals is required (privileges, profiles, permission, etc), because user or services interact with the system or among them. In this scenario, the security can be compromised of several ways, for example spoofing, sniffing, platform damage and other kinds of attacks.

In this case study, the candidate architecture uses the OSGi specification [3] and the security standard selected was security part of CIM [16].

5.1 Proposed Architecture

The proposed architecture is illustrated in the Fig 6 [18]. The control center is composed by web server, application server and a set of services, one of them deployment service. The service gateway uses some services defined in the OSGi specification (permission admin and user admin), dependencies resolver and management agent. The last two, the client part that request and deploy new services from control center to service gateway.

5.2 QAC Analysis

Several analyses were executed during QAC analysis [18]: *Context conformance, Architecture asset conformance, Relationships conformance, Practices conformance and Assertion conformance.* In this paper only *Architecture asset conformance* is presented.

The common assets RMDS and CIM (SCA ∩ SSA) are: Privilege, Identity, Organization, Resource, Policy, Setting-Data, UserAdmin, PackageAdmin, Device, PermissionAdmin, Log, Tracker and URL. Similar to the concepts, assets have not a precise mapping. The list of missed assets is shown in Table 1.

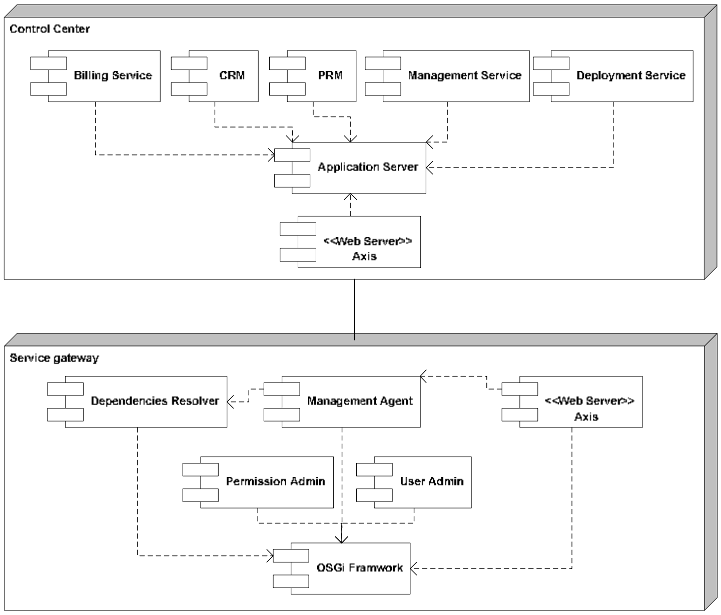


Fig. 6. Proposed architecture of the RMDS

Proposal for enhancement of RMDS (SSA-SCA): In the conceptual model the following elements are required:

- *OrganizationalEntity* is a type of ManagedElement that represents an Organization or an OrgUnit (organization unit or part of an organization), it could be composed of organizations or organization units (collections) with a defined structure.
- *Notary* is a service for credential management used in authentication service.
- *AdminDomain* describes the system domain (context).
- *AccountManagementService* is a type of security service in charge of managing the accounting issues in the system.

Table 1. Extracted requirements from conformance process between RMDS and CIM (DMTF)

	Conceptual model	**Static architecture**
SSA-SIA	OrganizationalEntity Notary AdminDomain AccountManagementService	Certificate Credential
SIA-SSA	Framework Device Manager	Provisioning service StartLevel service WireAdmin service

In the static architecture the next components are required:

- *Certificate authority* is a service for credential management used in the authentication service. It is a trusted third party organization or company that issues digital certificates used to create digital signatures and public-private key pairs (unsigned public key and public key certificate).
- *Credential*, is a type of ManagedElement. In cryptography, a credential is a subset of access permissions (developed with the use of media-independent data) attesting to, or establishing, the identity of an entity, such as a birth certificate, social security number, fingerprint, or others.

Proposal for CIM-DMTF standard (SCA-SSA): The following lacks of the standard have been detected:

In the conceptual model the next elements are required:

- *Framework*, A framework is a reusable, "semi-complete" application that can be specialized to produce custom applications.
- *Device Manager*, In OSGi, device manager service detects registration of Device services and is responsible for associating these devices with an appropriate Driver service.

In the static architecture the next components are required:

- *Provisioning service,* is a service registered with the Framework that provides information about the initial provisioning to the Management Agent.
- *StartLevel service,* allows Management Agent to manage a start level assigned to each bundle and the active start level of the Framework.
- *WireAdmin service,* is an administrative service that is used to control a wiring topology in the OSGi Service Platform.

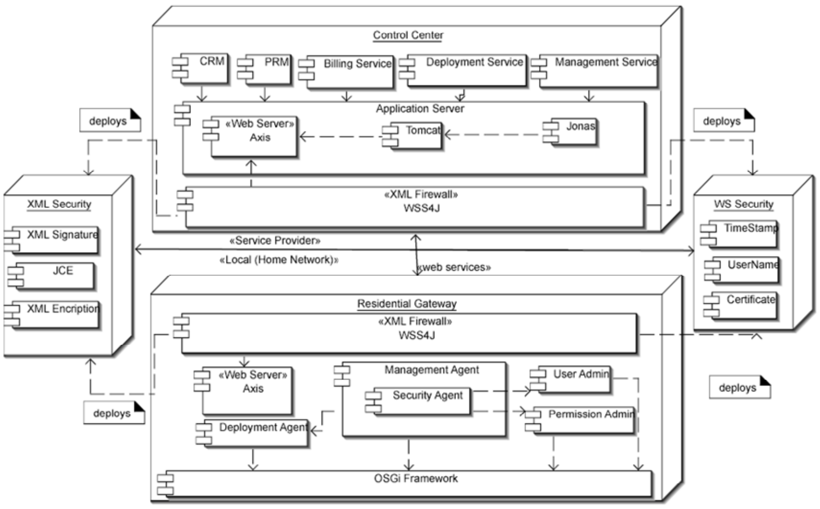


Fig. 7. Second candidate architecture for the scenario

5.3 QAC Agreement

Considering previous analysis, the main candidate architecture is so far to guarantee the security in a hostile environment. In consequence, other candidate architecture is required; this new architecture should consider services from a third party. In Fig. 7 it is illustrated a possible alternative using Web services and XML security [18].

6 Conclusions and Future Work

QAC process is a mechanism to evaluate, assess and check architectures. The main contributions of conformance process are: identification of improvements (lacks and new requirements) to architectures, identification of suggestion for standard recommendations, and identification of commonalities.

QAC is complementary assessment process and it is mechanism to learn from a standard. The standards are product of mature experiences, for this reason the effort in the comparison process is substantially reduced.

A methodological support has been presented for architecture conformance. It include: A conceptual model, a workflow method and a suit of methods, techniques and tools was presented.

The QAC model has been validated in a specific domain (Internet services) and applied over a quality characteristic (security). The validation was executed in some scenarios. As result of this validation; several recommendation for proposed RMDS system were introduced and some missed architectural assets of CIM standard were detected.

The new security requirements were detected with a real scenario (only some security aspects have been covered in the validation process). Other scenarios could be implemented.

Architecture conformance should consolidate methods and techniques that support conformance analysis. A set of tools should be also provided; they are part of future work in this discipline.

References

[1] ISO International Organization for Standardization, `http://www.iso.org/iso/en/ISOOnline.frontpage` (last visited date at 28/06/2010)

[2] ISO/IEC 25010-CD – JTC1/SC7. Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE). Internal ISO/IEC JTC1/SC7 Document, Currently at Commission Draft Stage, International Standardization Organization (2007)

[3] OSGi. Open Services Gateway Initiative. OSGI Service Platform, Specification Release 3.0 (March 2003)

[4] CMMI Product Team: Capability Maturity Model Integration (CMMI) Version 1.1: CMMI for Systems Engineering and Software Engineering (CMMI-SE/SW, V1.1), Staged Representation Technical Report CMU/SEI-2002-TR-002. Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA (2002)

[5] Kazman, R., Klein, M., Clemens, P.: ATAM: Method for Architecture Evaluation. Technical report. CMU/SEI-2000-TR-004. Software Engineering Institute, USA (2000)
[6] Obbink, J.H., Kruchten, P., Kozaczynski, W., Postema, H., Ran, A., Dominick, L., Kazman, R., Hilliard, R., Tracz, W., Kahane, E.: Software Architecture Review and Assessment (SARA) Report, Version 1.0 (February 2002)
[7] van der Linden, F., Bosch, J., Kamsties, E., Känsälä, K., Obbink, H.: Software Product Family Evaluation. In: Nord, R.L. (ed.) SPLC 2004. LNCS, vol. 3154, pp. 110–129. Springer, Heidelberg (2004)
[8] Stoermer, C., Bachmann, F., Verhoef, C.: SACAM: The Software Architecture Comparison Analysis Method. Technical Report. CMU/SEI-2003-TR-006 (2003)
[9] Lassing, N., Bengtsson, P., van Vliet, H., Bosch, J.: Experiences with ALMA: Architecture-Level Modifiability Analysis. Journal System Software 61(1), 47–57 (2002)
[10] Anastasopoulos, M., Bayer, J., Flege, O., Gacek, C.: A Process for Product Line Architecture Creation and Evaluation, PuLSE-DSSA, IESE, IESE-Report 038.00/E (June 2000)
[11] Emmerich, W., Finkelstein, A., Antonelli, S., Armitage, S., Stevens, R.: Managing Standards Compliance. IEEE Transactions on Software Engineering 25(6) (November/ December 1999)
[12] Sørumgård, S.: Verification of Process Conformance in Empirical Studies of Software Development. Ph.D. thesis, Norwegian University of Science and Technology (1997)
[13] Kim, D.: Evaluating conformance of UML Models to Design Patterns. IEEE, Los Alamitos (2005)
[14] Nguyen, T., Munson, E.: A Model for Conformance Analysis of Software Documents. In: Proceedings of the Sixth International Workshop on Principles of Software Evolution (IWPSE 2003). IEEE Computer Society, Los Alamitos (2003)
[15] Gnesi, S., Latella, D., Massik, M.: Formal Test-case Generation for UML Statecharts. In: Proceedings of the Ninth IEEE International Conference on Engineering Complex Computer Systems Navigating Complexity in the e-Engineering Age. IEEE, Los Alamitos (2004)
[16] Bass, L., Clemens, P., Kazman, R.: Software architecture in practice, 2nd edn. Addison Wesley Professional, Reading (2009)
[17] DMTF. Core Specification 2.9 (UML diagram Preliminary release) (July 2004)
[18] Arciniegas, J.: Contribution to quality driven evolutionary software development for service oriented architecture. PhD. Thesis presented to the Universidad Politécnica de Madrid (2006)

Trends in M2M Application Services Based on a Smart Phone

Jae Young Ahn[1], Jae-gu Song[2,4], Dae-Joon Hwang[3], and Seoksoo Kim[2,*]

[1] Electronics and Telecommunications Research Institute, Daejeon, Korea
[2] Department of Multimedia, Hannam University, Daejeon, Korea
[3] The School of Information & Communication Engineering, Sungkyunkwan University, Suwon, Korea
[4] School of Computing & Information System, Tasmania University, Hobart, Australia

ahnjy@etri.re.kr, bhas9@paran.com, djhwang@skku.edu, sskim0123@naver.com

Abstract. M2M, which stands for communications between machines, offers various services today thanks to advanced communication networks and sensor systems. Also, a powerful terminal such as a smart phone provides sufficient physical environments, not requiring a special device for the services. However, the smart phone M2M environment involves various complex technologies, and there have been no clear policies or standards for the technology. This study, therefore, analyzes the current status of M2M service introduction and the trends in M2M application services using a smart phone.

Keywords: M2M, Smart phone, Application, Communications.

1 Introduction

M2M, which refers to machine to machine, generally means wired or wireless communications, and communications between devices controlled by man and machines. Today, however, the word refers to wireless communications between devices [1]. Along with the growing popularity of smart phones, M2M is expected to be utilized in various fields by supporting communications between terminals and considered a blue ocean in the telecommunications industry [2, 3, 4, and 5]. This indicates that a multitude of electronic devices interact with smart phones in order to offer various application services.

M2M is an intelligent communication service between objects and telecommunication infrastructure that can be safely used anywhere in real time. So, there have been a lot of efforts to apply M2M to various fields such as telematics, exercise, navigation, smart meters, a vending machine, security services, and so on.

The M2M technology is being considered a core technology in the future telecommunication industry due to the increasing popularity of smart phones, and a

[*] Corresponding author.

lot of research is being done on various service models. This implies that the technology is ready to be commercialized and significant not only in terms of technology but application services[6].

In order to apply the M2M technology to various fields, basic research should be done on small-power telecommunication solutions such as mobile communications and the high-speed wireless Internet.

In this study the importance of M2M along with its concept and features is discussed. In addition, it explains relevant policies, the market trend, and related technologies and, then, the trends in development of M2M application technology based on a smart phone.

2 Concept and Features

M2M stands for machine-to-machine, which means automatic communications between machines without human intervention. This is known as pervasive computing. M2M also refers to interaction between a human and an object. That is, the technology makes possible diagnosis between machines, obtaining data and improving conditions. Thus, M2M is used to check locations, temperatures, maintenance data, productivity levels, and so on, conveying the data to humans for more efficient operations.

Meanwhile, M2P (Machine-to-Phone) or P2M refers to communications with a machine through a mobile phone and P2P (Phone-to-Phone), between mobile phones. In general, the communication flow of M2M is as follows.

Data collected from devices are sent to wireless data service providers through M2M hardware, and they are delivered to a company or others who need the information through the Internet or other channels.

The data received are used for various services such as remote control, remote inspection, security, home automation, telematics, RFID, location tracking, and so on.

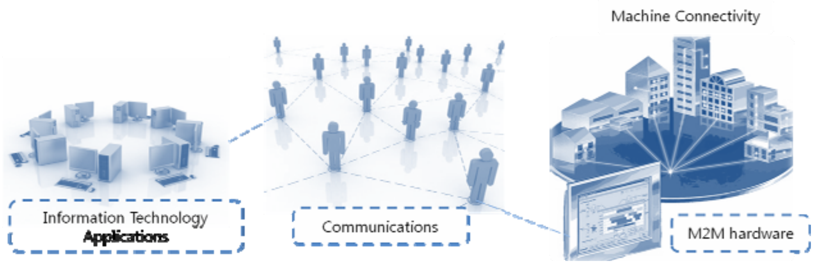


Fig. 1. Concept of M2M

Likewise, M2M is easy to be installed and maintained with low expenses for it uses mobile communication networks having the potential of being widely used in various areas.

3 Policies and Technical Trends

3.1 Trends in M2M-Related Policies

At present, ESTI (European Telecommunications Standards Institute) M2M standardization was made official in 2009 including the mandates given by European Commission. ESTI aims to provide comprehensive standards involving application areas, physical layers, service providers, and users.

M2M standardization is of growing interest because it provides a basic study on the new IT environment which will see a shift into the AToN(All Things on Network) system [7].

3.2 Trends in the M2M Technology

Analyzing the value chain of M2M will help understand its technical trends [8].

The value chain of M2M includes a chip, an M2M module/terminal, platform software, and community services. Details of each component are provided below.

Chip: The wireless transmitting-receiving chip measures electronic changes in the environment through a sensor and implements intelligent process using a micro-controller.

M2M module/terminal: A module packaging the wireless transmitting-receiving chip, a sensor, and a micro-controller and a terminal with several M2M modules installed by SW.

Platform software: Software which is inserted into a terminal and a server in order to control the M2M system.

Communication services: M2M service that controls the system in a comprehensive way

The services based on the M2M value chain is be expressed in Figure 2.

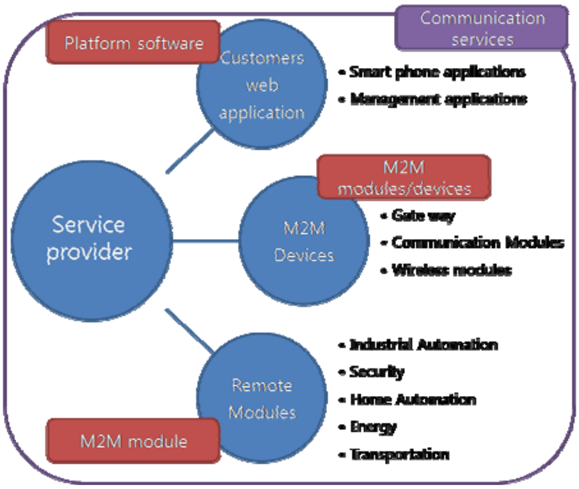


Fig. 2. Services based on the M2M Value Chain

4 Cases of M2M Technology Application

4.1 M2M Technology and Application Cases

Good examples of the technology application are telematics, logistics, intelligent meter reading, and security systems. Simplified M2M communication flow has promoted various application services.

Automobile telematics: M2M is used for tracking the location of a vehicle, vehicle maintenance data, mileage data, and provide an alarm for theft or emergency. All these data are sent to the platform in real time through GPRS (general packet radio services) while users can control the data for their needs [9]. The size of the data is not large also, significantly cutting down maintenance costs.

Logistics: Based on GPS data, M2M is used for tracking a location, checking conditions of a vehicle part, tracking goods, exchanging information with drivers, and optimizing transportation courses. Also, monitoring fuel can predict how much and when to supply the fuel, providing optimized materials for manufacturing as well [10].

Intelligent Meter Reading: This is the same as AMM (Advanced Meter Management) and AMI (Advanced Metering Infrastructure). The intelligent meter reading system is the third generation technology in the industry, a shift from manual reading to AMR (Automated Meter Reading) [11]. M2M can be applied to electricity and gas services, for example. The service has been extended to environmental monitoring, observing the level of water and air or water contamination. Also, the system can be used to control hazardous gases or chemicals.

Security: Major application cases include security or guard services. In particular, the security system senses intrusion and promptly reports to a control center to take measures. More recently, M2M mobile security services are offered to users on the move.

Besides, the technology can be applied to monitoring conditions of a moving vehicle such as a bus, a truck, a garbage truck, or a truck mixer, public services such as a load lamp, and a bridge or a tunnel.

Other good examples also include observing conditions of pipes of a smokestack or underground and temperatures/humidity of livestock sheds or greenhouses.

4.2 Smart Phone M2M Technology and Application Cases

Smart phone M2M is fast developing along with various wireless M2M applications [12].

During the last 4 years, growth of M2M services doubled, especially in the mobile industry. In view of the wide range of application today (medical equipment, automobiles, etc.), the technology is expected to be adopted to an increasing number of areas. Particularly, the introduction of a smart phone supports services requiring a large quantity of data, further expanding the application of M2M.

The strength of smart phone M2M service is that users can enjoy various application services without a special device, which has been needed for previous M2M services. In addition, no additional installation expenses are charged but users' current device is sufficient to receive new services.

3-Screen, which has recently appeared, and integration of TV-mobile-online is also supported by communications between hardware and cross- platform, making possible various M2M application services. And most research efforts are being made on the following areas.

Location tracking: Using the GPS module included in a smart phone, a location of a child or an elderly person can be tracked for their safety. The location-based product is of great importance in the study area related with tracking a person.

Individual security: Since a smart phone is always carried by an individual, it can be effectively used for personal security. For example, ZenieCall, a communication service provided by SECOM, a security service provider, is a smart phone application that makes an emergency call through iPhone or T-Omnia2 (Samsung) when one senses a danger [13].

Context-awareness: A device analyzes human data in order to meet the needs of a person and an object according to each circumstance [14]. The device can analyze a user's schedule saved in a smart phone and other data related with his tendencies. Communications are made between smart phone sensors in order to exchange data generated in real time according to context changes.

Management of the national industrial infrastructure: As TCP/IP communications and sensors are introduced to DNP3 and Modbus, mainly employed by industrial infrastructure based on SCADA (Supervisory Control and Data Acquisition), a smart phone is being applied to facility safety management solutions [15]. Thus, remote monitoring and machine operation has become possible.

Communication services: The tethering technology, commercialized by a smart phone, is a feature of using the phone as a modem, connecting IT devices such as a laptop computer to a mobile phone so as to use the wireless Internet. Through the technology a user can have a 3G connection without previously required communication environments. It may seem that an object, unable to have wireless communications, works as if it is connected with a modem.

The M2M application services using a smart phone have recently achieved standardization and compact sizes. Fast down/up loading of data has been made possible by advanced communication environments also. And integration of various additional features needed for M2M services made it easy for developers or system providers to maintain and control the services. Self-diagnosis and less expensive data fees also significantly increase the availability of the services. Besides, users only need software to use the service.

5 Conclusion

Smart phone M2M is a service focused on mobility, abstractness, and message processing, very useful for customized services. M2M, applied to a smart phone, can easily extend application services, for it has physical resources supporting mobility, context awareness, tracking, security, management, and other complex services as well as software resources that can be easily installed or removed.

When communications standards are completely established and a regular module with basic specifications is provided in detail, more various M2M services will be introduced. As a result, more development efforts will be given to offer a variety of applications. This will lead to new business opportunities such as unmanned inspection systems, remote monitoring, health care, and so on, creating profits in various areas.

References

1. Machine-to-Machine (M2M) Communications, `http://www.mobilein.com/M2M.htm`
2. Machine to Machine Service Provider, `http://smartphone.biz-news.com/news/2010/04/09/0006`
3. Smart Services – MHealth: Body Parts Make Phone Calls, `http://m2m.orangeom.com/`
4. Machine-to-Machine, `http://www.nokia.com/NOKIA_COM_1/About_Nokia/Press/White_Papers/pdf_files/m2m-white-paper-v4.pdf`
5. M2M Wireless Smart Devices Could Boost Mobile Carrier Revenues, `http://mobilebeyond.net/m2m-wireless-smart-devices-could-boost-mobile-carrier-revenues/`
6. Watson, D., Piette, M., Sezgen, O., Motegi, N.: 2004, Machine to Machine (M2M) Technology in Demand Responsive Commercial Buildings. In: Proceedings of the ACEEE 2004 Summer Study on Energy Efficiency in Buildings (2005)
7. Shaw, D.E., Deneroff, M.M., Dror, R.O., Kuskin, J.S., Larson, R.H., Salmon, J.K., Young, C., Batson, B., Bowers, K.J., Chao, J.C., Eastwood, M.P., Gagliardo, J., Grossman, J.P., Ho, R.C., Ierardi, D.J., Kolossváry, I., Klepeis, J.L., Layman, T., McLeavey, C., Moraes, M.A., Mueller, R., Priest, E.C., Shan, Y., Spengler, J., Theobald, M., Towles, B., Wang, S.C.: Anton, a special-purpose machine for molecular dynamics simulation. In: Proc. 34th Ann. Intl. Symp. on Computer Architecture, San Diego, pp. 1–12 (2007)
8. M2M Value Chain, `http://www.wireless-technologies.eu/index.php?page=m2m-value-chain`
9. Bettstetter, C., Vögel, H.-J., Eberspächer, J.: GSM Phase 2+ General Packet Radio Service GPRS: Architecture, Protocols, and Air Interface. IEEE Communications Surveys and Tutorials 2, 2–14 (1999)
10. Li, Y.-b., Zhang Z.-y.: Design of a visible logistics management information system based on GPS and GIS. Metallurgical Industry Automation (2005)
11. Li, Y.-x., Xu, J.-z., Liu, A.-b.: Application of GPRS technology in automatic meter reading system. Metallurgical Industry Automation (2003)

12. National Global Information, Inc., Wireless Machine-to-Machine.: An In-depth Study of Applications and Vertical Markets (2004)
13. ZenieCall, `https://www.s1.co.kr/biz/zeniecall/zeniecall_01.jsp`
14. Vale, S., Hammoudi, S.: Towards context independence in distributed context-aware applications by the model driven approach. In: Proceedings of the 3rd International Workshop on Services Integration in Pervasive Environments, Sorrento, Italy (2008)
15. Heung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., Valdes, A.: Using model-based intrusion detection for SCADA networks. In: Proceedings of the SCADA Security Scientific Symposium (2007)

Using ERP and WfM Systems for Implementing Business Processes: An Empirical Study

Lerina Aversano and Maria Tortorella

Department of Engineering
University of Sannio, Via Traiano 1 82100 Benevento Italy
(aversano,tortorella)@unisannio.it

Abstract. Software systems mainly considered from enterprises for dealing with a business process automation belong to the following two categories: Workflow Management Systems (WfMS) and Enterprise Resource Planning (ERP) systems. The wider diffusion of ERP systems tends to favourite this solution, but there are several limitations of most ERP systems for automating business processes. This paper reports an empirical study aiming at comparing the ability of implementing business processes of ERP systems and WfMSs. Two different case studies have been considered in the empirical study. It evaluates and analyses the correctness and completeness of the process models implemented by using ERP and WfM systems.

Keywords: Business process modelling, WfMSs, ERP systems, Empirical study.

1 Introduction

Fast changing business requirements forces enterprises to support their business processes with appropriate software applications in order to increase the process performances. The software systems mainly considered from enterprises for automating their business processes belong to the following two categories: Workflow Management Systems (WfMS) and Enterprise Resource Planning (ERP) systems.

The two alternative solutions deal with the processes by using different approaches. ERP systems essentially consist of multi-module applications integrating activities across functional departments including product planning, purchasing, inventory control, product distribution, order tracking, and so on [9, 11]. They are designed around the idea of applications that need to be configured with appropriate setting for achieving the solution that is most adequate to the enterprise requirements. Of course, higher the possibility of configuration is, more flexible the support for the business processes is. On the other side, WfMSs represent an information technology designed for automating business processes by coordinating and controlling the flow of activity and information between participants. When a WfMS is used, a workflow model is first defined for the specific requirements of the enterprise and, then, workflow instances are created for performing the actual activities described in the workflow model.

The aim of this paper is not to compare WfM and ERP systems, whose aim and support level are completely different and, then, not comparable. The paper focus on the analysis of the service offered by ERP systems regarding the management of the business process and compare it with the main functionality offered by WfMSs.

In particular, the paper analyses the workflow ability of ERP systems and WfMSs. The expression "workflow ability" is used for indicating the capability of a software system of effectively and efficiently supporting the modelling, execution and monitoring of a business process. In a previous paper [2], the authors observed several limitations in the functionalities of most ERP systems with reference to the automation and management of business processes. This paper reports an empirical study aiming at comparing the "workflow ability" of ERP systems and WfMSs in two different case studies.

The paper is organized as follow: Section 2 gives some background on WfMSs and ERP systems; Section 3 describes the design of the empirical study; Section 4 introduces the main results achieved; finally, conclusions are discussed in Section 5.

2 Theoretical Background

Workflow Management System is a technology mainly focused on the automation of business processes. It is widely adopted in the enterprises for supporting production activities executed by the employee [5, 7]. ERP systems, vice versa, mainly address the need of having an integrated database that serves different functional modules supporting specific tasks of the enterprises. In the literature, there are several definitions of ERP systems [1, 4, 8]. Basically, ERP systems overcome the data separation of multiple functional applications by allowing individual modules to share the same data. Moreover, most of them contains functionality for modelling, deploying and managing workflows. The WfMS "embedded" in the ERP system is a module which is a part of the core ERP architecture. Differences existing between ERP systems and WfMSs are presented in the literature and mainly focus on the definition of frameworks and approaches for facilitating their integration. A strategy proposed for their integration consists of the use of a WfMS as a mean for implementing a workflow controlling the ERP functionalities [3]. The problem of this approach was highlighted in [8] and mainly deals with the difficulties of managing inconsistencies between the two systems. Several other strategies address the problem of the integration by considering the WfMSs as a "middleware" orchestrating legacy applications and ERP systems. Newmann and Hansmann [6] developed an architecture for integrating WfMS and the planning functionality embedded in ERP systems. Brehm and Gomez proposed an approach for federating ERP systems exploiting an architecture based on a WfMS enacting Web Services.

The main advantages deriving from the use of WfMSs are the following [5]:

- *efficiency improvement*: Using WFMSs allows an enterprise accomplishing several objectives for improving its efficiency regarding the management of processes, resources, market, delegation, motivations.
- *process control improvement*: WfMSs introduce a standard way of organizing the work and include tools for validating the performed activities.

- *service quality improvement*: Possibility of selecting the resources by exploiting their specific during the workflow execution.
- *training costs reduction*: The costs for training human resources is reduced as the execution of the tasks is guided by the WfMSs.

While, the main advantages deriving from the adoption of ERP systems are [8]:

- *Reliable information access*: Common DBMS, consistent and accurate data, improved reports.
- *Reduced data and operations redundancy*: Modules access the same data in the central database, avoiding multiple data input and update operations.
- *Delivery and cycle time reduction*: Minimization of retrieval and reporting delay.
- *Improved scalability*: Structured and modular design.
- *Improved maintenance*: Vendor-supported long-term contract as part of the system procurement.
- *Global outreach*: Use of extended modules, such as CRM – Customer Relationship Management – and SCM – Supply Chain Management.

3 Design of the Study

The aim of the conducted empirical study was to understand the efficiency of WfM and ERP systems for managing business processes.

The subjects involved in the empirical study were students from the course of Enterprise Information Systems, in their last year of the master degree in computer science at the University of Sannio in Italy.

The *treatment variable* of the empirical study was the independent variable *type of system used*, whose values are: *ERP system* and *WfM system*. The first value regards the use of ERP systems for modelling business processes. The second value is referred to the use of WfM systems with the same aim. The two kinds of systems were used by two groups of subjects, each composed of six elements. The first group is indicated with *G_ERP* and used ERP systems; while the second group, called *G_WfM*, used WfM systems. This represented the first independent variable.

The empirical study was performed by using two ERP systems and two WfM systems. This permitted to avoid the strong dependence of the obtained results form just one WfM / ERP system. In particular, the considered ERP systems were:

1. **Compiere ERP/CRM** (http://www.compiere.com/): An open source ERP and CRM business solution for the Small and Medium-sized Enterprise (SME) for execution distribution, retail, service and manufacturing activities.
2. **OpenERP** (http://openerp.com/product.html): An Open Source enterprise management software. It covers and integrates enterprise needs and processes, such as: accounting, sales, CRM, purchase, stock, production, services and project management, marketing.

The two considered WfMSs were:

1. **Ultimus BPM Suite** (http://www.ultimus.com/): A solution for automating essential business processes. It handles the following services: order processes, purchase and claims processes, document reviews, etc.
2. **ProcessMaker** (http://www.processmaker.com/): An Open Source business process management software including tools for enabling management of operational processes across systems, including finance, human resources and operations.

The ERP and WfM systems were randomly assigned to the *G_ERP* and *G_WfM* subjects respectively. The second considered independent variable is referred to the *modelling task to be performed*. A process model is made of different elements: activities, relationships, resources, and so on. The modelling of each different element was considered as a different modelling task. Table 1 lists the modelling tasks that were identified. The first and second columns of Table 1 indicate the name and description of the tasks, respectively. For sake of clarity, Figure 1 shows an example of process model implemented by using ProcessMaker. The figure highlights the model components involved in the modelling tasks, that are pointed out by an arrows labelled with the name indicated in Table 1.

The third independent variable was the business *process to be modelled*. In particular, two business process were considered. The *CSA* (Centro Servizi Amministrativi) business process, regarding the definition of the teaching resources needed in the primary and secondary schools of Benevento, a town in Italy. The TRIBUTE process regarding the process executed at the local Public Administration of Maddaloni (a town in the district of Caserta in Italy), and aiming at managing the payment of taxes and tributes. The two business processes differed in complexity. In fact, Table 1 shows the numbers of the different model elements composing the two considered process models. It can be noticed that the TRIBUTE process was more complex than CSA in terms of activities, relations, control and required resources. Both processes were analysed by both groups with both WfM and ERP systems.

Another considered independent variable was the time spent for *training* the subjects to the usage of either WfM or ERP systems. As it will be later explained, due to the different complexity of the two kinds of systems, a longer time was spent for presenting and training the subjects to the use of the ERP systems. Then, the *G_ERP* subjects were involved in added training before being able of assigned modeling the business processes.

The two dependent variables considered were the *Correctness* and *Completeness*.

Correctness indicates how *correctly* the empirical subjects performed the modelling elements by using either WfM or ERP systems. It was calculated as the proportion of the correctly automated modelling business elements, *#CorrectIdentifiedBPElem* respect to the total number of automated elements, *#IdentifiedBPElem*. The correctness is evaluated as indicated in the following formula:

$$Correctness = \frac{\#CorrectIdentifiedBPElem}{\#IdentifiedBPElem}$$

Completeness measures how *completely* were automated the modelling business elements. It is evaluated as the proportion of the correctly automated modelling business elements, *#CorrectIdentifiedBPElem*, respect the total number of the modelling business elements to be traced, *#TotalBPElem*. It is computed as:

$$Completeness = \frac{\#CorrectIdentifiedBPElem}{\#TotalBPElem}$$

Table 1. Modelling tasks

NAME	DESCRIPTION	CSA PROCESS MODELLING ELEMENTS	TRIBUTE PROCESS MODELLING ELEMENTS
MT1	Modelling the activities of the business process	10	14
MT2	Modelling interrelationships between business process activities	11	19
MT3	Modelling the conditions constraining the business process control flow	4	6
MT4	Modelling the forms to be filled by the user during the business process execution	8	10
MT5	Modelling the user roles in the execution of the business process	4	5
MT6	Modelling process control flow	2	3
MT7	Modelling the access rights to the business process forms	8	10

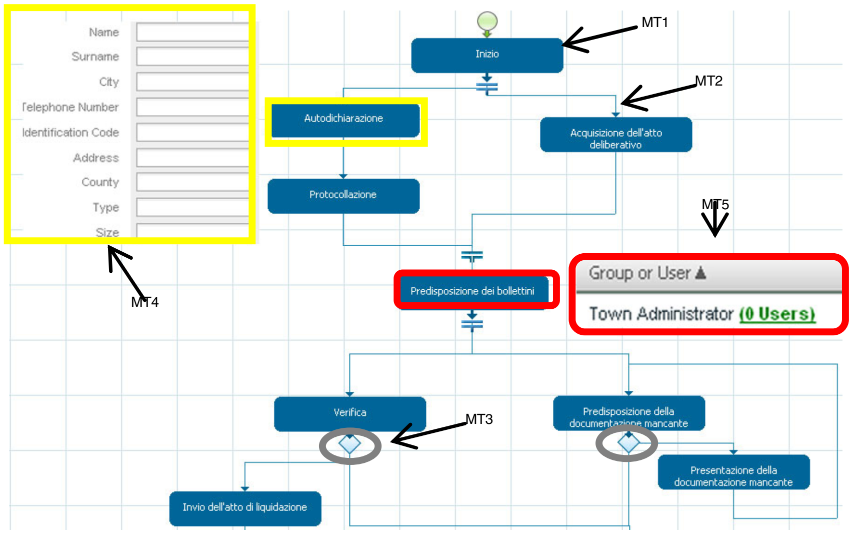

Name
Surname
City
Telephone Number
Identification Code
Address
County
Type
Size
MT4
MT1
Inizio
MT2
Autodichiarazione
Acquisizione dell'atto deliberativo
Protocollazione
MT5
Predisposizione dei bollettini
Group or User
Town Administrator (0 Users)
Verifica
MT3
Predisposizione della documentazione mancante
Presentazione della documentazione mancante
Invio dell'atto di liquidazione

Fig. 1. Example of process model

The evaluation were executed using two oracles represented by the actually adopted models of the considered business processes. They were defined and validated with the two groups of executors of the business processes in their daily execution. Table 1 lists the numbers of modelling business elements to be traced during each modelling task.

The evaluation of the *Correctness* and *Completeness* aimed at understanding if WfM and ERP systems helped modelling business processes and which of the two kinds of systems permitted to reach better modelling results in terms of conformance of the business process model to the operative business process. This goal was reached by evaluating the numbers of modelling elements correctly identified and traced in the business process model by using the two kinds of systems and comparing them.

The expected results were that the WFM systems permitted to obtain more complete and correct results than ERP systems. This would contribute to confirm the results obtained in [2], where the major workflow ability of WfMSs respect to ERP systems was demonstrated.

4 Results

This section reports the main results achieved in the proposed empirical study. The subjects involved in the study were asked to perform the modelling tasks listed in Table 1 for both case studies, regarding the modelling of the *CSA* and *TRIBUTE* business processes.

The first execution of the study was planned after the baseline training to the use of the selected ERP and WfM systems. Table 2 shows the number of sessions spent during the initial training for each kind of systems. It is worthwhile noticing that a longer time was spent for training to the use of the ERP systems. However, during the first execution of the empirical task, the subjects of the group *G_ERP* were not able to accomplish their modelling tasks by using the ERP systems. They did not perform at all the process implementation. On the contrary, the subjects belonging to *G_WfM* completely executed the modelling tasks by using the assigned WfM systems. As a consequence, a second session of training to the use of ERP systems was planned. Additional hours were, then, spent for training to the use of ERP systems, as shown in Table 2.

Once the additional training was completed, *G_ERP* subjects were asked again to perform the study, and, in this case, they were able to complete the modelling tasks. The data that will be later compared are those obtained by group *G_WfM*, obtained after the baseline training, and those reached by group *G_ERP* after the additional training.

Table 2. Training sessions

	ERP Training	**WfMS Training**
Baseline Training	12	4
Additional Training	4	0

Table 3. Correctness of the process model produced with *ERP* and *WfM* systems

Modelling Task	ERP	WfM
MT1	1	0,97
MT2	0,92	0,95
MT3	0,75	0,78
MT4	0,81	1
MT5	1	1
MT6	1	1
MT7	0,5	1

Table 3 reports the total results of *correctness* obtained by using either ERP or WfM systems during the execution of the modelling tasks in Table 1; while Table 4 shows the results achieved by the *completeness* per each system and modelling task. The value of *correctness* and *completeness* were computed as average of the results obtained by the subjects.

Table 4. Completeness of the process model produced with *ERP* and *WfM* systems

Modelling Task	ERP	WfM
MT1	0,94	0,94
MT2	0,875	0,92
MT3	0,75	0,86
MT4	0,78	1
MT5	1	1
MT6	0,75	0,91
MT7	0,5	1

Observing Tables 3 and 4, it can be noticed by that better results of both completeness and correctness have been obtained by using the WfM systems. These data are confirmed by analysing the analytical results obtained for each process and by each system.

Tables 5 and 6 highlight that for both processes better correctness results were obtained by using the WfM systems. Analogous data were reached for the completeness.

In a major detail, Compiere is the ERP system that permits to create more correct process models, with the exception of some modelling aspects. efficiently from the correctness point of view. The two modelling tasks that do not confirm this result are MT2 and MT4. This is mainly due to the lack of a graphical tool for modelling business processes, that contributes to facilitate the control flow specification. In addition, the specification of the forms for the activity execution entails the use of the entities' model that manages their access. Regarding the WfMSs, better results were obtained by Ultimus, except in the TRIBUTE process with reference to the MT3 modelling task. Actually, the control flow conditions is more complex in Ultimus than ProcessMaker. In the former, the use of a process variable in a condition requires its definition in an electronic spreadsheet.

Table 5. Correctness of the implementation of the TRIBUTE process

Modelling Task	ERP Systems		WfM systems	
	OpenERP	**Compiere**	**ProcessMaker**	**Ultimus**
MT1	1	1	0,91	1
MT2	0,86	0,81	0,86	1
MT3	0,5	0,5	0,75	0,62
MT4	0,87	1	1	1
MT5	1	1	1	1
MT6	1	1	1	1
MT7	0,5	1	1	1

Table 6. Correctness of the implementation of the CSA process

Modelling Task	ERP Systems		WfM systems	
	OpenERP	**Compiere**	**ProcessMaker**	**Ultimus**
MT1	1	1	1	1
MT2	1	1	1	1
MT3	1	1	0,80	0,90
MT4	1	0,56	1	1
MT5	1	1	1	1
MT6	1	1	1	1
MT7	0	0,5	1	1

Tables 7 and 8 synthesizes the analysis of the completeness values obtained for both business processes. Even in this case, better results were obtained by applying the WfM systems for both business processes. In addition, even for the completeness, Compiere is the mot efficient ERP system. A minor inconsistency can be evicted with reference to the modelling task MT1 in the TRIBUTE process, and the user form modelling MT4 for the CSA business process. This is due to the same reasons previously explained. Finally, the best analysed WfM system was Ultimus, with the exception of the activities and reciprocal relationship modelling tasks, MT1 and MT2, with reference to the CSA process. Actually, the modelling task of the joining activities is more complex with Ultimus. Overall, the values in Table 3 and 4 highlight that both the completeness and the correctness values are higher for the WfM systems than ERP systems. This could suggests that, the WfMSs guarantee a greater efficiency in the modelling tasks. This is even more relevant if related to the training sessions, that required a longer time in the case of the ERP Systems.

As it is shown in Table 3, the ERP systems exhibited a bigger efficiency just in the modelling task concerning the correct identification of the activities, MT1. This result is due to the fact that the TRIBUTE process included a synchronization activity that some subjects of group *G_WfM* managed adding a not needed dummy activity.

The longer time required for the training of the *G_ERP* group and lowest efficiency of the ERP systems for modelling business processes essentially derived from the following characteristics of this kind of systems:

1. ERP systems do not include an intuitive and simple graphical modeller for managing forms: they are generated by the entities of the entity/relationship model managed by the application. This characteristic introduces a major complexity ad difficulty in the form generation;
2. the form access rights have to be also assigned on the entity/relationship model that manages the persistence. Some subjects of G_ERP forgot to set there rights, causing the impossibility of executing the activities with associated forms;
3. ERP systems did not include a graphical modeller easy to use. This increases the difficulty of modelling the process activities and control flow.

Table 7. Completeness of the implementation of the TRIBUTE process

MODELLING TASK	ERP SYSTEMS		WFM SYSTEMS	
	OPENERP	COMPIERE	PROCESSMAKER	ULTIMUS
MT1	0,9	0,86	0,96	1
MT2	0,72	0,81	0,84	1
MT3	0,5	0,5	0,66	0,83
MT4	0,8	1	1	1
MT5	1	1	1	1
MT6	0,5	0,5	0,75	1
MT7	0,5	1	1	1

Table 8. Completeness of the implementation of the CSA process

MODELLING TASK	ERP SYSTEMS		WFM SYSTEMS	
	OPENERP	COMPIERE	PROCESSMAKER	ULTIMUS
MT1	1	1	1	0,87
MT2	1	1	1	0,93
MT3	1	1	1	1
MT4	1	0,56	1	1
MT5	1	1	1	1
MT6	1	1	1	1
MT7	0	0,5	1	1

5 Conclusions

Both WfMSs and ERP systems play a major role in the automation of enterprise's process. However, some research makes a theoretical comparison among these systems with reference to the business process management and concludes that ERP systems are less suitable than WfMSs for supporting such a kind of service. This conclusion has just a limited quantitative evidence [2].

The aim of this paper was to address this limitation. The empirical study presented aimed at understanding the effectiveness of ERP and WfM systems for modelling business processes. In particular, business process models obtained by using ERP and WfM systems have been compared. The comparison was referred to the models obtained by the execution of seven modelling tasks, performed by two groups of subjects by using two ERP systems and two WfM systems. In addition, the same modelling tasks were executed on two different case studies. Results of the study confirmed the formulated hypothesis and WFMSs highlighted a major workflow ability for process modelling than ERP systems. These results also confirmed the initial investigation performed in [2], where the highest workflow ability of WfM systems already emerged respect ERP systems.

In addition, the empirical experience permitted to highlight that: (i) the use of ERP systems requires a bigger effort for training than WfM systems; and, (ii) nevertheless the major training, better modelling capabilities were exhibited by WfM systems in terms of correctness and completeness of the obtained process models.

These results encourage to continue investigating the differences of performance existing between ERP and WfM systems. In addition, the authors will continue to search for additional evidence of their hypothesis with additional studies also involving subjects working in operative realities.

References

[1] Almadani, R., Alshawi, S., Themistocleous, M.: Integrating diverse ERP systems: a case study. The Journal of Enterprise Information Management 17(6), 454–462 (2004)

[2] Aversano, L., Intonti, R., Tortorella, M.: Assessing Workflow Ability of ERP and WfM Systems. In: Proceedings of the 21st International Conference on Software Engineering Knowledge Engineering, SEKE 2009, Boston, Massachusetts, USA, July 1-3 (2009)

[3] Bostrom, R.P., Cardoso, J., Sheth, A.: Workflow Management Systems and ERP Systems: Differences, Commonalities, and Applications. University of Georgia, Athens (2004)

[4] Davenport, T.H.: Putting the enterprise into the enterprise system. Harvard Business Review 76(4), 121–131

[5] Fischer, L.: Workflow Handbook. Future Strategies Inc. (2002)

[6] Hansmann, H., Neumann, S.: Workflow integrated ERP: an architecture model for optimizes coordination of intra and interorganizational production planning and control. In: ECIS 2002, Gdansk, Poland, June 6-8 (2002)

[7] Hollingsworth, D.: Workflow Management Coalition – The Workflow Reference Model, D.N. TC00-1003, 19 Gennaio (1995)

[8] Hossain, L., Patrick, J.D., Rashid, M.A.: Enterprise Resource Planning: Global Opportunities & Challenges, pp. 16–17. Idea Group Publishing, USA (2002)

[9] Kumar, K., Van Hillsgersberg, J.: ERP experiences and evolution. Communications of the ACM 43(4), 23–26

[10] Malamateniou, F., Poulymenopoulou, M., Vassilacopoulos, G.: Specifying Workflow Process Requirements for an Emergency Medical Service. J. Med. Syst. 27(4), 325–335 (2003)

[11] O'Leary, D.E.: Enterprise Resource Planning Systems: Systems, Life Cycle, Electronic Commerce, and Risk. Cambridge University Press, Cambridge (2000)

Mining Design Patterns in Object Oriented Systems by a Model-Driven Approach

Mario Luca Bernardi and Giuseppe Antonio Di Lucca

Department of Engineering - RCOST, University of Sannio, Italy
{mlbernar,dilucca}@unisannio.it

Abstract. In this paper we present an approach to automatically mine Design Patterns in existing Object Oriented systems and to trace system's source code components to the roles they play in the Patterns. The approach defines and exploits a model representing a Design Pattern by its high level structural Properties. It is possible to detect Pattern variants, by adequately overriding the Pattern structural properties. The approach was validated by applying it to some open-source systems.

Keywords: Design Patterns, Software Comprehension, Software Maintenance, Software Evolution, Source Code Analysis.

1 Introduction

Pattern-based design leads to better structured, understandable, maintainable and ultimately reusable software systems [11,6]. Unfortunately, the advantages of adopting the DPs are often reduced by the lack of adequate documentation reporting the DPs used in a software system. In this paper we propose an approach to detect the DPs defined and implemented in an Object Oriented (OO) software system by identifyng the source components coding them. The approach defines a metamodel to represent a DP through a set of high level properties. The mining of the patterns is performed by traversing a graph representing an instance of this meta-model and annotating the elements of the system Type hierarchy with information on the roles they play in the patterns. With respect to existing mining approaches, ours has the main advantage that it allows to identify variant forms of the classic DPs (as known in the literature). This is an important issue since DPs are present in real world systems with many different variants [14,13]. Our approach organizes the DPs catalog into a hierarchy of specifications, reusing existing ones. Thus a DP Variant (DPV) can be easily expressed as the modification of existing specifications by adding, removing or relaxing the properties modelling it.

The proposed approach allows to reduce the size of the search space on the graph traversal, resulting in a better efficiency than existing ones. Indeed, in our approach, the analyis of a Type marks the code elements of the Type's neighbour that are succesfully bounded to patterns' members. Existing bindings are then reused during the traversal and provide chances to reduce the need of re-evaluating properties again for all Types (reducing the search space).

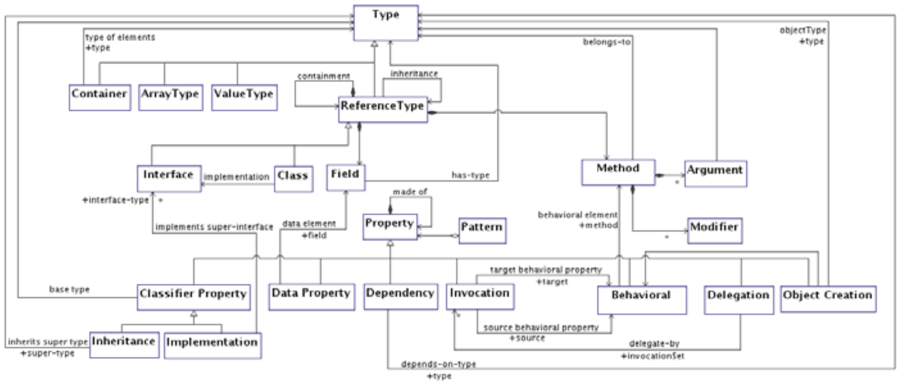


Fig. 1. The meta-model represented as a UML class diagram

Another issue affecting DP mining approaches is related to the dependence between the approach itself and the pattern to mine, while a mining approach should not be dependent on the particular pattern. The proposed approach is not dependent on the pattern to mine and the mining process is based on declarative specification. This allows to write new specifications deriving them from the existing ones (to detect variants) or to write them from scratch (to detect new patterns). To provide an automatic support to the approach the Design Pattern Finder (DPF) tool has been developed.

The approach has been assessed by applying it to some open source java systems. The results have been validated against those from the analysis of an expert and taking into account public benchmarks [7] or previous works available for the same systems.

The paper is structured as follows. Section 2 presents the meta-model defined to represent the DPs structure in terms of Properties. Section 3 presents the process to find DPs in a Java system, exploiting the instances of the proposed meta-model. Section 4 provides synthetic information about the DPF tool. Section 5 illustrates a case study carried out on some open-source Java systems. In the Section 6 relevant related works are discussed and some comparisons with the proposed approach are made. Finally, Section 7 contains some conclusive remarks and briefly discusses future work.

2 A Meta-model to Represent Design Pattern Specifications

The approach defines a meta-model to represent the Types (and Type's components) of an OO system, the structural relationships (e.g., inheritance, implementation and type nesting relationships) among the Types, the DPs, and the relationships among the DPs' code elements and the Types. The DPs are represented by a set of Properties modelling the structural elements of the patterns. The relationships among DPs and the Types are traced down to the DPs Properties and

Types' components. The Figure 1 shows, as a UML class diagram, the defined meta-model. An OO system is modeled as a set of Types (i.e., Container, Value, Reference, and Array Types[1]) where Reference Types are Interfaces and Classes, and an Interface is implemented by one or more Classes. Reference Types are composed by Fields and Methods, and a Method can have Arguments. A ReferenceType can inherit from another ReferenceType as well as can contain another ReferenceType (e.g. an inner class). A Pattern is defined by the aggregation of the Properties characterizing it. A *Classifier* Property, models a Type (Class or Interface) used in a pattern specification (or to modify an already existing Type). A Classifier models a role needed by the pattern with respect to its required internal structure and relationships with other Classifiers. Also, it allows to define constraints, if any, on its super-type or its implemented interfaces. A Classifier can be an Inheritance or an Implementation. The *Data* Property is used to define a field in an existing Classifier (or override an existing field). It can specify an existing Classifier as the field's type or a compound type of an existing Classifier (like an array for a generic Collection). The *Behavioral* Property allows to define a method (or to override a method's definition) in an existing Classifier. The definition of the method includes the definition of its return type, its arguments and, optionally, of the method itself or for any of its arguments. This property can be used to define one or more of the required (or optional) behaviours of the Classifiers introduced in a pattern specification. The *Dependency* Property describes the dependency between a pattern element (like a method) and another pattern element (as another method or a field). The *Invocation* Property models a call between methods already defined for some Classifiers in the pattern specfication. The *Delegation* Property specifies a mapping between a set of methods of a Class and a set of methods of an existing Classifier in the pattern specification. This allows to take into account the Delegation for the patterns that require it. For instance, Delegation is used to specify delegation in the Proxy pattern towards the proxied objects and for State pattern towards the ConcreteStates. The *Object Creation* Property models the creation constraints specifying the method or the field that needs the object creation and the Classifier of the created object. This happens for patterns expressing a mandatory object creation semantic, like in the case of creational patterns but also for many patterns in the other categories.

3 The Mining Process

The pattern mining process is structured in the following steps:

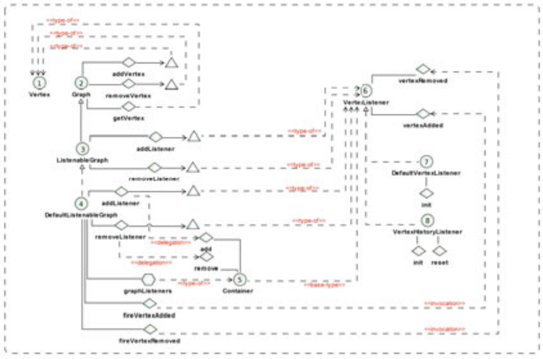
1. System source code analysis
2. Meta-model instantiation
3. Patterns' models matching

[1] Array types are threated as separate types since they must specify the type of array's components.

The source code of the system under study is parsed, and the resulting AST is used as input to the meta-model instantiation. Then a traversal of the AST is performed in order to generate an instance of the defined system model. Several techniques (Rapid Type Analysis - RTA, Class Flattening, and the Inlining of not-public methods) are exploited to build a system's representation to be used by the matching algorithm. RTA is used to handle late binding and hence the computed call graph reports a super-set of the actual calls that can be executed at run-time. The generated system model and the models of the DPs to be searched are the inputs of the detection step 3), that performs the matching algorithm.

The algorithm matches the models of the DPs against the model of the system. Models are represented by means of graphs: given a set of model graphs $P_1 \ldots, P_N$, representing DPs specifications, and an input graph S, representing the analyzed system, mining design patterns means to find all sub-graph isomorphisms from P_i to S. The approach uses a matching algorithm inspired to the one proposed in [10] and modified to adapt it to the context of DPs mining. The defined algorithm allows to: (*i*) detect more easily patterns' variants also (handled as new pattern models inheriting and overriding properties of a parent pattern model); (*ii*) reduce the search space by using constraints defined in the pattern specifications. Nodes' and edges' bindings are reused (across different pattern specifications) by marking traversed elements during the matching process, and the binding between already (and successfully) matched elements are no more considered (thus pruning the search space). The Figure 3 (lines 1-6) reports the core of the algorithm used for performing the match. As showed in Figure 3 in lines 7-15 (reporting the case of Classifier properties) the match function proceeds in forward until there are no property nodes that can be considered. For a match to be successful, both forward and backward steps must result in at least a binding (line 14). When the forward step fails, the backward step is not executed at all and the match is unsuccessful. Moreover existing bindings are reused while pending matches, suspended during the evaluation of matches for the properties in the neighbourhood, are considered as satisfied since they are part of the ongoing binding. More details can be found in [1].

In Figure 2 is an example about the algorithm beahaviour. The Figure shows, on the left, a small excerpt of a system model containing an instance of the Observer DP, and, on the right, the results of some of the matches that occurs during the graph traversal. In the model instance, a circle represents a Type, as specified by a Classifier property. Data properties are rendered as hexagons (see the Observer's fields connected to the ConcreteSubject). Methods and their arguments are rendered as diamonds and triangles respectively. The Property meta-classes are represented as stereotyped relationships. Due to the space constraints only one successful match will be entirely described (while in the other cases the result of the match is provided along with a brief explanation about the reasons it failed or succeeded). All the type nodes in Figure 2 are numbered just to refer to them in an easier way (nodes from system model have the prefix "S", nodes from pattern specification have the prefix "P"). The algorithm starts by scheduling matches among each couple of system and pattern type. As reported



```
match(S1,{P1,P2,P3,P4,P5}) {
 forward { ... } -> failed (not satisfied)
 backward not evaulated
} -> failed (constraints not satisfied)
match(S2,{P1,P2,P3,P4,P5}) {
 forward { ... } -> failed (not satisfied)
 backward not evaulated
} -> failed (constraints not satisfied)
match(S3,{P1,P2,P3}) {
 forward { ... } -> failed (not satisfied)
 backward not evaulated
}
match(S3,P4){
 forward{
  match(S3.addListener,P4.add){.}->true
  match(S3.addListener,P4.remove){.}->false
  match(S3.removeListener,P4.add){.}->false
  match(S3.removeListener,P4.remove){.}->true
 } -> true (other matches S6-P1,S4-P5,S5-P3)
 backward{
         match (S4,P5){
         }->true (already satisfied)
 } -> true (satisfied)
} -> satisfied (binding S3-P4 successful)
```

Fig. 2. An example of a single binding during the detection of the Observer Pattern on a small excerpt of a system model

```
1.Set
   T = all system classifier properties
   P = all pattern specifications
   M = Matches list
2.while (Property sp = T.getNext())
3.foreach (Pattern p : P )
4.foreach (Property pp:p.getNext()) {
5. Binding b = match(p,pp,sp);
6. M.updateBinding(b,p);
  }
-----------------------------------------------
7.match(PatternSpecification p,
        ClassifierProperty pcp,
        ClassifierProperty scp){
8.  foreach ( Property pp :
                 cp.getProperties() ){
9.   foreach ( Property sp :
                 scp.getProperties() ){
10.    match(p,pp,sp);
11.   }
12.  }
13.  backwardBinding(p,cp,scp);
14.  return (p.hasForwardBinding() &&
               p.hasBackwardBinding());
15.}
```

Fig. 3. A sketch of the detection algorithm

in the Figure 2, right side, the matches between S1/S2 and all pattern classifiers fail. The match between S1 and P1 fails on the backward step because the pattern requires a method that is not present in S1. The match among S1 and P2 fails on forward because the nodes are of different types (S1 is an interface while P2 is a class). The node S2 doesn't match with all P_i since it requires a match on three methods, one returning and two taking as argument a type matchable with S1 (i.e. "Vertex" in the system). These three matches fail (some in forward others in backward) for all pattern types (this can be verified on the graph looking at neighbourhood of nodes S2 and P_i for all i). The first successful match is the one that binds the node S3 with the node P4 of the pattern specification. In this case the matches on the required behavioral properties are satisfied for the couple of methods (addListener,add) and (removeListener,remove) but fail for the others (since they must satisfy the dependencies towards the add/remove methods of the node P3). The backward step considers the Inheritance and try to match P5 with the system node S4. This match succeeds both in forward and backward (and indirectly binds also S5 to P3): for this reason the initial match S3-P4 is satisfied. This means that the AbstractSubject and AbstractObserver interfaces along with at least a ConcreteSubject and a ConcreteObserver are identified in the system along with their members. The algorithm continues to try all remaining matches, excluding the ones that already succeeded, until all mandatory pattern nodes are bounded to at least one system element (in this case at least a pattern instance is found) or until the system types are all evaluated and the pattern instance is not found.

4 The DPF Tool

To provide an automatic support to the mining approach the Design Pattern Finder (DPF) tool has been developed, implementing all the steps of the mining process. It is an Ecplise plug-ins set based upon JDT (to extract information from source code), and upon the EMF framework (implementing the meta-model). The tool uses the JDT platform to extract detailed information on the systems' static structure, such as: type hierarchy, type's inner structure (attributes, their types and scopes and so on), method and constructors signatures, method calls, object creations, container support in order to express containment within types, static member information, delegation. The pattern specifications are organized in a pattern catalog. Each pattern specification can be standalone or override a base specification by changing only some of its properties. After the system has been parsed and the model built, the user can select which design patterns are to be detected. After the execution of the algorithm, the detected patterns are shown (including with their internal members). Each identified pattern instance is traced to the source code elements implementing it and a user can visualize, inspect and analyze such code components.

5 Case Study

The proposed approach has been validated by applying it to two open source java software systems: JHotDraw 6, a Graphical Editor framework (made up of 25,308 LOC, 48 Classes and 35 Interfaces, and 2898 Methods), and JUnit 3.7, a unit testing framework for Java programs (made up of 9,743 LOC, 16 Classes and 11 Interfaces, and 851 Methods). These systems were chosen because their development is explicitly based on design patterns and hence they are adequate to evaluate design pattern mining approaches. Moreover, they have been extensively studied in literature and hence benchmark already exists for these systems.

Table 1. Overview of the results: design patterns identified and quality of detection

	JhotDraw 6					JUnit				
Pattern	Detected	True	FP	Precision	Recall	Detected	True	FP	Precision	Recall
AbstractFactory	5	2	3	0,40	1,00	1	1	0	1,00	1,00
Adapter	7	1	6	0,14	1,00	1	1	0	1,00	1,00
Command	1	1	0	1,00	1,00	1	1	0	1,00	1,00
Composite	2	2	0	1,00	1,00	1	1	0	1,00	1,00
Observer	2	2	0	1,00	1,00	4	3	2	0,50	0,67
Prototype	8	2	7	0,13	0,50	2	1	1	0,50	1,00
Singleton	3	3	1	0,67	0,67	1	1	0	1,00	1,00
State	12	5	8	0,33	0,80	3	3	1	0,67	0,67
Visitor	1	1	0	1,00	1,00	1	1	0	1,00	1,00

Thus it is easier to evaluate precision and recall providing quantitative data on the quality of detection. The considered systems have different sizes and this let us also to assess how the proposed matching algorithm scales with respect to the system size.

To assess the effectiveness and the correctness of the approach, its results were compared with the ones indicated by an expert. The expert has validated the DPs identified by the tool by comparing them with the results he got by analysing the systems' code and documentation, and the results from other works known in the literature. The Table 1 reports, for each of the two analysed systems: the name of the DPs searched in the code (column 'Pattern'); the number of each searched Pattern detected by the proposed approach (column 'Detected'); the number of each searched Pattern the expert said to exist actually in the system (column 'True'); the number of False Positive (column 'FP'), i.e. the number of DPs instances detected by the tool but not validated by the expert; the values of precision and recall for the results by the tool (columns 'Precision' and 'Recall' respectively). Patterns like Command, Composite or Observer but also Visitor (that is based on double dispatch) are better identified since their specifications include both static and behavioral relationships. They have a lower number of false positives than those patterns with a less constrained structure or with limited or absent behavioral properties.

Some false negatives, i.e. pattern instances that exist in the system (as specified by the expert) but missed by the detection process, were found along the validation process. The false negatives were related to patterns that were implemented by a not standard way. Anyway the number of false negative was lower than the false positive one. False negatives occured for State, Singleton, and Prototype DPs in JHotDraw, and for State and Observer in JUnit. The occurence of false negatives was also due and related to DPs variants. Indeed, the number of false negatives was reduced by defining the specifications of new variants inheriting existing ones that took into account the structural differences. This allowed to detect those pattern instances that were not identified because of few small structural differences originating the variants.

For the JHotDraw system, the recall is optimal for most of the detected DPs, while in many cases the precision is less than 0.5; this was because some pattern specifications were particularly relaxed. For instance, in the case of the AbstractFactory DP, we considered as an AbstractFactory even a class owning a single method creating and returning an instance of a product. The analysis of JHotDraw gave better results for the precision. The proposed mining approach can help to distinguish among patterns that have same static structure but different behaviours. For example, in order to distinguish between the Command and Adapter (the object version) DPs, the approach uses information inside the invocation property requiring that the Execute method, in the concrete subclass, is implemented by invoking a method of a class still bound to a Command. The same is for Composite/Decorator where the Decorator is required to specify a delegation towards the decorated object. The approach identifies patterns from source code and hence all third party libraries doesn't participate into detection process. Anyway, the approach supports some of the standard Java environments meaning that we map all the Java collection framework classes and interfaces onto our notion of Container.

In the experiments we used a repository containing the specifications of three variants of the Observer DP, two variants for the Composite, and three variants for the Singleton. The first variant of the Observer (we call it as the variant A) uses the Java types (Observable class and Listener interface) while the second (B) uses a generic interface to be found within system classes and the third one (C) defines a multi-event Observer in which type of event is passed to the notify method. For the Composite pattern only two versions are defined: the classic one proposed in literature by Gamma [6] (version A) and a version in which an intermediate abstract class implementing the core method for components has been inserted in the Component hierarchy (version B). For the Singleton DP we have a first version (A) with static getter and private constructor; a second one (B) taking concurrency into account, and a third version (C) in which there is a single instance for each object identifier passed to the singleton itself. The two systems used different variants of the Composite and Observer patterns (the A variant for JHotDraw, and the B variant for JUnit), while they used the same variant of the Singleton pattern (the A one). Within the analysed systems we did not found instances of different variants of a same pattern. It would be interesting to perform such analysis on a wider set of systems to see if different variants of a same pattern actually coexists in the same system; future work will include this analysis.

The execution times for each step of the detection process were:

– JHotDraw: Step1=1281 ms - Step2= 7838 ms - Step3=11480;
– JUnit: Step1=157 ms - Step2= 1199 ms - Step3=3502;

The total time for JHotDraw was 20599 ms and 4858 ms for JUnit. The Step 3 'Patterns' models matching' resulted the most CPU time consuming. The average time (2288.78 ms for JHotDraw and 539.78 for JUnit) for a single pattern resulted to be comparable to the time of other approaches. The approach is more effective when each specification is focused on well defined patterns' variants with explicit and mutually exclusive constraints among them. The worst results were obtained for overlapping pattern specifications in which only one or two properties were different: when specifications are badly written (i.e., few and overlapping constraints) the performance of the algorithm degrades rapidly. Of course, the pattern detection is highly affected by the number of nodes in both system and pattern specifications. Moreover, since the algorithm is pivoted on matches among types, the number of types is the major characteristic affecting the overall performances.

One of the most important limitation regards the generation of behavioral properties in presence of late binding. In this case a call graph using Rapid Type Analysis (RTA) is built to reduce the set of possible callers. However the call graph still contains a super-set of the actual calls. In the property extraction algorithm we decided to take into account the sets of possible targets in order to perform the matches. In this way, surely we don't miss any possible binding but this exposes the algorithm to the presence of false positives (since the set of successful binding is a super-set of the actual ones). Further experimentation (with more strict policies) should be performed in order to assess if the behavior of the algorithm improves with respect to this conservative choice. The algorithm is faster when there are patterns to be found in the analysed system. The worst performances are obtained when there are no pattern instances in the system (since all matches need to be executed and no existing bindings are used to reduce the list of remain matches to be evaluated).

6 Related Work

A review on current techniques and tools for discovering architecture and design patterns from OO systems, are provided in [5].

In [2], De Lucia et al. present some case studies of recovering structural design patterns from OO source code. Di Penta et al. in [3] present an empirical study to understand whether in DPs' modifications, along with the maintenance/evolution of a system, there are roles more change-prone than others and whether there are changes that are more likely to occur for certain roles. In [4] an approach to discover design patterns is described by defining the structural characteristics of each DP in terms of weights and matrix. In [15], [8], Gueheneuc et al. propose an approach to semiautomatically identify microarchitectures that are similar to design motifs in source code and to ensure the traceability of these microarchitectures between implementation and design. Moreover in [9], Gueheneuc et al. present a study about classes playing zero or more roles in six different DPs.

A design pattern detection methodology based on similarity scoring between graph vertices is proposed in [14]; the approach is able to recognize also patterns that are modified from their standard representation. The main difference among this approach and the one we propose regards the matching algorithm: our representation is graph-based (instead of matrix-based) and is driven by a meta-model (instead to be hard-coded) that speficies the information to be taken into account during matching between pattern models and system models. In our approach pattern specifications can be written and inserted into a repository to became available to the pattern matching algorithm. Moreover specifications can be overridden to increase the catalog of detected patterns and to handle variants in an effective way.

An approach to overcome the scalability problems due to the many design and implementation variants of design pattern instances is proposed in [12]. It is based on a recognition algorithm working incrementally and requiring human intervention. Our approach differs from this one because it is completely based on information recovered from the defined meta-model and specified in the pattern models, and it does not involve user to drive the detection algorithm.

7 Conclusions and Future Work

An approach to detect DPs in existing OO systems has been presented. A meta-model has been defined to represent DPs by a set of Properties specifying each DP, and the system to mine. The identification of DPs is carried out by performing an algorithm that matches the models of the DPs against the model of the system to detect those components cooperating in a way that satisfies the model of a pattern. The method is able to detect multiple instances of the same pattern and to find several pattern instances even when collapsed into a single class. The approach also allows to identify DPs variants as modifications to pattern specifications already defined. The approach has been applied to two Java systems producing good results. Future work will consider the extension of the catalog of the patterns to identify, the evolution of the prototype of the DPF tool and the execution of further experimentation on a wider set of open source systems.

References

1. Bernardi, M.L., Di Lucca, G.A.: Model driven detection of design pattern. In: Proceedings of the 26th IEEE International Conference on Software Maintenance (ICSM 2010), ERA Track, Washington, DC, USA. IEEE Computer Society, Los Alamitos (2010)
2. Costagliola, G., De Lucia, A., Deufemia, V., Gravino, C., Risi, M.: Design pattern recovery by visual language parsing. In: Proc. of the Ninth European Conference on Software Maintenance and Reengineering, CSMR 2005, Washington, DC, USA, pp. 102–111. IEEE Computer Society, Los Alamitos (2005)
3. Di Penta, M., Cerulo, L., Gueheneuc, Y.-G., Antoniol, G.: An empirical study of the relationships between design pattern roles and class change proneness. In: Proc. IEEE Int. Conf. on Software Maintenance, ICSM 2008, pp. 217–226 (2008)
4. Dong, J., Lad, D.S., Zhao, Y.: Dp-miner: Design pattern discovery using matrix. In: Proc. 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS 2007, pp. 371–380 (2007)
5. Dong, J., Zhao, Y., Peng, T.: Architecture and design pattern discovery techniques - a review. In: Arabnia, H.R., Reza, H. (eds.) Software Engineering Research and Practice, pp. 621–627. CSREA Press (2007)
6. Johnson, R., Vlissides, J., Gamma, E., Helm, R.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, Reading (1998)
7. Fulop, L.J., Ferenc, R., Gyimothy, T.: Towards a benchmark for evaluating design pattern miner tools. In: Proc. 12th European Conference on Software Maintenance and Reengineering, CSMR 2008, April 1-4, pp. 143–152 (2008)

8. Gueheneuc, Y.-G., Antoniol, G.: Demima: A multilayered approach for design pattern identification. IEEE Trans. on Software Engineering 34(5), 667–684 (2008)
9. Khomh, F., Guéhéneuc, Y.-G., Antoniol, G.: Playing roles in design patterns: An empirical descriptive and analytic study. In: ICSM, pp. 83–92. IEEE, Los Alamitos (2009)
10. Kim, D.H., Yun, I.D., Lee, S.U.: Attributed relational graph matching based on the nested assignment structure. Pattern Recogn. 43(3), 914–928 (2010)
11. Philippsen, M., Prechelt, L., Unger-Lamprecht, B., Tichy, W.F.: Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. IEEE Trans. Softw. Eng. 28(6), 595–606 (2002)
12. Niere, J., Schäfer, W., Wadsack, J.P., Wendehals, L., Welsh, J.: Towards pattern-based design recovery. In: Proceedings of the 24th International Conference on Software Engineering, ICSE 2002, pp. 338–348. ACM, New York (2002)
13. Smith, J.M., Stotts, D.: Spqr: flexible automated design pattern extraction from source code. In: Proc. 18th IEEE International Conference on Automated Software Engineering, October 6-10, pp. 215–224 (2003)
14. Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., Halkidis, S.T.: Design pattern detection using similarity scoring. IEEE Transactions on Software Engineering 32(11), 896–909 (2006)
15. Gueheneuc, Y.-G., Sahraoui, H., Zaidi, F.: Fingerprinting design patterns. In: Proc. 11th Working Conference on Reverse Engineering, pp. 172–181 (2004)

Exploring Empirically the Relationship between Lack of Cohesion and Testability in Object-Oriented Systems

Linda Badri, Mourad Badri, and Fadel Toure

Software Engineering Research Laboratory
Department of Mathematics and Computer Science
University of Quebec at Trois-Rivières, Trois-Rivières, Québec, Canada
`{Linda.Badri,Mourad.Badri,Fadel.Toure}@uqtr.ca`

Abstract. The study presented in this paper aims at exploring empirically the relationship between lack of cohesion and testability of classes in object-oriented systems. We investigated testability from the perspective of unit testing. We designed and conducted an empirical study using two Java software systems for which JUnit test cases exist. To capture testability of classes, we used different metrics to measure some characteristics of the corresponding JUnit test cases. We used also some lack of cohesion metrics. In order to evaluate the capability of lack of cohesion metrics to predict testability, we performed statistical tests using correlation. The achieved results provide evidence that (lack of) cohesion may be associated with (low) testability.

Keywords: Software Attributes, Quality Attributes, Lack of Cohesion, Testability, Metrics, Relationship, Empirical Analysis.

1 Introduction

Cohesion is considered as one of most important object-oriented (OO) software attributes. Many metrics have been proposed in the last several years to measure class cohesion in object-oriented systems (OOS). Class cohesion (more specifically, functional cohesion) is defined as the degree of relatedness between members of a class. In OOS, a class should represent a single logical concept, and not to be a collection of miscellaneous features. OO analysis and design methods promote a modular design by creating high cohesive classes (Larman, 2003; Pressman, 2005; Sommerville, 2004). However, improper assignment of responsibilities in the design phase can produce low cohesive classes with unrelated members. The reasoning is that such (poorly designed) classes will be difficult to understand, to test and to maintain. However, there is no empirical evidence on these beliefs. In fact, studies have failed to show a significant relationship between, for example, cohesion metrics and software quality attributes such as fault-proneness or changeability (Briand, 1998; Briand, 2000; Kabaili, 2001). Moreover, studies have noted that cohesion metrics fail in many situations to properly reflect cohesion of classes (Aman, 2002; Chae, 2000; Chae, 2004; Kabaili, 2000; Kabaili, 2001).

One possible explanation of the lack of relationship between cohesion and some software quality attributes is, according to some authors (Briand, 1998; Briand, 2000; Henderson-Sellers, 1996; Stein, 2005), the difficulty of measuring cohesion from syntactic elements of code. We believe that, effectively, major of existing cohesion metrics (known as structural metrics) can give cohesion values that do not reflect actually the disparity of the code, in the sense that they capture some structural links between parts of code that may be conceptually disparate. Moreover, we believe also that cohesion metrics are based on restrictive criteria and do not consider some characteristics of classes. These weaknesses lead, in fact, in many situations to some inconsistency between the computed cohesion values and the intuitively expected ones (Badri, 2004; Chae, 2000; Chae, 2004; Kabaili, 2001). An empirical study conducted by Stein *et al.* (Stein, 2005) pointed, however, to a more basic relationship between cohesion and complexity: that a lack of cohesion may be associated with high complexity.

In this paper, we decided to explore empirically the relationship between lack of cohesion (disparity of the code) and testability of classes in OOS. The objective is also to get a better understanding of testability and particularly the contribution of (lack of) cohesion to testability. Testability refers to the degree to which a software artifact facilitates testing in a given test context (Voas, 1995; Freedman, 1991; Le Traon, 2000; Jungmayr, 2002). Software testability is related to testing effort reduction and software quality (Gao, 2005). It impacts test costs and provides a means of making design decisions based on the impact on test costs (Sheppard, 2001). Several software development and testing experts pointed out the importance of testability and design for testability, especially in the case of large and complex systems. Software testability is affected by many different factors, including the required validity, the process and tools used, the representation of the requirements, and so on (Bruntink, 2006). Yeh et *al.* (Yeh, 1998) state that diverse factors such as control flow, data flow, complexity and size contribute to testability. According to Zhao (Zhao, 2006), testability is an elusive concept, and it is difficult to get a clear view on all the potential factors that can affect it. Furthermore, Baudry et *al.* (Baudry, 2003) argue that testability becomes crucial in the case of OOS where control flows are generally not hierarchical but distributed over whole architecture.

We designed and conducted an empirical study to evaluate the capability of lack of cohesion to predict testability of classes. This paper investigates testability from the perspective of unit testing, where the units consist of classes of an OO software system. We focused on white box testing of classes. We used for our study two Java software systems for which JUnit test cases exist. To capture testability of classes, we used different metrics to measure some characteristics of the corresponding JUnit test classes. In order to test our hypothesis, we chose in our experiment two well-known lack of cohesion metrics: LCOM (Lack of COhesion in Methods) (Chidamber, 1994) and LCOM* (Henderson-Sellers, 1996). To facilitate comparison with our class cohesion measurement approach (Badri, 2004; Badri, 2008), and knowing that the selected cohesion metrics are basically lack of cohesion metrics (inverse cohesion measures), we derive a lack of cohesion measure (following the same approach of LCOM) from the cohesion metric we proposed. In order to evaluate the capability of the lack of cohesion metrics to predict testability, we used statistical tests using correlation.

The rest of the paper is organized as follows: Section 2 gives a brief survey on related work on (predicting) software testability. Section 3 presents an overview of major class cohesion metrics. Section 4 presents briefly our approach for class cohesion assessment. In section 5 we describe the experimental design, define the used metrics and discuss the statistical technique we used to investigate the relationship between lack of cohesion and testability metrics. Section 6 presents some characteristics of the used systems. We also present and discuss the obtained results. Finally, conclusions and some future work directions are given in section 7.

2 Software Testability

IEEE (IEEE, 1990) defines testability as the degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met. ISO (ISO, 1991) defines testability (characteristic of maintainability) as attributes of software that bear on the effort needed to validate the software product.

Fenton et *al.* (Fenton, 1997) define testability as an external attribute. Freedman introduced testability measures for software components based on two factors: observability and controllability (Freedman, 1991). Voas defines testability as the probability that a test case will fail if the program has a fault (Voas, 1992). Voas and Miller propose a testability metric based on the inputs and outputs domains of a software component (Voas, 1993), and the PIE (Propagation, Infection and Execution) technique to analyze software testability (Voas, 1995).

Binder (Binder, 1994) discusses software testability based on six factors: representation, implementation, built-in text, test suite, test support environment and software process capability. Khoshgoftaar et *al.* modeled the relationship between static software product measures and testability (Khoshgoftaar, 1995) and applied neural networks to predict testability from static software metrics (Khoshgoftaar, 2000). McGregor et *al.* (McGregor, 1996) investigate testability in OOS and introduce the visibility component (VC) measure. Bertolino et *al.* (Bertolino, 1996) investigate the concept of testability and its use in dependability assessment. Le Traon et *al.* (Le Traon, 1995; Le Traon, 1997; Le Traon, 2000) propose testability measures for dataflow designs. Petrenko et *al.* (Petrenko, 1993) and Karoui et *al.* (Karoui, 1996) address testability in the context of communication software. Sheppard et *al.* (Sheppard, 2001) focuses on formal foundation of testability metrics.

Jungmayr (Jungmayr, 2002) investigates testability measurement based on static dependencies within OOS. Gao et *al.* consider testability from the perspectives of component-based software construction (Gao, 2003), and address component testability issues by introducing a model for component testability analysis (Gao, 2005). Nguyen et *al.* (Nguyen, 2003) focused on testability analysis based on data flow designs in the context of embedded software. Baudry et *al.* addressed testability measurement (and improvement) of OO designs (Baudry, 2003; Baudry, 2004). Bruntink et *al.* (Bruntink, 2004) evaluated a set of OO metrics with respect to their capabilities to predict testability of classes of a Java system. More recently, Chowdhary (Chowdhary, 2009) focuses on why it is so difficult to practice testability in the real world.

3 Cohesion Metrics

Yourdon *et al.* (Yourdon, 1979) defined cohesion, in the procedural programming paradigm, as a measure of the extent of the functional relationships between the elements of a module. In the OO paradigm, Booch (Booch, 1994) described high functional cohesion as existing when the elements of a class all work together to provide some well-bounded behavior. There are several types of cohesion: functional cohesion, sequential cohesion, coincidental cohesion, etc. (Henderson-Sellers, 1996; Yourdon, 1979). In this work, we focus on functional cohesion.

Many metrics have been proposed in the last several years in order to measure class cohesion in OOS. The argument over the most meaningful of those metrics continues to be debated (Counsell, 2006). Major of proposed cohesion metrics are based on the notion of similarity of methods, and usually capture cohesion in terms of connections between members of a class. They present, however, some differences in the definition of the relationships between members of a class. A class is more cohesive, as stated in (Chae, 2000), when a larger number of its instance variables are referenced by a method ($LCOM^*$ (Henderson-Sellers, 1996), Coh (Briand, 1998)), or a larger number of methods pairs share instance variables (LCOM1 (Chidamber, 1991), LCOM2 (Chidamber, 1994), LCOM3 (Li, 1993), LCOM4 (Hitz, 1995), Co (Hitz, 1995), TCC and LCC (Bieman, 1995), DC_D and DC_I (Badri, 2004)).

These metrics are known as structural metrics, which is the most investigated category of cohesion metrics. They measure cohesion on structural information extracted from the source code. Several studies using the Principal Component Analysis technique have been conducted in order to understand the underlying orthogonal dimensions captured by some of these metrics (Aggarwal, 2006; Briand, 1998; Chae, 2000; Etzkorn, 2004; Marcus, 2005). Briand *et al.* (Briand, 1998) developed a unified framework for cohesion measurement in OOS that classifies and discusses several cohesion metrics. Development of metrics for class cohesion assessment still continues (Badri, 2008; Chae, 2004; Chen, 2002; Counsell, 2006; Marcus, 2005; Marcus, 2008; Meyers, 2004; Woo, 2009; Zhou, 2002; Zhou, 2003). Recent approaches for assessing class cohesion focus on semantic cohesion (De Lucia, 2008; Marcus, 2008). We focus in this work on structural cohesion metrics.

4 Class Cohesion Measurement

We give, in this section, a brief overview of our approach for class cohesion assessment. For more details see (Badri, 2004; Badri, 2008). The adopted approach for the estimation of class cohesion is based on different functional cohesion criteria:

- *Used Attributes*: Two methods M_i and M_j are directly related if there is at least one attribute shared by the two methods.
- *Invoked Methods*: Two methods M_i and M_j are directly related if there is at least one method invoked by the two methods. We also consider that M_i and M_j are directly related if M_i invokes M_j, or vice-versa.
- *Common Objects Parameters*: Two methods M_i and M_j are directly related if there is at least one parameter of object type used by the two methods.

Let us consider a class *C* with *n* methods. The number of methods pairs is [n * (n – 1) / 2]. Consider an undirected graph G_D, where the vertices are the methods of the class *C*, and there is an edge between two vertices if the corresponding methods are directly related. Let E_D be the number of edges in the graph G_D. The cohesion of the class *C*, based on the direct relation between its methods, is defined as: $DC_D = |E_D| / [n * (n - 1) / 2] \in [0,1]$. DC_D gives the percentage of methods pairs, which are directly related. As mentioned in Section1, we associate to a class *C* a lack of cohesion measure (not normalized) based on the direct relation given by: $LC_D = [n * (n - 1) / 2] - 2 * |E_D|$. When the difference is negative, LC_D is set to zero.

5 Experimental Design

We present, in this section, the empirical study we conducted to explore the relationship between lack of cohesion and testability. We performed statistical tests using correlation. The null and alternative hypotheses are:

- H_0 : There is no significant correlation between lack of cohesion and testability.
- H_1 : There is a significant correlation between lack of cohesion and testability.

The objective is to assess how extent the selected lack of cohesion metrics can be used to predict class testability. In this experiment, rejecting the null hypothesis indicates that there is a statistically significant relationship between lack of cohesion metrics and the used testability metrics (chosen significance level $\alpha = 0.05$).

5.1 Selected Metrics

Metrics related to lack of cohesion

In this experiment, we chose the lack of cohesion metrics LCOM (Chidamber, 1998), LCOM* (Henderson-Sellers, 1996), and LC_D. LCOM is defined as the number of pairs of methods in a class, having no common attributes, minus the number of pairs of methods having at least one common attribute. LCOM is set to zero when the value is negative. LCOM* is somewhat different from the LCOM metric. LCOM* is different also from the other versions of the LCOM metric proposed by Li *et al.* (Li, 1993) and Hitz *et al.* (Hitz, 1995). It considers that cohesion is directly proportional to the number of instance variables that are referenced by the methods of a class.

Metrics related to testability

The objective of this paper is to explore empirically to what extent lack of cohesion can be used to predict testability (in terms of testing effort) of classes in OOS. For our experiments, we selected from each of the used systems only the classes for which JUnit test cases exist. To indicate the testing effort required for a software class (noted C_s), we used two metrics, introduced by Bruntink et *al.* in (Bruntink, 2004), to quantify the corresponding JUnit test class (noted C_t).

JUnit[1] (www.junit.org) is a simple framework for writing and running automated unit tests for Java classes. A typical usage of JUnit is to test each class C_s of the program by means of a dedicated test class C_t. We used in our experiments each pair $<C_s, C_t>$, for classes for which test cases exist, to compare characteristics of C_s's code with characteristics of the corresponding test class C_t. The objective in this paper is to use these pairs to evaluate the capability of lack of cohesion metrics to predict the measured characteristics of the test classes C_t. To capture the testability of classes, we decided to measure for each test class C_t, corresponding to a software class C_s, two characteristics:

- *TNbLOC*: This metric gives the number of lines of code of the test class C_t. It is used to indicate the size of the test suite corresponding to a class C_s.
- *TNbOfAssert*: This metric gives the number of invocations of JUnit *assert* methods that occur in the code of a test class C_t. The set of JUnit *assert* methods are, in fact, used by the testers to compare the expected behavior of the class under test to its current behavior. This metric is used to indicate another perspective of the size of a test suite.

The metrics *TNbLOC* and *TNbOfAssert* have already been used by Bruntink et *al.* (Bruntink, 2004; Bruntink, 2006) to indicate the size of a test suite. Bruntink et *al.* based the definition of these metrics on the work of Binder (Binder, 1994). We assume, in this paper, that these metrics are indicators of the testability of software classes C_s. The used metrics reflect, in fact, different source code factors as stated by Bruntink et *al.* in (Bruntink, 2004; Bruntink, 2006): factors that influence the *number of test cases required* to test the classes of a system, and factors that influence the *effort required* to develop *each individual test case*. These two categories have been referred as *test case generation factors* and *test case construction factors*.

Metrics data collection

The metrics LCOM, LCOM* and TNbLOC have been computed using the Borland Together tool. The metrics LC_D and TNbOfAssert have been computed using the tool we developed.

5.2 Statistical Analysis

For the analysis of the collected data we used the Spearman's correlation coefficient. This technique, based on ranks of the observations, is widely used for measuring the degree of linear relationship between two variables (two sets of ranked data). It measures how tightly the ranked data clusters around a straight line. Spearman's correlation coefficient will take a value between -1 and +1. A positive correlation is one in which the ranks of both variables increase together. A negative correlation is one in which the ranks of one variable increase as the ranks of the other variable decrease. A correlation of +1 or -1 will arise if the relationship between the ranks is exactly linear. A correlation close to zero means that there is no linear relationship between the ranks. We used the XLSTAT software to perform the statistical analysis.

[1] `www.junit.org`

6 The Case Studies

6.1 Selected Systems

In order to achieve significant results, the data used in our empirical study were collected from two open source Java software systems. This selection was essentially based on the number of classes who underwent testing using the JUnit framework. The selected systems are:

- ANT (www.apache.org): a Java-based build tool, with functionalities similar to the unix "make" utility.
- JFREECHART (http://www.jfree.org/jfreechart): a free chart library for the Java platform.

Table 1. Some characteristics of the used systems

	# LOC	# Classes	Mean LOC	# Attributes	# Methods	# Test Classes	MeanLOC TestedCL	# LOC TestedCL
ANT	64062	713	89.85	2419	5365	115	153.52	17655
JFC	92077	795	115.82	1616	7308	230	231.00	53131

Table 1 summarizes some characteristics of ANT and FREECHART (JFC) systems: total number of lines of code, total number of classes, average value of lines of code, total number of attributes, total number of methods, number of JUnit test classes, average value of lines of code of the software classes for which JUnit test classes have been developed, total number of lines of code of the software classes for which JUnit test classes have been developed. These data will be used in the discussion section (Section 6.3).

6.2 Results

The analysis of the data sets is done by calculating the Spearman's correlation coefficients for each pair of metrics (TNbLOC-LCOM, TNbLOC-LCOM*, TNbLOC-LC_D) and (TNbOfAssert-LCOM, TNbOfAssert-LCOM*, TNbOfAssert-LC_D). We have a total of six pairs of metrics. Table 2 and Table 3 summarize the results of the correlation analysis. They show, for each system and between each distinct pair of metrics, the obtained values for the Spearman's correlation coefficients.

Table 2. Correlation values for ANT

	Variables	TNbOfAssert	TNbLOC
ANT	LC_D	**0.303**	**0.396**
	LCOM	**0.326**	**0.404**
	LCOM*	**0.218**	**0.237**

Table 3. Correlation values for JFREECHART

	Variables	TNbOfAssert	TNbLOC
JFR	LC_D	**0.392**	**0.315**
	LCOM	**0.424**	**0.379**
	LCOM*	**0.199**	0.117

6.3 Discussion

Table 2 and Table 3 give the obtained Spearman's correlation coefficients. They are all significant at $\alpha=0.05$ (indicated in bold) except for the pair of metrics LCOM*-TNbLOC for JFREECHART. Moreover, all measures have positive correlation. Since the used cohesion metrics are lack of cohesion measures, the positive coefficients indicate that the ranks of both TNbOfAssert and TNbLOC and lack of cohesion metrics increase together. The achieved results support the idea that there is a statistically significant relationship between lack of cohesion and testability, in the sense that the more the lack of cohesion of a class is high, the more important its testing effort is likely to be (which is reflected by the two metrics TNbOfAssert and TNbLOC). We reject then the null hypothesis.

For ANT, LCOM and LC_D metrics are significantly better predictors of the number of lines of code of test classes (TNbLOC) than the number of test cases (TNbOfAssert). By cons, for JFREECHART, LCOM and LC_D metrics are significantly better predictors of the number of test cases (TNbOfAssert) than the number of lines of code of test classes (TNbLOC). The results for JFREECHART also show that the correlation values are not significant for the metric LCOM * particularly with the number of lines of code of test classes (TNbLOC). Moreover, for both ANT and JFREECHART, LCOM is slightly better predictor of the number of test cases (TNbOfAssert) and the number of lines of code of test classes (TNbLOC) than LC_D, which gives better results than LCOM*.

By analyzing the values of the used lack of cohesion metrics more closely, we found that LC_D indicates, on average, a lower lack of cohesion value for both systems (ANT: 123. 37 and JFreeChart: 303.59) than LCOM (ANT: 151.53 and JFreeChart: 350,780). This difference is, in fact, explained by the difference between the cohesion criteria used by the two metrics (definition of the measures themselves). The two metrics share the attribute usage criterion. The metric LC_D uses, however, two other criteria as mentioned in Section 4. This makes that the metric LC_D captures more pairs of connected methods than the metric LCOM (also LCOM*). This difference leads in general to LC_D values that are lower than LCOM values. Indeed, several experiments in our previous work (Badri, 2004; Badri, 2008) have showed that the metric LC_D, by capturing more pairs of connected methods than LCOM, gives lower values of lack of cohesion (which is plausible). Moreover, we observed also that (for the considered case studies), overall, lack of cohesion values seem increasing with the size of classes (and systems), which is plausible. In effect, large classes tend to lack cohesion. These classes tend to have a (relatively) high number of attributes and methods. It is harder in this case to have a high number of pairs of related methods (according to cohesion criteria). By cons, cohesive classes tend to have a relatively low number of attributes and methods. This makes, in our opinion, the metrics LCOM and LC_D (and particularly LCOM) sensitive to size. This may explain why LCOM is slightly better correlated in some cases (than LC_D) with the used testability metrics. Also, by analyzing the source code of the JUnit test classes, we feel that some characteristics of test classes are not captured by the used testability metrics (like the response set of a test class which may indicate the effort required for testing the interactions with the other classes to which a class under test is coupled).

After these first observations, we decided to extend (and replicate) our experiments by introducing a complementary set of metrics for attempting to explain the first observations we made. We used, in fact, five other metrics. We used two metrics (TRFC and TWMPC) to capture additional dimensions of test classes. The TRFC metric gives the size of the response set for the test class C_t corresponding to a software class C_s (like the traditional RFC (Chidamber, 1994) metric for a software class). The RFC of a test class C_t is a count of methods of C_t and the number of methods of other classes that are invoked by the methods of C_t. It includes methods that can be invoked on other objects. A class C_t, which provides a larger response set than another will be considered as more complex. This will reflect another perspective of the testing effort corresponding to a class under test. The TWMPC metric (like the traditional WMPC (Chidamber, 1994) metric for a software class) gives the sum of the complexities of the methods of a test class, where each method is weighted by its cyclomatic complexity. Only methods specified in a test class are included. We used also three code based metrics to capture size of software classes for which JUnit test classes exist: LOC (lines of code), NOA (number of attributes) and NOM (number of methods).

Table 4. Correlation values between the used metrics for ANT

ANT	TNbOfAssert	TNbLOC	TRFC	TWMPC	LC_D	LCOM	LCOM*	LOC	NOA	NOO
TNbOfAssert	**1**									
TNbLOC	**0.667**	**1**								
TRFC	0.067	**0.477**	**1**							
TWMPC	0.140	**0.574**	**0.796**	**1**						
LC_D	**0.303**	**0.396**	**0.236**	**0.365**	**1**					
LCOM	**0.326**	**0.404**	0.164	**0.283**	**0.959**	**1**				
LCOM*	**0.218**	**0.237**	0.139	**0.244**	**0.677**	**0.669**	**1**			
LOC	**0.350**	**0.507**	**0.395**	**0.396**	**0.750**	**0.761**	**0.650**	**1**		
NOA	0.136	**0.303**	**0.316**	**0.344**	**0.679**	**0.635**	**0.731**	**0.709**	**1**	
NOO	**0.325**	**0.460**	**0.258**	**0.329**	**0.886**	**0.908**	**0.685**	**0.858**	**0.763**	**1**

Table 5. Correlation values between the used metrics for JFREECHART

JFR	TNbOfAssert	TNbLOC	TRFC	TWMPC	LC_D	LCOM	LCOM*	LOC	NOA	NOO
TNbOfAssert	**1**									
TNbLOC	**0.840**	**1**								
TRFC	**0.727**	**0.862**	**1**							
TWMPC	**0.570**	**0.732**	**0.792**	**1**						
LC_D	**0.392**	**0.315**	**0.332**	0.121	**1**					
LCOM	**0.424**	**0.379**	**0.399**	**0.203**	**0.952**	**1**				
LCOM*	**0.199**	0.117	0.125	-0.044	**0.625**	**0.583**	**1**			
LOC	**0.393**	**0.426**	**0.459**	**0.326**	**0.704**	**0.772**	**0.510**	**1**		
NOA	**0.357**	**0.353**	**0.275**	0.056	**0.731**	**0.719**	**0.720**	**0.678**	**1**	
NOO	**0.479**	**0.467**	**0.526**	**0.311**	**0.761**	**0.811**	**0.577**	**0.814**	**0.721**	**1**

We performed, for a second time, our experiments using all the selected metrics (lack of cohesion, testability and size metrics).The correlation values between the metrics are given in Table 4 and Table 5 respectively for systems ANT and FREECHART. The obtained correlations seem confirming our first observations (the significant values are indicated in bold). In effect, from Table 4 and Table 5 we can observe that:

- For ANT (Table 4), the three lack of cohesion metrics LCOM, LCOM* and LC_D are significantly correlated to the metric TWMPC, which indicates the cyclomatic complexity of a test class. In this case, the metric LC_D is better predictor of the cyclomatic complexity of the test class than the metrics LCOM and LCOM*. Moreover, the metric LC_D is also significantly correlated to the metric TRFC, which indicates the response set of a test class. By cons, the metrics LCOM and LCOM* are not correlated to the metric TRFC. For JFREECHART (Table 5), only the metric LCOM is significantly correlated with TWMPC. Moreover, the metrics LC_D and LCOM are significantly correlated with the metric TRFC. LCOM* is not correlated with the metric TRFC.
- For both ANT and JFREECHART, the three lack of cohesion metrics LCOM, LCOM* and LC_D are significantly correlated (and strongly correlated in some cases) to size metrics (LOC, NOA and NOM). Overall, the metric LCOM is better correlated to size metrics than LC_D and LCOM*. This was somewhat a surprising result. In effect, in a previous work (Badri, 2010), we demonstrated using several OOS that LC_D is better correlated to size metrics than LCOM. However, in the present work, as mentioned previously, we analyzed only software classes for which JUnit test cases have been developed. The number of tested classes for each of the used systems is given in Table 1 (115 classes for ANT and 230 classes for JFREECHART). This may affect the results of the study. Moreover, LCOM* is better correlated to the NOA size metric than LOC and NOO metrics. In effect, LCOM* considers that cohesion is directly proportional to the number of instance variables that are referenced by the methods of a class.
- For ANT, the correlation values between the metrics TRFC and TWMPC and the metric TNbOfAssert are not significant. By cons, the correlation values between the metrics TRFC and TWMPC and the metric TNbLOC are significant. For JFREECHART, the correlation values between the metrics TRFC and TWMPC and the metric TNbOfAssert are significant. It is also the case for the correlation values between TRFC and TWMPC and the metric TNbLOC. We can also observe that, in general, the correlation values between the testability metrics (TNbOfAssert, TNbLOC, TRFC and TWMPC) and the software classes' size metrics (LOC, NOA and NOM) are higher in the case of JFREECHART.

Table 6. Mean values of complexity and size metrics

	Mean LOC	Mean WMPC	Mean WMPC TestedCL	Mean LOC TestedCL
ANT	89.85	17.1	30.37	153.52
JFC	115.82	19.91	46.08	231.00

Consider now Table 6, which gives some descriptive statistics on the used systems: average number of lines of code of software classes of a system (MeanLOC), average cyclomatic complexity of software classes (MeanWMPC), average cyclomatic complexity of the tested software classes (MeanWMPCTestedCL), and average number of lines of code of the tested software classes (MeanLOCTestedCL). To collect data for lack of cohesion (and other) metrics we only used the classes for which JUnit test cases exist. As mentioned previously, for ANT we used 115 software classes and corresponding JUnit test cases (which represents 16 % of the classes of the system), and for JFREECHART we used 230 software classes and corresponding JUnit test cases (which represents 29 % of the classes of the system). Moreover, from Table 6 we can observe that the classes for which JUnit test cases have been developed, classes which we used in our experiments, are complex and large classes. This is true for both ANT and JFREECHART systems. Moreover, the tested classes of JFREECHART are more complex (and larger) than the tested classes of ANT. This may affect the results of our study in the sense that depending on the methodology followed by the developers while developing test classes and the criteria they used while selecting the software classes for which they developed test classes (randomly or depending on their size or complexity for example, or on other criteria) the results may be different. This may explain why the metric LCOM is (in many cases) slightly better correlated to the used testability (and size) metrics than LC_D. It would be interesting to replicate this study using systems for which JUnit test cases have been developed for a maximum number of classes. This will allow observing correlation values between the used metrics for different types of classes (small, medium and large classes). Moreover, by analyzing the source code of the JUnit test classes, we observed also (for ANT as well as for JFREECHART) that, in many cases, they do not cover all the methods of the corresponding software classes. This may also affect the results of the study.

7 Conclusions

This paper investigates the relationship between lack of cohesion and testability in OOS. The objective was also to get a better understanding of testability and particularly of the contribution of (lack of) cohesion to testability. As a first attempt, we designed and performed an empirical study on two open source Java software systems. We used various metrics related to lack of cohesion and testability. The obtained results support the idea that there is a statistically and practically significant relationship between lack of cohesion of classes and testability. The analysis performed here is correlational in nature. It only provides empirical evidence of the relationship between lack of cohesion and testability. Such statistical relationships do not imply causality. Only controlled experiments, which are difficult to perform in practice, could really demonstrate causality (Briand, 2000).

The study performed in this paper should be replicated using many other systems in order to draw more general conclusions. In fact, there are a number of limitations that may affect the results of the study or limit their interpretation and generalization. We investigated the relationship between lack of cohesion and testability using only two open source Java software systems, which is a relatively small number of systems. This may pose a threat to the scalability of the results. The study should be replicated on a large number of OOS to increase the generality of the results. It is also possible that facts such as the development process used to develop the analyzed systems and the development style of a given development team might affect the results or produce different results for specific applications. Moreover, knowing that software testability is affected by many different factors, it would be interesting to extend the used suite of metrics to better reflect the testing effort. Our experiments involved only three code-based (lack of) cohesion metrics. It would be interesting to extend this study by using other (structural and semantic) cohesion metrics, and comparing cohesion metrics to traditional object-oriented metrics (such as coupling, complexity, inheritance, etc.) in terms of predicting testability. Our study involved only software systems written in Java. While there is no reason to suspect that the results would be different with systems written in other object-oriented languages (such as C++), it would be interesting to study systems written in other languages. We hope, however, this study will help to a better understanding of what contributes to testability, and particularly the relationship between (lack of) cohesion and testability.

Acknowledgements

This project was financially supported by NSERC (National Sciences and Engineering Research Council of Canada).

References

Aggarwal, K.K., Yogesh, S., Arvinder, K., Ruchika, M.: Empirical study of object-oriented metrics. Journal of Object Technology 5(8) (2006)

Aman, H., Yamasaki, K., Yamada, H., Noda, M.T.: A proposal of class cohesion metrics using sizes of cohesive parts. In: Welzer, T., et al. (eds.) Knowledge-Based Sof. Engineering. IOS Press, Amsterdam (2002)

Badri, L., Badri, M.: A proposal of a new class cohesion criterion: An empirical Study. Journal of Object Technology 3(4) (2004); Special issue: TOOLS USA

Badri, L., Badri, M., Gueye, A.: Revisiting class cohesion, An empirical investigation on several systems. Journal of Object Technology 7(6) (2008)

Badri, L., Badri, M., Toure, F.: Exploring empirically the relationship between lack of cohesion in object-oriented systems and coupling and size. In: ICSOFT, Greece (July 2010)

Baudry, B., Le Traon, Y., Sunyé, G.: Testability analysis of a UML class diagram. In: Proceeding of the 9th International Software Metrics Symposium (METRICS 2003). IEEE Computer Society Press, Los Alamitos (2003)

Baudry, B., Le Traon, Y., Sunyé, G.: Improving the Testability of UML Class Diagrams. In: Proceedings of IWoTA (International Workshop on Testability Analysis), France (November 2004)

Bertolino, A., Strigini, L.: On the Use of Testability Measures for Dependability Assessment. IEEE Transactions on Software Engineering 22(2) (February 1996)

Bieman, J.M., Kang, B.K.: Cohesion and reuse in an object-oriented system. In: Proc. of the Symposium on Software Reusability (1995)

Binder, R.V.: Design for Testability in Object-Oriented Systems. Com. of the ACM 37 (1994)

Briand, L.C., Daly, J., Porter, V., Wuest, J.: A unified framework for cohesion measurement in object-oriented systems. Empirical Software Engineering 3(1) (1998)

Booch, G.: Object-Oriented Analysis and Design With Applications, 2nd edn. Benjamin/Cummings, Amsterdam (1994)

Briand, L.C., Daly, J., Porter, V., Wuest, J.: Exploring the relationships between design measures and software quality in object-oriented systems. Journal of Systems and Software (51) (2000)

Bruntink, M., Deursen, A.V.: Predicting Class Testability using Object-Oriented Metrics. In: Fourth Int. Workshop on Source Code Analysis and Manipulation (SCAM). IEEE Computer Society, Los Alamitos (2004)

Bruntink, M., Van Deursen, A.: An empirical study into class testability. JSS 79(9) (2006)

Chae, H.S., Kwon, Y.R., Bae, D.H.: A cohesion measure for object-oriented classes. Software Practice and Experience (30) (2000)

Chae, H.S., Kwon, Y.R., Bae, D.H.: Improving cohesion metrics for classes by considering dependent instance variables. IEEE TSE 30(11) (2004)

Chen, Z., Zhou, Y., Xu, B., Zhao, J., Yang, H.: A novel approach to measuring class cohesion based on dependence analysis. In: Proc. 18th International Conferrence on Software Maintenance (2002)

Chidamber, S.R., Kemerer, C.F.: Towards a Metrics Suite for Object-Oriented Design. Object-Oriented Programming Systems, Languages and Applications (OOPSLA), Special Issue of SIGPLAN Notices 26(10) (1991)

Chidamber, S.R., Kemerer, C.F.: A Metrics suite for OO Design. IEEE TSE 20(6) (1994)

Chidamber, S.R., Darcy, D.P., Kemerer, C.F.: Managerial use of metrics for object-oriented software: An exploratory analysis. IEEE TSE 24(8) (1998)

Chowdhary, V.: Practicing Testability in the Real World. In: International Conference on Software Testing, Verification and Validation. IEEE Computer Society Press, Los Alamitos (2009)

Counsell, S., Swift, S.: The interpretation and utility of three cohesion metrics for object-oriented design. ACM TSEM 15(2) (2006)

De Lucia, A., Oliveto, R., Vorraro, L.: Using structural and semantic metrics to improve class cohesion. In: International Conference on Software Maintenance (2008)

Etzkorn, L.H., Gholston, S.E., Fortune, J.L., Stein, C.E., Utley, D.: A comparison of cohesion metrics for object-oriented systems. Information and Software Technology 46 (2004)

Fenton, N., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach. PWS Publishing Company (1997)

Freedman, R.: Testability of software components. IEEE Transactions on Software Engineering 17(6), 553–564 (1991)

Gao, J., Tsao, J., Wu, Y.: Testing and Quality Assurance for Component-Based Software. Artech House Publishers, Boston (2003)

Gao, J., Shih, M.C.: A Component Testability Model for Verification and Measurement. In: Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC 2005). IEEE Computer Society, Los Alamitos (2005)

Henderson-Sellers, B.: Object-Oriented Metrics Measures of Complexity. Prentice-Hall, Englewood Cliffs (1996)

Hitz, M., Montazeri, B.: Measuring coupling and cohesion in object-oriented systems. In: Proc. of the Int. Symp. on Applied Corporate Computing (1995)

IEEE, IEEE Standard Glossary of Software Engineering Terminology. IEEE CSP, NY (1990)

ISO, International Standard ISO/IEC 9126. information technology: Software product evaluation: Quality characteristics and guidelines for their use (1991)

Jungmayr, S.: Testability Measurement and Software Dependencies. In: Proceedings of the 12th International Workshop on Software Measurement (October 2002)

Kabaili, H., Keller, R.K., Lustman, F., Saint-Denis, G.: Class Cohesion Revisited: An Empirical Study on Industrial Systems. In: Workshop on Quantitative Approaches OO Software Engineering (2000)

Kabaili, H., Keller, R.K., Lustman, F.: Cohesion as Changeability Indicator in Object-Oriented Systems. In: Proceedings of the Fifth European Conference on Software Maintenance and Reengineering (CSMR 2001), Estoril Coast (Lisbon), Portugal (2001)

Karoui, K., Dssouli, R.: Specification transformations and design for testability. In: Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM 1996), London (1996)

Khoshgoftaar, T.M., Szabo, R.M.: Detecting Program Modules with Low Testability. In: 11th ICSM, France (1995)

Khoshgoftaar, T.M., Allen, E.B., Xu, Z.: Predicting Testability of Program Modules Using a Neural Network. In: 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology (2000)

Larman, G.: Applying UML and Design Patterns, An introduction to object-oriented analysis and design and the unified process. Prentice Hall, Englewood Cliffs (2003)

Li, W., Henry, S.: Object-oriented metrics that predict maintainability. JSS 23 (1993)

Marcus, A., Poshyvanyk, D.: The conceptual cohesion of classes. In: Proc. 21th IEEE International Conference on Software Maintenance (September 2005)

Marcus, A., Poshyvanyk, D., Ferenc, R.: Using the Conceptual Cohesion of Classes for Fault Prediction in Object-Oriented Systems. IEEE TSE 34(2) (2008)

McGregor, J., Srinivas, S.: A measure of testing effort. In: Proceeding of the Conference on Object-Oriented Technologies, pp. 129–142. USENIX Association (June 1996)

Meyers, T.M., Binkley, D.: Slice-Based cohesion metrics and software intervention. IEEE WCRE (2004)

Nguyen, T.B., Delaunay, M., Robach, C.: Testability Analysis Applied to Embedded Data-Flow Software. In: Proceedings of the 3rd International Conference on Quality Software, QSIC 2003 (2003)

Petrenko, A., Dssouli, R., Koenig, H.: On Evaluation of Testability of Protocol Structures. In: Proceedings of the International Workshop on Protocol Est Systems (IFIP), Pau, France (1993)

Pressman, R.S.: Software Engineering, A practitioner's approach. McGraw Hill, New York (2005)

Sheppard, J.W., Kaufman, M.: Formal Specification of Testability Metrics in IEEE P1522. IEEE AUTOTESTCON, Pennsylvania (August 2001)

Sommervile, I.: Software Engineering (2004)

Stein, C., Cox, G., Etzkorn, L.: Exploring the relationship between cohesion and complexity. Journal of Computer Science 1(2) (2005)

Le Traon, Y., Robach, C.: Testability analysis of co-designed systems. In: Proceedings of the 4th Asian Test Symposium, ATS. IEEE Computer Society, Washington (November 1995)

Le Traon, Y., Robach, C.: Testability Measurements for Data Flow Design. In: Proceedings of the Fourth International Software Metrics Symposium, New Mexico (November 1997)

Le Traon, Y., Ouabdessalam, F., Robach, C.: Analyzing testability on data flow designs. In: Proceedings of ISSRE 2000, San Jose, CA, USA (October 2000)

Voas, J.M.: PIE: A dynamic failure-based technique. IEEE TSE 18(8) (August 1992)

Voas, J., Miller, K.W.: Semantic metrics for software testability. JSS 20 (1993)

Voas, J.M., Miller, K.W.: Software Testability: The New Verification. IEEE Software 12(3) (1995)

Yeh, P.L., Lin, J.C.: Software Testability Measurement Derived From Data Flow Analysis. In: Proceedings of 2nd Euromicro Conference on Software Maintenance and Reengineering (1998)

Woo, G., Chae, H.S., Cui, J.F., Ji, J.H.: Revising cohesion measures by considering the impact of write interactions between class members. Information and Software Technology 51 (2009)

Yourdon, E., Constantine, L.: Structured Design. Prentice Hall, Englewood Cliffs (1979)

Zhao, L.: A New Approach for Software Testability Analysis. In: 28th ICSE (May 2006)

Zhou, Y., Xu, B., Zhao, J., Yang, H.: ICBMC: An improved cohesion measure for classes. In: ICSM (2002)

Zhou, Y., Wen, L., Wang, J., Chen, Y., Lu, H., Xu, B.: DRC: dependence-relationships-based cohesion measure for classes. In: Proc. 10th APSEC (2003)

The Study of Imperfection in Rough Set on the Field of Engineering and Education

Tian-Wei Sheu[1], Jung-Chin Liang[2], Mei-Li You[3], and Kun-Li Wen[4]

[1,2] Graduate Institute of Educational Measurement and Statistics
National Taichung University, Taichung, Taiwan
[2] Department of Technological Product Design, Ling Tung University
Taichung, Taiwan
[3] Department of General Education, Chienkuo Technology University
Changhua, Taiwan
[4] Department of Electrical Engineering (Grey System Rough Center)
Chienkuo Technology University, Changhua, Taiwan
klw@ctu.edu.tw

Abstract. Based on the characteristic of rough set, rough set theory overlaps with many other theories, especially with fuzzy set theory, evidence theory and Boolean reasoning methods. And the rough set methodology has found many real-life applications, such as medical data analysis, finance, banking, engineering, voice recognition, image processing and others. Till now, there is rare research associating to this issue in the imperfection of rough set. Hence, the main purpose of this paper is to study the imperfection of rough set in the field of engineering and education. First of all, we preview the mathematics model of rough set, and a given two examples to enhance our approach, which one is the weighting of influence factor in muzzle noise suppressor, and the other is the weighting of evaluation factor in English learning. Third, we also apply Matlab to develop a complete human-machine interface type of toolbox in order to support the complex calculation and verification the huge data. Finally, some further suggestions are indicated for the research in the future.

Keywords: Imperfection, Rough set, Muzzle noise suppressor, English learning, Matlab toolbox.

1 Introduction

Because the main function of rough set theory is classification, according to the analysis characteristic of rough set theory, If we want to use rough set theory to find the weighting of the influence factor in system, we have to make sure two thing, one is those being analyzed data must be under the discrete condition, and the other is that the attribute factor and decision factor are under indiscernibility or discernibility condition, and these two points mentioned above are the imperfection in rough set theory[1]. According to past researches, The data in Taguchi method in Industry Engineering and questionnaire analysis method in Education field already in discrete condition[2,3]. Hence, we focus on the imperfection of indiscernibility and discernibility in our paper.

Dr. Pawlak presented the rough set in 1982, the basic topic of rough set includes: set theory; conditional probability; membership function; attributes analysis and uncertainty description of knowledge. And the main purpose of rough set is used the difference of lower approximations and upper approximations, it not only can find out the subjective result of clustered set, but also can use to find the weighting for factor in the system. Hence, in our research, we present rough set model to find the weighting in system, because after review the past research about this field, although the have many studies in this field[4], did not find any paper to discuss the limitation of rough set. Hence, we focus on this point, to study the limitation of rough set, hope to provide the new approach for the weighting analysis system.

In this paper, first, in section 2, we introduce the basic mathematical foundation of rough set model be our mathematics model. In section 3, we give two examples to verify our point, which are the study on noise in gun and the evaluation of English learning in education[3,5]. In section 4, the Matlab GUI toolbox is developed to verify our approach[6], Also in section 5, we make some advantages and suggestions for the further research in our study.

2 The Preview of Rough Set

In this section, we only simply introduce the basic concept of rough set[1].

1. Information system: $IS=(U,A)$ is called information system, where $U=\{x_1,x_2,x_3,......,x_n\}$ is the universe finite set of object, and $A=\{a_1,a_2,...,a_m\}$ is the set of attribute.
2. Information function: If exist a mapping $f_a:U\rightarrow V_a$, then V_a is the set of value of a, call the domain of attribute a .
3. Discrete: The mathematics model of equal interval width is shown in equation (1).

$$t=\frac{V_{\max.}-V_{\min.}}{k} \tag{1}$$

where: $V_{\max.}$: Maximum value in the data, $V_{\min.}$: Minimum value in the data.means the range of attribute value is $[V_{\max.},V_{\min}]$.

According to the result, we can get the interval corresponding to attribute value are

$$\{[d_0,d_1],[d_1,d_2],\cdots,[d_{k-1},d_k]\} \tag{2}$$

where: $d_0=V_{\min}$, $d_k=V_{\max}$, $d_{i-1}<d_i$, $i=1,2,3,\cdots,k$, k is the grade of discrete.

4. Lower approximations and upper approximations

If $A\subseteq U$, then the lower approximations is defined as

$$\underline{R}(A)=\{\mathrm{x}\in U\mid[x]_R\subseteq A\}=\bigcup\{[x]_R\in\frac{U}{R}\mid[x]_R\subseteq A\},\ [x]_R=\{y\mid xRy\} \tag{3}$$

and the upper approximations is defined as

$$\overline{R}(A)=\{\mathrm{x}\in U\mid[x]_R\cap A\neq\phi\}=\bigcup\{[x]_R\in\frac{U}{R}\mid[x]_R\cap A\neq\phi\},\ [x]_R=\{y\mid xRy\} \tag{4}$$

In other words, the lower approximation of a set is the set of all elements that surely belongs to *U*, whereas the upper approximation of *U* is the set of all elements that possibly belong to *U*.

5. Indiscernibility: An indiscernibility relation is defined as for any x_i and x_j , if x_i is identical to x_j , then x_i and x_j have all the same properties
6. Positive, negative and boundary: Base on the mentioned above, the positive, negative and boundary are defined as

$$pos_R(X) = \underline{R}(X)\,,\ neg_R(X) = U - \overline{R}(X)\,, bn_R(A) = \underline{R}(A) - \overline{R}(A) \qquad (5)$$

7. The dependents of attributes: The dependents of attributes is defined as

$$\gamma_c(D) = \frac{|posc(D)|}{U} \qquad (6)$$

means under $a \in C$, the ratio in the whole set.

8. The significant value of attributes: In decision system, $S = (U, C \bigcup D, V, f)$, under $a \in C$, the significant value of attributes is defined as

$$\sigma_{(C,D)}(a) = \frac{\gamma_c(D) - \gamma_{c-\{a\}}(D)}{\gamma_c(D)} \qquad (7)$$

means significant value of attributes can be imaged as the weighting in system for each factor.

3 Real Case Analysis

3.1 Attribute Factor-Muzzle Noise Suppressor

While the muzzle suppressor is an optional accessory for automatic weapons, a suppressor that performs well improves the weapon's effectiveness and helps to maintain the physical and psychological safety of the operator. Many overseas R&D units have therefore developed suppressors for all types of firearms to meet the different mission requirements. While this type of research is relatively lacking in Taiwan, the demand for this type of devices can be expected to increase as more emphasis is placed on lighter automatic weapons, commonality and operator safety. There is still a great deal of room for development in sound suppression in particular[5].

Based on the above, this study focused on the five potential control factors that influence the suppression device, including; the suppressor's front cover(with and without); the diameter of the side vents on the suppressor's external cylinder(1mm, 2mm and without); the position of side vents in the extension tube(15mm, 30mm and without); the external cylinder's internal diameter(30mm, 40mm and 50mm) and the external cylinder length(100mm, 125mm and 150mm). The 5.56mm carbine was used as the subject to analyze the influence of each factor for ranking and clustering.

According to these five control factors and their individual levels, our research used an $L_{18}(2^1 \times 3^7)$ orthogonal array. For the inner orthogonal array allocation of experimental factors, we assigned the selected control factors R_1, R_2 , R_3 , R_4 and R_5 to column 1, 2, 3, 4 and 5 of the $L_{18}(2^1 \times 3^7)$ orthogonal array, respectively.

Since our experiment focused on the noise of the rifle muzzle, the environmental and rifle conditions might affect the results. So we assigned both of them outer factors for the $L_{18}(2^1 \times 3^7)$ orthogonal array. They both had two levels. Condition of rifle for test firing Two-level factor: level 1 assigned to use the durable rifle(a rifle used for a long time but with good function); level 2 assigned to new rifle use.

The overall allocation of various experimental combinations in our study is shown in Table 1. And the measurement of muzzle flash involved using a camera with the B shutter (manual exposure) to take progressive pictures of the flash and maximum flash area at a set distance (3*m*). During the experiment, a randomly selected L_{18} (2^1x3^7) orthogonal table was used to select experimental combinations for live fire. And The photos of muzzle noise acquired through the experiment were processed using image processing software to count the number of pixels within the effective flame area. This was then used as the measurements for the quality characteristics as shown in Table 2.

Based on the characteristic of rough set, the data must be discrete, and we take four grades and take the values of dB in minimum the better, and are shown in Table 3.

Table 1. The orthogonal Table of muzzle flash suppressor

No	R_1	R_2	R_3	R_4	R_5
x_1	With	1mm	15mm	30mm	100mm
x_2	with	1mm	30mm	40mm	125mm
x_3	with	1mm	without	50mm	150mm
x_4	with	2mm	15mm	40mm	125mm
x_5	with	2mm	30mm	50mm	150mm
x_6	with	2mm	without	30mm	100 mm
x_7	with	without	15mm	30mm	150mm
x_8	with	without	30mm	40mm	100mm
x_9	with	without	without	50mm	125mm
x_{10}	without	1mm	15mm	50mm	125mm
x_{11}	without	1mm	30mm	30mm	150mm
x_{12}	without	1mm	without	40mm	100mm
x_{13}	without	2mm	15mm	50mm	100mm
x_{14}	without	2mm	30mm	30mm	125mm
x_{15}	without	2mm	without	40mm	150mm
x_{16}	without	without	15mm	40mm	150mm
x_{17}	without	without	30mm	50mm	100mm
x_{18}	without	without	without	30mm	125mm
	F	Durable rifle	New rifle	Durable rifle	New rifle
	Pixal(dB)	Y_1	Y_2	Y_3	Y_4

Control factor:
R_1: The suppressor's front cover.
R_2: The diameter of the side vents on the suppressor's external cylinder.
R_3: The position of side vents in the extension tube.
R_4: The external cylinder's internal diameter.
R_5: The external cylinder length.

Noise factor
F: The kinds of rifle.
Y_1 ~Y_4: The pixel of noise (dB).

Table 2. The data of muzzle noise and its average value

No	Y_1(dB)	Y_2(dB)	Y_3(dB)	Y_4(dB)	Average
x_1	127.08	118.45	117.34	115.76	119.66
x_2	116.85	116.65	115.96	115.85	116.33
x_3	115.42	128.39	114.32	114.45	118.14
x_4	117.04	117.07	115.92	116.46	116.62
x_5	117.32	118.88	115.84	116.05	117.02
x_6	137.32	136.76	128.53	125.02	131.91
x_7	114.36	114.34	114.38	114.39	114.37
x_8	114.84	114.12	116.39	116.20	115.39
x_9	113.80	113.68	114.18	113.47	113.78
x_{10}	125.22	123.05	114.32	114.35	119.23
x_{11}	115.43	115.65	115.99	115.83	115.72
x_{12}	115.40	115.36	114.56	114.69	115.00
x_{13}	117.27	116.76	125.00	128.29	121.83
x_{14}	126.33	124.53	126.84	122.44	125.04
x_{15}	127.43	130.12	124.02	139.18	130.19
x_{16}	113.82	113.68	123.46	113.86	116.20
x_{17}	113.84	113.92	113.65	113.73	113.78
x_{18}	114.54	114.22	114.65	114.75	114.54

Table 3. The data pre-processing

No	R_1	R_2	R_3	R_4	R_5	Average	Discrete
x_1	2	2	2	1	1	119.66	3
x_2	2	2	3	2	2	116.33	4
x_3	2	2	1	3	3	118.14	4
x_4	2	3	2	2	2	116.62	4
x_5	2	3	3	3	3	117.02	4
x_6	2	3	1	1	1	131.91	1
x_7	2	1	2	1	3	114.37	4
x_8	2	1	3	2	1	115.39	4
x_9	2	1	1	3	2	113.78	4
x_{10}	1	2	2	3	2	119.23	3
x_{11}	1	2	3	1	3	115.72	4
x_{12}	1	2	1	2	1	115.00	4
x_{13}	1	3	2	3	1	121.83	3
x_{14}	1	3	3	1	2	125.04	2
x_{15}	1	3	1	2	3	130.19	1
x_{16}	1	1	2	2	3	116.20	4
x_{17}	1	1	3	3	1	113.78	4
x_{18}	1	1	1	1	2	114.54	4

The calculation steps are shown below[1].

1. The dependents and significant, for condition attributes $\frac{U}{C}=\frac{U}{\{R_1,R_2,R_3,R_4,R_5\}}$ ={{ x_1 },{ x_2 },{ x_3 },{ x_4 },{ x_5 },{ x_6 },{ x_7 },{ x_8 },{ x_9 },{ x_{10} },{ x_{11} },{ x_{12} },{ x_{13} },{ x_{14} },{ x_{15} },{ x_{16} },{ x_{17} },{ x_{18} }}. For decision attributes: $\frac{U}{D}$ ={{ x_6 , x_{15} },{ x_{14} },{ x_1 , x_{10} , x_{13} }{ x_2 , x_3 , x_4 , x_5 , x_7 , x_8 , x_9 , x_{11} , x_{12} , x_{16} , x_{17} , x_{18} }={ X_1 , X_2 , X_3 , X_4 }. Hence, $pos_C(D)=$ { x_1 , x_2 , x_3 , x_4 , x_5 , x_6 , x_7 , x_8 , x_9 , x_{10} , x_{11} , x_{12} , x_{13} , x_{14} , x_{15} , x_{16} , x_{17} , x_{18} }.substitute into equation (7), we have $\gamma_c(D)=\frac{|pos_C(D)|}{|U|}=\frac{18}{18}=1$

2. Omit the attribute of R_1, the condition attributes

$$\frac{U}{C}=\frac{U}{\{R_2,R_3,R_4,R_5\}}=\{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},\{x_{10}\},\{x_{11}\},\{x_{12}\},\{x_{13}\},\{x_{14}\},\{x_{15}\},\{x_{16}\},\{x_{17}\},\{x_{18}\}\}$$

$\frac{U}{D}=\{\{x_6, x_{15}\},\{x_{14}\},\{x_1, x_{10}, x_{13}\}\{x_2, x_3, x_4, x_5, x_7, x_8, x_9, x_{11}, x_{12}, x_{16}, x_{17}, x_{18}\}=\{X_1, X_2, X_3, X_4\}$. Hence,

$pos_C(D)=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{17}, x_{18}\}$, substitute into the formula of dependents, we have $\gamma_{c-\{R_1\}}(D)=\frac{|pos_C(D)|}{|U|}$ $=\frac{18}{18}=1$, then the significant of R_1 is $\sigma_{(C,D)}(R_1)==\frac{1-1}{1}=0$.

3. We also omit R_2, R_3, R_4 and R_5, then the value of significant all are equal to 0.

3.2 Decision Factor- A Study of English Vocabulary Learning Strategies in Taiwan College Students

Recently, English has played a key role in the dissemination ideas and thoughts throughout the world. In order to foster a high quality international participation developing English capability has been one of main national objectives. In the "Education Policy 2005~2008" of Ministry of Education, all levels of schools are required to develop a test "General English Proficiency Test(GEPT)" to achieve a level of proficiency in English. In the highly competitive modern society of Taiwan, English language ability is a powerful asset in seeking employment and securing promotion[7]. Accordingly a plethora of organizations offer proficiency tests the most common being TOEIC, IELTS, TOEFL, GET (Global English Test), Cambridge Main Suite and FLPT (Foreign Language Proficiency Test). More and more colleges require students to pass GEPT or a similar English proficiency test before graduation. These requirements place a heavy psychological pressure on non-English major students. In addition, many colleges even require their professors to select textbooks in English and instruct in English. Therefore, listening and reading ability becomes more and more important. Vocabulary learning strategy cultivation can develop efficiency in reading ability.

In modern society, people are often judged not only by their appearance, but also by their ability to speak whether they are students or teachers, politicians or salesmen. According to the findings of domestic researchers, college students in Taiwan have serious problems with vocabulary learning. Educators and language test organizations expect the senior high school graduates to have a vocabulary size of 5,000 to 7,000 words in order to comprehend college English textbooks. But according to the past research, 171 college students and conducted a "Vocabulary Level Test" which was based on Laufer [8] and the Nation's vocabulary level test. The result of the study was 48.5% of the students could understand up to 1,000 words, 17% students with 2000 words. Only 2.3% of the students had an understanding of 3,000 words or more. This means that up to 32.5% of the students had a vocabulary of less than 1,000 words [9].

Most of time, students cannot comprehend the meaning of new words. Researchers believed that insufficient vocabulary will apparently lead to poor reading comprehension and subsequent academic achievement. Some foreign scholars also suggest that if language learners can have a larger vocabulary, they can understand more content and express themselves more clearly. They are also able to read broader and deeper on subjects. Vocabulary ability is a very important indicator of reading comprehension. In addition, good use of vocabulary learning strategies is an effective way to achieving reading comprehension. Hence, in this example, the score is based on Likert five points scale, and a total of 15 students were selected from each English proficiency groups, and calculate the weighting 50 factors.

The test results by 15 students are listed in Table 6 to Table 7, and the calculation steps are shown below.

Table 4. Likert five points scale

Score	1	2	3	4	5
Items	All the time	Most of the time	About half the time	Some of the time	Not at all

Table 5. The contents of questionnaire

No	Content	No	Contents
Q_1	I will analyze parts (verbs, nouns) of speech to judge the meaning.	Q_{13}	I will say a new word aloud when studying.
Q_2	I will guess the meaning from textual context.	Q_{14}	I will underline the new word to enhance the impression.
Q_3	I will consult a bilingual dictionary.	Q_{15}	I will remember root, prefix and suffix of the word.
Q_4	I will consult a monolingual dictionary.	Q_{16}	I will learn the whole phrase including the new word.
Q_5	I will consult a Chinese-English and English-Chinese dictionary.	Q_{17}	I will take notes in class.
Q_6	I will ask teachers for a sentence including the new word.	Q_{18}	I will use the vocabulary section in the textbook to learn new words.
Q_7	I will discover a new word's meaning through group work activities.	Q_{19}	I will listen to tapes of word lists.
Q_8	I will study and practice a new word's meaning with classmates.	Q_{20}	I will keep a vocabulary notebook to write down new words.
Q_9	I will interact with native speakers with new vocabularies.	Q_{21}	I will learn new words from watching English films.
Q_{10}	I will connect the word to its synonyms and antonyms.	Q_{22}	I will learn new words from reading English newspapers.
Q_{11}	I will use new words in sentences.	Q_{23}	I will learn new words from reading English articles.
Q_{12}	I will group words together within a storyline.	Q_{24}	I will learn new words from listening to English radio programs.

Table 6. The test results from question 1 to question 13

	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{13}
x_1	4	5	4	2	3	3	3	1	1	5	5	1	5
x_2	4	5	4	3	3	3	2	3	1	5	4	2	5
x_3	5	5	4	3	2	3	3	2	2	5	4	2	4
x_4	5	5	4	3	3	3	2	1	2	4	3	1	5
x_5	5	5	4	3	3	4	1	1	1	5	4	2	5
x_6	5	5	4	3	3	2	1	2	1	5	4	2	5
x_7	4	5	3	3	4	4	1	1	2	5	4	2	5
x_8	5	5	3	3	3	3	1	1	1	5	3	1	5

Table 6 (*continued*)

x_9	4	5	3	2	3	3	1	1	1	5	4	1	5
x_{10}	5	5	2	3	3	3	1	1	1	5	4	1	5
x_{11}	4	5	3	2	2	3	1	2	1	5	3	2	5
x_{12}	5	5	2	2	4	3	1	1	2	5	4	1	5
x_{13}	4	5	2	3	3	3	1	1	2	5	5	2	5
x_{14}	5	5	3	2	3	2	1	2	1	5	4	1	5
x_{15}	5	5	4	1	3	3	1	1	2	5	3	2	5

Table 7. The test results from question 14 to question 24, and the output

	Q_{14}	Q_{15}	Q_{16}	Q_{17}	Q_{18}	Q_{19}	Q_{20}	Q_{21}	Q_{22}	Q_{23}	Q_{24}	Q_{14}	Output
x_1	4	5	4	2	3	3	3	1	1	5	5	1	5
x_2	4	5	4	3	3	3	2	3	1	5	4	2	5
x_3	5	5	4	3	2	3	3	2	2	5	4	2	4
x_4	5	5	4	3	3	3	2	1	2	4	3	1	5
x_5	5	5	4	3	3	4	1	1	1	5	4	2	5
x_6	5	5	4	3	3	2	1	2	1	5	4	2	5
x_7	4	5	3	3	4	4	1	1	2	5	4	2	5
x_8	5	5	3	3	3	3	1	1	1	5	3	1	5
x_9	4	5	3	2	3	3	1	1	1	5	4	1	5
x_{10}	5	5	2	3	3	3	1	1	1	5	4	1	5
x_{11}	4	5	3	2	2	3	1	2	1	5	3	2	5
x_{12}	5	5	2	2	4	3	1	1	2	5	4	1	5
x_{13}	4	5	2	3	3	3	1	1	2	5	5	2	5
x_{14}	5	5	3	2	3	2	1	2	1	5	4	1	5
x_{15}	5	5	4	1	3	3	1	1	2	5	3	2	5

1. The dependents and significant, for condition attributes $\frac{U}{C}=\frac{U}{\{Q_1,Q_2,Q_3,\cdots,Q_{24}\}}=\{\{x_1\},\{x_2\},\{x_3\},\{x_4\},\{x_5\},\{x_6\},\{x_7\},\{x_8\},\{x_9\},$ $\{x_{10}\},\{x_{11}\},\{x_{12}\},\{x_{13}\},\{x_{14}\},\{x_{15}\}\}$.For decision attributes: $\frac{U}{D}=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}=\{X_1\}$. Hence, $pos_C(D)=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}$, substitute into equation (7), then, we have $\gamma_c(D)=\frac{|pos_C(D)|}{|U|}=\frac{15}{15}=1$.

2. Omit the attribute of Q_1

   The condition attributes= $\frac{U}{C}=\frac{U}{\{Q_2,Q_3,Q_4,\cdots,Q_{24}\}}=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}$. For decision attributes: $\frac{U}{D}=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}=\{X_1\}$. Hence, $pos_C(D)=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}, x_{13}, x_{14}, x_{15}\}$, substitute into the formula of dependents, then, we have $\gamma_{c-\{Q_1\}}(D)=\frac{|pos_C(D)|}{|U|}=\frac{15}{15}=1$, then the significant of Q_1 is $\sigma_{(C,D)}(Q_1)=\frac{1-1}{1}=0$.

3. We also omit $Q_2, Q_3, \cdots, Q_{24}$ respectively, then the value of significant all are equal to 0.

4 The Design of Toolbox

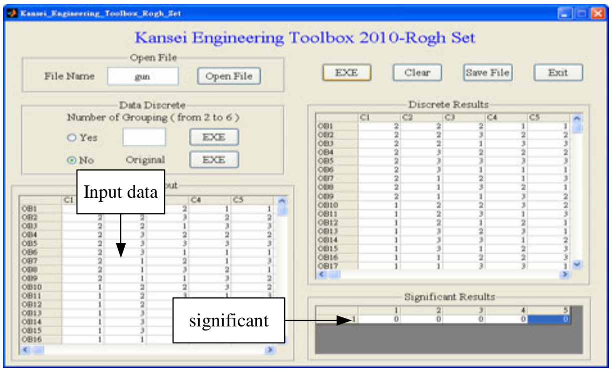
In our paper, we develop a toolbox based on Matlab, it not only can reduce the huge and complex calculation, but also can let the input data easily to calculate and make the results on the analysis system more convincing and more practical[12].

4.1 The Characteristics of Toolbox

The development of the toolbox has the following characteristics.

1. The Toolbox changes the current processing of rough set formula and methods into the GUI method.
2. The input interface adopts GUI in Matlab and reconciles with Microsoft so that it can input set numbers randomly. Therefore, the user can easily use this data and offer great help in dealing with examiners and subjects.
3. The requirements of toolbox are Window XP; Screen resolutions at least 1024×768; Matlab 2007/a version and data type is Excel.

4.2 The Calculation of Toolbox



	C1	C2	C3	C4	C5
OB1	[illegible]	[illegible]	2	1	1
OB2	2	2	3	2	2
OB3	2	2	1	3	3
OB4	2	3	2	2	2
OB5	2	3	3	3	3
OB6	2	3	1	1	1
OB7	2	1	2	1	3
OB8	2	1	3	2	1
OB9	2	1	1	3	2
OB10	1	2	2	3	2
OB11	1	2	3	1	3
OB12	1	2	[illegible]	[illegible]	[illegible]
OB13	1	3	[illegible]	[illegible]	[illegible]
OB14	1	3	[illegible]	[illegible]	[illegible]
OB15	1	3	[illegible]	[illegible]	[illegible]
OB16	1	1	[illegible]	[illegible]	[illegible]

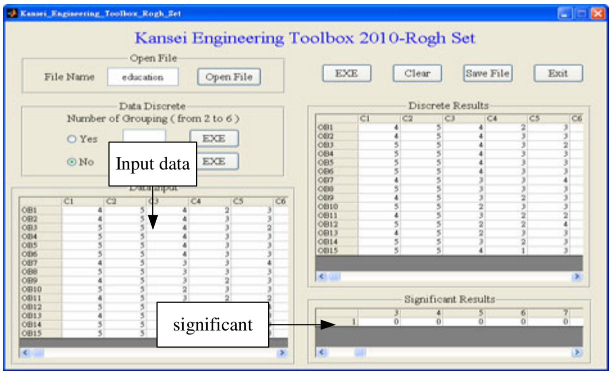
Discrete Results

	C1	C2	C3	C4	C5
OB1	2	2	2	1	1
OB2	2	2	3	2	2
OB3	2	2	1	3	3
OB4	2	3	2	2	2
OB5	2	3	3	3	3
OB6	2	3	1	1	1
OB7	2	1	2	1	3
OB8	2	1	3	2	1
OB9	2	1	1	3	2
OB10	1	2	2	3	2
OB11	1	2	3	1	3
OB12	1	2	1	2	1
OB13	1	3	2	3	1
OB14	1	3	3	1	2
OB15	1	3	1	2	3
OB16	1	1	2	2	3
OB17	1	1	3	3	1

Significant Results

	1	2	3	4	5
1	0	0	0	0	0

Fig . 1. The calculation in toolbox for muzzle noise



	C1	C2	C3	C4	C5	C6
OB1	4	5	4	2	3	
OB2	4	5	4	3	3	
OB3	5	5	4	3	2	
OB4	5	5	4	3	3	
OB5	5	5	4	3	3	
OB6	5	5	4	3	3	
OB7	4	5	3	3	4	
OB8	5	5	3	3	3	
OB9	4	5	3	2	3	
OB10	5	5	2	3	3	
OB11	4	5	3	2	2	
OB12	5	5	[illegible]	[illegible]	[illegible]	
OB13	4	5	[illegible]	[illegible]	[illegible]	
OB14	5	5	[illegible]	[illegible]	[illegible]	
OB15	5	5	[illegible]	[illegible]	[illegible]	

Discrete Results

	C1	C2	C3	C4	C5	C6
OB1	4	5	4	2	3	
OB2	4	5	4	3	3	
OB3	5	5	4	3	2	
OB4	5	5	4	3	3	
OB5	5	5	4	3	3	
OB6	5	5	4	3	3	
OB7	4	5	3	3	4	
OB8	5	5	3	3	3	
OB9	4	5	3	2	3	
OB10	5	5	2	3	3	
OB11	4	5	3	2	2	
OB12	5	5	2	2	4	
OB13	4	5	2	3	3	
OB14	5	5	3	2	3	
OB15	5	5	4	1	3	

Significant Results

	3	4	5	6	7
1	0	0	0	0	0

Fig. 2. The calculation in toolbox for English learning

5 Conclusions

As mentioned in the abstract, the main function of rough set is classification, and its limitation is that, when the discrete method and attribute factors of discernibility and of decision factors both are indiscernibility, it is impossible for us to get their weightings. Hence, in this paper, we use Taguchi method and questionnaires analysis method to verify that when the attribute factors is in discernibility condition and decision factor is indiscernbility condition, we can't classify them and get a weighting of each factor.

To sum up, the rough set theory is the new classification method in soft computing, some practical problems in relation with application of rough sets had been applied in many domains. In our paper, we have only provided two examples to verify the imperfection of indiscernibility or discernibility in rough set theory, but this is our main contribution in the paper. Besides, we also use Matab to develop a toolbox to help the complex calculation in huge data, and this is the other contribution in our paper.

Acknowledgment

The authors want to heartily thank Taiwan Kansei Information Association(TKIA), for provide the toolbox to verify our results.

References

1. Wen, K.L., Nagai, M.T., Chang, T.C., Wen, H.C.: An introduction to rough set and application. Wunam Published, Taipei (2008)
2. Chang, C.S., Liao, R.C., Wen, K.L., Wang, W.P.: A grey-based Taguchi method to optimal design of muzzle flash restraint device. Int. J. of Advanced Manufacturing Technology 24, 860–864 (2004)
3. Liang, H.Y.: A study of English vocabulary learning strategies in Taiwan college students via GM (0, N), Final Project Report of Chienkuo Technology University (2010)
4. Pawlak, Z.: Rough sets approach to multi-attribute decision analysis. European Journal of Operational Research 72, 443–459 (1994)
5. Wen, K.L., Lu, K.Y., Chang, C.S.: Apply GM (h, N) to the optimize design of muzzle noise suppressor. In: Proceedings of 2009 National Symposium on System Science and Engineering, Taiwan (2009)
6. Wen, K.L., You, M.L.: The development of rough set toolbox via Matlab. In: Current Development in Theory and Applications of Computer Science, Engineering and Technology, vol. 1, pp. 1–15 (2010)
7. Taiwan Ministry of Education, Education main Policy 2005-2008 (2008), `http://torfl.pccu.edu.tw/torfl8_2.htm`
8. Laufer, B.: How much lexis is necessary for reading comprehension? In: Be Joint, H., Arnaud, P. (eds.) Vocabulary and Applied Linguistics. MacMillan, London (1992)
9. Zhan, L.H.: What Reading Models EFL Teachers and Students Use in Freshman English Classes, Master thesis, Southern Taiwan University of Technology, Applied Foreign Languages (2009)

The Software Industry in the Coffee Triangle of Colombia

Albeiro Cuesta[1], Luis Joyanes[2], and Marcelo López[3]

[1] Nacional University – Alsus IT Group, Street 70 No 23B-41, Manizales - Colombia
[2] Pontificia University of Salamanca, Madrid Campus, P. Juan XXIII, 3, Madrid, Spain
[3] Caldas University, Street 65 No 26-10, Manizales - Colombia
alcuestame@unal.edu.co, luis.joyanes@upsam.net, mlopez@ucaldas.edu.co

Abstract. The so-called "Coffee Triangle" region is located in the Andean Region, in central Colombia, South America. This Andean Region is composed of the Departments of Caldas, Quindío and Risaralda. The Andean Region has been characterized by the production of coffee as a worldwide industry supported by high Quality and Research standards. These components have become the key bastions to compete in international markets. After the decline of the Coffee industry it is necessary to consider alternatives, supplemented by the success of the Software Industry at the global level. The strengthening of the Software Industry in the Coffee Triangle seeks to establish a productive alternative for regional growth in a visionary way, where knowledge, a fundamental input of the Software Industry, is emerging as one of the greatest assets present in this geographical area - Andean Region - of Colombia.

Keywords: Software Quality, Software products and services, development sectors, training.

1 Introduction

The software industry as well as the countries of India, Ireland, the United States and China among others has been successful in software industrialization. These countries have been characterized by promoting policies and institutions that aimed to strengthen the technological framework, regulate incentives and taxation, and promote the training of education and human talent, in the short, medium and long term. In addition, models have been incorporated to assure quality standards recognized worldwide and to create ways to attract foreign investment.

The research and the development of software engineering techniques and methods have been consolidating; the purpose is to advance in the problem solutions of software development. However, it is common that in the professional practice the recommendations of the software engineering are not included or otherwise are not applied carefully. If the software productive process in the Coffee Triangle were evaluated with maturity models such as Capability Maturity Model Integration, CMMI[1], we will find that the software development is usually in the beginning state. However by September 2009 there were six enterprises in Colombia certified in the model IT Mark[2] by the European Software Institute – ESI -, four of which were located in Bogotá. Alsus IT Group S.A. an enterprise from Manizales, becomes the first one with this important distinction in the Coffee Triangle[3]. By July 2010 31 enterprises were already evident in Colombia, two of them from the Coffee Triangle certified in IT Mark; six more enterprises in Manizales are in the process of certification.

The quality of a product, for organizations and enterprises, in general, depends directly on the processes, materials and the techniques used. Therefore, it is important that the software developing enterprises and groups have an approach for their software developing process, using the appropriate tools and techniques to ensure a better quality of the final product and greater customer satisfaction.

The Coffee Triangle region presents a countless number of natural resources with complex network interactions which determine the presence and behavior of human settlements. These human settlements in turn are composed of complex social networks, economic and symbolic relationships, which determine and modify substantially the processes and interrelationships with the natural platform.

The productive bids in the Coffee Triangle region are related to agricultural production, to tourism and to the Information and Communication Technologies ICT; in the same way the strategic software projects of the region are oriented towards the same[4].

There exists an emerging but still beginning software industry in the region. Also there are initiatives of generic software applications developing at the local and the national level. Various enterprises are even starting certification work using quality standards models. In addition, progress is being made in setting policies and regulations from the public sector to standardize certification.

2 The Software Industry in the Coffee Triangle

The information that follows was part of a Doctoral Thesis “A Model for the Software Industrialization in the Coffee Triangle of Colombia”. Fieldwork was carried out during 2009 primarily to locate and identify companies that make it clear that their corporate purpose is to develop Software in the Coffee Triangle region, as well as Informatics services and/or sales of software products. The identification of companies was supported with information gathered from the Chambers of Commerce located in the capitals of the three departments in the Coffee Triangle. This information allowed us to work with a total of 48 Software Developing Enterprises in the Coffee Triangle, 20 of them in Manizales, 16 in Pereira and 12 in Armenia. Subsequently, fieldwork was done in all of the 48 enterprises of the region.

2.1 Products and Services

The next table shows the results related to the “type of product or service” that the Coffee Triangle region (Manizales, Pereira and Armenia) Software enterprises offer. It also shows the partial results for each city and the corresponding tendencies.

Table 1. Type of Product or Service Offered by the Coffee Region Enterprises

	Outsourcing Services	*Packaged Software*	*Customized Software*	*Applications for Mobile Devices*	*Web Applications*	*Other*
Manizales	7	12	16	3	12	3
Pereira	6	16	12	6	8	4
Armenia	4	5	6	3	8	5
	17	**33**	**34**	**12**	**28**	**12**

Figure 1 shows the "type of product and service" the Coffee Triangle region enterprises offer, most have to do with "Customized Software", "Packaged Software" and "Web Applications." The percentages of those enterprises that offer these services are 25%, 24.26% and 20.59%, respectively. The products that are less offered in the Coffee Triangle region are "Mobile Applications" and "Outsourcing Services".

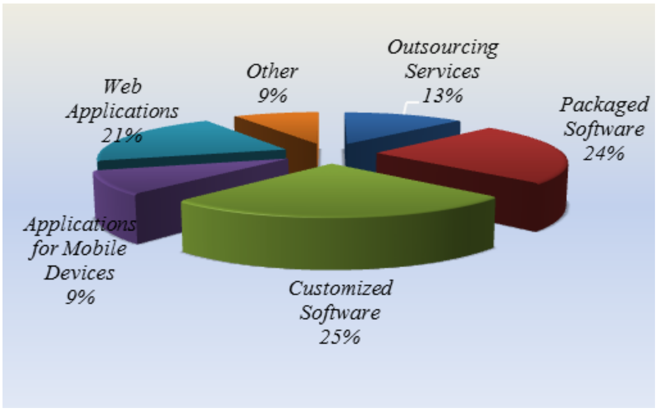


Fig. 1. Service or Product offered by the Coffee Region

2.2 Sectors for Which the Software Is Developed

Figure 2 shows the percentage of the economic sectors for which the enterprises develop software. It shows that the sectors with the greatest demand are those of services, industry and government, with percentages of 16.23%, 11.84%, and 11.40%, respectively.

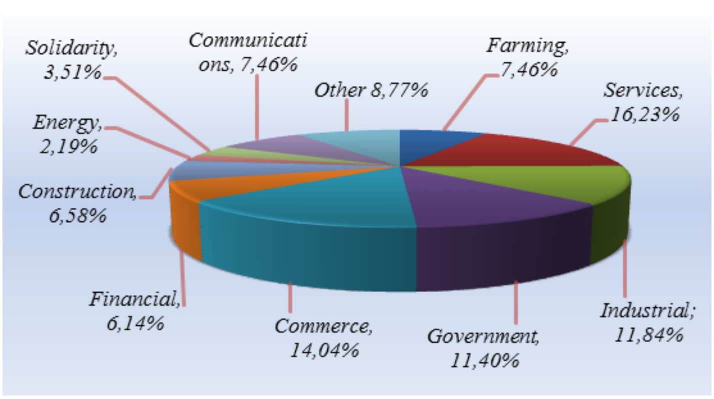


Fig. 2. Economic sectors for which the software is developed

2.3 Market Types

The software companies of the Coffee Triangle region sale more to the national market followed by local and international markets.

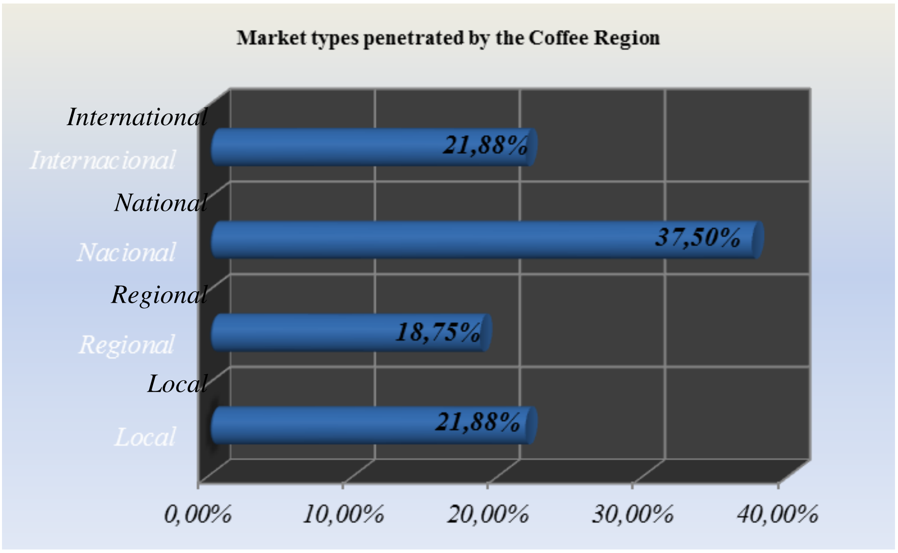


Fig. 3. Market types

2.4 Annual Sales Range

In relation to annual sales most companies, have a percentage of 35.29%, on sales of USD 26,317 - 52,632; 17.65% on sales of USD 52,633 - 105,263; 17.65% on sales of USD 105,264 - USD 131,579; 5.88% on sales greater than USD 263,158 as observed in Figure 4.

The number of enterprises in the Coffee Triangle region that record sales of less than USD 26,316 is almost equal in the three department capitals. The number of enterprises that made annual sales of USD 52,633 – 105,263 is located in Pereira. Also the majority of enterprises with sales of USD 105,264 -263,158 are located in Manizales. In addition, 80% of enterprises with annual sales greater than USD 263,158 are in Manizales, followed by Armenia with 20%, and the enterprises in Pereira did not record annual sales greater than USD 263,158.

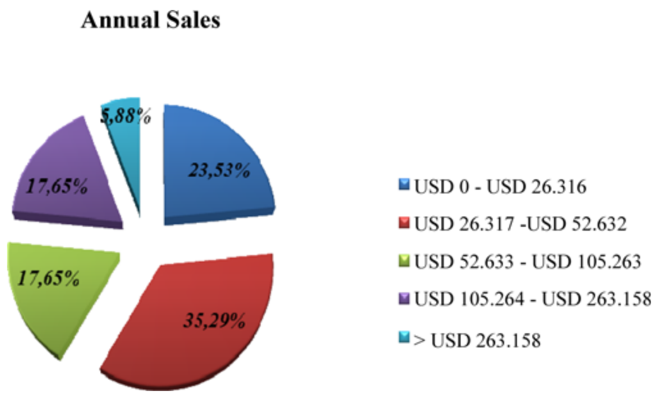


Fig. 4. Sales Rang

3 Academics in the Coffee Triangle

3.1 Academic Training

One of the main advantages for developing the Software Industry in the Coffee Triangle Region is the presence of important universities in the three capitals of the Coffee Region. These universities offer intensive academic offer programs related to Informatics Engineering and related professions.

Table 2 shows the summary of the academic training that the Coffee region is offering right now related to the Software Industry.

Table 2. Academic Training in the Coffee Region

Level	*Caldas*	*Risaralda*	*Quindío*	*Total*
Technology	*4*	*1*	*0*	*5*
Technological Specialization	*1*	*0*	*0*	*1*
Undergraduate	*8*	*8*	*8*	*24*
Bachelor Degree	*2*	*2*	*1*	*5*
Specialization	*8*	*3*	*3*	*14*
Mastery	*1*	*1*	*0*	*2*

3.2 Graduates

In 2007, a research project was done in Manizales aimed at supplementing the training of personnel in the Informatics area for Technical, Technological and Undergraduate levels.

Table 3. Technical training

Number of Graduates	*Degree name*	*Years*
Number of Graduates	*Systems Technician*	*1987 - 2007*

Table 4. Technological Training

Number of Graduates	*Degree name*	*Years*
1.145	*Information Systems Technology*	*1997 - 2007*

Table 5. Undergraduate Training

Number of Graduates	*Degree name*	*Years*
313	*Systems and Telecommunications Engineering*	*2004 – 2007*
10	*Systems and Computing Engineering*	*2006 – 2007*
731	*Systems Engineering*	*1989 – 2007*
203	*Information Systems Administration*	*1995 - 2007*

Because of individual characteristics it is easier in the software industry to have rapid expansion, in national as well as in international markets. Since it is a knowledge industry it does not require conventional raw materials and in the same way it does not require greater physical infrastructure. In addition the initial investment is relatively low by comparison to other sectors; furthermore, if the Software development is feasible the enterprise may become a global business resulting in large worldwide markets. Finally, since it is a knowledge industry, there is no comparative difference at the world level.

References

[1] Chrissis, M.B., Konrad, M., Shrum, S.: CMMI Guidelines for Process Integration and Product Improvement, Boston, p. 663. Addison-Wesley, Reading (2009) ISBN: 9788478290963

[2] ESI Zamudio: ESI, consultado en (Febrero 8, 2010), `http://www.esi.es/Products&Services/ITCompetitiveness/itmark/itmarkCompList.php.2010`

[3] ESI. List of IT Mark Certified Companies. Zamudio: European Software Institute, consultado en (September 2009), `http://www.esi.es/Products&Services/ITCompetitiveness/itmark/itmarkCompList.php.2009`

[4] Cuesta, A., López, M., Joyanes, L.: Experiences of digital territory in Manizales and Caldas. In: The IEEE Latin-American Conference on Communications - WorkShop, Medellin, Vol: PP: n/a, Editado por. IEEE, Los Alamitos (2009); ISBN: 978-1-4244-4388-8

Towards Maintainability Prediction for Relational Database-Driven Software Applications: Evidence from Software Practitioners

Mehwish Riaz, Emilia Mendes, and Ewan Tempero

Department of Computer Science,
The University of Auckland, New Zealand
mria007@aucklanduni.ac.nz,
{emilia,e.tempero}@cs.auckland.ac.nz

Abstract. The accurate maintainability prediction of relational database-driven software applications can improve the project management for these applications, thus benefitting software organisations. This paper presents an up-to-date account of the state of practice in maintainability prediction for relational database-driven software applications. Twelve semi-structured interviews were conducted with software professionals. The results provide both an account of the current state of practice in that area and a list of potential maintainability predictors for relational database-driven software applications.

Keywords: Maintainability, Relational Database-Driven Software, Prediction.

1 Introduction

Software maintainability is defined as "the ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment" [5]. Intrinsically associated with it is the process of software maintenance [15], which has long been known to consume the majority of the cost of a Software Development Life Cycle (SDLC) [1]. Software maintainability can significantly impact software costs [15]; therefore it is imperative to understand software maintainability in order to control and improve it.

In order to improve software maintainability, it is important to understand the factors that impact upon it and how they can be measured and used to predict it. A maintainability prediction model can enable organizations to make informed decisions about managing their maintenance resources and in adopting a defensive design [14].

Database-driven applications consist of a database, a database management system, and a set of applications that interact with the database through this management system [9]. When software requirements change, these applications undergo maintenance resulting in storing an increased number of data sources and relationships [11], leading to increased complexity of the database schema and coupling between the database and application [11]. Hence, both application-specific and database-specific features impact the maintainability of these applications.

Database-driven applications have gained substantial importance in the modern software development [2] and anecdotal evidence suggests that most commonly used databases are relational [4]. Due to the importance of relational database-driven applications within the scope of modern software development, it is imperative to be able to measure and forecast their maintainability. There is, however, little evidence in the literature relating to the maintainability of these types of applications [2]. The aim of this research is, therefore, to use evidence from industrial practice in order to help improve the field of maintainability prediction for relational database-driven applications. The evidence has been gathered from 12 interviews with software practitioners (six each from Pakistani and New Zealand organizations). This paper also extends the work in Riaz et al. [16] by combining their results with data obtained from six additional interviews conducted with software professionals in New Zealand.

The remaining of this paper is organised as follows. Section 2 presents the background, Section 3 details the research methodology, Section 4 gives results, and Section 5 discusses the results followed by conclusions and future work in Section 6.

2 Background

The research described herein was informed by the results of a Systematic Review (SR) on the topic 'software maintainability prediction and metrics' [15]. The results of the SR revealed very little evidence on software maintainability prediction. For detailed results of the SR, refer to Riaz et al. [15].The 15 studies selected by the SR [15] were further analyzed to assess if the datasets used in these studies completely or partially comprised relational database-driven software applications. Only three of these studies [3, 6, 7] had used relational database-driven software applications where only one [6] presented a maintainability prediction model but did not provide its prediction accuracy; the other two presented measures [3] and factors [7] impacting software maintainability. None of these studies proposed predictors or factors related specifically to a back-end database or to the interaction between a back-end database and the front-end application.

In addition to the SR above mentioned, a complementary literature review was carried out focusing on the topic of relational databases, details can be found in Riaz et al [16], which suggested that there was no evidence on maintainability metrics or prediction for relational databases or relational database schema.

The lack of existing literature focusing specifically on maintainability prediction for relational database-driven software applications prompted us to carry out an investigation with software practitioners in order to gather data on relational database-driven applications maintainability predictors and metrics used in practice.

3 Research Methodology

The research methodology used herein was mainly qualitative in nature and is detailed in the subsequent sub-sections.

3.1 Data Source

The data source used herein comprised interviews with 13 software professionals from 10 different software companies in Pakistan and New Zealand. A total of 12 interviews were conducted, six in each country. In one of the interviews conducted in Pakistan, two interviewees participated; however, the information provided by them overlapped; therefore, their responses are considered to come from a single interview. The interviewees were either Project Managers (PM), project Team Leads (TL) or Quality Assurance (QA) professionals. These roles, , according to our observations, are involved in all phases of SDLC and were therefore better able to reflect upon their experiences and lessons learnt.

The two countries, Pakistan and New Zealand, were chosen for the following reasons: i) the first author is originally from Pakistan; ii) most software companies in Pakistan are well established, follow commercial quality standards and practices, and deal with outsourced projects from the United States, United Kingdom, Denmark, etc.; iii) all the three authors are affiliated to the University of Auckland, New Zealand where the research is being carried out; and iv) we believe that the context of this research was not culture-sensitive as the focus was on software applications and not specific aspects related to people or their way of carrying out their work.

The interviewees were all at least at managerial positions in well established companies with an average experience of 10.3 years in software development and management. A summary of interviewees' experience for each role is given in table 1. All the companies, except one, where the interviewees worked developed only relational database-driven software applications.

Table 1. Interviewees' summary per role

Role	No. of Interviewees in Pakistan	No. of Interviewees in New Zealand	Total Years of Experience (Averaged per role)
PM	6	4	10.35
QA	1	0	7
TL	0	1	9
CEO & Founder	0	1	15

3.2 Semi-structured Interviews and Interview Procedure

Interviews are a commonly used qualitative data collection technique which can be used to collect opinions or impressions about a phenomenon of interest [8]. For this research, semi-structured interviews were conducted. This type of interviews were best suited for this research as these comprise both open and closed ended questions and aim eliciting expected information as well as unforeseen information [8].

Contacts were made with the companies and the participating employees both through email and telephone. The interviews were conducted in two steps. In the first step, six interviews were carried out with software professionals in Pakistan in July 2009. In the second step six interviews were conducted with software professionals in New Zealand between August and November 2009 which were informed by the responses obtained in the first step. All the interviews were face-to-face and recorded on two digital recorders (the second recorder was used as a backup). During the interviews, extensive notes were also taken.

Each of the interviews began with a short introduction to the research and its objectives. It was emphasized that the research focused at relational database-driven software applications. The questions asked during the interviews are given in the appendix. The format and order of questions was the same for each interview.

The format of the questions and their order was the same for each interview. The language used in the first step of interviews, conducted in Pakistan, was Urdu – the native language of the country – to enable the interviewees to fully express their thoughts and perceptions. In this step, the interviewer shared her perception of maintainability with the interviewees, after asking them about their own views on the topic. This was done so that both the interviewer and interviewee were on equal grounds to discuss further concepts. The interviewer also shared some predictors of maintainability from the SR [15] to give an example of the perception of predictors.

The second step of interviews was conducted in New Zealand in English. The same process was followed, except that some predictors specific to database-driven applications gathered from the first step of interviews were also discussed with the interviewees. In both phases, the duration of the interviews ranged from 20 to 60 minutes with an average duration of 38 minutes.

3.3 Data Analysis

Prior to analysing the data obtained from the interviews, all the conducted interviews had to be transcribed. The six interviews conducted in Pakistan had also to be translated from Urdu to English. All the interviews were translated and transcribed by the first author due to her familiarity with both languages. The translations and transcriptions were further verified by a volunteer who listened to parts of interviews at random and tallied them with the time stamped transcripts of the interviews.

The specific technique used for qualitative data analysis was ‘content analysis’, using the framework and components defined by Krippendorff [10]. The components that were used to proceed from text to results are as below:

- Unitizing – the systematic distinguishing of segments of text that are of interest [10]. This was done in accordance to the interview questions.
- Sampling – allows the analyst to economize on research efforts by limiting observations to a manageable subset [10]. It was carefully done by choosing appropriate companies and roles for a better population representation.
- Recording/Coding – transformation of unedited text or unstructured sounds into analyzable representations [10]. This was also carefully performed – recording done on reliable digital recorders and coding done with the help of NVivo [13] against the units of analysis.
- Reducing – serves the need for efficient representations [10]. This was done alongside coding.
- Inferring - bridges the gap between descriptive accounts of texts and what they mean [10]. This was done against the interview questions.
- Narrating - amounts to the researchers making their results comprehensible to others [10]. This has been done in following Section.

4 Results Informed by Industrial Practice

The results obtained by performing content analysis on the conducted interviews, are given below. Also it was ensured that the interviewees understood that the context of the questions was specific to relational database-driven software applications.

4.1 Perception of Maintainability

The most common perception of maintainability, reflected in 7 of the 12 interviews (58%), was to consider it as the ease or difficulty to implement changes in the software or to carry out maintenance tasks. Four of the interviewees perceived maintainability as the time taken to fix a software bug. The term ‘ease’ to them was a “*bit of a relevant term*”, and was understood in the context of time. Two of the interviewees also mentioned - “*availability or low down time*” in the context of database-driven applications.

4.2 Should Applications Be Maintainable?

All the interviewees agreed that software applications should be maintainable. The reasons provided included “*scalability*”; producing “*quality*” and “*mature*” products”; making an application “*customizable*”; reduced “*resource utilization*”, “*cost*”, “*effort*” and “*development time for adding new features*”; “*convenience to the developers and customers*”; and “*client satisfaction*”.

4.3 Practical Experience with Maintainability Prediction

The analysis revealed that software maintainability prediction is not a common concept in practice. This was reflected by the responses of all, except two, interviewees, who in answer to whether they have any experience with maintainability prediction said “*yes in order to improve*” and “y*es …due to the older technologies that we adopt (application) becomes obsolete and then we have to move forward*”. None of the interviewees had experience with predicting software maintainability whereby it is done by following any specific prediction model or formally done as part of the development or project planning processes. However, all the interviewees agreed that maintainability should be predicted and also stated the perceived benefits associated with its prediction, see Section 4.4.

4.4 Benefits of Predicting Maintainability

The benefits identified by the interviewees to predict software maintainability were: i) improved costs and effort; ii) to help organizations in “*decision making*” and “*resource planning*”; and iii) improved design. Note that the interviewees used the terms design and software architecture interchangeably.

Other benefits narrated by the interviewees included “*deep understanding of the application*”; ability to “*negotiate with client*” on cost, and allowing the team to spend “*more time on quality*” and reducing “*time spent on operations*”; and the ability to decide whether to bid on a project by considering “*not just the initial cost of project’s development but life time cost*”.

4.5 Is Maintainability Predicted in Practice?

The responses of the interviewees on whether maintainability was predicted in their respective companies were mixed. Eight interviewees (67%) gave a positive response and said that maintainability is predicted based on "*expert judgment*". One interviewee gave a negative response. Since his company developed and managed applications that interacted with large databases, his context was that of the features provided by the tools used to develop these applications and stated that "*the features are not available".* Two interviewees initially answered no; however they later added that they make a "*judgment that is very subjective*" and that it is not predicted using "*some measure but there are certain things that I do know*". None of the interviewees suggested any quantitative measures of maintainability.

In relation to at what project stage maintainability can be best predicted, seven interviewees (58%) supported the design stage. They said that they could make a better judgment based on design than by considering resources, deliverables, or the source code. The remaining interviewees gave no definite answer. The supporters of design chose it because it can be improved before the implementation as "*application is the last thing to be challenged*". In summary, this means that at least those we interviewed believed that maintainability of a relational database-driven application can be subjectively predicted based on an application's design or architecture.

4.6 Maintainability Measures

All the interviewees, except for one, believed software maintainability was nearly impossible to measure. They stated that "*we cannot say it on the basis of any measurement*". However, they were of the view that software maintainability can be judged by considering various factors such as those mentioned in Section 4.8. The only interviewee who believed maintainability could be measured supported the idea of "*measuring maintainability in terms of man hour resources*" or "*in terms of dollars*" as he believed that "*ultimately* (it) *comes down to it*".

4.7 Is There a Difference between Maintainability for Relational Database-Driven Applications and Applications that Do Not Have Any Back-End Database?

All the interviewees, except for one, corroborated that maintainability of relational database-driven applications is different from that of applications that do not have any back-end database by stating "*if we consider the database-driven applications, not only the factors related to database play a part but also those factors that are because of the technical aspects of the DBMS are introduced*". Another interviewee, a very strong supporter of database-driven applications, stated: "*I call those applications without a database more as tools, they are not applications. To me applications run on the database*". The interviewee who differed in opinion was of the view that "*if it is a database-driven application, then one of your modules is database*" and that implementing changes to a database should be considered the same as implementing changes to different parts of an application.

4.8 Factors That Should Be Considered When Predicting the Maintainability of Relational Database-Driven Software Applications

Several factors specific to predicting the maintainability of relational database-driven software applications that were identified by interviewees are given in Table 2.

Table 2. Factors for predicting maintainability of relational database-driven applications

Factors	#Interviewees Pakistan	#Interviewees New Zealand	Total
Application's architecture or design	5	5	10
Tools & technologies including development frameworks	5	5	10
Database design including the following:	5	4	9
1. proper normalization	5	2	7
2. database design complexity	1	3	4
3. correct definition of entities	3	1	4
4. de-normalization according to the needs	0	3	3
Follow processes, procedures and practices	4	4	8
Customizable or parameterized application	3	4	7
Documentation	3	4	7
Database performance	3	3	6
Application complexity	2	4	6
Maintenance effort	2	4	6
Experience and skill set of human resources	4	2	6
DB connectivity and interaction between DB & application	3	2	5
Understandability	1	4	5
Emphasis on properly conducting requirements phase	3	2	5
Choice of database engine	4	1	5
Amount of data	3	2	5
Use of stored procedures instead of inline queries	3	1	4
Code inspection & design and peer code reviews	2	2	4
Tiered approach for application development	2	2	4
Support for different database engines	1	3	4
Code quality	1	3	4
Coupling (both at design and code levels)	1	3	4
Code comments & their quality	1	3	4
Ability to foresee changes	3	1	4
Database availability	2	1	3
Number of application users	2	1	3
Project cost	1	2	3
Application size	1	2	3
Initial development effort	3	0	3
Following standards (coding and naming)	0	3	3
Implementation of basic checks at the database side	2	1	3

Note that Table 2 lists only the factors stated by three or more interviewees. There are 12 other factors that were reported by 2 interviewees and several others reported by only one. We argue that it is very likely that these factors would also be applicable to other software companies beyond the ones who participated in this research, given that most of the factors shown in Table 2 were selected by companies in both Pakistan and New Zealand, thus suggesting that they are applicable across countries.

The factors listed in Table 2 suggest that the most important factor for relational database-driven software application's maintainability is the application architecture, followed by the database design. This suggests that it is most important for organisations to design their applications correctly. In addition, other factors such as those related to code, design, software processes and people also seem to be relevant predictors of maintainability. Note that the factors listed in Table 2 may seem to be at different levels of abstraction and/or related; however we refrained from merging any factors in order to present exactly the factors as emphasized by the interviewees.

5 Discussion

This paper presents evidence from the state of practice on maintainability prediction for relational database-driven software applications. The results suggest that maintainability is understood in practice as a quality attribute but sometimes also in terms of time taken to correct issues in the software application. This can be attributed to the fact that quality itself is rather an intangible concept and so maintainability is dealt with as a relative concept understood in terms of time. Also, based on the evidence presented herein, we believe that it can be argued that the maintainability of relational database-driven software applications is perceived to be different from that of the applications that do not interact with a back-end database.

It also appears that a formal prediction model or approach to predicting the maintainability of relational database-driven applications is not used in practice. However, the interviewees did suggest that they relied upon expert opinion when making a judgment on the maintainability of such applications. Therefore the notion that maintainability is unlikely to be predicted in practice may be slightly misleading as 'expert opinion' also is a well-known prediction technique [12]. Nevertheless, the use of subjective means for prediction is known to present several problems [12]; therefore there is a need to formalize this process by proposing and validating maintainability prediction models that use predictors that are relevant in practice to enable their adoption by the practitioners.

An analysis of the interviews shows that the answers given by the interviewees of both countries are very similar; implying that maintainability prediction for relational database-driven software applications may not be sensitive to culture, and in addition, results may also be generalized outside the sample population used in this research.

As with most research, this research work also has some limitations. A first limitation of possible missing details is due to the reason that the interviews are based on the interviewees' recollection of experiences. This was addressed for most part by using appropriate prompts during the interviews to probe further on important details.

A second limitation can be that the interviewees during each step were told some results from previous steps of the research – SR and interviews conducted in the first step. While this had a disadvantage of introducing the interviewer's own perceptions, after much contemplation this was considered important especially in cases where it was absolutely necessary to enable the discussion with the interviewees to take place.

A third limitation can be due to inaccurate hearing of the recorded interviews. This was minimized by spending a generous amount of time on extensive pause-and-play to hear unclear words, and by having a volunteer validate a subset of the transcripts.

A fourth limitation relates to the external validity of the results. This is because the results come only from twelve interviews, which can be argued to be a small sample; however the nature of the study is qualitative and does not involve statistical analysis which requires large volumes of data. In addition, we also had to take into account the time available to carry out this work.

6 Conclusions

This paper presents the current state of practice on software maintainability prediction for relational database-driven software applications. The research involved conducting twelve interviews with software professionals.

The results suggest that maintainability for relational database-driven applications is understood in practice as a quality attribute; it is not measured or predicted using a formal metric or prediction technique; the practitioners believe that it should be predicted; the practitioners also believe that its prediction is different from that of the applications that do not have a database back-end; and it is predicted in the domain of relational database-driven applications based on expert judgment. The fact that expert judgment is made on maintainability because it is required for these applications and there is no formal prediction model for quantifying these judgments, suggests a strong need for a formal technique that can be used to quantify maintainability prediction. The predictors provided herein take formalizing the process of maintainability prediction for the mentioned type of applications one step forward. These factors can result in improved software maintainability, as the collected evidence suggests.

The future work involves further verification of these factors with the help of a survey, followed by the use of these factors in software maintainability prediction models built from real project data gathered from software companies.

References

1. Bhatt, P., et al.: Influencing Factors in Outsources Software Maintenance. SIGSOFT SEN 31(3), 1–6 (2006)
2. Fadlalla, A., et al.: Survey of DBMS Use and Satisfaction. Info. Sys. Mgmt. 14(2) (1973)
3. Ferneley, E.H.: Design Metrics as an Aid to Software Maintenance: An Empirical Study. J. Softw. Maint.: Res. Pract. 11, 55–72 (1999)
4. Gardiokiotis, S.K., et al.: A Structural Approach towards the Maintenance of Database Applications. In: IDEAS 2004, pp. 277–282 (2004)
5. IEEE Std. 610.12-1990: Standard Glossary of Software Engineering Terminology. IEEE Computer Society Press, Los Alamitos, CA (1993)
6. Genero, M., Olivas, J., Piattini, M., Romero, F.: Using Metrics to Predict OO Information Systems Maintainability. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 388–401. Springer, Heidelberg (2001)
7. Lim, J.S., et al.: An Empirical Investigation of the Impact of OO Paradigm on the Maintainability of Real-World Mission-Critical Software. J. Syst. Soft. 77, 131–138 (2005)
8. Seaman, C.: Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering 25(4), 557–572 (1999)
9. Kapfhammer, G.M., Soffa, M.L.: A Family of Test Adequacy Criteria for Database-Driven Applications. In: ESEC/FSE 2003, pp. 98–107 (2003)

10. Krippendorff, K.: Content Analysis – An Introduction to Its Methodology, 2nd edn. Sage Publications, Thousand Oaks (2004)
11. Maule, A., et al.: Impact Analysis of DB Schema Changes. In: ICSE 2008, pp. 451–460 (2008)
12. Mendes, E., Mosley, N.: Web Engineering. Springer, Heidelberg (2006) ISBN-10 3-540-28196-7, ISBN-13 978-3-540-28196-2
13. QSR International Pty Ltd.: Nvivo version 8 (2010), `http://www.qsrinternational.com/products_nvivo.aspx`
14. Oman, P., Hagemeister, J.: Construction and Testing of Polynomials Predicting Software Maintainability. J. Syst. Software 24, 251–266 (1994)
15. Riaz, M., Mendes, E., Tempero, E.: A Systematic Review of Software Maintainability Prediction and Metrics. In: ESEM 2009, pp. 367–377 (2009)
16. Riaz, M., Mendes, E., Tempero, E.: Maintainability Prediction for Database-Driven Software Applications – Preliminary Results from Interviews with Software Professionals. In: SEDE 2010 (2010)

Appendix: Interview Questions

1. What is your understanding of maintainability?
2. Do you believe that software applications should be maintainable? If yes, why?
3. What is your view and/or experience with software maintainability prediction?
4. What, in your opinion, may be the benefits of predicting software maintainability?
5. Is maintainability measured and/or predicted in your company, even if it is based on expert judgment?
   a. If yes:
      i. What are the factors that you take into account when predicting maintainability?
      ii. How do you measure maintainability?
   b. If no:
      i. Why not?
      ii. What would you consider in terms of artefacts e.g., design, resources, deliverables, source code etc. in order to predict maintainability?
      iii. What factors would you consider in order to predict maintainability?
6. When dealing with maintainability, do you differentiate between database-driven and non database-driven applications? Why? What are the additional factor(s) that need to be considered when developing a database-driven software application?

Software and Web Process Improvement – Predicting SPI Success for Small and Medium Companies

Muhammad Sulayman and Emilia Mendes

Department of Computer Science, Private Bag 92019,
The University of Auckland, New Zealand
msul028@aucklanduni.ac.nz,
emilia@cs.auckland.ac.nz

Abstract. This study revisits our previous work in SPI success factors for small and medium companies [26] in order to investigate separately SPI success factors for Web companies that only develop Web applications (Web development companies (12 companies and 41 respondents)) from SPI success factors for Web companies that develop Web as well as software applications (Web & software development companies (8 companies and 31 respondents)). We have replicated Dyba's theoretical model of SPI success factors [12] in [26], and later also in this paper. However, the study described herein differs from [12] and [26] in that Dyba used data from both software and Web companies, and Sulayman and Mendes used data from Web companies that developed Web applications and sometimes also software applications by employing quantitative assessment techniques. The comparison was also performed against the existing theoretical model of SPI success factors replicated in this study. The results indicate that SPI success factors contribute in different ratios for both types of companies.

Keywords: Success Factors, Web Companies, Software Process Improvement.

1 Introduction

Software processes facilitate software development companies by introducing mature and structured practices [1]. Nearly all software development companies face time and cost related constraints in their projects [2] [9]. To cater their needs, companies embark on software process improvement (SPI) (see [3] to have a view of different models and techniques) to improve processes and engineer quality software [4]. Companies invest in SPI to obtain benefits i.e. better quality software, lesser development time, enhanced productivity [4], improved overall company flexibility and client contentment [5] [6] [10]. Numerous researchers have also investigated SPI success factors [11] [12] [13] [14] and people issues as they adopt new processes and also support in approving existing ways of performing tasks [6].

It has been also been observed that small and medium software development companies formulate a large proportion of the overall quantity of software companies. A recent survey has reported that 99.2% of the overall software development companies are either small or medium [7] many of them are working in the domain of Web development [8].

This paper investigates the suitability of an existing theoretical model of SPI success factors, proposed for small and large software companies [12] to two different types of companies i.e., small and medium 'Web' and 'Web and software' development companies. 'Web development' companies are considered as companies that are engaged in pure Web and Web application development, whereas 'Web and Software development' companies are considered to develop Web applications and also other types of applications, which include telecommunications, desktop applications, system software etc. For the purpose of this paper, the theoretical model of SPI success factors proposed by Dyba [46] was applied to two different datasets separately. Finally, a comparison between the results from using these two datasets and those from Dyba's [12] and our previous replicated study [26] was carried out. The reasons to choose two datasets include: (1) a tangible difference between traditional software and Web engineering; (2) to investigate whether similar factors would predict SPI success for both types of companies as both are engaged totally or partially in managing 'Web' projects.

The main contribution of this paper is to investigate the extent to which SPI success factors proposed for software companies are also applicable to small and medium 'Web' and 'Web and Software' companies. This paper also revisits our previous work [26] by splitting its single dataset into two datasets – one containing data solely from Web companies, and another containing data from companies that develop both Web and software applications. In addition, this paper also reports on the differences between SPI success factors for small and medium 'Web' companies and SPI success for small and medium 'Web and Software' companies.

2 Literature Review Theoretical Model of SPI Success Factors

The authors of this paper conducted a systematic literature review (SLR) on SPI for Small and Medium Web Companies' [23] and one of the aim was to gather evidence on specific SPI success factors for small and medium Web companies. No evidence was found on this aspect; however, only one of the studies found in the primary search phase of the SLR had presented SPI success factors within a broader context – that of small and large companies (software and Web companies) [15]. This was the primary reason for its selection for replication in comparison to the other selected studies of the SLR. Other publications of the same author further clarified the model replicated by us [12] [16]. Population in [12] [15] and [16], had a very small sample which comprised Web development companies and no specialized success factors of the SPI success for small and medium Web companies were presetned. Dyba's theoretical model consists of seven independent variables/hypotheses – 'business orientation', 'involved leadership', 'employee participation', 'concern for measurement', 'exploitation of existing knowledge', and 'exploration of new knowledge' (hypotheses 1-6) [12]. These six variables independently and collectively measure 'SPI success' as a dependent variable (hypothesis 7) [12] [15]. Two moderating variables – 'organizational size' and 'environmental conditions' - are also a part of the model [12] [15].

3 Research Process

3.1 Population and Subjects of the Research

The population of this research comprises small and medium ‘Web development’ and ‘Web & Software development’ companies, whereas participants comprise professionals involved in Web development, and sometimes also in software development. The summary of the respondents’ and companies characteristics for both company types are given in appendix.

Primary reason of selecting data from Pakistani companies was that Pakistan has a growing and vibrant Web development industry because of the outsourcing market. Initially, CEOs and managers from a random sample of 28 small and medium Pakistani ‘Web development’ and ‘Web & Software development’ companies were contacted. The management of 20 companies (12/17 ‘Web’ development companies, 8/11 ‘Web & Software’ development companies) agreed to be surveyed, signifying a response rates of 70.58% and 72.72% respectively, which is much better than the specified minimum range of 40% [17]. The difference between two company types as well as the context of the research was elaborated to the contacted CEOs. It was requested from the CEOs of ‘Web & software’ development companies that the participants they choose from within their companies should be involved in both ‘Web’ and ‘Web & software’ projects. As a result, a total of 41/68 ‘Web’ and 31/47 Web & software development professionals participated in the investigation, showing response rates of 60.29% and 65.95% respectively.

3.2 Variables and Measurement Scales

The conducted survey based on [12] and [26] had three sections. First two sections gathered characteristics and demographics of companies and participants. The last section questioned about SPI success factors in the surveyed company. Detailed questionnaire is presented in the appendix.

The same variables (dependent, independent and moderating) and the similar scale types by Dyba [12] [16] and Sulayman and Mendes [26] were used to determine important conditions to find the differences in ‘Web development’ and ‘Web & Software development’ organizations in the context of SPI success.

3.3 Reliability of the Measurement Scales and Detailed Item Analysis

Cronbach alpha [18] was used to determine the reliability of scales in this research. (as also in [12] [16] [26]). Values of α for each independent variable for both company types are presented in the appendix. According to Cronbach [18], $\alpha > 0.7$ for a variable represents sufficient reliability. Hence, all the scales used in this research for both company types are reliable.

Similar to Dyba [12] [16] and Sulayman and Mendes [26], we performed a detailed item analysis based on Nanually’s method [19] to evaluate item assignments for both ‘Web development’ and ‘Web & software development’ companies separately. The variables which do not show significant correlation values are removed from the analysis [19] [58]. The suggested cut-off value is ≈ 0.3 [19] [58]. Correlated matrix for the variables of both company types, and proposed by Dyba [12] is given in appendix.

Items 14, 29, 30 and 36 did not meet the cut-off criterion for ‘Web & software development’ companies whereas, items 11,13,29,36 and 37 did not meet the cut-off criteria for ‘Web development’ companies (see appendix) and were hence removed from the further analysis steps.

3.4 Validity of the Measurement Scales and Statistical Data Analysis

The concept of validity of the measurement scales determines that how accurate measurement scales are. We have validated the construct and criteria validity of the data using the same procedures as used by Dyba [16] and previously used by us [26]. The content validity was also satisfied as this study is based on two previously conducted studies [16] [26].

Scree tests [60] and eigenvalue rules [20] were used to check construct validity in [12] and [26], hence we have also used same measures for that purpose. Eigenvalues must be represented by positive numbers, whereas excellent item loading ranges are great or equal to 0.6; and good item loading ranges are between 0.4 and less than 0.6 are considered to be fair [20]. The calculated valyes for eigenvalues and item loadings are given in appendix. Scree plot for all variables in both company types represented elbow shaped curves which is the passing requirement of tests (see appendix).

Criterion validity determines the independent relationship of the each dependent variable with the independent variables. All six independent variables were positively correlated with the dependent variables for both cases (proposed in [12] [16] and used in [26]), which is relative criteria for us. We have used the the same statistical techniques used in [12] and which we have already applied in [26]: Multiple regression analysis, Pearson correlation coefficient, T-tests and F statistic. Prior to performing the analysis, we ensured that values for K-S tests, skewness, and kurtosis were falling within the prescribed limits.

4 Empirical Findings

As in [12] and [26], we also present bivariate and partial order correlations for the independent variables calculated for both company types (Tables 1 and 2).

Four correlation scores out of a total fifteen scores showed higher values than 0.5, which is a significant outcome for small and medium ‘Web development’ companies. ‘Leadership involvement’ and ‘Exploration of Existing Knowledge’ showed the highest correlation with ‘Business Orientation’ for ‘Web development’ companies as given in Table 1.

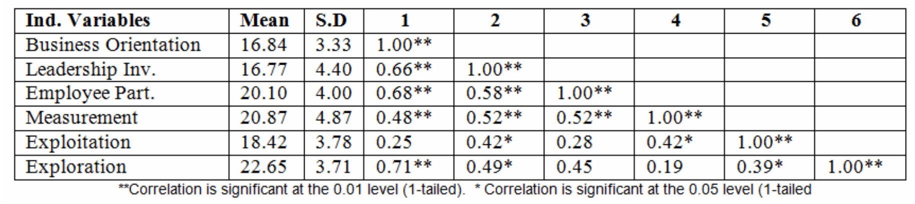
Table 1. Correlations among Independent Variables (Web)

Ind. Variables	Mean	S.D	1	2	3	4	5	6
Business Orientation	17.30	3.06	1.00**					
Leadership Inv.	18.09	4.16	0.74**	1.00**				
Employee Part.	17.20	2.87	0.34*	0.29	1.00**			
Measurement	22.41	3.90	0.36**	0.50*	0.88	1.00**		
Exploitation	18.85	2.73	0.23*	0.20**	0.12**	0.20*	1.00**	
Exploration	21.17	4.11	0.71**	0.65**	0.35**	0.25*	0.38	1.00**

**Correlation is significant at the 0.01 level (1-tailed). * Correlation is significant at the 0.05 level (1-tailed

For small and medium ‘Web & Software development’ companies six out of fifteen correlation scores were higher than 0.5. ‘Leadership Involvement’, ‘Employee Participation’ and ‘Exploration of Existing Knowledge’ showed the highest correlation with ‘Business Orientation’ as given in Table 2.

Table 2. Correlations among Independent Variables (Web & Software)



Ind. Variables	Mean	S.D	1	2	3	4	5	6
Business Orientation	16.84	3.33	1.00**					
Leadership Inv.	16.77	4.40	0.66**	1.00**				
Employee Part.	20.10	4.00	0.68**	0.58**	1.00**			
Measurement	20.87	4.87	0.48**	0.52**	0.52**	1.00**		
Exploitation	18.42	3.78	0.25	0.42*	0.28	0.42*	1.00**	
Exploration	22.65	3.71	0.71**	0.49*	0.45	0.19	0.39*	1.00**

**Correlation is significant at the 0.01 level (1-tailed). * Correlation is significant at the 0.05 level (1-tailed

4.1 Measurements for Success Factors

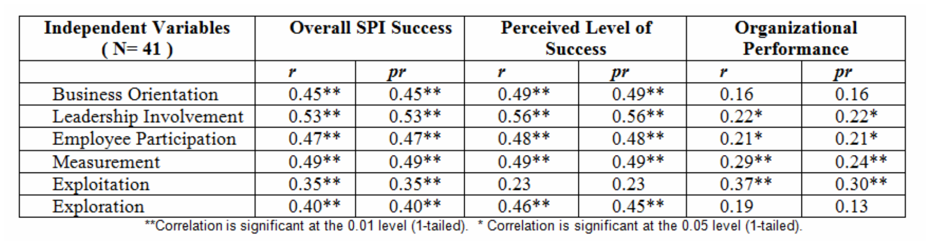
Success factors or hypotheses were tested similar to the way presented in [12] and [26], which using bivariate correlations. Bivariate correlations included zero orders and partial correlations. Regarding zero order correlations (r), the moderating variables were considered constant and for partial correlations (p_r) exact values of moderating variables were considered for the data sets of both company types.

Results (see Tables 3 & 4) showed that all correlations, which were positive, showed high values and the values were significant for both company types. Table 5 and 6 show the regression analysis results for overall SPI success of small and medium ‘Web development’ and ‘Web & Software development’ companies. As in [12] and [26], stepwise regression analysis was employed for both types of companies to find which factors influenced SPI success more significantly. In case of small and medium ‘Web development’ companies ‘Leadership Involvement’ was the most influential factor, contributing 26% towards SPI in the first model. ‘Employee Participation’ along with ‘Leadership Involvement’ contributed 35% towards SPI success. ‘Concern for Measurement’ with ‘Employee Participation’ and ‘Leadership Involvement’ contributed 42% towards SPI success. For ‘Web and software development’ companies ‘Business Orientation’ contributed 26% towards SPI success.

All the independent variables presented high and significant values of correlations for both company types. Along with that, the three variables ‘Leadership Involvement’, ‘Employee Participation’ and ‘Concern for Measurement’ were also selected by the stepwise regression in case of ‘Web development’ companies and ‘Business Orientation’ was selected for ‘Web & software development’ companies.

Hypotheses 7 was tested using the same technique used in [12] and [26] - Cohen’s coefficient $(f^2 = R^2/(1- R^2))$ [21], to check the variance of the independent variables in SPI success. In our case f^2 was 0.40 for ‘Web & software development’ companies and 0.87 (Model 3) in the case of ‘Web development’ companies, demonstrating a high effect. Very high values of test statistic (C_r) were found, demonstrating a normal distribution, when we used Konishi’s extension [22] to Fisher R-to-Z transformation, which supported our hypotheses 7 for the two data sets.

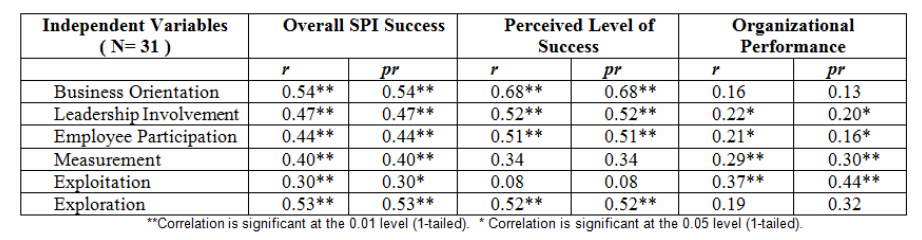
Table 3. Hypotheses 1-6/Success Factors Correlations (Web)



Independent Variables (N= 41)	Overall SPI Success		Perceived Level of Success		Organizational Performance	
	r	*pr*	*r*	*pr*	*r*	*pr*
Business Orientation	0.45**	0.45**	0.49**	0.49**	0.16	0.16
Leadership Involvement	0.53**	0.53**	0.56**	0.56**	0.22*	0.22*
Employee Participation	0.47**	0.47**	0.48**	0.48**	0.21*	0.21*
Measurement	0.49**	0.49**	0.49**	0.49**	0.29**	0.24**
Exploitation	0.35**	0.35**	0.23	0.23	0.37**	0.30**
Exploration	0.40**	0.40**	0.46**	0.45**	0.19	0.13

**Correlation is significant at the 0.01 level (1-tailed). * Correlation is significant at the 0.05 level (1-tailed).

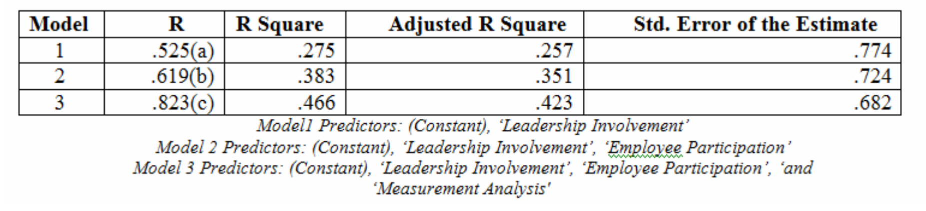
Table 4. Hypotheses 1-6/Success Factors Correlations (Web and Software)



Independent Variables (N= 31)	Overall SPI Success		Perceived Level of Success		Organizational Performance	
	r	*pr*	*r*	*pr*	*r*	*pr*
Business Orientation	0.54**	0.54**	0.68**	0.68**	0.16	0.13
Leadership Involvement	0.47**	0.47**	0.52**	0.52**	0.22*	0.20*
Employee Participation	0.44**	0.44**	0.51**	0.51**	0.21*	0.16*
Measurement	0.40**	0.40**	0.34	0.34	0.29**	0.30**
Exploitation	0.30**	0.30*	0.08	0.08	0.37**	0.44**
Exploration	0.53**	0.53**	0.52**	0.52**	0.19	0.32

**Correlation is significant at the 0.01 level (1-tailed). * Correlation is significant at the 0.05 level (1-tailed).

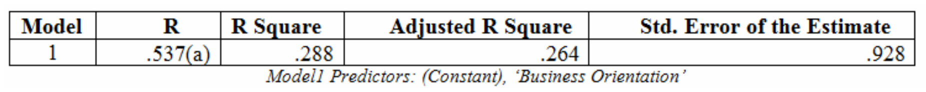
Table 5. Stepwise Regression Analysis Results (Web)



Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.525(a)	.275	.257	.774
2	.619(b)	.383	.351	.724
3	.823(c)	.466	.423	.682

Modell Predictors: (Constant), 'Leadership Involvement'
Model 2 Predictors: (Constant), 'Leadership Involvement', 'Employee Participation'
Model 3 Predictors: (Constant), 'Leadership Involvement', 'Employee Participation', 'and 'Measurement Analysis'

Table 6. Stepwise Regression Analysis Results (Web and Software)



Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.537(a)	.288	.264	.928

Modell Predictors: (Constant), 'Business Orientation'

4.2 Comparison of SPI Success Factors between Small and Medium Companies

Small and large software companies' SPI success factors were compared by Dyba [16]. We, in our previous study [26], differentiated between small and medium Web and software development companies. However, in this study we have differentiated between small and medium 'Web development' and between small and medium 'Web & software development' companies. Based on the expert opinion of academics and industry practitioners [15] [24] [25] [67] we have used the variable 'Organizational size' to differentiate between small and medium companies for both data sets, as follows: according to Steel [67] companies having less than twenty development staff are small; and companies having twenty to hindered development staff were said to be medium sized.

Similar to [16] and [26], we have employed 2-tailed –tests for the comparison of means for the data sets from two companies', where considering $\alpha = 0.05$. The results are shown in appendix and there were no significant differences between small and medium companies for any of the independent variables for the data sets from both company types.

5 Discussion

This study revisits our previous work [26] on success factors for SPI in small and medium Web companies by splitting the original data set into two data sets – one for 'Web development' and second for 'Web & software development' small and medium companies. The study was conducted to identify whether any differences exist between the SPI success factors for the two types of companies. The intention was to separate the SPI success factors for the two types of companies so that future research can be conducted as well as SPI initiatives can be taken by taking these differences into consideration.

A comparison of the overall findings of Dyba's study [12], our previously conducted study [26], and this study are given in table 13. The table clearly shows that the extent to which most factors ('Business Orientation', 'Leadership Involvement', 'Employee Participation', 'Measurement') explain SPI success for small and medium 'Web' companies is different to the extent the same factors explain it for small and medium 'Web and Software' companies. This difference suggests that SPI success is not explained for the two types of companies mentioned herein in the same manner. The differences also clearly indicate that the approach towards SPI for the mentioned types of companies should be different as the same factors do not play the same role in both types of companies' SPI success.

Table 7. Comparison of Results for the Support of Hypotheses between the Three Studies

Hypothesis	Dyba's results [12]	Sulayman & Mendes' results [26]	Our results (Web)	Our results (Web & software)
1 (Business Orientation)	Strong	Partial	Partial	Strong
2 (Leadership Involvement)	Partial	Strong	Strong	Partial
3 (Employee Participation)	Strong	Strong	Strong	Partial
4 (Measurement)	Strong	Strong	Strong	Partial
5 (Exploitation)	Strong	Partial	Partial	Partial
6 (Exploration)	Partial	Partial	Partial	Partial
7 (Overall SPI Success)	Strong	Strong	Strong	Strong

In addition to the differences highlighted in table 13, the results of the stepwise regression analysis (see tables 9 and 10) for both types of companies discussed in this paper and in Sulayman and Mendes [26] indicate very low scores. This means that SPI success for small and medium 'Web' and 'Web and Software' companies may not be explained only on the basis of the six independent variables presented in Dyba [12] and investigated by this study's authors. This means that there are more factors that may contribute to SPI success for the mentioned types of companies that have not been investigated as yet. This suggests a need for conducting an inductive study to investigate these factors in the given context, which is a line of our future work.

The level of support for the 7 hypotheses was in most cases different for the three studies and 4 discussed cases Dyba's results [12], our previous findings [69 and findings of this study for two data sets, as shown in table 7. One important finding of this study is that the strength of the support for all the hypotheses in case of small and medium 'Web development' companies is exactly the same as that of Sulayman and Mendes [26]. This further validates the findings given in [26]. The results for small and medium 'Web development' and 'Web & software development' companies converge for hypotheses 5-7 but differ for hypotheses 1-4. Further analysis of the results (see Table 13) reveal that the hypotheses 5-7 converge and 1-4 differ between 'Web & software development' companies and our findings in [26], whereas hypotheses 1-2, 6-7 converge while 3-5 diverge from Dyba's findings [12].

The study uses the same theoretical model, techniques, questionnaire and statistics proposed in [12] and used in [26]. Also, the data set used was the same as [26] but it was split for the two types of companies, as mentioned above. The data was split based on the responses gathered from the participants where the participants mentioned the types of applications developed by their respective companies. All the techniques for data analysis were then applied on the two data sets separately but simultaneously. The similarities and differences were noted between the two data sets and compared with Dyba's findings [12] [15] [16] and the findings of our previously conducted study [26].

The population of interest for this study is different from that of Dyba's [12] where the population of interest was small and large software companies. This study's population of interest is small and medium 'Web development' and 'Web & software development' companies. Note that the dataset used herein is the same as that of our previous study [26] in which a combined data analysis was performed due to the fact that all participating companies developed Web applications.

Various similarities as well as differences were observed between this study's findings, the findings from Dyba's studies [12] [15] [16] and those from our previous study [26]. In terms of similarities, similarly to previous studies [12] [26], all the hypotheses (1-7) were also supported herein. Dyba [12] removed sub-factor 29 and our previous study [26] removed sub-factors 29 and 36 (see appendix) from the corresponding scales due to their low correlation scores. However, for this study sub-factors 14, 29, 30 and 36 were removed for 'Web & software development' companies and sub-factors 11,13,29,36 and 37 were removed for 'Web development' companies, from their corresponding scales (Section 3.4). This was the first point of differentiation between the SPI success factors for the two types of companies used herein.

The main reason, in our opinion, which may have contributed towards the differences in our results may relate to the different nature of the software developed in the two types of companies. However, the difference from Dyba's findings [12] may be attributed to contextual factors given that he investigated small and large companies, where some developed Web applications.

There are certain threats to the validity of this research. We surveyed seventy two respondents from twenty 'Web development' and 'Web & Software Development' companies in Pakistan, which represented a relatively small sample size, all from a single country. Due to the difference of cultures and different environments, there can be different or other SPI success factors in other countries of the world. Further replications of the same model in other countries may clarify the applicability of the model further.

6 Conclusions

This research investigated the effects of SPI success factors on the specialized domain of 'Web development' and 'Web & software development' providing useful results to these types of companies which are willing to engage in SPI activities. The study has empirically obtained evidence in support of the theoretical model of SPI success factors proposed by Dyba [12] and already validated in [26]. This study performed a quantitative assessment of SPI success factors for 12 small and medium 'Web development' and 8 'Web & software development' companies by revisiting our previous work [26], which replicated the work presented by Dyba [12].

References

1. Glass, R.: Software creativity. Prentice-Hall, Inc., Upper Saddle River (1994)
2. Cugola, G., Ghezzi, C.: Software Processes: a Retrospective and a Path to the Future. Software Process: Improvement and Practice 4(3), 101–123 (1998)
3. Thomson, H., Mayhew, P.: Approaches to software process improvement. Software Process: Improvement and Practice 3(1), 3–17 (1997)
4. Zahran, S.: Software process improvement: practical guidelines for business success. Addison-Wesley, Reading (1998)
5. Florac, W., Park, R., Carleton, A., INST, C.-M. U. P. P. S. E.: Practical software measurement: Measuring for process management and improvement (1997)
6. Abrahamsson, P.: Rethinking the concept of commitment in software process improvement. Scandinavian Journal of Information Systems 13, 69–98 (2001)
7. Fayad, M., Laitinen, M., Ward, R.: Thinking objectively: software engineering in the small. Communications of the ACM 43(3), 118 (2000)
8. Allen, P., Ramachandran, M., Abushama, H.: PRISMS: an approach to software process improvement for small to medium enterprises (2003)
9. Salo, O.: Improving Software Development Practices in an Agile Fashion. Agile Newsletter 2 (2005)
10. van Solingen, R.: Measuring the ROI of software process improvement. IEEE Software 21(3), 32–38 (2004)
11. Niazi, M., Wilson, D., Zowghi, D.: Critical success factors for software process improvement implementation: an empirical study. Software Process: Improvement and Practice 11(2), 193–211 (2006)
12. Dyba, T.: An empirical investigation of the key factors for success in software process improvement. IEEE Transactions on Software Engineering 31(5), 410–424 (2005)
13. Alexandre, S., Renault, A., Habra, N., Cetic, G.: OWPL: A Gradual Approach for Software Process Improvement In SMEs (2006)
14. Wilson, D., Hall, T., Baddoo, N.: A framework for evaluation and prediction of software process improvement success. Journal of Systems and Software 59(2), 135–142 (2001)
15. Dybå, T.: Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context (2003)
16. Dyba, T.: An instrument for measuring the key factors of success in software process improvement. Empirical Software Engineering 5(4), 357–390 (2000)
17. Baruch, Y.: Response rate in academic studies—A comparative analysis. Human Relations 52(4), 421–438 (1999)

18. Cronbach, L.: Coefficient alpha and the internal structure of tests. Psychometrika 16(3), 297–334 (1951)
19. Nunnally, J., Bernstein, I.: Psychometric theory New York (2004)
20. Kaiser, H.: The application of electronic computers to factor analysis. Educational and Psychological Measurement 20(1), 141 (1960)
21. Cohen, J.: Statistical power analysis for the behavioral sciences. Lawrence Erlbaum, Mahwah (1988)
22. Konishi, S.: Normalizing transformations of some statistics in multivariate analysis. Biometrika 68(3), 647–652 (1981)
23. Sulayman, M., Mendes, E.: A Systematic Literature Review of Software Process Improvement in Small and Medium Web Companies. In: Advances in SE (ASEA 2009), pp. 1–8 (2009)
24. Cater-Steel, A., Toleman, M., Rout, T.: Process improvement for small firms: An evaluation of the RAPID assessment-based method. IST 48(5), 323–334 (2006)
25. El Emam, K., Birk, A.: Validating the ISO/IEC 15504 measures of software development process capability. Journal of Systems and Software 51(2), 119–149 (2000)
26. Sulayman, M., Mendes, E.: Quantitative assessments of key success factors in software process improvement for small and medium web companies (2010)

Appendix

Respondents and companies' characteristics, all the measurement scales and their sub factors, coefficient alpha (α), eigenvalues and item loading ranges, scree plots, variables' correlated matrix and comparison of small and medium 'Web' and 'Web & Software' Development Companies that are used in this study are mentioned in this appendix which can be browsed at:
`http://www.cs.auckland.ac.nz/~mria007/Sulayman/Repstudy2`

Test Prioritization at Different Modeling Levels

Fevzi Belli[1] and Nida Gökçe[1,2]

[1] University of Paderborn, Department of Computer Science, Electrical Engineering and Mathematics, 33098 Paderborn, Germany
[2] Mugla University, Faculty of Arts and Sciences, Department of Statistics, 48000, Mugla, Turkey
{belli,goekce}@adt.upb.de

Abstract. Validation of real-life software systems often leads to a large number of tests; which, due to time and cost constraints, cannot exhaustively be run. Therefore, it is essential to prioritize the test cases in accordance with their importance the tester perceives. This paper introduces a model-based approach for ranking tests according to their preference degrees, which are determined indirectly, through event classification. For construction of event groups, Gustafson - Kessel clustering algorithms are used. Prioritizing is performed at different granularity levels in order to justify the effectiveness of the clustering algorithm used. A case study demonstrates and validates the approach.

Keywords: Model- based test prioritization, fault detection, clustering.

1 Introduction and Related Works

Testing is the traditional validation method in the software industry. For a productive generation of tests, model based techniques focus on particular, relevant aspects of the requirements of the system under test (SUT) and its environment. Once the model is established, it 'guides' the test process to generate and select test cases (*test suites*). The test selection is ruled by an *adequacy criterion,* which provides a measure of how effective a given set of test cases is in terms of its potential to reveal faults [7]. Most of the existing adequacy criteria are *coverage-oriented;* they rate the portion of the system *specification* (behavioral aspects) or *implementation* (code) that is covered by the given test case set when it is applied to exercise the SUT [10].

In this paper, we focus on model based specification and coverage-oriented testing. In this context, event sequence graphs (ESG) [11] are favored. ESG approach views the system behavior and user actions as events.

The costs of testing often tend to run out the limits of the test budget. In these cases, the tester may request a complete test suite and attempt to run as many tests as affordable, testing the most important items first. This leads to the Test Case Prioritization Problem, a formal definition of which is represented in [4] as follows:

Given: A test suite *T*; The set *PT* of permutations of *T;* A function *f* from *PT* to the real numbers that represents the preference of the tester while testing.

Problem: Find $T' \in PT$ such that $(\forall T'')$ $(T'' \neq T')$ $[f(T') \geq f(T'')]$

Existing approaches to solve this problem usually suggest constructing a density covering array in which all pair-wise interactions are covered [1, 4]. In order to capture significant interactions among pairs of choices, the importance of pairs is defined as the "benefit" of the tests. Every pair covered by the test contributes to the total benefit of a test suite by its individual benefit. Therefore, the tests given by a test suite are to be ordered according to the importance of corresponding pairs. In order to generate a test set achieving arc coverage from a given ESG, a variant of Chinese Postman Problem [2] is solved. Thus, a minimum test set is constructed.

In our previous work, we suggest a prioritized testing approach that attempts to improve the testing capacity of the algorithm described in [3] and provides the ranking of the test cases to be run. For this aim, to each of the tests, a degree of preference is assigned. This degree is indirectly determined through qualification of events by several attributes [5, 6]. These attributes depend on the system and their values are justified by their significance to the user. We give some examples how to define such attributes and assign values to them, based on a graphical representation of corresponding events. We suggest [5, 6] representing those events as an unstructured multidimensional data set and dividing them into groups which correspond to their importance.

Clustering methods, such as adaptive competitive learning [5], fuzzy c-means [8, 9] and Gustafson Kessel [12] clustering algorithms are used to derive event groups. The advantage of the proposed approach is that no priori information is needed about the tests, e.g., as in regression testing [4]. Most, if not all, of the existing works require such prior information [1, 4].

Section 2 summarizes background information on the modeling with ESG, test case generation and Gustafson Kessel clustering method based on soft computing. Section 3 explains the approach to model-based test prioritization. Section 4 presents the test selection strategies at different levels based on prioritization, and a case study demonstrates and validates the approach. Finally, Section 5 discusses the results of the case study.

2 Background

2.1 Event Sequence Graphs

An event, an externally observable phenomenon, can be a user stimulus or a system response, punctuating different stages of the system activity. Event Sequence Graphs (ESG) graphically represents the system behavior interacting with the user's actions. In following, only a brief introduction into ESG concept is given that is necessary and sufficient to understand the test prioritization approach represented in this paper; more detail has been published before ([11]).

Mathematically speaking, an ESG is a directed, labeled graph and may be thought of as an ordered pair ESG=(V, E), where V is a finite set of nodes (vertices) uniquely labeled by some input symbols of the alphabet Σ, denoting events, and E: V $\rightarrow$V, a precedence relation, possibly empty, on α. The elements of E represent directed arcs (edges) between the nodes in V. Given two nodes a and b in V, a directed arc ab from a to b signifies that event b can follow event a, defining an event pair (EP) ab

(Fig. 1(a)). The remaining pairs given by the alphabet Σ, but not in the ESG, form the set of faulty event pairs (FEP), e.g., ba for ESG of Fig.1(a). As a convention, a dedicated, start vertex e.g., [, is the entry of the ESG whereas a final vertex e.g.,] represents the exit. Note that [and] are not included in Σ. The set of FEPs constitutes the complement of the given ESG ($\overline{ESG}$ in Fig.1(a)).

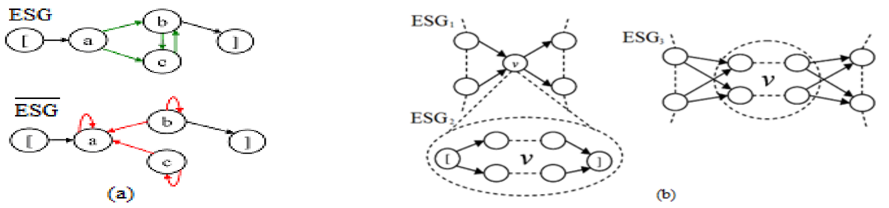


Fig. 1. (a) is an event sequence graph ESG and $\overline{ESG}$ as the complement of the given ESG; (b) is refinement of a vertex v and its embedding in the refined ESG

A sequence of *n+ 1 consecutive event* that represents the sequence of *n* arcs is called an *event sequence (ES) of the length n+1*, e.g., an *EP* (*event pair*) is an ES of length 2. An ES is *complete* if it starts at the initial state of the ESG and ends at the final event; in this case it is called a *complete ES* (*CES*). Occasionally, we call CES also as *walks* (or *paths*) through the ESG given [3].

Given an ESG, say $ESG_1 = (V_1, E_1)$, a vertex $v \in V_1$, and an ESG, say $ESG_2 = (V_2, E_2)$. Then replacing v by ESG_2 produces a *refinement* of ESG_1, say $ESG_3 = (V_3, E_3)$ with $V_3 = V_1 \cup V_2 \backslash \{v\}$, and $E_3 = E_1 \cup E_2 \cup E_{pre} \cup E_{post} \backslash E_{1replaced}$ (\: set difference operation), wherein $E_{pre} = N^-(v) \times \Xi(ESG_2)$ (connections of the predecessors of *v* with the entry nodes of ESG_2), $E_{post} = \Gamma(ESG_2) \times N^+(v)$ (connections of exit nodes of ESG_2 with the successors of *v*), and $E_{1replaced} = \{(v_i, v), (v, v_k)\}$with $v_i \in N^-(v)$ and $v_k \in N^+(v)$ (replaced arcs of ESG_1) [3].

Fig. 1(b) shows that ESG_1 is 'mother' ESG of higher level abstraction and ESG_2 is refinement of the vertex *v* of ESG_1. ESG_3 represents resultant refined 'mother' ESG_1; the refinement of its vertex *v* is embedded in the context. As Fig. 1(b) illustrates, *every* predecessor of vertex *v* of the ESG of higher level abstraction points to the entries of the refined ESG. In analogy, *every* exit of the refined ESG points to the successors of v. The refinement of *v* in its context within the original ESG of higher level abstraction contains no pseudo vertices *[* and *]* because they are only needed for the identifycation of entries and exits of the ESG of a refined vertex [3].

2.2 Test Cases Generation

Since, in this work, testing is performed based on a model, to generate test cases from ESGs, arc coverage is used. In this perspective, arc coverage provides a measure on how many of all possible event transitions in the system have been covered. Fulfilling of arc coverage criteria in the ESG modeling structure leads to, Chinese Postman Problem (CPP) [2].

2.3 Test Cost

The number and length of the event sequences are the primary factors that influence the cost of the testing. The test costs for CESs are defined as [3], where ψ_{CES} is the maximum number of CESs, $\beta \in \mathbb{R}$ as the weight factor for conducted multiple tests, $\#\, CES$ as the number of CESs, α the cost of each click and l_{cov} as the sum of lengths of CESs to cover all ESs of up to a given maximum length l_{max}.

$$\psi_{CES} = \beta .\#\, CES + \alpha .\, l_{cov} \tag{1}$$

2.4 Clustering

Clustering is used to generate an optimal partition of a given data set into a predefined number of clusters. Mathematically speaking, clustering partitions dataset $X = \{x_1, \ldots, x_i, \ldots, x_n\} \subset \mathbb{R}^p$ into c number clusters or groups S_k in the form of hyper spherical clouds of input vectors $x_{i1} = \{x_{i1}, \ldots, x_{ij}, \ldots, x_{ip}\} \in \mathbb{R}^p$, where $\mathbb{R}$ is the set of real numbers, p is the number of dimension.

We chose Gustafson Kessel clustering (soft clustering) as it allows elements to belong to several clusters, which reflects the situation in testing.

2.5 Gustafson Kessel Clustering

Gustafson and Kessel (GK, [12]) extended the standard fuzzy c-means algorithm [10] by employing an adaptive distance norm in order to detect clusters of different geometrical shapes in one data set. GK algorithm associates each cluster with both a point and a matrix, respectively representing the cluster centre and its covariance. Whereas fuzzy c-means algorithms make the implicit hypothesis that clusters are spherical, GK algorithm is not subject to this constraint and can identify ellipsoidal clusters [12]. The objective function of the GK algorithm is defined by

$$J(X; U, V, \{A_i\}) = \sum_{k=1}^{c} \sum_{i=1}^{n} (u_{ki})^m D_{kiA_k}^2 \tag{2}$$

Each cluster has its own norm-inducing matrix A_k, which yields the following inner-product norm:

$$D_{kiA_k}^2 = (x_i - v_k)^T A_k (x_i - v_k) \tag{3}$$

The matrices A can be defined as the inverse of the covariance matrix of X. A_k are used as optimization variables in the c-means functional, thus allowing each cluster to adapt the distance norm to the local topological structure of the data.

3 Model-Based Test Case Prioritization

The ordering of the CESs as test cases is in accordance with their importance degree that is defined indirectly, i.e., by estimation of events that are the nodes of ESG and represent objects (modules, components) of SUT. Events are presented as a *multidimensional data vector* $x_i = \{x_{i1}, \ldots, x_{ip}\}$ where p is the *number of its attributes*. A data set $X = \{x_1, \ldots, x_i, \ldots, x_n\} \subset \mathbb{R}^p$ where n is the number of events, is constructed which being an unstructured one is divided into c groups. Table 1 exemplifies the approach for qualifying an event corresponding to a node in ESG, introducing eleven attributes (p=11) that determine the *dimension* of a data point.

Table 1. The Attributes Table

No	*Attributes*
x_{i1}	The number of level in which the graph exist
x_{i2}	The number of sub-windows to reach an event from the entry [(gives its distance to the beginning).
x_{i3}	The number of incoming and outgoing edges (invokes usage density of a node, i.e., an event).
x_{i4}	The number of nodes that are directly and indirectly reachable from an event except entry and exit (indicates its “traffic” significance).
x_{i5}	The maximum number of nodes to the entry [(its maximum distance in terms of events to the entry).
x_{i6}	The number of nodes of a sub-node as sub-menus that can be reached from this node (maximum number of sub-functions that can be invoked further).
x_{i7}	The total number of occurrences of an event (a node) within all CESs, i.e., walks.
x_{i8}	The averaged frequencies of the usage of events $(Avrf(x_i))$(4) within the CESs (all graphs). (Determine the averaged occurrence of each event within all CESs (See (4)).
x_{i9}	The *balancing degree* determines balancing a node as the sum of all incoming edges (as plus (+)) and outgoing edges (as minus (-)) for a given node.
x_{i10}	The averaged frequencies of the usage of events (4) within the CESs (only one graph), where $f_q(x_i)$ is frequency of occurrence of event ith within CESq and $l(CES_q)$ is length of CESq, d is determined that events belonging to number of CESs as $d \leq$ r.
x_{i11}	The number of FEPs connected to the node under consideration (takes the number of all potential faulty events entailed by the event given into account).

$$Avrf(x_i) = \frac{1}{d}\left(\sum_{q=1}^{r} \frac{f_q(x_i)}{l(CES_q)}\right) \quad q = 1, \ldots, r \in N; \quad i = 1, \ldots, n \in N; \quad d \in N \tag{4}$$

where $f_q(x_i)$ is frequency of occurrence of event ith within CES_q and $l(CES_q)$ is length of CES_q, d is determined that events belonging to number of CESs as$d \leq r$.

3.1 Importance Degree of Groups

The importance degree of groups $(ImpD(S_k))$ has been assigned by ordering length of the groups mean vector $l(\bar{x}_k)$ from higher to lower. The group with the highest $l(\bar{x}_k)$ value has importance degree 1 (the most important group). *Importance index* $Imp(x_i)$ of *ith* event belonging to kth group is defined as follows:

$$Imp(x_i) = c - ImpD(S_k) + 1 \tag{5}$$

where c is the optimal number of the groups; $ImpD(S_k)$ is the importance degree of the group S_k (7) the *ith* event belongs to.

$$PrefD(CES_q) = \sum_{i=1}^{n} Imp(x_i)\, \mu_{S_k}(x_i) f_q(x_i) \tag{6}$$

where r is the number of CESs, $Imp(x_i)$ is importance index of the *ith* event (5), $\mu_{S_k}(x_i)$is membership degree of the *ith* event belonging to the group S_k (7), and $f_q(x_i)$ is frequency of occurrence of event *ith* within CES_q.

$$S_k = \{x_i | \mu_{ki} > \mu_{ji}, \ k \neq j \quad k = 1, \ldots, c\} \tag{7}$$

Finally, the CESs are ordered, choosing the events with their degree of membership from the ordered groups, and using importance index of events. Their descending preference degrees begin from the maximal one. The assignment of preference degrees to CESs is based on the rule that is given as follows:

- The CES under consideration has the highest degree if it contains the events that belong to the "top" group(s) with utmost importance degrees, i.e., that is placed within the highest part of group ordering.
- The under consideration has the lowest degree if it contains the events which belong to the group(s) that are within the lowest part of the "bottom" group(s) with least importance degree i.e., that is placed within the lowest part of group ordering.

Priorities of the test cases are determined by ranking PrefD(CES_q) from higher to lower. The path with the highest PrefD(CES_q) value has priority 1, (which is the highest priority value). The proposed test prioritizing algorithm is presented as follows.

Indirect Determination of the Test Prioritizing Algorithm:

```
Step 1  Construction of a set of events $X = \{x_{ij}\}$ where $i = 1, \dots, n ; i \epsilon N$ is an event index, and
        $j = 1, \dots, p ; j \epsilon N$ is an attribute index.
Step 2  Clustering the events using GK clustering algorithms.
Step 3  Classification of the events into c fuzzy groups.
Step 4  Determination of importance degrees of groups to length $l(\bar{x}_k)$ of group means vectors for all
        type of groups.
Step 5  Determination of importance index of event groups (5) with respect to crisp groups and fuzzy
        qualified groups.
Step 6  An ordering of the CESs as test cases using the preference degree (PrefD($CES_q$)) (6) for prioritiz-
        ing the test process.
```

4 Case Study

ISELTA is an online reservation system for hotel providers and agents. It is a cooperative product of the work between a mid-size travel agency (ISIK Touristik Ltd.) and University of Paderborn. For the test case study, a relatively small part of ISELTA called "Special Module" is used to prioritize the test cases. Through this module, one is able to add special prices to the specified number of rooms of certain type for the determined period of time in the given hotel. Consequently, one can edit existing specials and also remove them.

4.1 Modeling

SUT is tested using combination of sub-graphs of three levels. The legends of the nodes are given in Table 2. In Fig 2 (a), IS represents the first level of specials module. S0, S1, and S2+ (symbolically represented such as 1,3,6) are used to denote the current condition of the system. In case of S0, no entries exist in the system. In S1, there is only one special and finally S2+ means that there are two or more specials in the system.

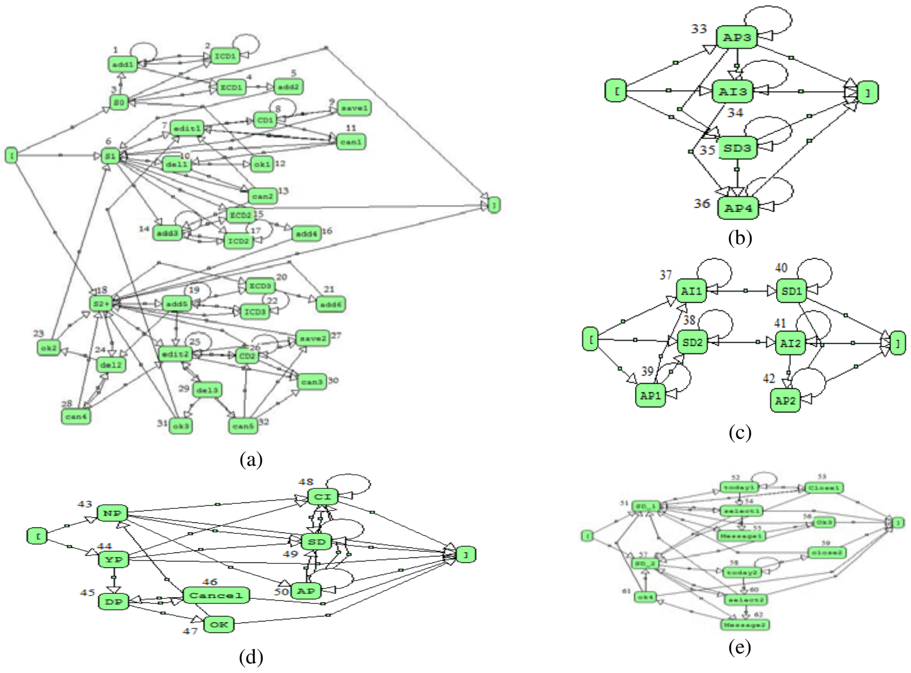


Fig. 2. ESGs of (a) ISELTA Special Module (IS); (b) Incomplete Data (ICD); (c) Enter Complete Data Module (ECD); (d) Change Data (CD); (e) Select Date (SD)

Table 2. Legend of the nodes

Nodes	*Legends*	*Nodes*	*Legends*	*Nodes*	*Legends*
S0	no special	save	click save button	OK	click ok button
S1	in a special	can	click cancel button	AI	Add Information
S2+	in two or more specials	NP	There isn't Photo	SD_1	select the arrival date
ICD	Incomplete Data	YP	There is Photo	SD_2	select the departure date
ECD	Enter Complete Data	DP	Delete Photo	today	click today button
CD	Change Data	CI	Change Information	select	select a date
add	click add button	SD	Select Date	Message	error message
edit	click edit button	AP	Add Photo	close	click close button
Ok	click ok button	Cancel	click cancel button		

In second level, the nodes ICD (2, 17, 22), ECD (4, 15, 20) and CD (8, 26) are refinement (Fig. 2(b), Fig. 2(c) and Fig. 2(d) respectively. Fig. 2(b)), ICD (Incomplete Data Module), shows how to enter missing data and SD (Select Date) gives in detail in third level as in Fig. 2(e). Fig. 2(c), ECD (Enter Complete Data Module), represents complete data entry, and Fig. 2(d) explains editing an existing special.

4.2 Clustering

Test case generation algorithm is mentioned in Section 3 and described in [3] in detail. CESs generated are listed in Table 3. As a follow-on step, each event, i.e.,

Table 3. List of the Test Cases

Levels	ESGs	No	CESs	Preference Degrees
First Level	**IS**	CES_1	[3 1 1 4 5 6 7 8 8 9 6 7 8 11 6 10 12 3 4 5 6 10 13 6 10 13 7 9 6 14 14 15 16 18 19 19 20 21 18 19 25 25 26 25 26 26 27 18 20 21 18 25 27 18 25 30 18 25 29 31 18 25 29 31 7 11 7 11 10 13 14 15 16 18 25 29 32 25 29 32 26 30 18 25 29 32 27 18 24 28 18 24 28 25 29 32 30 18 24 28 24 23 6 15 16 18]	355,03
		CES_2	[18 22 22 19 22 19 24 23 18]	31,78
		CES_3	[3 2 2 1 2 1 4 5 6]	31,1
		CES_4	[6 17 17 14 17 14 15 16 18]	31,07
		CES_5	[3]	3
Second Level	**ICD (2, 17, 22)**	CES_6	[33 33 34 34 36 36]	5,94
		CES_7	[33 35 35 36]	3,96
		CES_8	[35]	0,99
		CES_9	[34]	0,99
		CES_{10}	[33]	0,98
	ECD (4, 15, 20)	CES_{11}	[37 37 40 40 42 42]	6
		CES_{12}	[38 38 41 41 42]	5
		CES_{13}	[39 39 38 41]	4
		CES_{14}	[39 37 40]	3
	CD (8, 26)	CES_{15}	[43 48 48 49 48 50 48]	7
		CES_{16}	[44 45 46 45 46]	5,88
		CES_{17}	[43 49 49 50 49]	5
		CES_{18}	[44 45 47 43]	4,44
		CES_{19}	[44 45 47]	3,44
		CES_{20}	[43 50 50]	3
		CES_{21}	[44 49]	2
		CES_{22}	[44 48]	2
		CES_{23}	[44]	1
Third Level	**SD (35, 38, 40, 49)**	CES_{24}	[51 52 52 54 51 52 53 51 54 57 58 58 59 51 54 55 56 51 54]	37,7
		CES_{25}	[57 58 60 51 53 57 60 57 60 62 61 51 53]	25,88
		CES_{26}	[51 55 56 57 58 59]	11,94
		CES_{27}	[57 62 61 57 62 61]	12
		CES_{28}	[51 55 56]	5,94
		CES_{29}	[57 60]	4

corresponding node in the ESG, is represented as a multidimensional data point using the values of eleven attributes as defined in the previous section.

For the dataset gained from the case study, the optimal number c of clusters is determined to be 4. In order to divide the given dataset into clusters, we apply the clustering algorithm mentioned in Section 3. After the clustering process, the importance degrees of groups and so the events are determined by using (5). The preference degrees of the CESs are determined indirectly by (6) that depend on the importance degree of the events (5) and frequency of event(s) within the CES. Finally, we determine an ordering of CESs according to their descending preference degrees. The assignment of preference degree is based on the rule that is given as follows: The CES has the highest preference degree is that contains the events which belong to the most important groups, and the CES has the lowest preference degree is that contains the events which belong to the less important groups. Table 3 presents the results of the analysis.

4.3 Testing

Test process is performed by executing test sequences CES_1-CES_5 (see Table 3) by refining them in three levels. Table 4 gives the cost of testing (ψ_{CES}) required for these three types of refinements. With the number of levels, test cost excessively increase (Table 3). We assume it is not possible to execute all test cases of second and third levels. Thus, we select test sequences detecting faults. The number of faults occurred during testing and test cases numbers are given in Table 4.

4.4 Test Selection Strategies Based on Prioritization

Strategy 1. In the first level, in case of no refinements, test sequences are ranked according to the preference degrees, and they are executed by these rankings. In our case, CES_1, CES_2, CES_3, CES_4, CES_5, are executed in the given order.

Strategy 2. 1st level test sequences are refined using the highest priority test sequences from the second level, i.e., each composite event is always replaced by the same highest priority test sequence related to it.

Example 1. In 1st + 2nd level, tests are executed as follow:

In CES_1, 4, 8, 15, 20 and 26 are composite events. Thus, during execution, they are replaced by CES_{11}, CES_{15}, CES_{11}, CES_{11}, CES_{15}, respectively.

Table 4. Cost of CESs

Refinement Levels	ψ_{CES}	Mean of l_{cov}	# CES	Test cases	Detected faults	Additional faults
1st (no refinement)	$5\beta + 134\alpha$	27	5	59	8	8
1st + 2nd	$8.10^9\beta+1{,}5.10^{13}\alpha$	198	~8.10^9	656	11	3
1st + 2nd + 3rd	$10^{30}\beta+1{,}6.10^{33}\alpha$	1424	~10^{30}	5368	12	1

Strategy 3. 1st level test sequences are refined using the highest priority test sequences from the second and third levels, i.e., each composite event is always replaced by the same highest priority test sequence.

Example 2. In 1st + 2nd + 3rd levels, tests are executed as follow:

In CES_1, 4, 8, 15, 20 and 26 are composite events. Thus during execution, they are replaced by CES_{11}, CES_{15}, CES_{11}, CES_{11}, CES_{15}, respectively. In addition, in CES_{11}, 40 and in CES_{15}, 49 are composite events, so that they are replaced by CES_{24}, CES_{24}.

4.5 Results

The execution first test sequence (CES_1) revealed seven faults in the system (see Table 5). The second test sequence (CES_2) revealed one additional fault, and the other test sequences revealed no new fault. In the second level, only test sequences having the highest priority (CES_6, CES_{11}, CES_{15}) are replaced into the first level test sequences. Three new faults are revealed; two of them are detected by refined CESs. In

Table 5. List of Faults

Levels	CESs	Faults
First	CES_1	• The number of special offers available can be set to zero. • Today button does not work • In existing offers the arrival date can be set to dates into the past. • In existing offers the departure date can be set to dates into the past. • Departure date is falsely autocorrected • Departure date is not autocorrected. • Missing warning for wrong file types.
	CES_2	• A special can be inserted with missing start date
Second	CES_1	• If a departure date in the past is set to another day in the past, the arrival is set to the current date. • Missing warning for incorrect prices and number of special offers
	CES_2	• Missing warnings for empty dates.
Third	CES_2	• Departure and arrival date can be selected previously date.

second and third level, test sequences with the highest priorities are replaced into the second and first level refinements. As a result, only one fault is detected.

5 Conclusions, Limitations, Scalability Aspects, and Future Work

A model-based, coverage-and specification-oriented approach is presented for ordering test cases according to their degree of preference at different modeling levels. No prior knowledge about the tests carried out before is needed. The priorities of test sequences are determined using GK clustering technique.

Note that the approach heavily relies on model used. As models usually focus on some selected features of SUT and thus neglect the others, the results can largely differ if other features dominate by modeling.

The case study (Section 4) used a relatively small application; for applying the approach to larger modules, rudimentary tools are available. The completion and improvement of these tools are planned to increase the scalability of the approach.

We plan to extend the approach by considering *n*-tuple events coverage (with *n*=3, 4, 5, …) instead of the pair-wise event coverage (*n*=2) as studied in this paper. Additionally, more empirical research is planned.

References

1. Bryce, R.C., Colbourn, C.C.: Prioritized Interaction Testing for Paire-wise Coverage with seeding and Constraints. Information and Software Technology 48, 960–970 (2006)
2. Edmonds, J., Johnson, E.L.: Matching, Euler Tours and the Chinese Postman. Math. Programming, 88–124 (1973)
3. Belli, F., Budnik, C.J., White, L.: Event-based Modelling, analysis and testing of user Interactions- Approach and Case Study. J. Software Testing, Verification & Reliability 16(1), 3–32 (2006)
4. Elbaum, S., Malishevsky, A., Rothermel, G.: Test Case Prioritization: A Family of Empirical Studies. IEEE Transactions on Software Engineering 28(2), 182–191 (2002)

5. Gökçe, N., Eminov, M., Belli, F.: Coverage-Based, Prioritized Testing Using Neural network Clustering. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 1060–1071. Springer, Heidelberg (2006)
6. Belli, F., Eminov, M., Gökçe, N.: Coverage Oriented Prioritized Testing- A fuzzy Clustering Approach and Case Study. In: Bondavalli, A., Brasileiro, F., Rajsbaum, S. (eds.) LADC 2007. LNCS, vol. 4746, pp. 95–110. Springer, Heidelberg (2007)
7. Binder, R.V.: Testing Object-Oriented Systems. Addison-Wesley, Reading (2000)
8. Bezdek, C.J.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
9. Hoppner, F., Klawonn, F., Kruse, R., Runkler, T.: Fuzzy Cluster Analysis. John Wiley, Chichester (1999)
10. Yhu, H., Hall, P.A.V., May, J.H.R.: Software Unit Test Coverage and Adequacy. ACM Computing Surveys 29(4) (December 1997)
11. Belli, F.: Finite-State Testing and Analysis of Graphical User Interfaces. In: Proc. 12th Int. Symp. Software Reliability Eng. (ISSRE 2001), pp. 34–43 (2001)
12. Gustafson, D.E., Kessel, W.C.: Fuzzy clustering with a fuzzy covariance matrix. In: Proc. IEEE Conf. Decision Contr., San Diego, CA, pp. 761–766 (1979)

Adoption of Requirements Engineering Practices in Malaysian Software Development Companies

Badariah Solemon[1], Shamsul Sahibuddin[2], and Abdul Azim Abd Ghani[3]

[1] College of IT, Universiti Tenaga Nasional, Km 7 Jalan Kajang-Puchong, 43009 Kajang, Selangor, Malaysia
badariah@uniten.edu.my
[2] Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia
shamsul@utm.my
[3] Universiti Putra Malaysia, Selangor, Malaysia
azim@fsktm.upm.edu.my

Abstract. This paper presents exploratory survey results on Requirements Engineering (RE) practices of some software development companies in Malaysia. The survey attempted to identify patterns of RE practices the companies are implementing. Information required for the survey was obtained through a survey, mailed self-administered questionnaires distributed to project managers and software developers who are working at software development companies operated across the country. The results showed that the overall adoption of the RE practices in these companies is strong. However, the results also indicated that fewer companies in the survey have use appropriate CASE tools or software to support their RE process and practices, define traceability policies and maintain traceability manual in their projects.

Keywords: Requirements Engineering (RE), RE practices, adoption.

1 Introduction

Good RE practices, according to Davis and Zowghi [1], can "either reduces the cost of the development project or increases the quality of the resulting project when used in specific situation". There exist in the literature several empirical research that study the RE practices in different companies in different parts of the world. Such empirical research include those by El Emam and Madhavji in 1995 [2], a study of 60 (12 interviews and 48 document inspection) cases in Canada; Nikula and colleagues. in 2000 [2], a survey of 15 respondents in twelve small and medium enterprises (SMEs) in Finland; Neill and Laplante in 2003, as explained in [4], a survey of 194 respondents from diverse mix of industries in US; and Damian and colleagues in 2004 [5], a study within a single Australian company. Findings from these studies not only show the different types of RE practices implemented but also provide empirical evidence that improved RE practices help in improving the software development projects.

However, results from most of these surveys may not appropriate to be generalized from such a relatively small samples used. In addition, the situation in Malaysia was not quite known as there was no research to study both the state of the RE problems and RE practices in the country. Hence, we designed and conducted a survey to find out the current RE problems and practices amongst these software development companies. The survey is adapted from three surveys as reported in [2], [6], and [7]. One of the objectives of the survey is to investigate the pattern of RE problems and practices amongst the software development companies in Malaysia. In this paper, however, the focus is only at presenting the pattern of RE practices implemented in the software companies. In the next two sections, the materials and data collection methods, and the results on valid responses, and the analyses performed to interpret the results of the study are explained.

2 Materials and Methods

To help investigate the patterns of current RE problems and practices, the survey method was used. Self-administered, mailed questionnaire was chosen as the instrument of the survey. The questionnaire was constructed with four sections to capture the required information: Section A, Section B, Section C and Section D. Section A focuses at the background information on the companies, the respondents' profile and attitude to the importance of the RE process and software process improvement (SPI). Section B was intended to identify RE problems, while Section C was looking for information related to the adoption and adequacy of RE practices implementations in the companies. Finally, Section D allows respondents to add other RE practices not addressed in the questionnaire but are important to ensure software development project success in their organizations, and to comment on any aspect they choose. This paper aims to present only the results of section C. Meanwhile, results of the first two sections have been reported in other articles including in [8], [9], and [10].

The survey questionnaires were mailed to 500 randomly selected samples of software development companies in Malaysia. A software development company in the survey is defined following the definition by Sison and colleagues [11] as "[...] a for-profit organization whose main business is the development of software for customers external to the organization. Included in this definition are developers of generic software products (e.g., Microsoft) as well as providers of software solutions for specific domain (e.g., Infosys). Excluded in this definition are IT departments that cater primarily to the needs of the organization of which they are part". As mentioned in [8], the survey data was collected through February and March 2008. Although a total of 90 responses were received, making up 18% response rate, only 64 responses are complete and considered valid for analysis. Most are excluded mainly due to incomplete answers. Although this response rate is fairly low, we decided to proceed with analyzing the responses because according to [12], a low response rate of about 5% would already be sufficient for an exploratory study of this kind. Moreover, the 13% response rate is virtually the same with that reported in [11].

3 Results and Discussion

As mentioned earlier, Section C of the questionnaire was looking for information related to the adoption and adequacy of RE practices implementations in the companies. The information was obtained using a revised list of 72 RE practices items. These items are categorized into 6 key practices: requirements elicitation (Item 3.1-Item 3.13), requirements analysis and negotiation (Item 4.1 – Item 4.9), requirements specifications and documentation (Item 5.1 – Item 5.20), requirements verification and validation (Item 6.1 – Item 6.9), requirements management (Item 7.1 – Item 7.10), and other practices (Item 8.1 – Item 8.11). The RE practices were gathered from the literature such as [7], [13], [14], [15], [16], and [17]. Each item in the list uses a 0 to 3 Likert scale, which will be scored based on the REAIMS assessment (0 = never, 1 = used at discretion of project manager, 2 = normal used, 3 = standardized) as proposed in [14]. A zero scale also mean a non-adoption of the RE practice, whereas the other scale numbers represent different adequacy level of the RE practice implementation in the respondent's organization. However, due to limited space, this paper focuses at presenting only the results related to the adoption level of the RE practices.

3.1 Adoption of RE Practices

First, we queried respondents on their requirements elicitation practices as this key practice is an especially critical part of the RE process [18]. Table A-1 in Appendix A summarizes the results on requirements elicitation practices. The mean overall adoption level of the requirements elicitation practices in the companies in our survey is 96.03%, which is expectably high for such important practices. All 100% of the responding companies define the system's operating environment (Item 3.5), use business concerns to drive requirements elicitation (Item 3.6), and define operational processes (Item 3.12) as part of their requirements elicitation practices. The practices with the lowest three adoption level are prototype poorly understood requirements (Item 3.10), be sensitive to organizational and political considerations (Item 3.2), and look for domain constraints (Item 3.7).

Next, table A-2 summarizes the results of the requirements analysis and negotiation practices. From this table, the mean overall adoption level of the requirements analysis and negotiation practices is relatively high at 89.58%. It can be observed that the most top adopted requirements analysis and negotiation practice is define system boundaries (Item 4.1). The second most adopted practice in this key practice is Item 4.6, prioritize requirements. This is followed by Item 4.7, classify requirements into classes or sections. Meanwhile, the lowest three adopted requirements analysis and negotiation practices are use interaction matrices to find conflicts and overlaps (Item 4.8), provide software or tools to support negotiations (Item 4.4), and assess requirements risks (Item 4.9).

Table A-3 lists the adoption level of the requirements specification and documentation practices. The mean overall adoption level of the requirements specification and documentation practices is high too (92.7%). At least one out of 20 requirements specification and documentation practices (i.e., apply the standard structure to all of your requirements documents) is rated with 100% adoption by the responding companies. The top three most adopted practices are Item 5.3 which is apply the standard structure to all of your requirements, Item 5.8 which is layout the document for readability, and Item 5.12 which is model the system architecture showing the entire system, subsystem and the links between them. Whilst the bottom three adopted practices are Item 5.2 which is tailor a requirements standard such as IEEE Std 830.1998, Item 5.20 which is specify non-functional requirements quantitatively, and Item 5.15 which is shows traceability between stakeholder requirements and system models.

From table A-4, it can be observed that the mean overall adoption level of the requirements verification and validation practices is 89.06%. All 100% of the responding companies check that the requirements document meets their standard (Item 6.1). The other two top most adopted practices in this RE key practices include perform requirements reviews (Item 6.2), and produce requirements test cases (Item 6.8). Meanwhile, the lowest three adopted requirements verification and validation practices are Item 6.7 which is write a draft user manual at early stage in the process, Item 6.4, which is set-up inspections team and perform formal inspections for requirements and Item 6.9, which is paraphrase system models.

Table A-5 summarizes the adoption level of the requirements management practices. The mean overall adoption level of the requirements management practices (87.97%) is the lowest adoption level of all the 6 key practices. Nevertheless, Item 7.1, Uniquely identify each requirement is being implemented by all of the responding companies). Meanwhile, the other two highly adopted requirements management practices are Item 7.10, Define the real (actual) requirements, and Item 7.7, Identify reusable system requirements. The bottom three adopted requirements management practices are Item 7.5, Use a database to manage requirements, Item 7.4, Maintain traceability manual, and Item 7.3, Define traceability policies.

Finally, it can be observed from Table A-6 that the mean overall adoption level of the other RE practices is high too (89.90%). The top three adopted RE practices under this key practice category are define the requirements process using best practices (Item 8.5), assign skilled project managers and team members to RE activities (Item 8.3), and establish and utilize a joint team responsible for the requirements (Item 8.4). The lowest adopted practice under this key practice category is use appropriate CASE tools for RE support and others (Item 8.1), (Item 8.11), and (Item 8.2).

3.2 Top and Lowest Ten RE Practices

The mean overall adoption level of the all the RE practices presented is high (91.21%). Table 1 summarizes the top and lowest 10 RE practices adopted in the responding companies. At least six out of 72 RE practices are rated with 100% adoption by the responding companies. These practices include define the system's operating environment (Item 3.5), use business concerns to drive requirements elicitation (Item 3.6), define operational processes (Item 3.12), apply the standard structure to all of your requirements documents (5.3), check that the requirements document meets your standard (Item 6.1), and uniquely identify each requirements (Item 7.1). Data in the table also suggest that majority of these top 10 RE practices adopted are actually categorized under the requirements elicitation, and requirements specification and documentation key practices. At least two out of the top ten RE practices fall under the requirements verification and validation, and requirements management key practice categories.

According to the survey results, three of the lowest RE practices are related to adoption of software or tool to support RE process and its practices. They are Item 8.1

which is use appropriate CASE tools for RE support and others, Item 4.8 which is use interaction matrices to find conflicts and overlaps requirements and system models, and Item 4.4 which is provide software or tools to support negotiations. It is also interesting to see that at least two out of the ten lowest RE practices are attributed to the traceability related practices i.e., maintain traceability manual (Item 7.4), and define traceability policies (Item 7.3).

Table 1. Top and lowest ten RE practices

Top ten practices		Lowest ten practices	
3.5	Define the system's operating environment	5.2	Tailor a requirements standard such as IEEE Std 830.1998
3.6	Use business concerns to drive requirements elicitation	8.1	Use appropriate CASE tools for RE support and others
3.12	Define operational processes	8.2	Use interaction matrices to find conflicts and overlaps
5.3	Apply the standard structure to all of your requirements documents	7.5	Use a database to manage requirements
6.1	Check that the requirements document meets your standard	6.7	Write a draft user manual at early stage in the process
7.1	Uniquely identify each requirements	7.4	Maintain traceability manual
5.8	Layout the document for readability	4.4	Provide software or tools to support negotiations
7.10	Define the real (actual) requirements	5.20	Specify non-functional requirements quantitatively (put a figure on)
3.3	Identify and consult system stakeholders	7.3	Define traceability policies
5.12	Model the system architecture showing the entire system, subsystem and the links between them	6.4	Set-up inspections team and perform formal inspections for requirements

3.3 Findings

The high mean overall adoption level of the all the RE practices, as presented earlier, suggests that the overall adoption of the practices in these companies is actually strong. It also indicates that the companies had implemented almost all of the practices. This is true especially for practices that received 100% rating such as define the system's operating environment, use business concerns to drive requirements elicitation, define operational processes, apply the standard structure to all of your requirements documents, check that the requirements document meets your standard, and uniquely identify each requirements. Data in the table also suggest that majority of these top 10 RE practices are actually categorized under the requirements elicitation, and requirements specification and documentation categories. Two of the top ten activities fall under the requirements verification and validation, and requirements management categories.

Meanwhile, the results in the survey also suggest that there is a relatively low adoption of software or CASE tool to support RE process and practices similar to those findings discovered in other related studies (e.g., [19], and [20]). This low adoption may be due to reasons such as cost, lack of measurable returns and unrealistic expectations as suggested in [21]. According to Kannenberg and Saiedian [22], requirements traceability has been demonstrated to provide many benefits to software companies. In spite of the benefits that traceability offers to the RE process and software engineering industry, less companies in the survey define traceability policies or maintain traceability manual in their projects. Perhaps this has to do with the low CASE tool adoption amongst these practitioners as poor tool support is the biggest challenge to the implementation of requirements traceability as discussed in [22].

4 Conclusion

As presented earlier, this part of the survey has identified and analyzed the adoption level of RE practices implementation specifically in software development companies operated in Malaysia. The overall adoption of the RE practices, which was described according to the 6 key RE practices, has been presented in this paper. Also, the top ten and the bottom ten RE practices adopted in these companies have been listed. It concludes that the overall adoption of the RE practices in these companies is strong. However, the results also indicated that fewer companies in the survey have use appropriate CASE tools or software to support their RE process and practices, define traceability policies and maintain traceability manual in their projects.

As stated in [23], "surveys are of course based on self-reported data which reflects what people say happened, not what they actually did or experienced". Because we surveyed project managers and software developers, the results are limited to their knowledge, attitudes, and beliefs regarding the RE problems, RE practices and the software development projects with which they have taken part. While every care has been taken to ensure the validity and reliability of the information gathered, its representativeness cannot be 100% guaranteed as the data were obtained from sampled population. To mitigate the possible threats to empirical validity of the survey results, several measures have been taken to evaluate the four criteria (i.e., construct validity, internal validity, external validity, and reliability) for validity suggested in [24]. While we would not assume that the survey results are typical of all software development companies, we believe that they are reasonably typical of software development companies in Malaysia.

Acknowledgement

We would like to thank the respondents for their cooperation and willingness to share their experiences and data. Also, we would like to give special acknowledgment to the Universiti Tenaga Nasional, which provided funding for this work through grant number J5100-14-018.

References

1. Davis, A.M., Zowghi, D.: Good Requirements Practices are neither Necessary nor Sufficient. In: Requirements Eng., vol. 11, pp. 1–3. Springer, London (2006)
2. El Emam, K., Madhavji, N.H.: A Field Study of Requirements Engineering Practices in Information Systems Development. In: 2nd IEEE International Symposium of Requirements Engineering, pp. 68–80. IEEE Press, New York (1995)

3. Nikula, U., Sajaniemi, J., Kalviainen, H.: A State-of-the-Practice Survey on Requirements Engineering in Small- and Medium-Sized Enterprises. Technical Report. Telecom Business Research Center, Lappeenranta University of Technology, Finland (2000)
4. Laplante, P.A., Neill, C.J., Jacobs, C.: Software Requirements Practices: Some Real Data. In: Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop, SEW-27 2002 (2003)
5. Damian, D., Zowghi, D., Vaidyanathasamy, L., Pal, Y.: An industrial case study of immediate benefits of requirements engineering process improvement at the Australian Center for Unisys Software. Empirical Software Engineering Journal 9(1-2), 45–754 (2004)
6. Beecham, S., Hall, T., Rainer, A.: Software Process Improvement Problems in Twelve Software Companies: An Empirical Analysis. Empirical Software Engineering 8(1), 7–42 (2003)
7. Niazi, M., Shastry, S.: Role of Requirements Engineering in Software Development Process: An Empirical Study. In: IEEE INMIC 2003, pp. 402–407 (2003)
8. Solemon, B., Sahibuddin, S., Ghani, A.A.A.: Requirements Engineering Problems and Practices in Software Companies: An Industrial Survey. In: International Conference on Advanced Software Engineering and Its Applications, ASEA 2009 Held as Part of the Future Generation Information Technology Conference, FGIT 2009, Jeju Island, Korea, December 10-12, pp. 70–77 (2009)
9. Solemon, B., Sahibuddin, S., Ghani, A.A.A.: Requirements Engineering Problems in 63 Software Companies in Malaysia. In: International Symposium on Information Technology 2008 (ITSIM 2008), August 26-28 (2008)
10. Solemon, B., Sahibuddin, S., Ghani, A.A.A.: An Exploratory Study of Requirements Engineering Practices in Malaysia. In: 4th Malaysian Software Engineering Conference, Universiti Malaysia Terengganu (UMT), Terengganu (2008)
11. Sison, R., Jarzabek, S., Hock, O.S., Rivepiboon, W., Hai, N.N.: Software Practices in Five ASEAN Countries: An Exploratory Study. In: The 28th International Conference in Software Engineering, ICSE 2006, pp. 628–631. ACM, China (2006)
12. Lethbridge, T.C., Sim, S.E., Singer, J.: Studying Software Engineers: Data Collection Techniques for Software Field Studies. In: Empirical Software Engineering, vol. 10, pp. 311–341. Springer Science + Business Media, Inc., The Netherlands (2005)
13. Kotonya, G., Sommerville, I.: Requirements Engineering. Processes and Techniques. John Wiley & Sons, Chichester (1997)
14. Sommerville, I., Sawyer, P.: Requirements Engineering. A Good Practice Guide. John Wiley & Sons, Chichester (1997)
15. Hofmann, H.F., Lehner, F.: Requirements Engineering as a Success Factor in Software Projects. IEEE Software, 58–66 (2001)
16. Young, R.R.: Effective Requirements Practices. Addison-Wesley, Boston (2001)
17. Beecham, S., Hall, T., Rainer, A.: Defining a Requirements Process Improvement Model. Software Quality Journal 13, 247–279 (2005)
18. Pfleeger, S.L., Atlee, J.M.: Software Engineering. In: Theory and Practice, 3rd edn., Pearson Prentice Hall, New Jersey (2006)
19. Aaen, I., Siltanen, A., Sørensen, C., Tahvanainen, V.-P.: A Tale of Two Countries: CASE Experiences and Expectations. In: Kendall, K.E., Lyytinen, K., DeGross, J. (eds.) The Impact of Computer Supported Technologies on Information Systems Development, IFIP Transactions A-8, pp. 61–91. North-Holland, Amsterdam (1992)
20. Kusters, R.J., ja Wijers, G.M.: On the Practical Use of CASE-tools: Results of a Survey, CASE 1993. In: Proceedings of the 6th International Workshop on CASE, Singapore, pp. 2–10. IEEE Computer Society Press, Los Alamitos (1993)

21. Lending, D., Chervany, N.L.: The Use of CASE Tools. In: Proceedings of the 1998 ACM Special Interest Group on Computer Personnel Research Annual Conference, Boston, Massachessetts, US, pp. 49–58 (1998)
22. Kannenberg, A., Saiedian, H.: Why Software Requirements Traceability Remains a Challenge. The Journal of Defense Software Engineering (July/August 2009), `http://www.stsc.hill.af.mil/crosstalk/2009/07/0907KannenbergSaiedian.html` (retrieved July 10, 2010)
23. Verner, J., Cox, K., Bleistein, S., Cerpa, N.: Requirements Engineering and Software Project Success: An Industrial Survey in Australia and the U.S. Australasian Journal of Information Systems 13(1), 225–238 (2005)
24. Easterbrook, S., Singer, J., Storey, M.-A., Damian, D.: Selecting Empirical Methods for Software Engineering Research. In: Shull, F., Singer, J. (eds.) Guide to Advanced Empirical Software Engineering, pp. 285–311. Springer, London (2007)

Appendix A

Table A-1. Adoption of requirements elicitation practices

	Item	Valid Responses	Adoption Level
			%
3.1	Assess system feasibility	64	96.88
3.2	Be sensitive to organizational and political considerations	64	92.19
3.3	Identify and consult system stakeholders	64	98.44
3.4	Record requirements sources	64	96.88
3.5	Define the system's operating environment	64	100.00
3.6	Use business concerns to drive requirements elicitation	64	100.00
3.7	Look for domain constraints	64	92.19
3.8	Record requirements rationale	63	93.65
3.9	Collect requirements from multiple viewpoints (sources)	64	96.88
3.10	Prototype poorly understood requirements	64	87.50
3.11	Use scenarios to elicit requirements	64	96.88
3.12	Define operational processes	64	100.00
3.13	Reuse requirements	64	96.88

Table A-2. Adoption of requirements analysis and negotiation practices

	Item	Valid Responses	Adoption Level
			%
4.1	Define system boundaries	64	98.43
4.2	Use checklists for requirements analysis	64	95.31
4.3	Use operational definitions to define requirements	64	93.75
4.4	Provide software or tools to support negotiations	64	78.13
4.5	Plan for conflicts and conflict resolution	64	89.07
4.6	Prioritize requirements	63	96.83
4.7	Classify requirements into classes or sections	64	96.87
4.8	Use interaction matrices to find conflicts and overlaps	64	70.32
4.9	Assess requirements risks	64	89.06

Table A-3. Adoption of requirements specification and documentation practices

	Item	Valid Responses	Adoption Level
			%
5.1	Define a standard requirements document structure	64	96.87
5.2	Tailor a requirements standard such as IEEE Std 830.1998	64	62.50
5.3	Apply the standard structure to all of your requirements documents	63	100.00
5.4	Explain how to use the document	63	85.94
5.5	Include a summary of the requirements (in an overview section)	63	95.32
5.6	Make a business case for the system (showing the systems part in the business)	63	95.31
5.7	Define specialized terms	63	96.88
5.8	Layout the document for readability	63	98.44
5.9	Make the document easy to change	63	95.31
5.10	Develop complementary system models	64	95.31
5.11	Model the system's environment	64	96.88
5.12	Model the system architecture showing the entire system, subsystem and the links between them	64	98.43
5.13	Use systematic approaches for systems modelling	64	93.75
5.14	Use a data dictionary	64	93.76
5.15	Shows traceability between stakeholder requirements and system models	64	82.81
5.16	Define standard templates for describing requirements	64	90.62
5.17	Use simple and concise language	64	96.87
5.18	Use diagrams appropriately	63	95.31
5.19	Supplement natural language with other descriptions of requirements	63	90.62
5.20	Specify non-functional requirements quantitatively (put a figure on)	64	79.69

Table A-4. Adoption of requirements verification and validation practices

	Item	Valid Responses	Adoption Level
			%
6.1	Check that the requirements document meets your standard	64	100.00
6.2	Perform requirements reviews	64	98.43
6.3	Use multi-disciplinary teams to review requirements	64	85.93
6.4	Set-up inspections team and perform formal inspections for requirements	64	81.26
6.5	Define and document validation checklists	64	92.19
6.6	Use prototyping to validate the requirements	64	89.06
6.7	Write a draft user manual at early stage in the process	64	75.01
6.8	Produce requirements test cases	64	96.88
6.9	Paraphrase system models (convert system models into natural language)	64	82.82

Table A-5. Adoption of requirements management practices

	Item	Valid Responses	Adoption Level
			%
7.1	Uniquely identify each requirements	64	100.00
7.2	Define policies for requirements management	64	85.94
7.3	Define traceability policies	64	81.25
7.4	Maintain traceability manual	64	78.12
7.5	Use a database to manage requirements	64	73.44
7.6	Define change management policies	64	89.06
7.7	Identify reusable system requirements	64	96.88
7.8	Identify volatile requirements	64	92.19
7.9	Record rejected requirements	64	84.38
7.10	Define the real (actual) requirements	64	98.44

Table A-6. Adoption of other RE practices

	Item	Valid Responses	Adoption Level
			%
8.1	Use appropriate CASE tools for RE support and others	63	65.62
8.2	Allocate 15% to 30% of total project effort to RE activities	63	85.93
8.3	Assign skilled project managers and team members to RE activities	63	93.75
8.4	Establish and utilize a joint team responsible for the requirements	63	93.75
8.5	Define the requirements process using best practices	63	95.31
8.6	Use and continually improve a requirements process	63	93.74
8.7	Iterate the system requirements and the system architecture repeatedly	63	93.74
8.8	Use a mechanism to maintain project communication	63	89.06
8.9	Select familiar methods and maintain a set of work products	63	90.62
8.10	Conduct careful and objective post-mortem analysis to project	63	89.06
8.11	Conduct Software Quality Assurance (SQA) activities in this RE process	63	82.81

Minimum Distortion Data Hiding*

Md. Amiruzzaman[1], M. Abdullah-Al-Wadud[2], and Yoojin Chung[3,**]

[1] Department of Computer Science
Kent State University, Kent, Ohio 44242, USA
{mamiruzz,peyravi}@cs.kent.edu

[2] Department of Industrial and Management Engineering,
Hankuk University of Foreign Studies,
Kyonggi, 449-791, South Korea
wadud@hufs.ac.kr

[3] Department of Computer Science,
Hankuk University of Foreign Studies,
Kyonggi, 449-791, South Korea
chungyj@hufs.ac.kr

Abstract. In this paper a new steganographic method is presented with minimum distortion, and better resistance against steganalysis. There are two ways to control detectability of stego message: one is less distortion, and another is less modification. Concerning the less distortion, this paper focuses on DCT rounding error, and optimizes the rounding error in a very easy way, resulting stego image has less distortion than other existing methods. The proposed method compared with other existing popular steganographic algorithms, and the proposed method achieved better performance. This paper considered the DCT rounding error for lower distortion with possibly higher embedding capacity.

Keywords: Steganography, JPEG image, coefficient, rounding error, distortions, information hiding.

1 Introduction

As more and more data hiding techniques are developed and improved, Steganalysis techniques are also advanced. As the Steganalysis advances, the steganography becomes more complicated. Among all steganographic method, the Least Significant Bit (LSB) modification method is considered as a pioneer work of steganography. The LSB modification and LSB matching have two different application areas. LSB modification is popular for uncompressed domain, while LSB matching is popular for compressed domain. It is found that detection processes of these techniques are also different. Nowadays, steganographic techniques are getting more secure against statistical attacks and undetectable by other different attacks. Many innovative steganographic algorithms are developed within last decade. Among them, [5], [6], [7], [8], [9] are most popular.

* This work was supported by Hankuk University of Foreign Studies Research Fund of 2010.

** Corresponding author.

However, many researchers are also having interest to break steganographic schemes. There are several steganalysis methods invented within last decade [3], [10]. Among them statistical attack [10] is one of the most popular and effective attacks in steganographic world. Another famous attack is the calibrated statistics attack [1], [2]. However, there are several other attacks available. Thus, data hiding methods have to be designed to make them secure from statistical attack because this attack is relatively easy to combat. Simple solution against this attack is keeping the same or similar stego image histogram to the original image histogram. However, keeping the same shape of a magnitude histogram is not easy to achieve as long as the coefficient magnitudes are altered. Note that one branch of past history of steganography was inventing methods to preserve the original histogram perfectly. LSB overwriting methods including OutGuess [5] can preserve the original histogram almost perfect (in fact, not absolutely perfect). This method modifies half of the nonzero coefficients and corrects the distorted histogram by adjusting with the rest of unused coefficients. In general, perfect preservation is not possible because data pattern is not ideal, but random. The most popular and revolutionary method is F5 by Westfeld [10]. F5 method [10] also tries to narrow the gap between original and modified histograms by decrementing nonzero JPEG coefficients to 0 and applying matrix embedding and permutative straddling. Sallee models the marginal distribution of DCT coefficients in JPEG-compressed images by the generalized Cauchy distribution [6]. Thus, the embedded message is adapted to the generalized Cauchy distribution using arithmetic coding. Arithmetic coding transforms unevenly distributed bit streams into shorter, uniform ones. This procedure is known as MBS1. One weak point of the MBS1 is that block artifact increases with growing size of the payload. MBS2 has presented a method to overcome this weakness [7]. The MBS2 embeds message in the same way as MBS1 does, but its embedding capacity is only half of that of MBS1. The other half of the nonzero DCT coefficients is reserved for de-blocking purpose.

For the first time, in [4], mentioned about the distortion by rounding operation in JPEG image processing. In their paper they gave a detail description of rounding errors. They also proposed an embedding technique by LSB modification with a modified matrix embedding. In this proposed method instead of matrix a group/block of coefficients are used. The main advantage of the proposed method is variable length of group/block, easy to implement and easy to control the hiding capacity.

The rest of this paper is organized as follows: In Section 2, existing method (i.e. F5, OutGuess, MBS1, MBS2 etc.). Section 3, the proposed method. In Section 4, summarizes experimental results, and Section 5 concludes the paper.

2 Existing Methods

Sallee models the marginal distribution of DCT coefficients in JPEG-compressed images by the generalized Cauchy distribution (i.e., MBS1) [6]. Thus, the embedded message is adapted to the generalized Cauchy distribution using arithmetic coding. Arithmetic coding transforms unevenly distributed bit streams into shorter, uniform ones. One weak point of the MBS1 is that block artifact increases with growing size of the payload. MBS2 has presented a method to overcome this weakness [7]. The MBS2 embeds message in the same way as MBS1 does, but its embedding capacity is only half of that of MBS1. The other half of the nonzero DCT coefficients is reserved for de-blocking purpose. F5 steganographic method proposed by Westfeld [10], perhaps which is first method of data hiding with less modification. F5 algorithm based on matrix encoding technique, where among seven nonzero AC coefficients only one coefficient was modified to hide three bit of hidden message. The time when F5 algorithm was introduced, F5 was secure against existing steganalysis techniques. Still now, F5 is considering as a good steganographic method. However, F5 method has several limitations, such as F5 algorithm has no freedom to select position for modification (i.e., positions are coming from matrix). F5 is not modifying the LSB in a smart way, as a result number of zeros are increasing. Existing, F5 algorithm does not minimize the distortion. In [5], Provos kept the histogram shape of the DCT coefficients by compensating distortion of modified image (after data hiding). He has divided the set of the DCT coefficients into two disjoint subsets: first subset is used for data hiding, and second subset is used for compensating the modification distortions after data hiding to the first subset. As result, the histogram of the DCT coefficients after data hiding has the same shape as original histogram. Methods presented in [12] and [13] used similar approach.

2.1 F5 Method

F5 steganographic method proposed by Westfeld [10], perhaps which is first method of data hiding with less modification. F5 algorithm based on matrix encoding technique, where among seven nonzero AC coefficients only one coefficient was modified to hide three bit of hidden message. The time when F5 algorithm was introduced, F5 was secure against existing steganalysis techniques. Still now, F5 is considering as a good steganographic method. However, F5 method has several limitations, such as F5 algorithm has no freedom to select position for modification (i.e., positions are coming from matrix). F5 is not modifying the LSB in a smart way, as a result number of zeros are increasing. Existing, F5 algorithm does not minimize the distortion.

Let, the nonzero AC coefficients LSBs are denoted by c_i (where, $i = 1, 2, 3, \cdots, 7$, is a one row and seven column martix), hidden message bits are denoted by b_i (where, $i = 1, \cdots, 3$). As b_i is three bit message which may represent like following way (denoted by H).

$$H = \begin{pmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix}$$

Matrix H is multiplying with the transpose matrix nonzero AC coefficients LSB matrix c_i (see Eq. 1).

$$H * c_i^T \tag{1}$$

Encoding. To get the position to change the coefficient Eq. 1 need to subtract from message bits b_i (see Eq. 2). After getting the information of modified position (p_i), that particular coefficient's LSB will be modified and produce c'_i.

$$p_i = b_i - H * c_i^T \tag{2}$$

Decoding. from the modified coefficients LSB matrix c'_i, decoder can extract hidden message bits b_i by following equation (see Eq. 3).

$$b_i = H * c'_i \tag{3}$$

Example. Suppose, seven nonzero AC coefficients are (5 2 3 1 -2 -5 -1) and corresponding LSB matrix $Coef$ = (1 0 1 1 0 1 1), message bits b_i = (1 0 1). Thus,

$$H * c_i^T = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

and the position of the coefficient is

$$b_i - H * c_i^T = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

(need to modify), From the matrix H, the coefficient position is second (i.e., 2), After modification the coefficients will be (5 1 3 1 -2 -5 -1) and corresponding modified LSB matrix c'_i = (1 1 1 1 0 1 1).

2.2 Model Based Steganographic Method

Model based steganography is general framework for constructing steganographic systems that preserve a chosen *model* of the cover media [6], [7]. Suppose, the cover media is c_i, and c_i is modeled as a random variable which is split in to 2 components (c_{inv}, c_{emb}), where c_{inv} is invariant to embedding and c_{emb} is for modification during embedding.

2.3 OutGuess Method

The OutGuess is popular steganographic algorithm, works as a 2-pass procedure. First pass embeds secret message bits along a pseudo-random walk of the LSBs of the DCT coefficients, skipping coefficients whose magnitude is zero or unity. Second pass, corrections are made to the magnitudes of the coefficients in order to ensure that the histogram of the DCTs of the stego image match those of the cover image.

Prior to embedding, outguess calculates the maximum length of a randomly spread message that can be embedded in the image during the first pass, while ensuring that one will be able to make corrections to adjust the histogram to the original values during the second pass.

2.4 StegHide Method

At first, the secret data is compressed and encrypted. Then a sequence of positions of pixels in the cover file is created based on a pseudo-random number generator initialized with the pass-phrase (the secret data will be embedded in the pixels at these positions). Of these positions those that do not need to be changed (because they already contain the correct value by chance) are sorted out. Then a graph-theoretic matching algorithm finds pairs of positions such that exchanging their values has the effect of embedding the corresponding part of the secret data. If the algorithm cannot find any more such pairs all exchanges are actually performed. The pixels at the remaining positions (the positions that are not part of such a pair) are also modified to contain the embedded data (but this is done by overwriting them, not by exchanging them with other pixels). The fact that (most of) the embedding is done by exchanging pixel values implies that the first-order statistics (i.e. the number of times a color occurs in the picture) is not changed. For audio files the algorithm is the same, except that audio samples are used instead of pixels.

2.5 JP Hide and Seek Method

$JPHide$ and $JPSeek$ are programs which allow you to hide a file in a jpeg visual image. There are lots of versions of similar programs available on the internet but $JPHide$ and $JPSeek$ are rather special. The design objective was not simply to hide a file but rather to do this in such a way that it is impossible to prove that the host file contains a hidden file. Given a typical visual image, a low insertion rate (under 5%) and the absence of the original file, it is not possible to conclude with any worthwhile certainty that the host file contains inserted data. As the insertion percentage increases the statistical nature of the jpeg coefficients differs from "normal" to the extent that it raises suspicion. Above 15% the effects begin to become visible to the naked eye.

3 Proposed Method

The proposed method is so simple, and easy to implement. At first proposed method collected all the nonzero AC coefficients in a single array which is called $coefficient\ stream$ (denoted by c_l). The $coefficient\ stream$ is divided into small coefficient blocks (denoted by $block_i$, Where $l, i = 1, 2, 3 \cdots, n$), and one block is used to represent one bit of hidden message. The number of blocks are same with number of hidden message bits, number of hidden message bits are denoted by α (see Eq. 4).

$$block_i = \lfloor \frac{c_l}{\alpha} \rfloor \tag{4}$$

To hide secret message, this method collected all LSB value from the $block_i$, and made sum of LSBs. The LSB value is checked, whether it is odd or even. A message representation scheme is introduced in way so that odd sum can represent hidden message bit $b_i = 1$, while even sum can represent $b_i = 0$.

When, hidden message requires to modify the LSB sum to represent message bit then only one LSB modification is enough.

$$1001101$$

$$\text{number of 1's are 4}$$

$$4 \bmod 2 = 0$$

$$1101101$$

$$\text{number of 1's are 5}$$

$$5 \bmod 2 = 1$$

As the proposed method has freedom to modify any one LSB from the block, thus this method modified that particular AC coefficient (i.e. LSB) which makes less distortion. To minimize the modification distortion, this method apply a pre-processing before embed the secret data.

3.1 Minimizing the Distortion

After the DCT quantization, all nonzero AC coefficients before the rounding operation collected in a array which is denoted by r_i (where, $i = 1, 2, 3 \cdots, n$), and after the rounding collected in r'_i (where, $i = 1, 2, 3 \cdots, n$) array (see Eq. 5, and Eq. 6)

$$r_i = \text{nonzero AC coefficients before rounding} \tag{5}$$

$$r'_i = \text{nonzero AC coefficients after rounding} \tag{6}$$

As before the rounding operation the DCT values are having floating point numbers, and after the rounding DCT values are becoming DCT coefficients, which means floating point value to integer value. Thus, the rounding operation is causing some error (i.e., rounding error). The rounding errors are denoted by e_i (where, $i = 1, 2, 3 \cdots, n$), and calculated by equation 4 (see Eq. 7),and saved that value in rounding error record array.

$$e_i = r'_i - r_i \tag{7}$$

Later, the rounding error record array used to choose AC coefficient for modification with minimum distortion. Using e_i value, the minimum distortion method [4] modification by following equation.

$$r''_i = \begin{cases} r'_i + 1, if\ e_i > 0,\ and\ r_i \neq -1 \\ r'_i - 1, if\ e_i < 0,\ and\ r_i \neq 1 \\ r'_i + 1, if\ e_i < 0,\ and\ r_i = 1 \\ r'_i - 1, if\ e_i > 0,\ and\ r_i = -1 \end{cases} \tag{8}$$

Now the modification distortion of the chosen group of coefficients (i.e., $block_i$) can be obtain by following equation (see Eq. 9)

$$e'_i = r''_i - r_i \tag{9}$$

and from the above equation, the optimized distortion can be found if the method will follow like (see Eq. 10),

$$d_i = || e'_i | - | e_i || \tag{10}$$

while the least distortion can be found by using the equation given bellow.

$$\min\{d_j \ \forall d_j \ \in \{d_i\}\} \tag{11}$$

Using the minimum distortion obtained from equation (11)in a given $block_i$, this method modified the LSB bit and hides secret data.

3.2 Encoding

During the encoding only nonzero coefficients was considered. A number of nonzero coefficients are representing one bit of secret message. During simulation, a set of random bit was generated by Pseudo Random Generator (PRG). Hidden bits are nothing but 0 and 1.

The encoding scheme is very simple and easy to implement. encoding scheme working as follows:

(1) Make $block_i$ same with hidden number of hidden message bits. Make sum of all LSBs of blocks. If sum is odd then that sum can represent hidden bit 1, and if sum is even then that can represent hidden message bit 0.
(2) Modify one LSB following the less distortion rule (if necessary).

3.3 Decoding

As number of nonzero coefficients are representing one bit of secret message. Thus, during the decoding only the LSB sum of nonzero coefficients was considered. Extracted bits are nothing but 0 and 1 as well (same with encoding).

Decoding scheme is simplest than encoding technique. Decoding scheme working as follows:

(1) Make $block_i$ same with hidden number of hidden message bits. Make sum of all LSBs of blocks. If sum is odd then hidden bit is 1, and if sum is even then hidden message bit is 0.

Example, suppose in a certain hiding capacity an AC coefficients group/block has five nonzero value, such as -0.6994, 0.8534, 0.7352, 1.6229, -2.6861 (before rounding), and -1, 1, 2, 2, -3 (after rounding). So, the rounding errors are -0.3006, 0.1466, 0.2648, 0.3771, -0.3139,and modification can be done by adding as -1, 1, 1, -1, 1. However, the modification will occur errors like, 0.6994, -0.8534, -0.7352, 1.3771, -1.3139. If the proposed method wants to make a choice to make minimum distortion from following distortions record, and then the best choice will be 0.6994 (i.e. changing the first nonzero coefficient value by adding -1).

4 Experimental Results and Comparisons

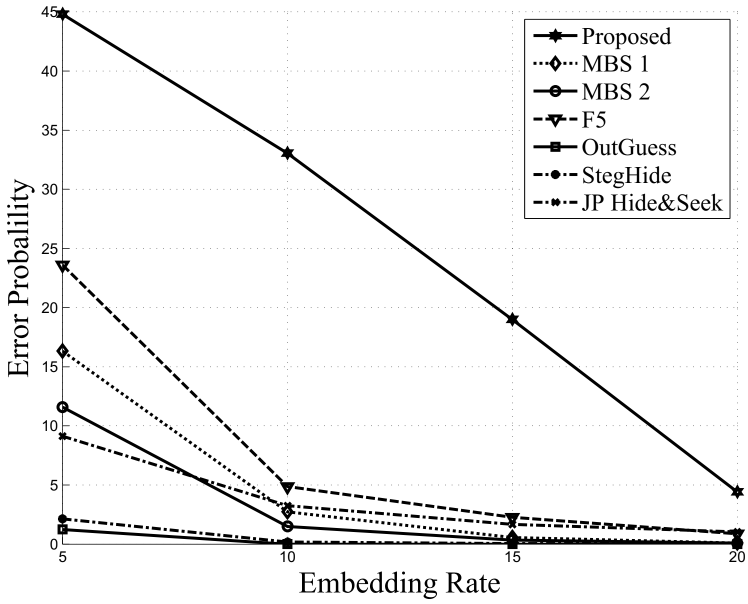
The proposed method tested over 1173 gray scale image and successfully obtained better results (see Table 1, 2). The proposed method and F5 method compared with two different image quality factor (QF). In case of QF = 50, it is found that F5 algorithm has higher Steganalysis error probability than the proposed method.

Table 1. Error Probability (EP) comparisons

Steganalysis by Error Probability (EP)					
		5%	10%	15%	20%
QF = 50	Proposed	44.80	33.04	18.99	4.42
	F5 method	23.80	4.59	2.0443	0.51
	MBS1	16.31	2.73	0.55	0.09
	MBS2	11.58	1.49	0.34	0.09
	OutGuess	1.24	0.00	0.00	0.00
	StegHide	2.13	0.17	0.04	0.04
	JP Hide&Seek	9.11	3.24	1.66	1.02
QF = 75	Proposed	44.97	33.30	17.46	3.83
	F5 method	18.39	2.12	0.68	0.25
	MBS1	11.46	1.96	0.47	0.09
	MBS2	8.73	1.11	0.26	0.04
	OutGuess	0.51	0.04	0.00	0.00
	StegHide	1.79	0.13	0.00	0.00
	JP Hide&Seek	10.31	4.81	2.43	1.45

Table 2. Mean Distortion (MD) comparisons

Steganalysis by Mean Distortion (MD)					
		5%	10%	15%	20%
QF = 50	Proposed	120.80	460.60	903.70	1,701.00
	F5 method	818.13	2,132.90	3,201.53	4,780.75
	MBS1	1,461.58	2,924.14	4,385.47	5,846.26
	MBS2	2,032.42	4,044.73	6,034.74	8,006.70
	OutGuess	2,270.50	4,558.26	6,849.15	9,129.43
	StegHide	1,846.62	3,416.75	4,979.28	6,549.80
	JP Hide&Seek	1,023.23	2,050.42	3,066.44	4,073.56
QF = 75	Proposed	167.20	637.60	1,252.00	2,347.00
	F5 method	1,137.03	2,977.09	4,465.49	6,709.05
	MBS1	2,196.04	4,391.61	6,586.95	8,785.02
	MBS2	3,004.81	5,990.65	8,949.79	11,889.28
	OutGuess	3,281.20	6,578.67	9,867.18	13,149.89
	StegHide	2,612.12	4,950.25	7,284.26	9,626.10
	JP Hide&Seek	1,737.09	3,475.43	5,202.58	6,929.67



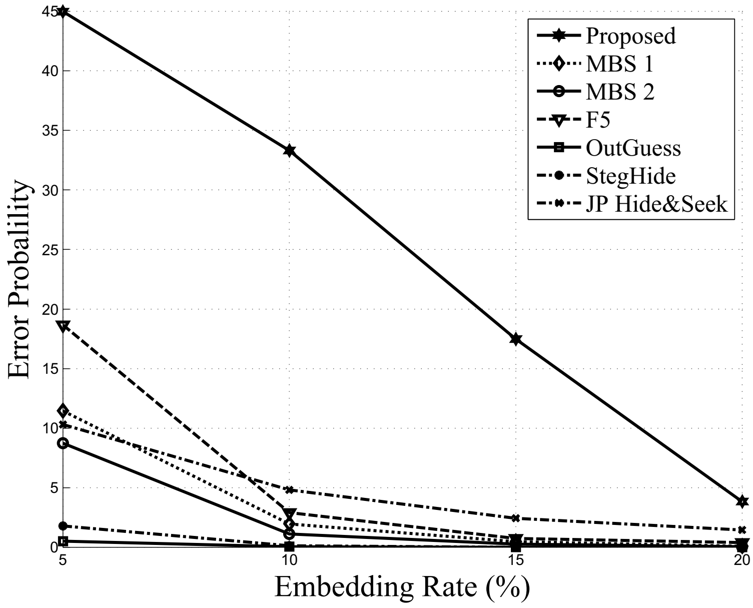
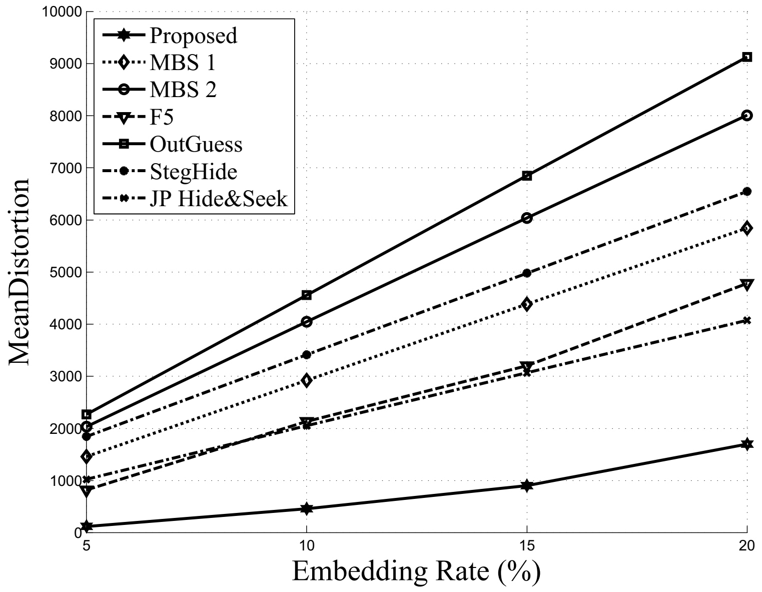


Fig. 1. Steganalysis comparison by support vector machine (SVM) of the proposed method with other existing algorithms (Top: with quality factor 75, and Bottom: with quality factor 80)



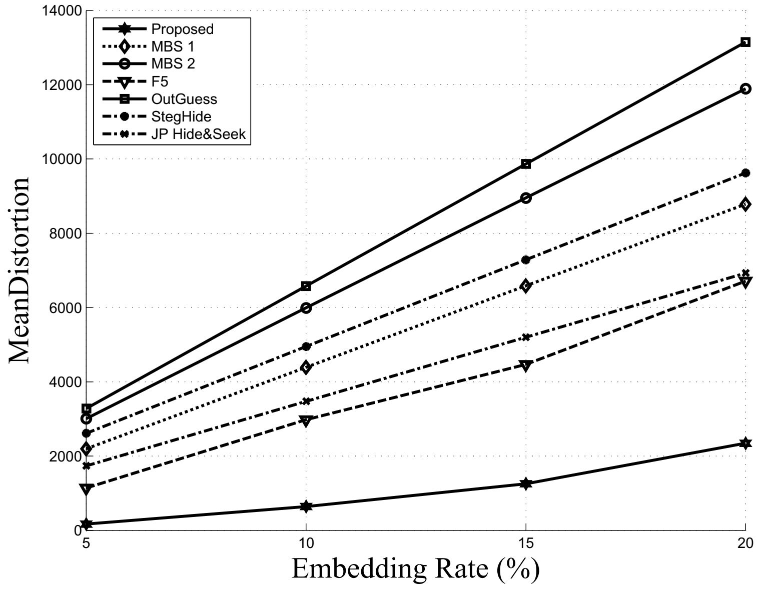


Fig. 2. Distortion comparison by Mean distortion (MD) Vs Embedding rate (ER) of the proposed method with other existing algorithms (Top: with quality factor 75, and Bottom: with quality factor 80)

4.1 Experimental Results

With 5% data hiding capacity F5 algorithm has 23.085 Steganalysis error probability, while the proposed method has 44.8041 (for the best situation the error probability is 50%). With 10% data hiding capacity and with QF = 50, F5 algorithm has 4.5997 Steganalysis error probability, while the proposed method has 33.0494. With 15% data hiding capacity and with QF = 50, F5 algorithm has 2.0443 Steganalysis error probability, while the proposed method has 18.9949. Again, with 20% data hiding capacity and with QF = 50, F5 algorithm has 0.5111 Steganalysis error probability; while the proposed method has 4.4293 (see Table 1, 2). Similarly, with 5% data hiding capacity and QF = 75, F5 algorithm has 18.3986 Steganalysis error probability, while the proposed method has 44.9744. With 10% data hiding capacity and with QF = 75, F5 algorithm has 2.1995 Steganalysis error probability, while the proposed method has 33.3049. With 15% data hiding capacity and with QF = 50 F5 algorithm has 0.6814 Steganalysis error probability, while the proposed method has 17.4617. Again, with 20% data hiding capacity and with QF = 50 F5 algorithm has 0.2555 Steganalysis error probability; while the proposed method has 3.8330 (see Table 1, 2).

4.2 Comparisons

The proposed method and F5, was tested with support vector machine to detect Steganalysis probability, the following comparison are prepared after getting the Steganalysis detection result (see Fig 1 and Fig 2).

During the performance testing, the error probability and embedding rate was considered with QF = 50, and QF = 75. With both QF, the proposed method has achieved better performance than F5 and proposed method (see Fig 1).

During the performance testing, the mean distortion and embedding rate was considered with QF = 50, and QF = 75. With both QF, the proposed method has achieved better performance than F5 and proposed method (see Fig 2).

5 Conclusions

The proposed method has better resistance against steganalysis than popular steganographic method F5. In case of steganography attacks are more important than capacity, while this method has better hiding capacity also. The main advantage of this proposed method is freedom of modifying any coefficients. Resulting better quality of stego image and higher resistance against attacks. The F5 [10], reduced the modification or flip not the modification distortion, while the proposed method reduced the modification distortion. Proposed method did not increased number of zeros, while F5 method did. As a future work less modification or less flipping can be considered.

References

1. Fridrich, J., Goljan, M., Hogea, H.: Attacking the Out-Guess. In: Proc. of the ACM Workshop on Multimedia and Security, pp. 967–982 (2002)

2. Fridrich, J., Goljan, M., Hogea, H.: Steganalysis of JPEG image: Breaking the F5 algorithm. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 310–323. Springer, Heidelberg (2003)
3. Fridrich, J.: Feature-based steganalysis for JPEG images and its implications for future design of steganographic schemes. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 67–81. Springer, Heidelberg (2004)
4. Pevny, T., Fridrich, J.: Merging Markov and DCT Features for Multi-Class JPEG Steganalysis. In: Proc. of SPIE Electronic Imaging, Photonics West, pp. 03–04 (2007)
5. Provos, N.: Defending against Statistical Steganalysis. In: Proc. of the 10th USENIX Security Symposium, pp. 323–335 (2001)
6. Sallee, P.: Model-based steganography. In: Kalker, T., Cox, I., Ro, Y.M. (eds.) IWDW 2003. LNCS, vol. 2939, pp. 154–167. Springer, Heidelberg (2004)
7. Sallee, P.: Model-based methods for steganography and steganalysis. International Journal of Image and Graphics 5(1), 167–190 (2005)
8. Solanki, K., Sarkar, A., Manjunath, B.S.: YASS: Yet another steganographic scheme that resists blind steganalysis. In: Furon, T., Cayre, F., Doërr, G., Bas, P. (eds.) IH 2007. LNCS, vol. 4567, pp. 16–31. Springer, Heidelberg (2008)
9. Westfeld, A., Pfitzmann, A.: Attacks on steganographic systems. In: Pfitzmann, A. (ed.) IH 1999. LNCS, vol. 1768, pp. 61–75. Springer, Heidelberg (2000)
10. Westfeld, A.: F5: A steganographic algorithm: High capacity despite better steganalysis. In: Moskowitz, I.S. (ed.) IH 2001. LNCS, vol. 2137, pp. 289–302. Springer, Heidelberg (2001)
11. StegHide Steganographic Method, `http://www.stegui.sourceforge.net/steghide.html`
12. Eggers, J., Bauml, R., Girod, B.: A communications approach to steganography. In: SPIE, Electronic Imaging, Security, and Watermarking of Multimedia Contents, San Jose, CA (2002)
13. Noda, H., Niimi, M., Kawaguchi, E.: Application of QIM with dead zone for histogram preserving JPEG steeganography. In: Processing ICIP, Genova, Italy (2005)

Appendix

If in a particular block non zero AC Coefficients are (-1, 1, 2, 2, 3), their LSBs are (1, 1, 0, 0, 1). So, the sum of all LSBs is 3, which is a odd number. Now, if the hidden bit is 0 then the sum value need to be modify. To modify the sum value (i.e., odd to even) the proposed method modifies on LSB. As in the given example, it is already defined that first LSB modification by adding -1 has less distortion thus the method modifies according to that (see Fig 3, and Fig 4). The modified AC coefficients are now, (-2, 1, 2, 2, -3), and corresponding LSBs are (0, 1, 0, 0, 1) (sum value of LSBs is 2). As, the proposed method has less modification distortion, thus the embedded images are having almost same quality like original JPEG images (see Fig 5).

-1	1	2	2	-3
1	1	0	0	1

Fig. 3. Before modification

-2	1	2	2	-3
0	1	0	0	1

Fig. 4. After modification

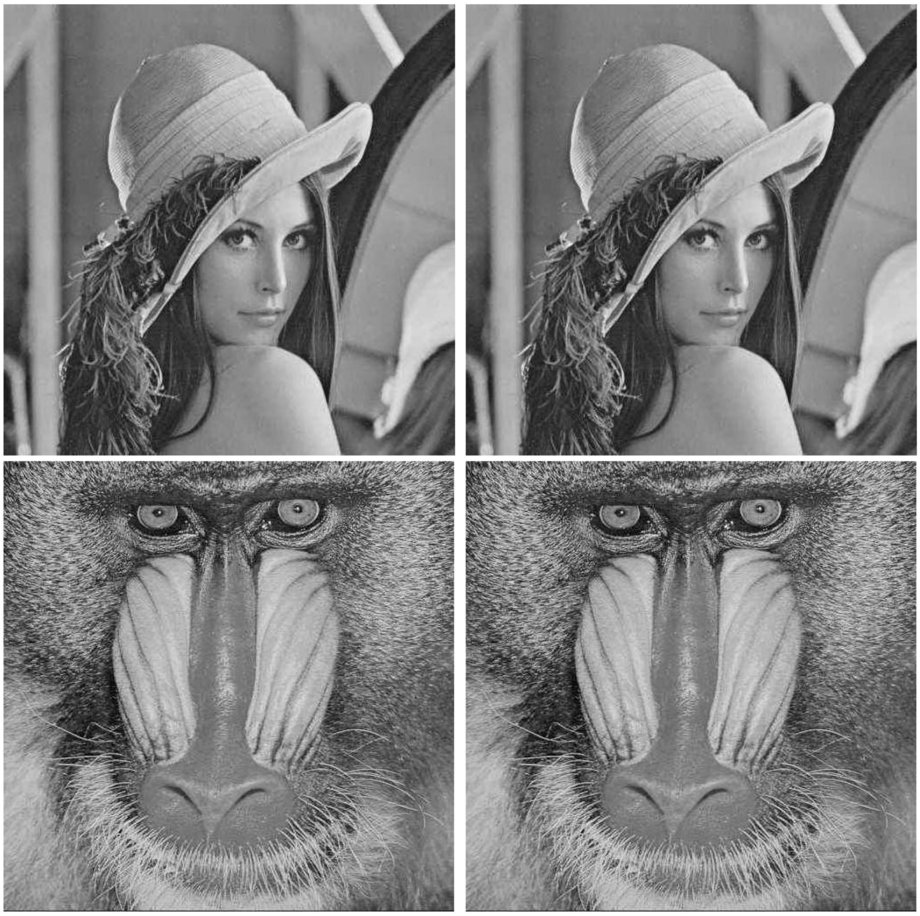


Fig. 5. Top: Original Lena (left) Modified Lena (right) image (after data embedding by the proposed method, with 5% of total embedding capacity). Bottom: Original Baboon (left) Modified Baboon (right) image (after data embedding by the proposed method, with 5% of total embedding capacity).

Model-Based Higher-Order Mutation Analysis

Fevzi Belli[1], Nevin Güler[2], Axel Hollmann[1], Gökhan Suna[3], and Esra Yildiz[3]

[1] University of Paderborn, Department of Electrical Engineering and Information Technology, Paderborn, Germany
[2] University of Mugla, Department of Statistics, Mugla, Turkey
[3] Izmir Institute of Technology, Department of Computer Engineering, Izmir, Turkey
{belli,ngueler,hollmann}@adt.upb.de,
gokhansuna44@hotmail.com, esrayildiz@gmail.com

Abstract. Mutation analysis is widely used as an implementation-oriented method for software testing and test adequacy assessment. It is based on creating different versions of the software by seeding faults into its source code and constructing test cases to reveal these changes. However, in case that source code of software is not available, mutation analysis is not applicable. In such cases, the approach introduced in this paper suggests the alternative use of a model of the software under test. The objectives of this approach are (i) introduction of a new technique for first-order and higher-order mutation analysis using two basic mutation operators on graph-based models, (ii) comparison of the fault detection ability of first-order and higher-order mutants, and (iii) validity assessment of the coupling effect.

Keywords: Software Testing, Higher-Order Mutation Analysis, Coupling Effect, Basic Mutation Operators, Event Sequence Graphs.

1 Introduction and Related Work

One of the major goals of software testing is to reveal defects/bugs by comparing the observed behavior of the software with its expected behavior. There are many approaches used in software testing. Mutation analysis (MA), originally proposed by DeMillo et al. [4] and Hamlet [7], is a method for analyzing the adequacy of a test set to reveal faults [17], [20]. It bases on the idea of seeding artificial defects (mutations) into the system under test (SUT).

MA has two main steps. In the first step, faults are injected into the software by applying mutation operators to the original software. The mutation operators cause syntactic changes and represent certain fault models. Each of the faulty versions of the software generated is called a “mutant”. In the second step, a test set is executed against the mutants. A mutant is considered "killed" if the expected output predicted by the test set differs from the actual output of the mutant. Otherwise, the mutant is “live”. In that case, the fault(s) could not be revealed or the mutant is equivalent to the original version. If all mutants are killed by a test set, the test set is considered to be adequate in detecting the faults that were injected into the mutants.

MA relies on two assumptions: the *competent programmer hypothesis* and the *coupling effect* [4], [15]. The competent programmer hypothesis claims that experienced programmers tend to implement software that is almost correct. The coupling effect assumes that a test set that detects all simple faults defined by first-order mutants (FOM) is also able to detect complex faults defined by higher-order mutants (HOM) [4]. The coupling effect has been studied by many researchers. Offutt ([15], [16]) described empirical investigations about the coupling effect over a specific class of software faults and showed that the coupling effect hypothesis is valid. Wah [19] presented a theoretical study of fault coupling, based on a simple model of fault-based testing. He shows that fault coupling occurs infrequently, at least when only two faults are involved. In another study, Wah [18] dealt with the coupling effect from a theoretical standpoint and he found out that the hypothesis of a coupling effect is largely valid.

Although many studies have shown the validity of the coupling effect, Harman et al. [8] argued that HOMs are potentially better able to simulate real faults. However, higher-order mutation analysis has been believed to be expensive and impractical. Therefore, they argue that higher-order mutation analysis can be available using a process that searches fitter mutants from the space of all possible mutants. Jia et al. ([10], [11]) also investigated HOMs. They introduced the concept of a subsuming HOMs that denote subtle fault combinations. The objective of that study is to seek valuable HOMs using automated search-based optimization techniques.

MA can be very expensive since there is a great number of different mutation operators for generating mutants. Therefore, another important issue of MA is the selection of appropriate mutation operators. Several studies have been carried out to find a smaller set of mutation operators without significant loss of test effectiveness. This idea was proposed by Mathur [12]. In his study, two mutation operators, array reference for scalar variable replacement (ASR) and scalar variable replacement (SVR) that generated most of the mutants are omitted to reduce the number of generated mutants. This idea was extended by omitting four mutation operators (4-selective mutation) and omitting six mutation operators (6-selective mutation). Offutt et al. [14] have shown that selective mutation is an effective alternative to non-selective mutation by presenting empirical results that support the hypothesis. Mresa and Bottaci [13] proposed a different selective mutation based on assigning a score to each mutation operator.

In this paper we suggest reducing the broad variety of mutation operators known from literature to two basic operators and their combination. MA is usually applied to the source code of SUT. However, its source code is not always available or easily modifiable in case of hardware. To overcome this problem, this paper applies mutation analysis to a model that describes the relevant aspects of SUT. Belli et al. [1] introduced event sequence graphs (ESGs) for modeling the behavior of an SUT. ESGs consist of vertices (nodes) representing externally observable phenomena ("events") and directed edges (arcs) defining allowed sequences among events. For our approach, an ESG model is mutated by using mutation operators such as insertion and omission of directed edges or vertices. Thus, fault models of SUT are obtained. For each mutant a model-based test case generation algorithm is used to generate a test set. Concluding, SUT is executed against each test set and the mutants are classified as killed or live.

This paper focuses on an investigation of FOMs and HOMs based on mutants generated from ESG and comparison of higher-order mutation operators with basic mutation operators. The primary objective of this paper is to answer the following questions:

- *Do test sets that detect all FOMs detect most of the HOMs, i.e., does the coupling effect hold?*
- *Are basic mutation operators sufficient for measuring test effectiveness of higher-order mutation analysis?*

Now, the question might arise why ESG is used for modeling, and not UML diagrams. Formally speaking, ESG is directed graph enriched by elementary semantics to represent sequential processes, having the same representative power as finite-state automata, which are equivalent to type-3 (regular) grammars [1], [3]. The formalism used in ESG notation enables the direct adoption of mathematical results known from graph theory, automata theory, etc., as needed by our approach.

The rest of this paper is organized as follows: Section 2 presents modeling with ESG. Section 3 introduces basic mutation operators for ESG and a process for ESG-based mutation analysis. A case study in Section 4 validates the approach using a large commercial web-based system. Section 5 concludes the paper summarizing the results and outlines future work.

2 Modeling with Event Sequence Graphs

An event is an externally observable phenomenon, such as an environmental or a user stimulus, or a system response punctuating different stages of system activity. ESGs [1] represent system behavior and user-system interaction focusing on events.

Definition 1. An *event sequence graph* $ESG=(V, E, \Xi, \Gamma)$ is a directed graph with V as a finite set of vertices (events), $E \subseteq V \times V$ a finite set of *arcs*, and $\Xi, \Gamma \subseteq V$ as finite sets of distinguished vertices called *entry* events and *exit* events.

The semantics of an ESG is as follows: Any $v \in V$ represents an event. For two events $v,v' \in V$, the event v' must be enabled after the execution of v if $(v, v') \in E$. $\Xi(ESG)$, $\Gamma(ESG)$ represent the *entry* and *exit* events of a given ESG. To mark the entry and exit events, every $\xi \in \Xi$ is preceded by a pseudo event $[\notin V$ and every $\gamma \in \Gamma$ is followed by $] \notin V$. For each $v \in V$ there is at least one sequence of vertices $(\xi, v_1, \ldots, v_m)$ from $\xi \in \Xi$ to $v_m = v$ and one sequence of vertices $(v_1, \ldots, v_m, \gamma)$ from $v_1 = v$ to $\gamma \in \Gamma$ with $(v_i, v_{i+1}) \in E$, for $i=1,\ldots,m-1$ and $v \neq \xi$ and $v \neq \gamma$. Fig. 1 shows an example of an ESG with $V=\{a,b,c\}$, $E=\{(a,b), (a,c), (b,c), (c,b)\}$, $\Xi=\{a\}$, and $\Gamma=\{b\}$.

Definition 2. Let $ESG=(V, E, \Xi, \Gamma)$ be an ESG. Any sequence of vertices $(v_1,\ldots,v_m)$ is called an *event sequence* (ES) if $(v_i,v_{i+1}) \in E$, for $i=1,\ldots,m-1$.

Let α and ω be the functions to determine the entry vertex and exit vertex of an ES. For example, given $ES=(v_1,\ldots,v_m)$, the entry and exit vertices are $\alpha(\text{ES})=v_1$ and $\omega(ES)=v_m$, respectively. Note that the pseudo vertices [,] are not included in the *ES*. An $ES = (v_i, v_j)$ of length 2 is called an *event pair* (EP).

The function *l* (*length*) determines the number of vertices of an ES. In particular, if *l*(*ES*)=1 then it is an event sequence of length 1. The pseudo vertices [and] are not included in any ES. Neither are they considered to determine the entry and exit vertices, or to compute the length of an ES.

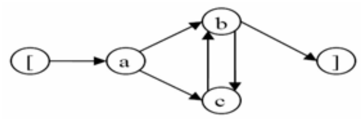


Fig. 1. Example of simple event sequence graph

Definition 3. An ES is a *complete event sequence* (*CES*) if $\alpha(ES) \in \Xi$ and $\omega(ES) \in \Gamma$.

Each CES represents a *walk* from an entry of the ESG to an exit realized by the form: (initial) user inputs→ (interim) system responses → … → (final) system response. Based on the notion of CES a test case generation algorithm Φ is described by Algorithm 1. Applying algorithm Φ to an ESG (denoted by Φ(*ESG*)) delivers a test set *T* that is an ordered pair of (*input to SUT*, *expected output of SUT*) to be executed against SUT. Details of an algorithm as well as minimization of test sets have been published in previous work ([1]).

Algorithm 1. Test case generation Φ

```
Input:   $ESG = (V, E, \Xi, \Gamma)$,  $len$ := maximum length of CES to be covered
Output:  Test report of succeeded and failed test cases
FOR $i$ := 1 TO $len$
BEGIN
    Cover all ESs of $ESG$ of length $i$ by means of CESs;
END
Apply the test cases given by CESs to SUT; Observe output of SUT;
```

3 ESG-Based Mutation Analysis

Based on ESG this section defines basic ([2]) and higher-order mutation operators to generate faulty models (mutants) and defines a mutation analysis process using ESG and these operators.

3.1 Basic Mutation Operators

A given ESG specifies the expected, desirable behavior of SUT. An ESG can be changed by manipulating either the arcs or the events resulting in a faulty model ESG*. There are two basic mutation operators that can be defined for different model elements – *insertion* (*I*) and *omission* (*O*).

Definition 4. Basic Mutation operators.

- An *arc insertion operator* (*aI*) extends an ESG by inserting a new arc $\alpha \notin E$ into the given ESG: $aI(ESG, \alpha) := (V, E \cup \{\alpha\}, \Xi, \Gamma)$.

- An *arc omission operator* (*aO*) removes an arc $\alpha \in E$ from the given ESG: $aO(ESG, \alpha) := (V, E \backslash \{\alpha\}, \Xi, \Gamma)$.
- An *event insertion operator* (*eI*) extends an ESG by inserting an event *e* into the given ESG: $eI(ESG, e) := (V \cup \{e\}, E, \Xi, \Gamma)$.
- An *event omission operator* (*eO*) removes an event $e \in E$ from the given ESG: $eO(ESG, e) := (V \backslash \{e\}, E, \Xi, \Gamma)$.

The *eI*-operator requires adding extra arcs to/from the inserted event from/to other nodes. After applying an *eO*-operator, adjacent arcs of event *e* have to be removed.

The set of first-order basic mutation operators is thereby given by $\Delta_1 := \{aI, aO, eI, eO\}$. Higher-order mutants are constructed as follows:

Definition 5. The set of all *n*-order ($n>1$) mutation operators are given by $\Delta_n := \Delta_1^n$

Traditional mutation operators as defined in the literature (e.g. [5],[6]) can be directly represented by the basic operators or as a combination. As an example, the operators manipulating the events of an FSM as defined in [6]) can be represented as follows: "event-missing" by basic operator $eO \in \Delta_1$, "event-extra" by basic operator $eI \in \Delta_1$, and "event-exchanged" as a second-order operator $(eO, eI) \in \Delta_2$. The direction of an arc of an ESG is changed by a second-order operator $cdA := (aO, aI)$.

3.2 ESG-Based Mutation Analysis

It is assumed that test cases generated by test case algorithm Φ (Algorithm 1) based on ESG do no reveal any more faults in SUT and that the source code of SUT is not available. Therefore, mutants are constructed based on the model and not the system itself. The goal is to evaluate the fault detection effectiveness of algorithm Φ and its test cases generated, i.e. its capability to distinguish the mutants from SUT. Algorithm 2 describes an ESG-based mutation analysis.

Algorithm 2. ESG-based mutation analysis

```
Input:  ESG that describes SUT,  $\Delta \subseteq \Delta_1 \cup \ldots \cup \Delta_n$ (set of mutation operators)
        $k$ := maximum number of mutants to be generated by each operator
        $\Phi$ (test generation algorithm)
Output: Killed and live mutants
FOREACH $\delta \in \Delta$
BEGIN
  FOR $i$ := 1 TO $k$
  BEGIN
    $ESG^*_{\delta,i} := \delta(ESG)$;
    $T^*_{\delta,i} := \Phi(ESG^*_{\delta,i})$;
    Execute SUT against $T^*_{\delta,i}$;
    Compare expected output contained in $T^*_{\delta,i}$ with actual output of SUT;
  END
END
```

An ESG representing SUT and a set of mutation operators $\Delta \subseteq \Delta_1 \cup \ldots \cup \Delta_n$ is used to generate up to k mutants for each mutation operator $\delta \in \Delta$. For each mutant $ESG^*_{\delta,i}$ test case generation algorithm Φ is used to generate a test set $T^*_{\delta,i}$. SUT is executed against this set to compare the predicted output contained in $T^*_{\delta,i}$ with the output observed by SUT. If the output differs, the mutant $ESG^*_{\delta,i}$ is killed. Otherwise, the mutant remains live or is equivalent to the original model.

Definition 6. Given an ESG, a set of mutation operators Δ, the number of mutants k, and a test case generation algorithm Φ, the *test generation mutation score* is defined as: $TGMS(ESG, \Delta, k, \Phi)$:= No. of killed mutants / (all mutants – no. of equivalent mutants).

4 Case Study

To analyze the practicability, characteristic features, and limitations of our approach, an experimental case study based on a commercial web-based system has been conducted. Our main focus is to answer the research questions from the introduction.

4.1 Modeling, Test Generation and Execution

SUT is a commercial web portal (ISELTA – Isik‘s System for Enterprise-Level Web-Centric Tourist Applications [9]) for marketing touristic services. ISELTA enables hotels and travel agencies to create individual search masks. These masks are embedded in existing homepages of hotels as an interface between customers and the system. Visitors of the website can then use these masks to select and book services, e.g., hotel rooms or special offers. The system also provides other search mask to book arrangements. This part of the system will be used within the case study. Arrangements part of system consists of two parts: additional services and special offers. To set up an additional service and special offer, the dedicated website as shown in Fig. 1 and Fig. 2, respectively, are used. Examples of ESG model for additional services and special offer are given Fig. 3 and Fig. 4, respectively.

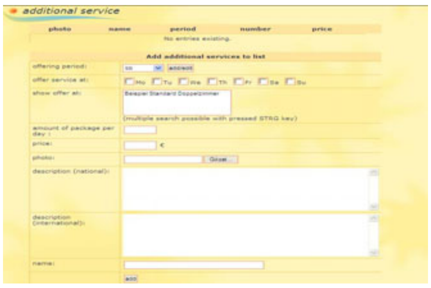

additional service

photo	name	period	number	price
No entries existing.				

Add additional services to list

offering period: [illegible] select
offer service at: Mo Tu We Th Fr Sa Su
show offer at: Beispiel Standard Doppelzimmer
(multiple search possible with pressed STRG key)
amount of package per day :
price: €
photo: Görse...
description (national):
description (international):
name:
add

Fig. 1. Website to set up additional service

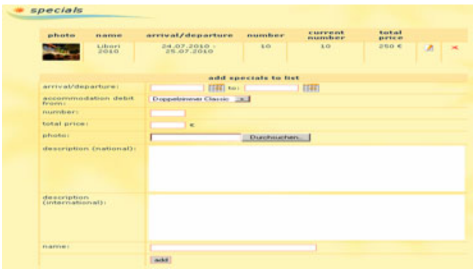

specials

photo	name	arrival/departure	number	current number	total price		
	Libori 2010	24.07.2010 - 25.07.2010	10	10	250 €		

add specials to list

arrival/departure: to:
accommodation debit from: Doppelzimmer Classic
number:
total price: €
photo: Durchsuchen...
description (national):
description (international):
name:
add

Fig. 2. Website to set up special offer

Test cases were generated for each mutant using the algorithm Φ (as described in Algorithm 1) for *len* = 2, that is, to cover all events and pairs of events by CES. Equivalent mutants were not observed in the case study.

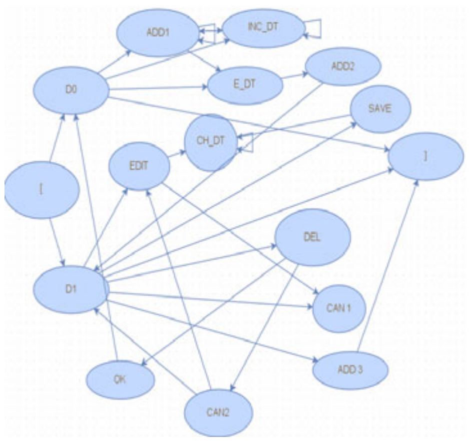


Fig. 3. Example of ESG for additional service

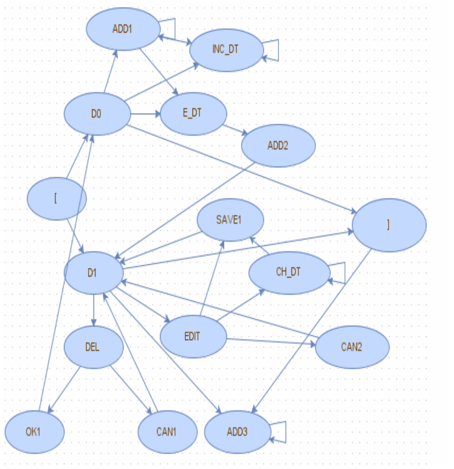


Fig. 4. Example of ESG for special offer

D0 : No Special **D1** : in a Special **INC_DT** :Incomplete Data **EDIT:** Click Edit
CH_DT : Change Information **E_DT** : Enter Complete Data
SAVE : Click Save **CAN** :Click Cancel **OK** : Click OK **ADD** : Click ADD

4.2 Results and Discussion

Table 1, Table 2 and Table 3 summarize the results of mutation analysis. 50 of first-order mutants, 108 of second-order mutants, and 83 of third-order mutants were generated for analysis.

Table 1. Results of First-Order Mutation Analysis

Mutation Operator/ Mutant Type	Killed	Alive	TGMS
All Mutation Op.	**17**	**33**	**0,34**
eO	0	16	0
aI	17	2	0,89
aO	0	15	0

To check the validity of the coupling effect, means of TGMS with respect to the order of the mutants was compared by using One Way ANOVA test. It is a statistical test to determine whether there are differences between different levels of an independent variable or not. In this case study, order of mutants (first-, second-, and third-order) is used as the independent variable.

<u>Coupling Effect Hypothesis</u> (Confidence level is 95%)

H_0: Tests that find first-order mutants also find higher-order mutants
H_1: Higher-order mutation is better in revealing the mutants.

Table 2. Results of Second-Order Mutation Analysis

Mutation Operator / Mutant Type	Killed	Alive	TGMS
All Mutation Op.	**50**	**58**	**0,46**
eO, eO	4	10	0,29
eO, aI	19	3	0,86
aO, aI	16	5	0,76
cdA	9	12	0,43
aI, aI	0	2	0
aO, aO	0	11	0
eO, eO	2	15	0,12

Table 3. Results of Third-Order Mutation Analysis

Mutation Operator / Mutant Type	Killed	Alive	TGMS
All Mutation Op.	**67**	**16**	**0,81**
cdA, aO	12	2	0,86
cdA, aI	12	1	0,92
aO, aI, aI	14	4	0,78
cdA, eO	11	9	0,55
aI, aI, eO	4	0	1
aI, aI, eO	14	0	1

Table 4 shows the results of One Way ANOVA. According to Table 4, H_0 hypothesis can not be accepted at 95% confidence level, since sig. value (0.03) is less than 0.05. This states that there is significant difference between first-order mutation analysis and higher-order mutation. In other words, coupling effect does not hold at 95% confidence for this case study.

Table 4. One Way ANOVA

	Sum of Squares	Mean Square	F	Sig.
Between Groups	1,007	0,504	4,647	**0,03**
Within Groups	1,409	0,108		
Total	2,416			

H_0: Tests that find first-order mutants also find second-order mutants
H_0: Tests that find first-order mutants also find third-order mutants
H_0: Tests that find second-order mutants also find third-order mutants
H_1: There is significant difference between test adequacies.

Table 5. Multiple Comparison Test

(I) MO (J) (MO)	Mean Difference (I-J)	Std. Error	Sig.
First-Order - Second Order	-0,055	0,2272	0,813
- Third Order	**-0,555****	0,233	**0,033**
Second-Order -Third-Order	**-0,5002***	0,183	**0,017**

If H_0 hypothesis can not be accepted as a result of one way ANOVA, the question arises “Which means of groups are significantly different from which other means of groups”. To answer the question, Multiple Comparison Tests are used. These tests compare all possible pairs of means of groups and produce which pairs significantly different under selected confidence level. According to Table 5, significant difference between first-order mutation analysis and third-order mutation analysis was found since sig. value (0,033) is less than 0.05. Similarly, it can be said that there is significant difference between second-order mutation analysis and third-order mutation analysis with respect to TGMS (Sig = 0.017 <0.05). Lastly, to decide whether basic mutation operators are sufficient for measuring test effectiveness of higher-order mutation analysis or not, mean of TGMS values are compared with 1, since the test set is said to be effective in detecting faults that that were injected into the mutants, if its TGMS value is 1. To carry out the comparison, One-Sample T-Test that is a statistical test technique used to compare the mean of a sample to a known value was performed. Table 6 shows the results of the comparison. As a result of One-Sample T-Test (Table 6), significant difference between TGMS value of higher-order mutation analysis and 1 was not found.

Table 6. One Sample T-Test

	T	Df	Sig. (2-tailed)	Mean Difference
TGMS	-2,134	5	0,086	-0,1483

5 Conclusions, Future Research

This paper introduced higher-order ESG-based mutation analysis by using two basic operators, insertion and omission, and checked the validity of coupling effect. Analysis was carried out on a case study using the web portal ISELTA. Test sets for first-order, second-order, and third-order mutants were generated and executed on SUT to determine the number of killed and live mutants. Statistical “One Way ANOVA” method enabled following:

- Means of TGMS values were compared to check validity of coupling effect.
- There are significant differences between means of first, second, and third-order mutants were found at 95% confidence level.
- Coupling effect could not be confirmed for this case study. However, it cannot be excluded that the results are affected by potential bias between the operators chosen. The coupling effect may be confirmed when using other higher-order mutation operators or all possible combinations of basic operators as higher-order operators.

In a second step, Multiple Comparison test (LSD) was performed to answer the question “which means of groups (order of mutants) are significantly different from which other means of groups”. Founding is:

- No significant difference between first-order and second-order mutation analysis.
- Significant difference between first/second-order and third order mutation analysis.

The additionally performed One-Sample T-Test concluded that higher-order mutation analysis carried out by using combinations of basic mutation operators in this study are adequate in detecting the injected faults. Work planned is to combine and iterate basic operators for creating more sophisticated mutants to be used in further empirical research.

References

[1] Belli, F., Budnik, C.J., White, L.: Event-based Modeling, Analysis and Testing of User Interactions: Approach and Case Study. Journal of Software Testing, Verification and Reliability 16(1), 3–32 (2006)

[2] Belli, F., Budnik, C.J., Wong, W.E.: Basic Operations for Generating Behavioral Mutants. In: Proc. 2nd Workshop on Mutation Analysis (2006)

[3] Belli, F.: Finite-State Testing and Analysis of Graphical User Interfaces. In: Proc. 12th IEEE International Symposium on Software Reliability Engineering, pp. 34–43 (2001)

[4] DeMillo, R.A., Lipton, R.J., Sayward, F.G.: Hints on Test Data Selection: Help for the Practicing Programmer. Computer 11(4) (1978)

[5] Fabbri, S.C.P.F., Maldonado, J.C., Delamaro, M.E., Masiero, P.C.: Mutation Analysis Testing for Finite-State Machines. In: Proc. 5th IEEE International Symposium on Software Reliability Engineering, pp. 220–229 (1994)

[6] Fabbri, S.C.P.F., Maldonado, J.C., Sugeta, T., Masiero, P.C.: Mutation Testing Applied to Validate Specifications Based on Statecharts. In: Proc. 10th IEEE International Symposium on Software Reliability Engineering, pp. 210–219 (1999)

[7] Hamlet, R.G.: Testing Programs with the Aid of a Compiler. IEEE Transactions on Software Engineering 3(4) (1977)

[8] Harman, M., Jia, Y., Langdon, W.B.: A Manifesto for Higher Order Mutation Testing. In: Proc. 5th International Workshop on Mutation Analysis (2010)

[9] ISELTA, `http://www.iselta.de/`

[10] Jia, Y., Harman, M.: Constructing Subtle Faults Using Higher Order Mutation Testing. In: SCAM, pp. 249–258 (2009)

[11] Jia, Y., Harman, M.: Higher Order Mutation Testing. Journal of Information and software Technology 51(10), 1379–1393 (2009)

[12] Mathur, A.P.: Performance, Effectiveness and Reliability Issues in Software Testing. In: Proc. of 5th International Computer Software and Applications Conference, Tokyo, Japan, pp. 604–605 (1991)

[13] Mresa, E.S., Bottaci, L.: Efficiency of Mutation Operators and Selective Mutation Strategies: An Empirical Study. Software Testing, Verification and Reliability 9(4), 205–232 (1999)

[14] Offutt, A.J., Rothermel, G., Zapf, C.: An Experimental Evaluation of Selective Mutation. In: Proc. 15th International Conference on Software Engineering (ICSE 1993), pp. 100–107. IEEE Computer Society Pres., Baltimore (1993)

[15] Offutt, A.J.: Investigations of the Software Testing Coupling Effect. ACM Transactions on Software Engineering and Methodology 1(1), 5–20 (1992)

[16] Offutt, A.J.: The Coupling Effect: Fact or Fiction? In: Proc. 3rd Symposium on Software Testing, Analysis and Verification, Key West, FL, pp. 131–140 (1989)

[17] Schuler, D., Dallmeier, V., Zeller, A.: Efficient Mutation Testing by Checking Invariant Violations. In: Proc. 2009 International Symposium on Software Testing and Analysis, Chicago, pp. 69–80 (2009)

[18] Wah, K.S.H.T.: An Analysis of the Coupling Effect I: Single Test Data. Science of Computer Programming 48(2-3), 119–161 (2003)

[19] Wah, K.S.H.T.: A Theoretical Study of Fault Coupling. Software Testing, Verification & Reliability 10(1), 3–45 (2000)

[20] Zhu, H., Hall, P.A.V., May, J.H.R.: Software Unit Test Coverage and Adequacy. ACM Computing Surveys 29(4), 366–427 (1997)

ISARE: An Integrated Software Architecture Reuse and Evaluation Framework

Rizwan Ahmad[1], Saif ur Rehman Khan[2], Aamer Nadeem[3], and Tai-hoon Kim[4]

[1] Department of Computer Science
Shaheed Zulifqar Ali Bhutto Institute of Science and Technology, Islamabad
ch_rizwanahmad@yahoo.com
[2] Department of Computer Science
Comsats Institute of Information Technology, Islamabad
saif_rehman@comsats.edu.pk
[3] Center for Software Dependability
Mohammad Ali Jinnah University (MAJU), Islamabad
anadeem@jinnah.edu.pk
[4] Dept. of Multimedia, Hannam University
133, Ojeong-dong, Daedeok-gu, Daejeon, Korea
taihoonn@hannam.ac.kr

Abstract. Quality is an important consideration in the development of today's large complex software systems. Software architecture and quality play a vital role in the success or failure of any software system. Similarly to maintain the qualities of a software system during development and to adapt the quality attributes as the software requirements changes, software architecture is necessary. This paper discusses software quality attributes and the support provided by software architecture to achieve the desired quality. A novel *Software Architecture Reuse and Evaluation* framework is proposed on the basis of existing software architecture evaluation methods with respect to quality requirements. A case study is used for experimental validation of the ISARE. The results show that ISARE ensures the required level of quality requirements in the software architecture and automatable.

Keywords: Software quality, software architecture, quality of service.

1 Introduction

Software Architecture (SA) being a core of the software system plays a vital role in the success or failure of any software system as it deals with the base structure, sub-systems and interactions among these sub-systems [1]. Software systems exhibit the global properties derived from these architectural styles. Basically it serves as a roadmap which guides the quality development of a software system.

SA provides the ground to foresee the end product's quality. Since it falls in the early phases of *Software Development Life Cycle* (SDLC) so if it is ensured that the SA which to be implemented satisfies the desired quality attributes then the quality of the end product would be high. More formal efforts are concentrated on ensuring that the quality is addressed at the architectural level [2]. Different researchers have discussed SA from different viewpoints. Bass et al. [3] discuss the importance of quality attributes, how to specify required quality attributes and how these can be applied to the design and evaluation of software architectures.

In this paper, we propose a novel framework called *Integrated Software Architecture Reuse & Evaluation* (ISARE) *Framework* that ensures higher level of Quality of Service (QoS) required by the stakeholders. This framework integrates the strengths of some existing SA evaluation methods and quality models. In order to evaluate the ISARE, we use the *Shifa International Hospital Management System* (SIHMS) specification [26]. The results show that the proposed framework ensures the required quality attributes in the SA and is automatable.

The rest of this paper is organized as follows: Section 2 presents a thorough related literature review: section 3 discusses proposed ISARE framework: section 4 evaluates the proposed framework using a SIHMS case study: while conclusion and future work are presented in section 5.

2 Literature Review

This section discusses the basic terms, existing SA methods and their analysis.

2.1 Quality Attributes (QA)

Non-functional characteristics of a component or a system are called quality attributes. Software quality is defined in IEEE 1061 as "*the degree to which software possesses a desired combination of attributes*" [4]. It is very important to differentiate between the *Software Quality* and *Quality of SA*. Quality of Software is derived from the SA whereas the quality of SA is explicitly required to be measured. Piattini et al. [5] conclude that most of the research has focused on *Software Quality* but there is much research work needed on the quality of SA.

2.2 Software Architecture (SA)

All software systems have SA that describes the fundamental organization of the system [4]. Designing SA to achieve required QA is one of the most demanding tasks [7]. The SA captures early design decisions and reflects major quality concerns, including functionality [10]. It recognizes functional requirements of a system [8] and is an important aspect of producing high quality software systems [9].

Bass et al. [10] define SA of a system as "*the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them*". SA must ensure the level of quality to be delivered to the user, referred to as QoS [8]. Ladan et al. [14] propose quality-driven software re-engineering process to support QoS.

2.3 Software Architecture with Respect to QoS

To construct software architectural structure for a system that fulfils the desired QA's is often a challenging task. In addition to functional requirements, QA's also affect the selection of appropriate SA. An appropriate architecture is not only governed by functional requirements but, to a large extent, by QA's [15], [16], [17]. There are usually more than one QA involved in a system and the knowledge of the pros and cons of different architectural structures with respect to different QA's is not mature enough [15].

In [15] Svahnberg et al. propose a decision support method that provides aid in the understanding of different architecture structures and to choose best-suited architecture to meet the quality requirements of the software system from a set of candidate SA structures. The proposed method enables a quantified understanding of different architecture candidates for a software system. The set of candidate SA structures and required QA's are the main inputs of the process. The output of the process is an appropriate SA structure that fulfils the required QA's and uncertainty indicator.

2.2 Overview of Analysis Methods

This section provides an overview of different existing SA methods.

Scenario-Based Architecture Analysis Method (SAAM). SAAM appeared in 1993 [19] with the trend for a better understanding of general architectural concepts, as a foundation for proof that a software system meets more than just functional requirements [11], [18]. The goal of this method is to verify basic architectural assumptions and principles against the documents describing the desired properties of an application [11]. The strength of this method is to concentrate on any QA in the form of scenarios. The main inputs of SAAM are problem description, requirements statement and architecture description(s).

SAAM Founded on Complex Scenarios (SAAMCS). SAAMCS is an extended version of SAAM that is directed to the way of looking for the scenarios and, to where their impact is evaluated. The important goal of SAAMCS is risk assessment. It considers that the complexity of scenarios is the most important factor for risk assessment [11], [20].

Extending SAAM by Integration in the Domain (ESAAMI). ESAAMI is a combination of analytical and reuse concepts and is achieved by integrating the SAAM in the domain-specific and reuse-based development process [11], [20]. The SAAM is the evaluation technique; the quality attributes and SA description are same in both SAAM and ESAAMI. However, ESAAMI adds the reuse of domain knowledge defined by SA's and analysis templates.

Software Architecture Analysis Method for Evolution and Reusability (SAAMER). SAAMER is an extended version of SAAM that supports two particular QAs. i.e., (i) evolution, and (ii) reusability [11], [22]. It provides a better support of how a system could restrain each of the quality objectives or the risk levels for evolution or how to reuse it [11]. It provides a framework of activities for architecture analysis process. This framework consists of gathering information about stakeholders, SA, quality, and scenarios, modeling usable artifacts, analysis, and evaluation activities [11].

Architecture Trade-Off Analysis Method (ATAM). ATAM is used to for architectural analysis of individual quality attributes. It was considered as a spiral model of designs [23]. The goal of ATAM is to provide a principal way of understanding SA capability with respect to multiple competing QA's [24].

Scenario-Based Architecture Re-engineering (SBAR). SBAR not only supports architecture design but also scenario-based evaluation of the software qualities of detailed architecture of a system [11], [25]. The goal of SBAR is to estimate the potential of the designed architecture to reach the software quality requirements [11].

Architecture Level Prediction of Software Maintenance (ALPSM). The ALPSM analyzes maintainability of software system by looking at the impact of scenarios at the SA level [11]. The inputs of the method are the requirements statement, the description of the architecture, expertise from software engineers and possibly historical maintenance data. It consists of six steps: (i) identification of categories of maintenance tasks, (ii) synthesis scenarios, (iii) assignment of a weight to each scenario, (iv) estimation of the size of all elements, (v) scripting the scenarios, and (vi) calculation of the predicted maintenance effort.

A Software Architecture Evaluation Model (SAEM). The goal of SAEM is to establish the basis for the SA quality evaluation and prediction of the final system quality [11]. SA quality requirements evaluation process is rigorously formalized, especially in relation to metrics in the model described in [12]. The elements required for quality evaluation process of a software system requires, standard specification, quality model, a method for evaluation, metrics, and the supporting tools.

Table 1 provides a comparison of existing SA methods. For analysis and comparison purpose, we identify following analysis parameters: (i) Support for Reuse of Existing Knowledge, (ii) Support for Scenarios, and (iii) Support for Multiple Attributes. From the table, it is concluded that only ESAAMI and ATAM support all of the identified analysis parameters.

Table 1. Existing Software Architecture Methods

Software Architecture Evaluation Methods	Support for Reuse of Existing Knowledge	Support for Scenarios	Support for Multiple Attributes
SAAM	No	Yes	Yes
SAAMCS	No	Yes	Yes
ESAAMI	Yes	Yes	Yes
SAAMER	No	Yes	No
ATAM	Yes	Yes	Yes
SBAR	No	Yes	Yes
ALPSM	No	Yes	No
SAEM	No	No	Yes

3 Proposed Framework

ISARE is a novel framework to ensure the higher level of QoS. It is a rationale approach to integrate the quality aspects of software development into the SA development phase in order to minimize the quality related risks involved throughout the process. ISARE also provides a mechanism to utilize the benefits of existing techniques developed to ensure the quality aspects in a very simple way with very limited resources. Key features of ISARE are highlighted below:

- It determines the quality concerns of different stakeholders, right from the beginning of the process and ensures proper mapping of those concerns in the selected SA.
- It reuses the available architecture structures/styles for selection of the candidate SA from its repository on the basis of QA requirements.
- It supports the decision making process for selection of better software architecture with respect to the prioritized QA requirements.
- It provides a recursive SA improvement process (during the architecture evaluation process) to ensure the right end product. It also provides an integrated mechanism to reuse the main part of existing techniques available for its sub- processes.
- It utilizes an ever-growing repository of reusable SA structures/styles, descriptions, analysis results with respect to QA requirements and the decisions made in the past.
- It also incorporates the quality attributes requirements at all phases of the framework including SA *development*, *evolution*, *selection* and *evaluation* to make sure that selected SA meets the quality goals.

Figure 1 shows a high-level architecture of ISARE. There are two main components of ISARE Framework; (i) Reuser and (ii) Evaluator. Each component performs different tasks to support overall framework objectives. The ISARE framework is designed to support the higher level of QoS through SA design and evaluation.

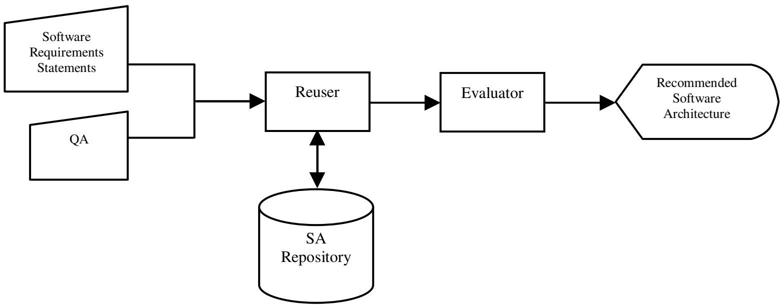

Software Requirements Statements
QA
Reuser
Evaluator
Recommended Software Architecture
SA Repository

Fig. 1. High Level Architecture of ISARE Framework

Figure 2 depicts a detailed architecture of each component including the sequence of information flow in the form of a number followed by an alphabet (i.e., 2b means

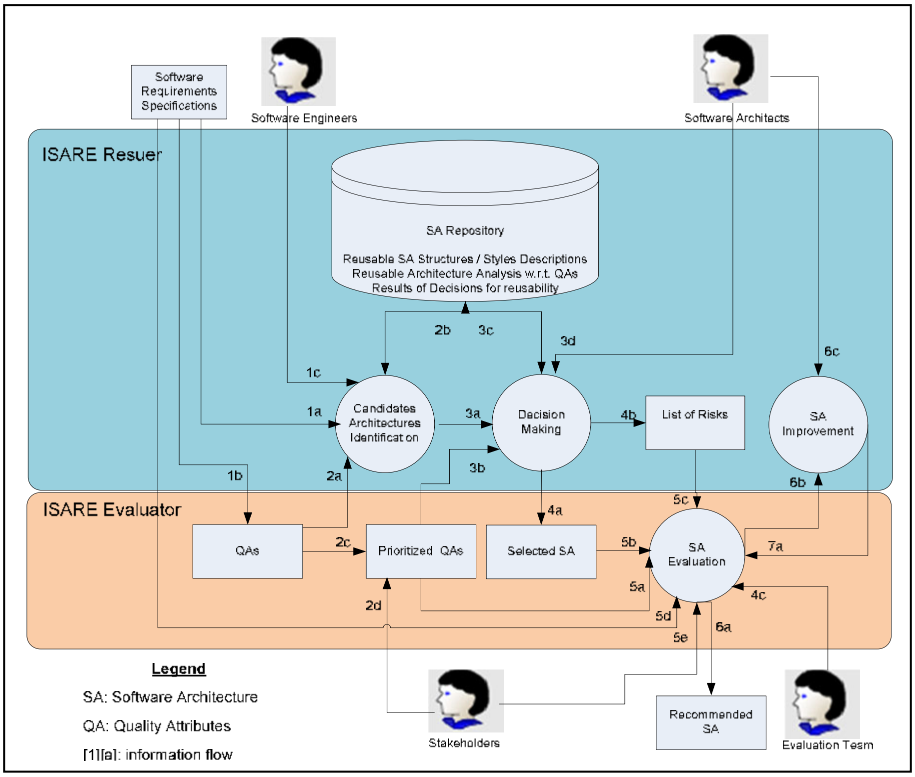


Fig. 2. Detailed Architecture of ISARE Framework

that control will be transferred here when all of its descendant activities; like 1a, 1b etc, have been performed). Now in following subsection we will discuss the basic ISARE components in detail:

3.1 ISARE Reuser

This component facilitates the complete process of SA from *Requirements Specification Document* to *SA Selection*. It utilizes the SA repository to reuse the available SA structures/styles, architecture descriptions, their analysis results with respect to, previously obtained quality attributes for the selection of appropriate SA. Basically, it provides a platform to fully utilize the available knowledge according to the situation and provides a mechanism for software engineers and software architects to incorporate brainstorming during the decision making process to analyze the candidate architectures for final SA selection. Now we will discuss the elements of this component in detail:

ISARE Reuser Human Element. Human element is an important aspect of this framework. Roles of software engineer and software architect are described below:

(i) *Software Architects*: they are the authoritative decision makers for analyzing candidate architectures against prioritized quality attributes, requirements and risks involved with those architectures. They are responsible for deciding which candidate SA is appropriate and fulfills the maximum level of QoS requirements.

(ii) *Software Engineers*: they play a very critical role during candidate SA selection process. These candidate SA's are the base for whole process till the final SA selection. They also analyze quality requirements from the given software requirements specifications.

ISARE Reuser Processes. The backbone of the automation will be the "*Software Architecture Repository*" which will contain all existing software architecture styles/patterns and their corresponding support for quality attributes. This Repository will contain Qualitative and Quantitative results of software architecture styles. The first process of our ISARE Framework "*Candidate Architectures Identification*" provides the results based on qualitative approaches and if we get multiple candidate architectures, "*Decision Making*" process will be applied on these identified candidate architectures which provides us with the list of prioritized candidate architectures according to the ratio of their support to meet required level of quality based on quantitative approaches as discussed in Decision Making Process. Since SA Repository is the core of ISARE framework, so it needs to be developed very carefully. Once SA Repository is developed, it can provide 100% automation for the ISARE Reuse Component.

ISARE Reuser consists of three major processes as discussed below:

(i) *Candidate Identification*: This process is responsible for extracting the reusable SA's from the available SA's available in the repository on the basis of available quality requirements. It takes two inputs: (a) functional requirements and (b) software quality attributes. This process then co-ordinates with the SA repository for available candidate architecture according to the list of QA's provided. The output of the process is a set of candidate architectures extracted from SA repository.

(ii) *Decision Making Process*: It analyzes the candidate architectures provided as an input. It determines the suitability of those architectures on the basis of prioritized QA's. It takes two inputs: (a) candidate architectures and (b) prioritized quality attributes. Finally, it suggests the suitable SA along with the uncertainty factor involved in those candidate architectures with respect to prioritized QA's. Since there is a possibility to have multiple candidate architectures, so we need an exact level of support provided by those candidate architectures against prioritized set of QA's. Steps performed in this process are taken from [15] to integrate the strength of quantitative measurement of SA candidates. The main steps are to (a) determine the uncertainty in the identified candidate architecture, and (b) identify SA structure. Decision process then provides the suitable SA with the lowest uncertainty factor in respect of QoS requirements and results will be stored in *SA Repository* for future reuse purpose.

(iii) *SA Improvement Process*: reviews the un-recommended SA from the ISARE *Evaluator* component. It provides a mechanism to enhance the SA in order to fulfill the desired level of QoS requirements.

ISARE Reuser Software Architecture Repository. This is the core of ISARE *Reuser* component which takes the benefit of available SA structures and provide ease in selecting candidate architectures on the basis of quality requirements. Since quality requirements are the basic drivers of the SA so it takes the quality requirements as an input and provides available SA structures/styles accordingly. It also maintains the descriptions of existing SA styles and their relevance with QA requirements.

3.2 ISARE Evaluator

The main task of ISARE Evaluator component is to facilitate the complete process of SA selection and evaluation in accordance with the QoS requirements. It provides a comprehensive mechanism to ensure that QA's are properly *documented*, *prioritized*, *communicated* and *incorporated* in selected SA. It provides a platform to thoroughly analyze the selected architectures against stakeholder(s) quality needs and authorize the evaluation team to reject the SA until all the scenarios are properly mapped on selected architecture.

ISARE Evaluator does not need to evaluate the selected architecture with existing methods because we have already included it in *Decision Process*, so now it just depends upon the scenario *brainstorming*, *prioritization*, *refinement* and *mapping* onto the selected architectures. It takes necessary input of the stakeholder(s) to define scenarios for the evaluation of selected software architecture with respect to prioritized QA's. The following subsection discusses the main elements of this component:

ISARE Evaluator Human Element. Human element plays a vital role in this framework because scenarios are required to be developed on the basis of prioritized set of QA's. Stakeholders are helpful in generating those scenarios. Roles of stakeholders and evaluation team are described below:

(i) *Stakeholders*: provide the quality requirements and help in assigning the priorities to those quality requirements. Stakeholders also actively participate in scenario brainstorming and prioritization with the close liaison with SA evaluation team to ensure that their quality needs are properly incorporated.
(ii) *Evaluation Team*: has the most responsible and sensitive role. They need to evaluate the selected SA against stakeholder's quality needs and ensure that they are fulfilled in selected architecture. Otherwise they can reject the architecture until it is enhanced to meet the desired level of quality.

ISARE Evaluator Processes. Software Architecture Evaluation Process is based on scenarios and scenarios are based on quality requirements which vary between different systems. So it is not possible to provide 100% automation. But as we have repository available, so we can also store basic quality scenarios which are generic in nature and use those scenarios for evaluation process to some extent.

ISARE Evaluator consists of only one process as discussed below:

(i) *Software Architecture Evaluation:* is the only process in ISARE Evaluator which ensures that selected SA should incorporate the quality requirements specified by stakeholders according to the priority.

Evaluation team is responsible for the desired level of quality in selected SA. It takes three inputs: (a) selected SA, (b) prioritized QA's and (c) list of related risks. The output of the process is a decision about recommendation of selected architecture or SA improvement to incorporate the required QoS requirements in that architecture.

The scenario brainstorming is performed with the help of stakeholders to identify the major scenarios of the system according to the prioritized quality requirements. This activity provides a list of scenarios. After scenario development, stakeholders prioritize these scenarios according to their requirements and then refine those scenarios. Finally scenarios are mapped on selected SA to ensure the higher level of QoS.

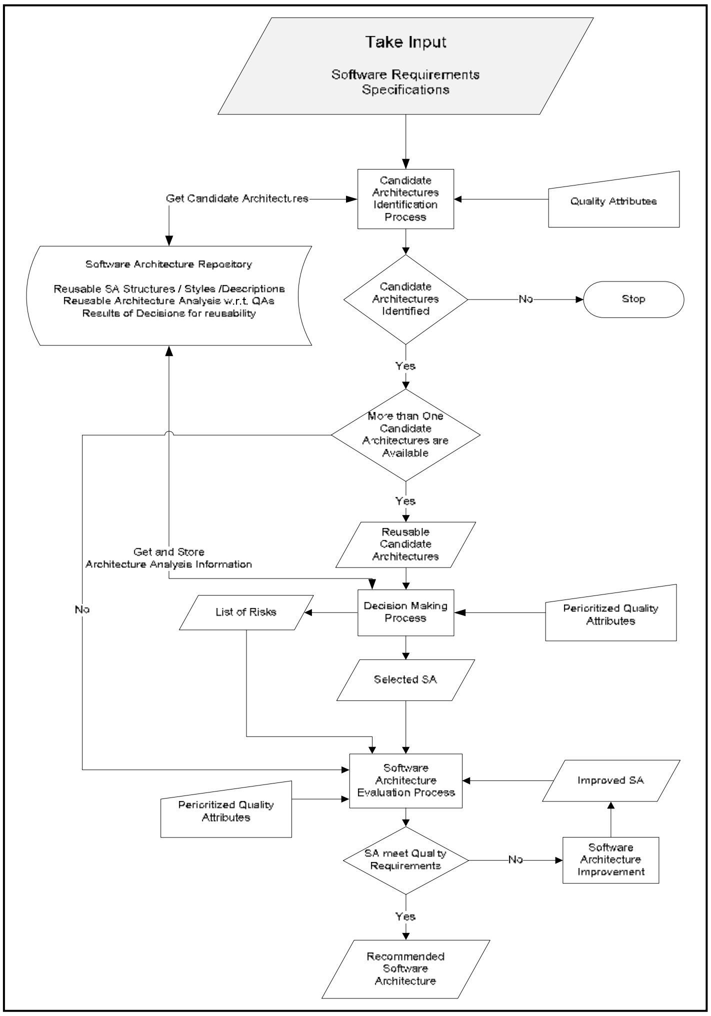

Take Input
Software Requirements Specifications
Candidate Architectures Identification Process
Get Candidate Architectures
Quality Attributes
Software Architecture Repository
Reusable SA Structures / Styles /Descriptions
Reusable Architecture Analysis w.r.t. QAs
Results of Decisions for reusability
Candidate Architectures Identified
No
Stop
Yes
More than One Candidate Architectures are Available
Yes
Reusable Candidate Architectures
Get and Store Architecture Analysis Information
No
List of Risks
Decision Making Process
Perioritized Quality Attributes
Selected SA
Software Architecture Evaluation Process
Improved SA
Perioritized Quality Attributes
SA meet Quality Requirements
No
Software Architecture Improvement
Yes
Recommended Software Architecture

Fig. 3. ISARE Workflow

3.3 Working of ISARE

Main input of the ISARE framework is the software requirements document provided by stakeholders. In addition, requirement statements provide the list of QA's according to the stakeholders needs. This document is the basis for selection of appropriate candidate SA's with respect to quality requirements from available SA styles in the SA repository. On the basis of these candidate architectures, best suitable SA is chosen according to prioritized QA's by applying a decision process. The prioritization of QA's is performed with the help of stakeholders. Final selection of SA for a given system also considers the existing reusable architectures.

After selection of SA, it is evaluated to ensure that required QA's are properly incorporated in the selected SA. If yes, then the selected architecture is recommended. Otherwise, it is forward for enhancement till the desired level of quality is achieved. Figure 3 depicts the workflow of ISARE framework.

4 Evaluating ISARE Framework

We have evaluated ISARE Framework using a case study. We used SIHMS with its detailed Software Requirements Specifications (SRS) including functional and non-functional requirements [26]. These requirements will be given as an input to ISARE Framework and SA will be selected as an output. Below is the evaluation process applied on SIHMS:

4.1 Brief Description of SIHMS

The aim of SIHMS is to automate all the functions of Shifa International Hospital and to provide best facilities to its stakeholders. It will replace the current manual methods followed for routine tasks i.e., data collection, compilation, storage and reporting, etc.

4.2 User Classes and Characteristics

We have identified following four user classes:

(i) *Directors:* will be able to manage/monitor the hospital workings, facilities, doctors and all staff. SIHMS will help the directors to make efficient decisions on the basis of accurate results and reports.

(ii) *Doctors*: will be in the field practically for using the system. They will execute their routine tasks for example patient treatment, history, diagnosis, medication, and prescriptions, and update al this information into the system. This information will be available to the all concerns on a single click.

(iii) *Staff*: of the Shifa International Hospital will use this system mostly. They will work for payments, appointments, patient services, and inventory, etc.

(iv) *Patients*: will use this system to avail the facilities like online appointments, and it will help the people a lot in saving their time.

4.3 Quality Requirements

SIHMS incorporate the following system attributes:

- *Availability*
- *Scalability*
- *Reliability*
- *Modifiability*
- *Reusability*
- *Security*
- *Testability*
- *Extendibility*

4.4 ISARE Evaluation for SIHMS

The functional and non-functional requirements of SIHMS are provided as an input to ISARE framework. Since the QA's are the basic drivers of SA so ISARE *Reuser* and ISARE *Evaluator* components will coordinate to provide the candidate SA according to the information available in the SA Repository.

On the basis of available quality criteria, "*Identify Candidate Architectures*" process of the ISARE Reuser component extracts the candidate architecture from SA Repository according the information stored in SA Repository in the form of Table 2. (Similarly, other architecture may be selected based on the QA's prioritization as suggested by the stakeholders.)

Table 2. SA Repository Information

	Analysis Parameters								
Software Architecture Style	Testability	Scalability	Reliability	Modifiability	Security	Portability	Availability	Reusability	Performance
Layers	+	-	+	+	+	+	+	+	+
Client-Server	-	+	-	-	+	-	-	+	+
Pipe and Filter	+	-	-	+	-	-	-	+	-
Object Oriented	-	+	-	+	-	-	-	+	+
Event Systems	-	-	-	+	-	-	-	+	-
Blackboard Repository	-	+	-	+	-	-	-	+	-
Main Program and Subroutine	-	+	-	+	-	-	-	-	+
Implicit Invocation	-	-	-	+	+	+	+	+	-

Finally using quality requirements of SIHMS and available candidate architecture, the SA Repository provide only one candidate architecture named "Layers".

According to the working of ISARE, if there is only one candidate architecture available on the given set of requirements, it does not forwarded for the decision process for the quantitative evaluation. Instead it is considered as "*Selected Software Architecture*" and forwarded for the ISARE Evaluator component for "*Software Architecture Evaluation*" process as discussed in Section 3.2. As a result, ISARE Reuser component will stop its processing on selection of "Layers" architecture style and ISARE Evaluator will start to evaluate the selected SA style against the prioritized set of quality attributes.

5 Conclusion and Future Work

We have presented a framework on the basis of a survey on quality attributes related aspects of SA. We have also focused on the reuse of available, proven techniques in efforts to maximize the results. The case study results show that ISARE ensures the required level of quality requirements in the SA and is automatable.

The research should generally aim at extending and reusing the existing proven SA evaluation techniques. Researchers should investigate and reuse the strengths of those important areas which have great impact on overall software quality concerns.

Another research direction is to strengthen the SA repository and automate the software architecture selection and evaluation methods for efficient and reliable results. The main input to this repository is the set of architectural styles/artifacts and their comprehensive analysis against the set of quality attributes. Utilizing existing quantitative approaches for SA analysis could be the best possible option to move further in the direction of reusing existing information.

Acknowledgement

This work was supported by the Security Engineering Research Center, granted by the Korean Ministry of Knowledge Economy.

References

[1] Ionita, M.T., Hammer, D.K., Obbink, H.: Scenario-based Software Architecture Evaluation Methods: An Overview. In: Workshop on Methods and Techniques for Software Architecture Review and Assessment at the International Conference on Software Engineering, ICSE (2002)

[2] Dobrica, L., Niemelä, E.: A Survey on Software Architecture. IEEE Transactions on Software Engineering 28(7) (2002)

[3] Bass, L.: Principles for Designing Software Architecture to Achieve Quality Attribute Requirements. In: Proceedings of the 4th International Conference on Software Engineering Research, Management and Applications, SERA 2006 (2006)

[4] IEEE Standard 1061-1992, Standard for Software Quality Metrics Methodology, Institute of Electrical and Electronics Engineers, New York (1992)

[5] Piattini, M., Calero, C., Astudillo, H.: Classifying Software Architecture Quality Research. In: Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA 2005), pp. 195–196. IEEE Press, New York (2005)

[6] Bachmann, F., Bass, L., Klein, M., Shelton, C.: Designing Software Architectures to Achieve Quality Attribute Requirements. In: IEEE Proceedings of Software, vol. 152(4). IEEE Press, New York (2005)

[7] Bruin, H., Vliet, H.: Quality Driven Software Architecture Composition. Journal of Systems and Software, Special Issue: Software Architecture-Engineering Quality Attributes, 269–284 (2003)

[8] Rakic, M.M., Malek, S., Medvidovic, N.: Architecture Driven Software Mobility in Support of QoS Requirements. In: Proceedings of the 1st International Workshop on Software Architectures and Mobility (SAM 2008), Leipzig, Germany, pp. 195–196 (2008)

[9] Schougaard, K.R., Hansen, K.M., Christensen, H.B.: SA@Work - A Field Study of Software Architecture and Software Quality at Work. In: Proceedings of the 15th Asia-Pacific Software Engineering Conference (APSEC 2008), pp. 411–418 (2008)

[10] Kazman, R., Kruchten, P., Nord, R., Tomayko, J.E.: Integrating Software-Architecture Centric Methods into the Rational Unified Process. Technical Report, CMU/SEI-2004-TR-011 (2004), `http://www.sei.cmu.edu/library/abstracts/reports/04tr011.cfm`

[11] Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley Publishing Co., Reading (2003) ISBN-10: 0-321-68039-1

[12] Duenas, J.C., Oliveira, W.L., Puente, J.A.: A Software Architecture Evaluation Model. In: van der Linden, F.J. (ed.) Development and Evolution of Software Architectures for Product Families. LNCS, vol. 1429, pp. 148–157. Springer, Heidelberg (1998)

[13] Bengtsson, P.O., Bosch, J.: Architecture Level Prediction of Software Maintenance. In: Proceedings of the 3rd European Conference on Software Maintenance and Reengineering, pp. 139–147 (1999)

[14] Tehvidari, L., Kontogiannis, K., Mylopoulos, J.: Quality Driven Software Re-engineering. The Journal of Systems and Software, 225–239 (2003)

[15] Svahnberg, M., Wohlin, C., Lundberg, L., Mattsson, M.: A Method for Understanding Quality Attributes in Software Architecture Structures. In: Proceedings of the Software Engineering and Knowledge Engineering (SEKE 2002), Ischia, Italy, pp. 819–826 (2002)

[16] Bosch, J.: Design & Use of Software Architecture– Adopting and Evolving a Product Line Approach. Addison-Wesley, Harlow (2000) ISBN-10: 0201674947

[17] Hofmeister, C., Nord, R., Soni, D.: Applied Software Architecture. Addison-Wesley, Reading (2000)

[18] Kazman, R., Abowd, G., Bass, L., Clements, P.: Scenario-Based Analysis of Software Architecture. IEEE Software 13(6), 47–55 (1996)

[19] Kazman, R., Abowd, G., Bass, L., Webb, M.: Analyzing the Properties of User Interface Software Architectures. Technical report, CMU-CS-93-201, Carnegie Mellon University, School of Computer Science (1993)

[20] Lassing, N., Rijsenbrij, D., Vliet, H.: On Software Architecture Analysis of Flexibility, Complexity of Changes: Size isn't Everything. In: Proceedings of the 2nd Nordic Software Architecture Workshop (NOSA 1999), pp. 1103–1581 (1999)

[21] Molter, G.: Integrating SAAM in Domain-Centric and Reuse-Based Development Processes. In: Proceedings of the 2nd Nordic Workshop Software Architecture (NOSA 1999), pp. 1103–1581 (1999)

[22] Lung, C., Bot, S., Kalaichelvan, K., Kazman, R.: An Approach to Software Architecture Analysis for Evolution and Reusability. In: Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research, Canada, pp. 1–11 (1997)
[23] Kazman, R., Klein, M., Barbacci, M., Lipson, H., Longstaff, T., CarrieÁre, S.J.: The Architecture Tradeoff Analysis Method. Technical report, CMU/SEI-98-TR-008, ESC-TR-98-008 (1998), http://www.pst.ifi.lmu.de/lehre/WS0102/architektur/VL9/ATAM.pdf
[24] Barbacci, M., Carriere, S., Feiler, P., Kazman, R., Klein, M., Lipson, H., Longstaff, T., Weinstock, C.: Steps in an Architecture Tradeoff Analysis Method: Quality Attribute Models and Analysis. Technical report, CMU/SEI-97-TR-029 ESC-TR-97-029 (1998)
[25] Bengtsson, P.O., Bosch, J.: Scenario-Based Software Architecture Re-engineering. In: Proceedings of the 5th International Conference on Software Reuse (ICSR 5), pp. 308–317 (1998)
[26] http://www.shifa.com.pk

Cognitive Informatics for New Classes of Economic and Financial Information Systems

Lidia Ogiela[1] and Marek R. Ogiela[2]

AGH University of Science and Technology
[1] Faculty of Management
[2] Institute of Automatics
al. Mickiewicza 30, PL-30-059 Krakow, Poland
{logiela,mogiela}@agh.edu.pl

Abstract. This publication discusses intelligent systems for cognitive data categorisation with a particular emphasis on image analysis systems used to analyse economic data. This type of systems used to interpret, analyse and reason work following the operating principles of cognitive information system. Cognitive systems interpret complex data by extracting semantic levels from it, which they use to determine the meaning of the data analysed, to cognitively understand it, as well as to reason and forecast changes in the area of the phenomena researched. Thus the course of human processes of learning about the described phenomenon becomes the foundation for developing automatic cognitive systems which are called cognitive data analysis systems.

Keywords: Cognitive informatics, cognitive processes, cognitive information systems, UBMSS systems (*Understanding Based Managing Support Systems*).

1 Introduction

Cognitive systems are currently developed very rapidly, and the operating algorithms applied in such systems and illustrating their processes of data analysis and interpretation increasingly frequently use the semantic layers contained in data/information sets as well as techniques of linguistic data description. Until recently, processes of this type were based on classical cognitive analysis processes, but today they are being replaced by methods of the extended cognitive analysis, which not only unanimously identifies the relationships between cognitive resonance and the data understanding process, but also shows that the system is capable of learning based on the results of the completed data analysis in order to optimise analysis processes. This situation is presented in Figure 1.

In the cognitive data analysis process, the process whereby the system learns new solutions which may impact the decision-making solution obtained is of key importance. So far, in the cognitive categorisation processes, the understanding of analysed data was based on the classical cognitive analysis process, whereby connections were indentified between pairs of consistent expectations of the system acquired from expert knowledge bases and characteristic features extracted from analysed datasets, and

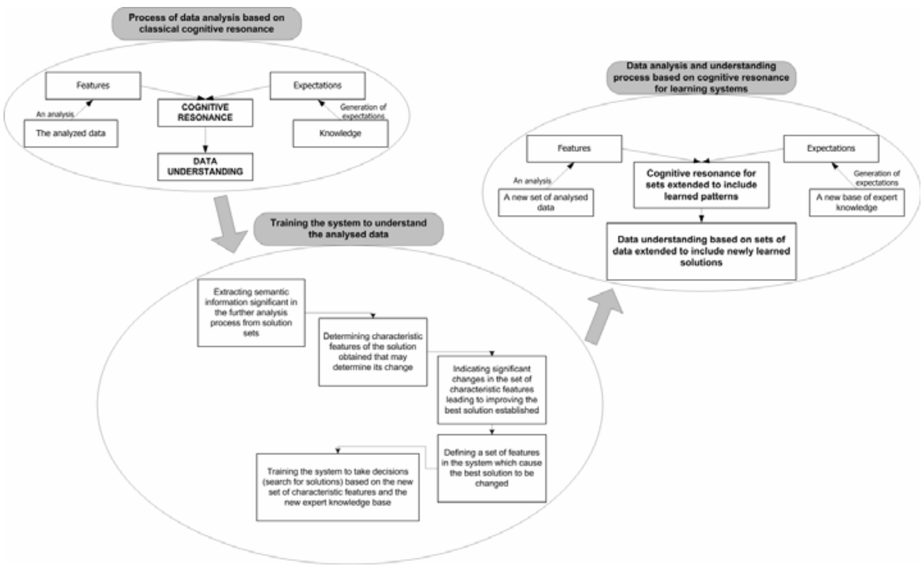

Process of data analysis based on classical cognitive resonance
Features
Expectations
COGNITIVE RESONANCE
An analysis
The analyzed data
Generation of expectations
Knowledge
DATA UNDERSTANDING
Training the system to understand the analysed data
Extracting semantic information significant in the further analysis process from solution sets
Determining characteristic features of the solution obtained that may determine its change
Indicating significant changes in the set of characteristic features leading to improving the best solution established
Defining a set of features in the system which cause the best solution to be changed
Training the system to take decisions (search for solutions) based on the new set of characteristic features and the new expert knowledge base
Data analysis and understanding process based on cognitive resonance for learning systems
Features
Expectations
Cognitive resonance for sets extended to include learned patterns
An analysis
A new set of analysed data
Generation of expectations
A new base of expert knowledge
Data understanding based on sets of data extended to include newly learned solutions

Fig. 1. Cognitive resonance in the process of data analysis and understanding enhanced with the process of training the system in new solutions

this led to cognitive resonance during which the above connections were determined as consistent or inconsistent. Only pairs that were completely consistent were selected for further analysis and a group of solutions could be defined which included the identified consistent pairs. This definition of the group made it possible not only to recognise the analysed data by naming it correctly, but also made the understanding of data complete, which lead to determining semantic features of the analysed data.

Research has shown that in this solution, pairs of characteristic features of the analysed data and expectations generated based on the expert knowledge base collected in the system which were not consistent were omitted at further analysis stages. This made it possible to envisage a situation in which the system encounters a solution it does not know and which is not defined at all in its bases. The question is, is it possible to recognise this type of a situation? Yes, the solution proposed in this publication shows that it is possible to introduce a stage at which the system is trained in solutions new to the system. This process is possible only when the set of solutions obtained (both optimum ones and those eliminated from further analysis) is used to create a set of features of analysed data and a set of new expectations not defined in the original bases of the system. The new features and expectations are input into the system base in which data is re-analysed, this time using the much broader expert knowledge set containing new patterns learned by the system. Such patterns constitute an extended expert knowledge base which the system uses to generate a set of expectations, and these are compared to the set of characteristic features of the analysed data. This process thus becomes an enhanced process of cognitive analysis based on cognitive resonance for learning systems. A system can be trained in any situation and this training can be multiplied depending on the needs and the necessity of extending the knowledge bases built into the system.

2 Cognitive Systems and Cognitive Informatics

Cognitive systems are those in which types of solutions modelled on data analysis processes taking place in the human brain have been applied to analyse complex data using layers of the semantic essence of data contained in every analysed set.

Processes of data analysis executed in cognitive systems based on cognitive resonance are very often equated with processes which form the cornerstone of cognitive informatics. However, we have to clearly distinguish between these two concepts associated with cognitive science and used in similar fields.

Cognitive informatics is a science dealing with the combination (one is tempted to say) of the traditional hardware informatics and cognitive science, that is the sciences concerned with learning, namely psychology, neurobiology, philosophy etc. Such combinations are possible if common ground or overlapping fields joining different scientific disciplines are found. Detailed solutions in the area of cognitive science, and in particular cognitive machine science, to which cognitive informatics belongs, make it necessary to look for specific applications. Such applications are created as system solutions combining the use of IT and cognitive tools. It is those combinations which lead to designing new classes of IT systems – cognitive systems.

This paper presents new solutions proposed for cognitive data analysis systems, showing how learning systems can enhance the image analysis processes executed by UBMSS (*Understanding Based Managing Support Systems*). UBMSS systems have been described by the authors previously, and are detailed in the following publications [2], [3], [4], [6].

3 UBMSS as an Example of Class of Cognitive Systems

UBMSS systems, as cognitive data analysis systems, can be used not just to analyse the economic figures of a company, but can also to supplement the analysis of information from data in the health sector. These systems are thus beginning to support the financial and strategic analysis of health-care providers (hospitals, clinics, medical companies offering various health services). What is characteristic of UBMSS system is that they conduct a financial analysis of a company using elements of cognitive data analysis.

In this paper was presented an example UBMSS system illustrating cognitive data interpretation methods for the efficient management of the investment process. UBMSS systems can be used for the cognitive analysis of economic ratios, particularly financial or macroeconomic ones. UBMSS systems can, for example, conduct analyses using the following ratios:

1. Liquidity ratios:
   - COGS (cost of goods sold),
   - EBIT (earnings before deducting interest and taxes),
   - NPV (net present value),
   - CR (current ratio),
   - QR (quick ratio),
   - cash ratio,

- inventory turnover,
- ACP (average collection period),

2. Profitability ratios:

- gross margin,
- profit margin,
- operating margin,
- net profitability ratio,
- gross profitability ratio,
- ROA (return on assets),
- ROE (return on equity),
- ROI (return on investment),
- ROIC (return on invested capital),
- ROS (return on sales),
- NPM (net profit margin),
- ROCE (return on capital employed),
- RONA (return on net assets),
- IRR (internal rate of return),
- WACC (weighted average cost of capital).

The analysis of economic ratios conducted in UBMSS systems allows the semantic contents of the analysed data to be used to determine the nature of that data, its impact on the current situation of the company and the extent of changes they cause to the company and its environment taking into account the information currently possessed. Such an analysis is possible due to semantic information contained in the analysed data. Semantic information may relate to:

- The scale (value) of analysed economic ratios,
- The frequency of their changes,
- The manner of their changes,
- The regularity of repetition,
- The number of changes observed,
- The type of changes observed.

This publication is an attempt at defining a UBMSS system for the cognitive analysis of investments on the basis of three key financial indicators, which include: NPV – net present value (symbol: W1), r – discount rate (W2), IRR – internal rate of return (W3).

For the proposed UBMSS systems, a sequence grammar of the following form has been defined:

$$G_{INV} = (\Sigma_N, \Sigma_T, P, S)$$

where:

Σ_N – the set of non-terminal symbols

Σ_N = {INVESTMENT, W1, W2, W3, WEAK_ACCEPT, ACCEPT, STRONG_ACCEPT, NO_ACCEPT, A, B, C, D, E}

Σ_T – the set of terminal symbols

Σ_T = {′a′, ′b′, ′c′, ′d′, ′e′}, and the particular elements were defined as follows: a = {0%}, $b \in$ (0%, 15%], $c \in$ (15%, 45%), $d \in$ [45%, 100%), $e \in$ (-100%, 0%) (Fig. 2.)

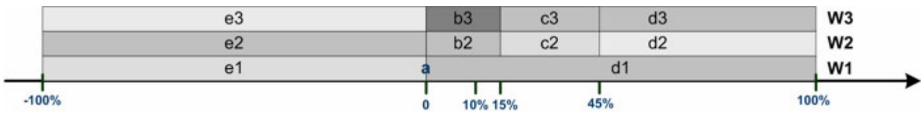


Fig. 2. The set of terminal symbols

S – the start symbol, S $\in \Sigma_N$, S = INVESTMENT

P – the set of productions shown below:

1. INVESTMENT→ WEAK_ACCEPT I ACCEPT I STRONG_ACCEPT I NO_ACCEPT
2. WEAK_ACCEPT→ W1 W2 W3 *//if (w1 & w2 & w3 = weak accept) final_decision:= weak accept*
3. ACCEPT→ W1 W2 W3 *//if (w1 & w2 & w3 = accept) final_decision := accept*
4. STRONG_ACCEPT → W1 W2 W3 *//if (w1 & w2 & w3 = strong accept) final_decision := strong accept*
5. NO_ACCEPT → W1 W2 W3 *//if (w1 & w2 & w3 =not akcept) final_decision := not accept*
6. W1 → A I B I C I D I E // *w1=decision*
7. W2 → A I B I C I D I E // *w2=decision*
8. W3 → A I B I C I D I E // *w3=decision*
9. A → a // *decision:= weak*
10. B → b // *decision:= weak*
11. C → c // *decision:= accept*
12. D → d // *decision:= strong*
13. E → e // *decision:= not accept*

The example UBMSS system discussed here can conduct a cognitive analysis of selected financial and economic ratios, which will make it possible to take the best strategic decision for the selected (analysed) company. Figures 3-5 show example results of the operation of the UBMSS system proposed for meaning-based analyses and interpretations stemming from understanding the analysed set of three financial ratios: the net present value, the discount rate, the internal rate of return.

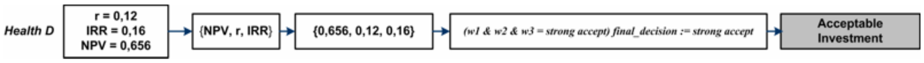


Fig. 3. Example UBMSS system for analysing and assessing the acceptability of an investment based on selected economic ratios

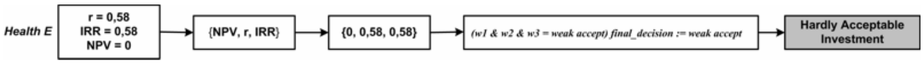

Health E
r = 0,58
IRR = 0,58
NPV = 0
{NPV, r, IRR}
{0, 0,58, 0,58}
(w1 & w2 & w3 = weak accept) final_decision := weak accept
Hardly Acceptable Investment

Fig. 4. Example UBMSS system for analysing and assessing the acceptability of an investment

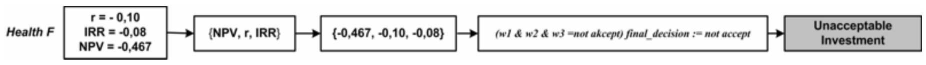


Fig. 5. Example UBMSS systems

Figure 3 shows a situation in which the UBMSS system analyses the economic data of an acceptable investment, in Figure 4 the investment is very good and deserving full acceptance, and in Figure 5 the system analysed ratios describing an unacceptable investment.

All cases of analysed economic and financial ratios demonstrate that it is of utmost importance to determine their impact on and their significance for the decision taken by the system.

Based on the values of the selected economic/financial ratios, the UBMSS system shows what strategic decision is best when it takes the said ratios into the analysis process. This decision is taken by comparing the analysed values with the values, kept by the system in particular knowledge bases, which have been defined based on optimum ratio values assumed by experts.

The above semantic information associated with the analysed economic data presented in the form of financial ratios allows a detailed identification of the type of situation (whether it is pathological or is a phenomenon expected and accepted by the company management with regard to the considered investment) prevailing within the company.

It must be borne in mind that changes taking place inside companies are brought about by various types of situations, phenomena and determinants. These situations may be either external or internal. This is why defining the right patterns applied to UBMSS systems which will be taking strategic and business decisions is very difficult, as it requires analysing a whole range of various factors that can have a significant impact on the decision-making process. It is because of this fact that the UBMSS systems presented in this paper for supporting the right decision whether to make (or forego) a given investment greatly help choose the best decision and determine whether the investment under consideration is acceptable or not; and if the decision is acceptable, then whether the acceptance is unconditional, or whether there is a certain danger (risk) inherent in implementing it (this situation is illustrated by the minimum permissible values of financial ratios selected for analysing).

UBMSS systems are of great help in understanding the analysed economic, financial and strategic situation with regard to the analysed company, investment and strategy. So they are systems which perform a very important type of analysis – a cognitive, interpretational, reasoning and forecasting analysis based on mechanisms of the linguistic and meaning-based description of data.

4 Conclusion

Cognitive analysis systems, also referred to as cognitive systems, are constantly developing. Increasingly frequently attempts at their improvement bring about successive improvements of the effectiveness of the analysis conducted and the increased utility of these systems.

Cognitive systems have been extended to include new solutions aimed at improving not just the effectiveness of the analysis process, but mainly the reliability of the semantic reasoning and forecasting processes. This last stage of analysis, aimed at correctly forecasting changes in the analysed data, is significant in that during a failed attempt to understand data this stage is omitted altogether. Implementing the capacity of learning new solutions in the system leads to a successful determination during subsequent attempts of the analysis undertaken, then to data understanding, and once that is successful, it supports making further forecasts.

Learning systems are therefore becoming a new class of cognitive data analysis systems and they now seem to have a bright future.

Acknowledgement. This work has been supported by the Ministry of Science and Higher Education, Republic of Poland, under project number N N516 196537.

References

1. Meystel, A.M., Albus, J.S.: Intelligent Systems – Architecture, Design, and Control. John Wiley & Sons, Inc., Chichester (2002)
2. Ogiela, L.: UBMSS (Understanding Based Managing Support Systems) as an Example of the Application of Cognitive Analysis in Data Analysis. In: IEEE Proceedings 6th International Conference CISIM 2007 – Computer Information Systems and Industrial Management Applications, CISIM 2007, Ełk, Poland, June 28-30, pp. 77–80 (2007)
3. Ogiela, L.: Modelling of Cognitive Processes for Computer Image Interpretation. In: Al-Dabass, D., Nagar, A., Tawfik, H., Abraham, A., Zobel, R. (eds.) EMS 2008 European Modelling Symposium, Second UKSIM European Symposium on Computer Modeling and Simulation, Liverpool, United Kingdom, September 8-10, pp. 209–213 (2008)
4. Ogiela, L., Ogiela, M.R.: Cognitive Techniques in Visual Data Interpretation. Studies in Computational Intelligence, vol. 228. Springer, Heidelberg (2009)
5. Ogiela, M.R., Tadeusiewicz, R.: Modern Computational Intelligence Methods for the Interpretation of Medical Images. Springer, Heidelberg (2008)
6. Tadeusiewicz, R., Ogiela, L.: Selected Cognitive Categorization Systems. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 1127–1136. Springer, Heidelberg (2008)
7. Tadeusiewicz, R., Ogiela, L., Ogiela, M.R.: The automatic understanding approach to systems analysis and design. International Journal of Information Management 28, 38–48 (2008)
8. Wang, Y.: The Theoretical Framework and Cognitive Process of Learning. In: Proc. 6th International Conference on Cognitive Informatics (ICCI 2007), pp. 470–479. IEEE CS Press, Lake Tahoe (2008)

An Automated Approach to Testing Polymorphic Features Using Object-Z

Mahreen Ahmad[1], Aamer Nadeem[1], and Tai-hoon Kim[2]

[1] Center for Software Dependability,
Mohammad Ali Jinnah University (MAJU), Islamabad, Pakistan
mahreen.ahmad@gmail.com, anadeem@jinnah.edu.pk
[2] Dept. of Multimedia, Hannam University
133, Ojeong-dong, Daedeok-gu, Daejeon, Korea
taihoonn@hannam.ac.kr

Abstract. Formal methods have proven their worth in different fields specially software engineering. Although object-oriented design features like inheritance and polymorphism improve the quality of design, these features may introduce new types of faults. However, the current research on formal specification based testing primarily focuses on unit level testing only. There is very little work on formal specification based inheritance and polymorphic testing. This paper describes a novel approach for testing of polymorphic relationships using an Object-Z specification. The proposed approach is based on the idea of coupling based testing. Tool support for the proposed technique has also been provided and is empirically evaluated by a real life example.

Keywords: Polymorphism, Inheritance, Formal Specification, Specification based testing, coupling based testing, Object-Z.

1 Introduction

Object-oriented (OO) design has an emphasis on defining abstractions to model the concepts in a particular domain [12]. These abstractions appear in software as user-defined types that have both state and behavior. Although these abstractions help in achieving higher quality design but OO features such as inheritance and polymorphism also complicate testing process and can potentially bring new types of faults in OO software [9].

Formal methods have proven their worth in different fields specially software engineering. The motivation for using formal methods in software engineering is that time spent on specification and design is paid back in the form of a higher quality product, and ultimately it reduces the cost of rework later on. This is because formal methods eliminate ambiguities and avoid errors in the desired software product.

Specification based testing, also known as black-box testing, has major advantage that the testing activity can be started at an earlier stage of software development life cycle, i.e., at the time of Software Specification, because in specification based testing, a test suite is produced on the basis of a specification [14].

Inheritance and polymorphism are commonly used in object-oriented software. But whenever inheritance and polymorphism are used in any program, the code becomes difficult to comprehend and is prone to errors and faults [9]. For example in inheritance, the child class must be compatible with its parent class; it should not violate the parent class invariant. Moreover, having inheritance relationship, any child class can be substituted for its parent class, this substitution may cause inconsistent type use error [6]. Dynamic binding, or knowing the type at runtime, is achieved through polymorphism. When polymorphism is used, it becomes hard to understand all possible bindings of an object and taking care of all possible interactions between caller and callee is quite a challenging task. Besides this, overridden methods could be faulty in the child class context or overriding methods might have different or conflicting pre and post conditions as compared to overridden methods in parent classes [6].

The scope of this paper is testing of inheritance and polymorphic features in object-oriented design using formal specification. The proposed approach detects certain polymorphic faults described in Offut et al.'s fault model [6]. Taking Object-Z specification and Finite State Machine (FSM) of each class in the hierarchy as an input, our approach flattens the Object-Z class hierarchy, and determines message call sequences (MCS) from FSM. Using Object-Z specification and MCS, it identifies the def-use of coupling variables and identifies coupling sequences to generate test cases. The novel approach is a specification based testing technique which is based on the idea of Alexander's coupling based testing technique for code [7][4][5].

2 Relevant Work

Murray et al. have extended Test Template Framework (TTF) for inheritance, in object-oriented programs using Object-Z specifications [13]. TTF is a formal and abstract model used for testing of procedural programs. Using this framework a test information hierarchy is build having valid input space and output space, but the TTF cannot be used for execution of test cases directly. They have also suggested that when test cases of parent class(s) should be reused for testing child class; in same the way as object oriented design facilitates reusability. They have defined three options for reusability of TTF information for child class i.e. when testing information can be inherited without change, when it is inherited with modifications and when it must be derived from scratch.

Liu and Miao have proposed a specification based approach of testing inheritance and polymorphism using Object Z specification [3]. They have addressed the combined effect of inheritance, polymorphism and aggregations covering intra-method, intra-class and inter-class testing criteria. They have provided an algorithm for intra-class testing and have suggested a coverage criterion for inter-class testing. They have automated their approach and have empirically evaluated their approach using a small example. They have empirically evaluated only four types of faults of Offut et al. fault model, i.e., SDA, SDIH, SDI, and IC.

Nadeem and Lyu have proposed a technique for generation of test cases for inheritance testing, using a VDM++ formal specification [1]. In this technique, the idea of flattening is used in VDM++ specification class, and then operation sequences are generated from the trace structure specified in the VDM++ specification. The input space for each operation is partitioned, and a test model is constructed from the operation sequences and the input partitions. Test paths are generated from the test model, which cover the different operation sequences as well as the partitions. Coverage criteria for test path generation have also been proposed in this paper.

Nadeem et al. have presented an approach to automate the generation of test cases from a VDM++ specification [2]. They have combined partition testing approach with testing of operation sequences to apply on testing of inheritance and polymorphic faults at specification level. They are using Offut et al. fault model as basis for faults but their approach does not cover all of the faults in the fault model, due to the higher abstraction level of specifications. The faults that can be captured by their approach are SDA, SDI, SVA and IC.

3 Specification Based Testing of Polymorphism and Inheritance

The proposed approach is based on the idea of Coupling Based Testing (CBT) proposed by Alexander [4] [5] [7]. He has introduced a technique for testing polymorphic features in Object-oriented programs [4]. This work is based on their ongoing research on coupling based testing (CBT) [11]. The presented technique is basically a data flow testing technique to identify testable coupling sequences among different call sequences in a class hierarchy. They have extended their work and have proposed new polymorphic coupling sequences as the testable paths for test cases.

The existing approach presented by Alexander is a code based testing technique, while our proposed approach applies the idea of coupling based testing on a formal specification. Coupling based testing states that the program be executed from definitions of actual parameters through calls to uses of the formal parameters, as a result different coupling paths are defined, and combination of these coupling paths form coupling sequences [11]. There are four structural types of coupling sequences, Type I, II, III and Type IV coupling sequences [7]. This paper covers only type I coupling sequence, due to the limitation of Object Z specifications which does not provide any information of sequence of statements execution [10].

Coupling based testing is a data flow based technique to discover and represent state space interactions between pairs of method calls in case of polymorphic relationships. Coupling sequences in CBT help in static and dynamic analysis of the programs and highlights the areas to be tested, for methods under test. This will help developers in analyzing and understanding the critical polymorphic interactions in the system.

The novel approach does not cover all the aspects of CBT based on specifications, due to the limitation of specifications itself. For example, every coupling sequence has an associated set of coupling variables and coupling paths. The proposed approach does not identify the coupling paths and identifies the coupling sequences without finding coupling paths, as extracting coupling paths require the statements execution sequence, and this information lacks in Object-Z specification [15]. Alternatively, due to the limitation of Object-Z specification, the proposed approach keeps abstraction level higher and identifies the def-use methods rather than finding coupling paths. And instead of defining absolute coupling paths, the approach defines the abstract paths between pair of method calls, as message call sequences can be extracted from the FSM(s) given as input to the system.

The fault model presented by Offutt et al. has 9 types of faults due to inheritance and polymorphism [6]. Making this fault model as basis of polymorphic fault types, the proposed approach can detect 6 faults, i.e., Inconsistent Type Use (ITU), State Definition Anomaly (SDA), State Definition Inconsistency due to state variable hiding (SDIH), State Defined Incorrectly (SDI), Indirect Inconsistent State Definition (IISD), Incomplete Construction (IC) and State Visibility Anomaly (SVA). The faults like ACB1 and ACB2 cannot be detected by the proposed approach, because these two types of faults need interaction of more than one method, while the presented approach can only detect those types of polymorphic faults that require calling client method interacting with the polymorphic methods, they are called as intra-method coupling sequences [7].

The focus of this paper is on testing polymorphic interactions resulting from pairs of method invocations within the same method, i.e., intra-method coupling sequences. There can be certain other interactions that may occur between methods and these interactions are not the result of invocation from the same method, these types of interactions represent inter-method coupling sequences or indirect coupling sequences and are left as future work.

4 The Proposed Approach

The proposed approach uses Object-Z specification for the class hierarchy. Object-Z is an extension of Z language which facilitates the formal specification of object-oriented programs [10] [15]. Our approach flattens the Object-Z class hierarchy, and takes FSM of each class in the hierarchy as input to find message call sequences (MCS). Using Object-Z specification and MCS, it identifies the def-use of coupling variables and identifies coupling sequences to generate test cases. Graphical representation of the approach is shown in Fig 1. This section describes the approach in detail.

4.1 Class Flattening

The input of the proposed approach is an error-free Object-Z specification having complete class hierarchy. By error-free we mean that Object-Z specification should not contain any syntax, semantic and design errors. The input class hierarchy is flattened using class flattening technique. Flattening is the process of including all the features of parent class into the child class without using inheritance feature [16]. In the proposed approach, each derived class is flattened in such a way that the predicates and variables of child class schemas (state & operation) are a conjunction of predicates and variables of both derived and base class schemas [10][15].

After flattening, all the parent class methods are added in the child class except the overridden methods, i.e., those methods that are redefined in the derived class.

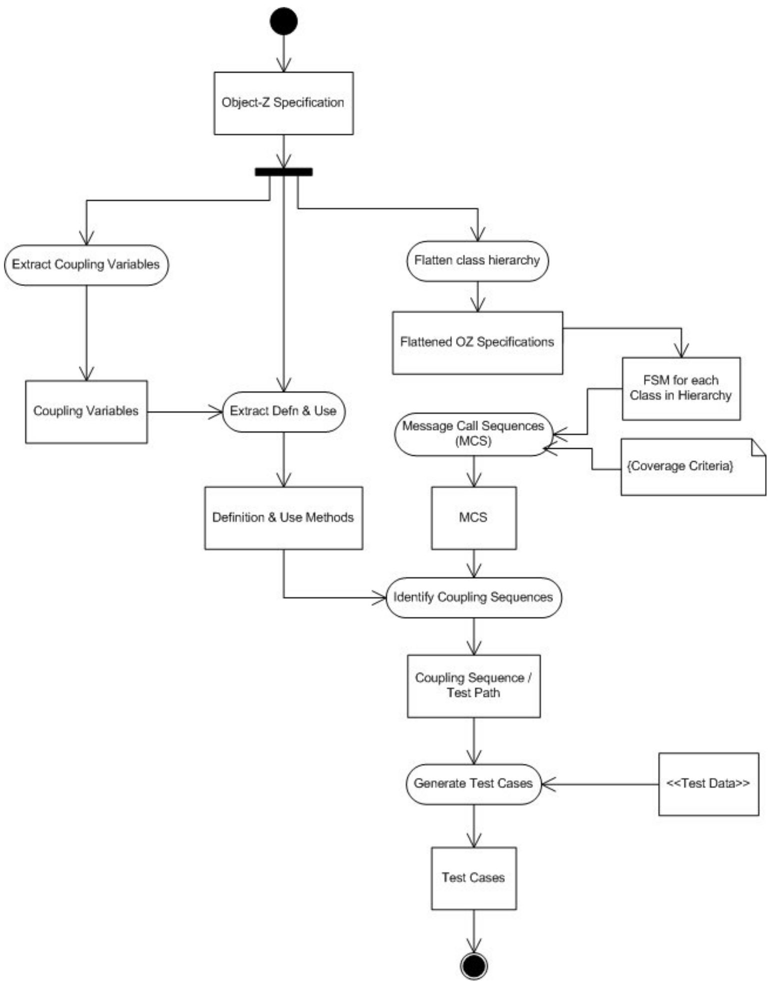
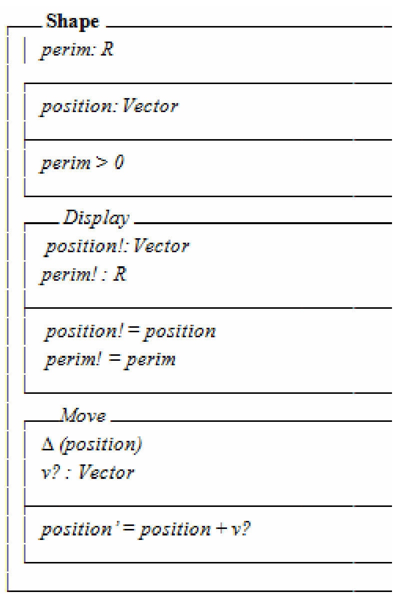


Fig. 1. Architecture of the proposed approach

Operation schemas having common names in parent and child class will be conjoined in the child class, until and unless common named operation is redefined in the child class. Operation schemas can be redefined in two ways, by renaming the common named operations and redefining them with new names, operations and variables both can be renamed. But redefining using rename facility will not allow polymorphic feature in the child class, to allow polymorphism the common named operation schema should be refined in the child class using *redef* comma separated list. If redefinition of operations is not required then common named operations will be extended in the child class and will be conjunction of parent and child class operation schemas. Consider following shape example in Fig 2 [10]. Polygon class inherits shape class. Fig 3, 4 shows unflattened and flattened polygon class respectively.



Shape

perim: R

position: Vector

perim > 0

Display

position!: Vector

perim! : R

position! = position

perim! = perim

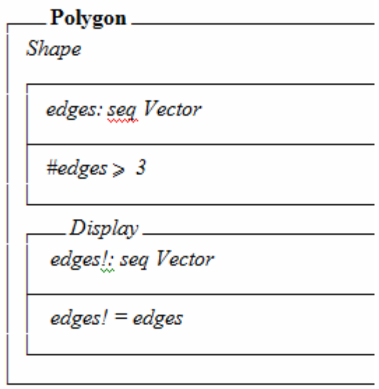
Move

Δ (position)

v? : Vector

position' = position + v?

Fig. 2. Object Z Specification of Shape



Polygon

Shape

edges: seq Vector

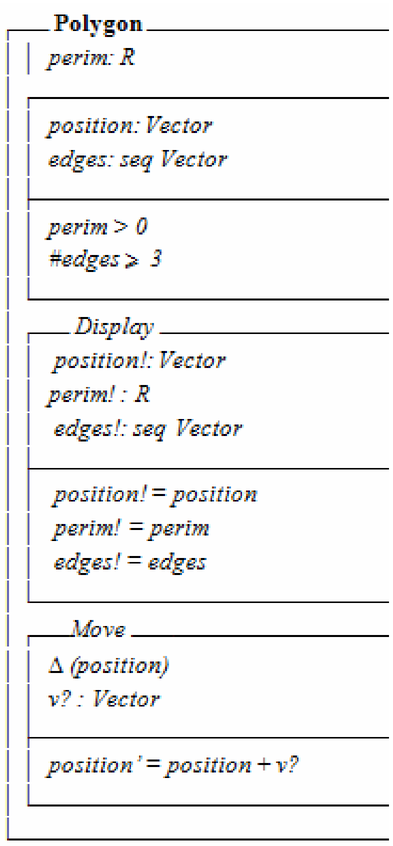
#edges ≥ 3

Display

edges!: seq Vector

edges! = edges

Fig. 3. OZ Specification of Polygon



Polygon

perim: R

position: Vector

edges: seq Vector

perim > 0

#edges ≥ 3

Display

position!: Vector

perim! : R

edges!: seq Vector

position! = position

perim! = perim

edges! = edges

Move

Δ (position)

v? : Vector

position' = position + v?

Fig. 4. Flattened Polygon Class

4.2 Extracting Message Call Sequences (MCS)

Finite State Machine (FSM) for each flattened class in the hierarchy is given as input to the system in the form of State transition table. FSM is used to find the sequence of message calls; because object-Z specification does not provide any information about the sequence of method calls neither does it provide any information about sequence of statement execution. Using certain coverage criteria we need to make a tree e.g. round trip tree to find the message call sequences. The coverage criteria for generating MCS from FSM is also given as input to the system because weaker coverage criteria reveals less MCS and stronger coverage criteria results in more

MCS. Different coverage criteria produce varying result. For this paper we have used all transition coverage criteria. The algorithm for extracting message call sequences is adapted by Round Trip Tree algorithm presented by Binder [8]. The algorithm states that tree be started from alpha node and every node should have one incoming and one outgoing transition. Each node should be split into child nodes equal to the number of outgoing transitions, if there are more than one outgoing transitions on that node, whenever an existing node appears as child of another node, make it a leaf node, repeat the process until omega or final state is reached. Consider FSM of shape example in Fig 5: -

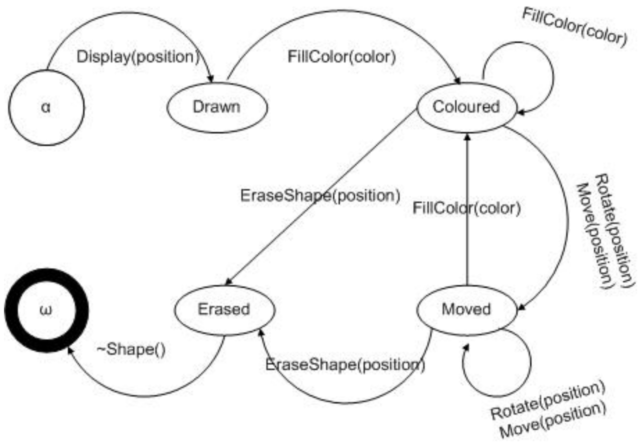
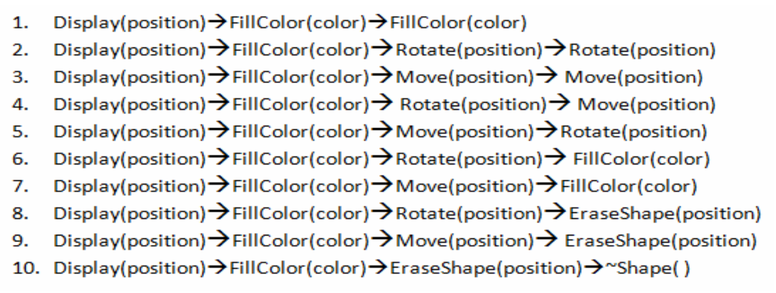


Fig. 5. FSM for Shape Class

For Shape example we'll make round trip tree to extract message call sequences from FSM, after making round trip tree we need to set some coverage criteria for finding maximum coverage of message calls, the round trip tree for Shape FSM is shown in Fig 7. Now to extract message call sequences, we will select all unique message sequences. All unique message call sequences of Shape class are shown in fig 6. The format for message call sequence is as follows: -

[Message(parameterList)→NextMessage(parameterList)]



1. Display(position)→FillColor(color)→FillColor(color)
2. Display(position)→FillColor(color)→Rotate(position)→Rotate(position)
3. Display(position)→FillColor(color)→Move(position)→ Move(position)
4. Display(position)→FillColor(color)→ Rotate(position)→ Move(position)
5. Display(position)→FillColor(color)→Move(position)→Rotate(position)
6. Display(position)→FillColor(color)→Rotate(position)→ FillColor(color)
7. Display(position)→FillColor(color)→Move(position)→FillColor(color)
8. Display(position)→FillColor(color)→Rotate(position)→EraseShape(position)
9. Display(position)→FillColor(color)→Move(position)→ EraseShape(position)
10. Display(position)→FillColor(color)→EraseShape(position)→~Shape( )

Fig. 6. Message Call Sequences of Shape FSM

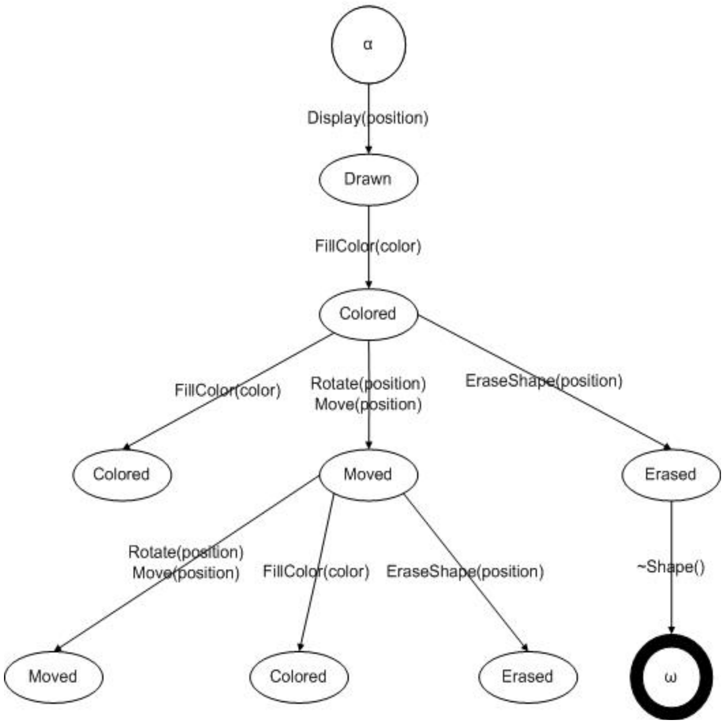


Fig. 7. Round Trip Tree for Shape FSM

4.3 Extracting Coupling Variables

As mentioned earlier, every coupling sequence has an associated set of class variables. These class variables or state variables are called as coupling variables. Coupling variables of each class in the hierarchy need to be extracted from Object-Z specification. In Object-Z Specification the coupling variables are specified in the state schema of class specification. After finding the coupling variables we need to see that which of these coupling variables are defined and used in the Object-Z specification. Considering shape example in Fig 2 the coupling variable of Shape class is *position*, in Fig 3 coupling variable of Polygon class is *edges*, and in Fig 4 coupling variables of flattened Polygon class is *position* and *edges*.

To determine coupling variables of each class we will search/parse the declaration part of state schema of each class, and will extract all the declared variables in this part.

4.4 Identifying Def-Use Methods

Due to the lack of information about sequence of statement execution in Object-Z specification, the proposed approach identifies def-use methods instead of finding coupling paths to define more abstract paths between pair of method calls from calling client method.

Finding def-use means that we need to extract only those coupling variables that are defined i.e., assigned any value and then used in any method. We will be finding definition and use of coupling variables from the object-Z specification and identify those methods that are defining or using any coupling variable. In Object Z equal to '=' symbol is not an assignment operator, rather it is a relational operator. Therefore it is not obvious that the variable to the left of = symbol will be defined. For example, statement (1) and (2) semantically mean same in Object Z specification: -

$$\text{position}' = \text{position} + v? \tag{1}$$

$$\text{position} + v? = \text{position}' \tag{2}$$

Hence, to find the definition of any coupling variable we need to check whether the coupling variable is appearing in predicate part of operation schema with or without prime. If coupling variable is primed then it is defined and If coupling variable is unprimed then it is used, no matter where it appears before or after the = symbol. Or we can also check the Δ-list in each operation schema of the class, if the coupling variable is present in the Δ-list then it means that the coupling variable is defined otherwise it is used if variable is present in the predicate part of operation schema and not present in the Δ-list. Hence in this way we will identify def-use of coupling variables along with the methods they are defined and used in. Considering shape example in Fig 2, position is the coupling variable of shape class, variable position is present in the Δ-list and predicate part of operation schema Move, which means that variable position is defined and used in this method, because it appears both, with and without prime. Moreover, variable position is not present in the Δ-list of Display operation schema, but it is there in the predicate part of operation schema Display, therefore variable position is used in this method. After finding the def-use of coupling variables and identifying the def-use methods we need to find the coupling sequences by using message call sequences.

4.5 Identifying Coupling Sequences

When the FSM and the coverage criteria is provided as input, we will extract all unique message call sequences of all methods in each class. After getting all unique MCS we need to select our desired coupling sequences, and the selected coupling sequences will be the test paths for test cases. Now to find a coupling sequence it is required to see that which coupling variable is defined in a method and then the same coupling variable is used in another method. This means we will consider only that sequence when two methods define and use the same coupling variable, and both methods are called by a client method using same instance. We need to find def-use pair among these message call sequences and we will select only those MCS that have this def-use pair. The def-use pair should be definition clear with respect to the coupling variable, in a message call sequence. A def-clear path with respect to variable X is a path that does not contain any definition of X in between the path [11].

In proposed approach we will consider type I coupling sequence, which means that the definition and the use of coupling variables will be in calling methods not the client method.

While selecting call sequences to identify coupling sequences there can be different scenarios that need to be handled. The cases are shown in Table 1.

Table 1. Def-Use pair Cases in Message Call sequences

Serial No.	Invalid Cases	Serial No.	Valid Cases
I	use – use	V	def – use
II	def – def	VI	def – def – use
III	use – def	VII	def – use – def
IV	use – def – def	VIII	use – def – use

Out of these eight cases of def-use, case I, II, III and IV will be omitted completely because they do not make a valid def-use pair, like the case V. For cases VI & VII and VIII we need to consider def clear path with respect to a particular coupling variable. So, in case VI and VIII we will consider later part and omit first def/use and in case VII we will consider first part and omit last def.

Taking example of Shape class with its message call sequences in Fig 6 and having the information of def-use methods and coupling variables, also keeping above mentioned scenarios of def-use in mind, we need to extract coupling sequences. For example class Shape specification in fig 2, the coupling variable is *position*, def and use methods are **Move** and **Display** respectively. Please note that variable *position* is both defined and used in method **Move**, while *position* is used only in **Display** method. Now from the extracted MCS we will select only those MCS in which position is defined in one method and then without redefinition it is used in another method, e.g., consider MCS 1 in fig 6, i.e.,

Display(position) →FillColor(color) →FillColor(color)

In above mentioned call sequence we can see coupling variable *position* in the parameter list of ‘Display’ method but it is not present in parameter list of ‘FillColor’ method, which means it, is neither defined nor used in ‘FillColor’. Also from specifications in fig 2 we can see that *position* is used and not defined at all, so we will not select this call sequence because it does not make a valid def-use pair.

Display(position) →FillColor(color) →Move(position) → Move(position)

In above call sequence, coupling variable *position* is used in ‘Move’ method and then defined as well. While ‘Display’ only uses coupling variable *position*. In this call sequence valid def-use pair lies in the later part of the sequence, so we will select this call sequence because this sequence resembles case VIII. So we will select MCS 3, 4, 5, 8 and MCS 9 from the listed MCS in Fig 7 as our coupling sequences or test paths.

In object oriented design when the instance of child class is created the call jumps from constructor of child class and calls the parent class constructor and then child class constructor is called after creation of parent class instance. This is why MCS like the following one will be seen in the call sequences, which shows two Init in the call sequence, the first Init shows parent class constructor and the second Init shows child class constructor.

Acc::Init()→SA::Init()→SA::Credit(amount)→Acc::Debit(amount)

As discussed earlier, the focus of this paper is on intra-method coupling sequences therefore the faults that can be catered by the proposed approach are the ones that occur due to the call from client method. Only the type of above call sequence can be catered in the presented approach.

4.6 Generating Test Cases

After identifying and extracting coupling sequences we have those call sequences that need to be tested i.e., the Test Paths. Now we need to provide test data to make test cases. Test data will be provided manually and then these test cases will be executed on the code for testing purpose. We also need to provide mapping of specification to code since our test cases will be executed on code. Here we assume that the code uses same naming convention for classes, operations and attributes as specified in the input Object-Z specification of class hierarchy, to avoid the effort of specification to code mapping. The general format for the test case will be as follows:

[{Test Paths}, {Test data list}] ==>
[{Message(parameterList)→NextMessage(parameterList)}, {{Test data list}}]

The test case requires test path and the test data. The selected coupling sequences are actually the test paths for test cases. We need to provide test data manually for the following test path. In this case we need value for *position* which comprises of two values i.e., x, y positions.

[{Display(position)→FillColor(color)→Move(position)→ Move(position)},
{(3,4), (-3,4)}]

5 Evaluation

The approach presented in this paper is automated by developing a prototype tool that generates test paths from an Object-Z specification. The tool is developed using C# on Microsoft .NET framework. The system takes Latex source files, for Object-Z specification, as input and generates test paths whereas test data is manually provided to the system.

We have also empirically evaluated our approach on a case study, and found it more effective than existing specification based approaches presented in literature. We have evaluated our approach by identifying certain evaluation parameters. The most important of these criteria is the percentage of *fault coverage*. Majority of the polymorphism faults are discussed in Offutt et al., fault model [6]. The existing approaches like [2] and [3] can identify polymorphic faults up to 40% and 50% respectively. While, keeping the higher abstraction level of specification in view, our approach can identify up to 70% of the polymorphic faults. If the specification language related limitations are omitted, our approach can be extended for the remaining faults as well.

6 Conclusion and Future Work

This paper discusses specification based testing of polymorphic features in object-oriented design using formal specification. The proposed approach is based on the idea of Alexander's coupling based testing for code. Using Object-Z specification as input, the proposed approach detects certain polymorphic faults presented by Offut et al., and generates test cases that can be executed on code. Due to which the faults can be identified at an early stage of software development. We have developed the prototype tool to verify the practicality of our approach.

The novel approach can detect faults that may occur due to intra-method calls in polymorphic relationships, i.e. when pairs of methods are called by the calling client method. This limitation is because of higher abstraction level of the specification language, Object-Z. Inter-method coupling sequences are left as future work. To avoid the limitation of the specification language we plan to use CSP-OZ in future for handling type II, type III and type IV coupling sequences.

Acknowledgement

This work was supported by the Security Engineering Research Center, granted by the Korean Ministry of Knowledge Economy.

References

[1] Nadeem, A., Lyu, M.R.: A Framework for Inheritance Testing from VDM++ Specifications. In: 12th Pacific Rim International Symposium on Dependable Computing, PRDC, pp. 81–88 (December 2006)

[2] Nadeem, A., Malik, Z.I., Lyu, M.R.: An Automated Approach to Inheritance and Polymorphic Testing using a VDM++ Specification. In: 10th IEEE International Multitopic Conference (INMIC 2006), Islamabad, Pakistan (December 2006)

[3] Liu, L., Miao, H.: A Specification-Based Approach to Testing Polymorphic Attributes. In: Davies, J., Schulte, W., Barnett, M. (eds.) ICFEM 2004. LNCS, vol. 3308, pp. 306–319. Springer, Heidelberg (2004)

[4] Alexander, R.T., Offutt, J.: Coupling-based Testing of O-O Programs. Journal of Universal Computer Science (JUCS) 10(4) (April 2004)

[5] Alexander, R.T., Offutt, J., Bieman, J.M.: Fault detection capabilities of coupling-based OO testing. In: Thirteenth International Symposium on Software Reliability Engineering (ISSRE 2002). IEEE Computer Society, Annapolis (2002)

[6] Offutt, J., Alexander, R., Ye, W., Quansheng, X., Chuck, H.: A Fault Model for Subtype Inheritance and Polymorphism. In: The Twelfth IEEE International Symposium on Software Reliability Engineering (ISSRE 2001), Hong Kong, pp. 84–95 (November 2001)

[7] Alexander, R.T.: Testing the Polymorphic Relationships of Object-oriented Programs: Phd. Dissertation, George Mason University (2001)

[8] Binder, R.V.: Testing Object Oriented Systems, Models, Patterns and Tools. The Addison-Wesley Object Technology Series (2000) ISBN 0-201-80938-9

[9] Binder, R.V.: Testing object-oriented software: a survey. Journal of Software Testing, Verification and Reliability, 125–252 (January 1999)

[10] Smith, G.: The Object-Z Specification Language. Kluwer Academic Publishers, USA (1999)

[11] Jin, Z., Offutt, A.J.: Coupling-based Criteria for Integration Testing. The Journal of Software Testing, Verification, and Reliability, 133–154 (1998)

[12] Meyer, B.: Object-Oriented Software Construction. Prentice Hall, Englewood Cliffs (1997)

[13] Murray, L., Carrington, D., MacColl, I., Strooper, P.: Extending test templates with inheritance. In: Software Engineering Conference, Australia, pp. 80–87 (October 1997)

[14] Ammann, P., Offutt, J.: Using Formal Methods To Derive Test Frames in Category-Partition Testing. In: Reggio, G., Astesiano, E., Tarlecki, A. (eds.) COMPASS 1994. LNCS, vol. 906, pp. 69–80. Springer, Heidelberg (1995)

[15] Smith, G.P.: An Object-Oriented Approach to Formal Specifications: Phd. Dissertation, University of Queensland (1992)

[16] Meyer, B.: Lessons from the design of the Eiffel libraries, pp. 68–88. Interactive Software Engineering Inc., Goleta (1990)

IDMS: A System to Verify Component Interface Completeness and Compatibility for Product Integration

Wantana Areeprayolkij[1], Yachai Limpiyakorn[1], and Duangrat Gansawat[2]

[1] Department of Computer Engineering,
Chulalongkorn University, Bangkok 10330, Thailand
[2] National Electronics and Computer Technology Center,
Pathumthani 12120, Thailand
Wantana.a@student.chula.ac.th, Yachai.L@chula.ac.th,
duangrat.gansawat@nectec.or.th

Abstract. The growing approach of Component-Based software Development has had a great impact on today system architectural design. However, the design of subsystems that lacks interoperability and reusability can cause problems during product integration. At worst, this may result in project failure. In literature, it is suggested that the verification of interface descriptions and management of interface changes are factors essential to the success of product integration process. This paper thus presents an automation approach to facilitate reviewing component interfaces for completeness and compatibility. The Interface Descriptions Management System (IDMS) has been implemented to ease and fasten the interface review activities using UML component diagrams as input. The method of verifying interface compatibility is accomplished by traversing the component dependency graph called Component Compatibility Graph (CCG). CCG is the visualization of which each node represents a component, and each edge represents communications between associated components. Three case studies were studied to subjectively evaluate the correctness and usefulness of IDMS.

Keywords: component interface compatibility, component dependency graph, product integration, software process improvement.

1 Introduction

Nowadays, the extension of Object-Oriented scale brings to the Component-Based approach. Component-Based software Development (CBD) is a discipline which has emerged as a new approach to deliver software engineering from the in-house industrial system into larger system for Information Technology [1]. It is recommended to promote the quality dimensions of system components in terms of interoperability and reusability during system architectural design.

However, one of today project delay problems often occurs in the component integration process. Many software applications encounter similar difficulties to effectively integrate the implemented component subsystems. The integration failure usually results from the flaw of the system architectural design that lacks effective cooperation and communications among the constituent subsystems. Therefore, the product integration plan for each subsystem should be well-established during system architectural design [2].

The investigation of the essence of the product integration process in software development was reported in the work of Larsson et al. [3]. The authors implemented some case studies based on the best practices contained in worldwide well-known reference models and standards for product integration process, such as ISO/IEC 12207, EIA-632, CMMI, EIA-731.1 and ISO/IEC 15288. The results suggested that the verification of interface descriptions and management of interface changes are important factors that impact on the successful implementation of the product integration process.

An approach of using interface descriptions as a set of important characteristics for system analyses such as architectural analysis, risk analysis, and change management analysis was discussed in [4]. The research work implemented a system called *Interface Descriptions for Enterprise Architecture (IDEA)* to assist the management of system architectural description within a large corporation. Furthermore, the work also addressed the usefulness of using a dependency graph for determining the entire dependency network required to complete all use cases for a single system.

Instead of developing the support tool for enterprise-level system, the focus of this research is on developing the tool to support reviewing interface descriptions of a system unit. The tool automatically extracts the input components' interface descriptions and their compatibility with associated components. The extracted interface descriptions can then be used in the static review process. The approach in this work required the detailed format of UML component diagram, i.e. white box view [5] so as to provide sufficient information for the construction of the Component Compatibility Graph (CCG) proposed in this research. The automation of the verification of component interface compatibility is then carried out by graph traversal.

2 Product Integration Best Practices

Referring to Capability Maturity Model® Integration (CMMI®) [6], the cluster of activities of product integration is considered as one of the groups of activities that affect project success. The recommended practices contained in Product Integration (PI) process area, so called Specific Practice (SP) are shown in Table 1. According to CMMI Glossary, a process area is a cluster of related practices in an area that, when implemented collectively, satisfy a set of goals considered important for making improvement in that area [6]. From Table 1, the Product Integration process area contains three goals so called Specific Goal (SG). In order to satisfy each SG, a set of associated SPs needs to be implemented successfully based on an individual organizational culture. Once all three SGs have been satisfied, it implies that the organization has a potential to assemble the product from the product components, ensuring that the product as integrated, functions properly, and deliver the product.

In this paper, we focus on developing a tool to facilitate the implementation of SP2.1 and partial of SP2.2. That is, SP2.1 requires the implementation of reviewing the interface descriptions periodically to ensure that each interface description is complete and fulfill the requirements by all stakeholders. Example evidence of successful implementation of SP2.1 should exist the mapping of the interfaces to the product component. Regarding SP2.2, the practice emphasizes on interface management to ensure that they will be controlled to be compatible throughout the product life cycle to reduce late discovery of mismatch. Example evidence of successful implementation of SP2.2 would be the existence of a table of relationships among the different product components.

Table 1. Specific Goals and Specific Practices of Product Integration Process Area [6]

SG 1	Prepare for Product Integration	
	SP 1.1	Determine Integration Sequence
	SP 1.2	Establish the Product Integration Environment
	SP 1.3	Establish Product Integration Procedures and Criteria
SG 2	Ensure Interface Compatibility	
	SP 2.1	Review Interface Descriptions for Completeness
	SP 2.2	Manage Interfaces
SG 3	Assemble Product Components and Deliver the Product	
	SP 3.1	Confirm Readiness of Product Components for Integration
	SP 3.2	Assemble Product Components
	SP 3.3	Evaluate Assembled Product Components
	SP 3.4	Package and Deliver the Product or Product Component

3 Approach Methodology

The automation approach to verifying component interface compatibility in this paper follows the methodology proposed in [7]. The method consists of three main steps (Fig.1) which are described in the following sections.

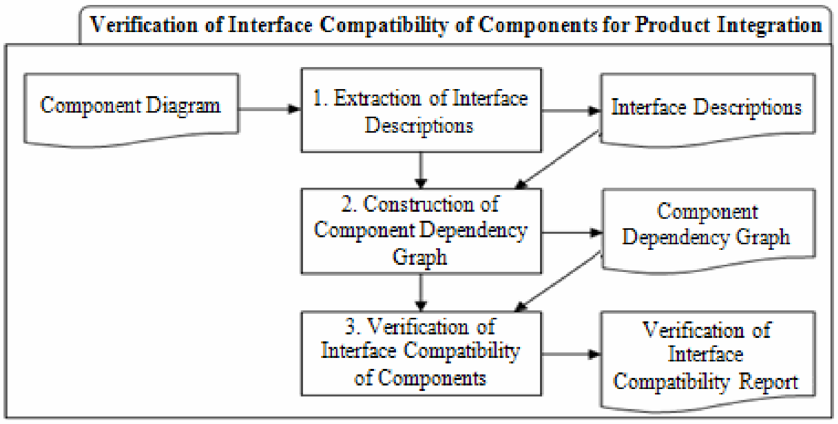

Verification of Interface Compatibility of Components for Product Integration
Component Diagram
1. Extraction of Interface Descriptions
Interface Descriptions
2. Construction of Component Dependency Graph
Component Dependency Graph
3. Verification of Interface Compatibility of Components
Verification of Interface Compatibility Report

Fig. 1. Approach methodology [7]

3.1 Extraction of Interface Descriptions

The first step is to extract all interface data from the source components illustrated in UML Component diagrams [5], which can be converted into the Extensible Markup Language (XML) Metadata Interchange format [8] for automatically generating the interface descriptions as the output from this step.

The data extracted can be classified into two categories: 1) the data directly extracted from each component, e.g. component name, and 2) the interface data i.e. “Required” and “Provided” of each component, class names which have realization relation inside the component and the data extracted from class which are names of class interfaces, the method of class that the realized instance is identified by parameter types and variables from interface class. The generation of these component interface descriptions could facilitate the implementation of SP2.1 of Product Integration to review interface descriptions for completeness as it is easier compared to directly review for interface descriptions completeness using design documents.

3.2 Construction of Component Dependency Graph

This paper has adapted the approach of Component Architecture Graph (CAG) presented in the work of Kanjilal et al. [9] combining with dependency tree [10] to generate the component dependency graph so called *Component Compatibility Graph (CCG)* in this work. Each node of CCG represents a component and it contains interface information to support interface compatibility verification. For example, names of methods, returned values of method, the number of names and parameter types. CCG is useful since it visualizes the cluster of associated components by addressing the dependency relationships among components through component’s edges.

3.3 Verification of Interface Compatibility of Components

The graph visualization of components could help clustering the components for ordering the sequences of integration plan. The traversal on CCG through each edge automatically inspects the interface compatibility of the associated components. The report will be generated at the end of this step.

4 Interface Descriptions Management System

In this work, Interface Descriptions Management System (IDMS) is implemented to automate parts of product integration process. That is, the system facilitates reviewing component interfaces for completeness and compatibility. Details are described as follows.

4.1 Design

The functionalities of IDMS are illustrated by UML Use-case diagram as depicted in Fig. 2. Further design of the system at the component level complying with UML 2.0 [5] is shown in Fig. 3.

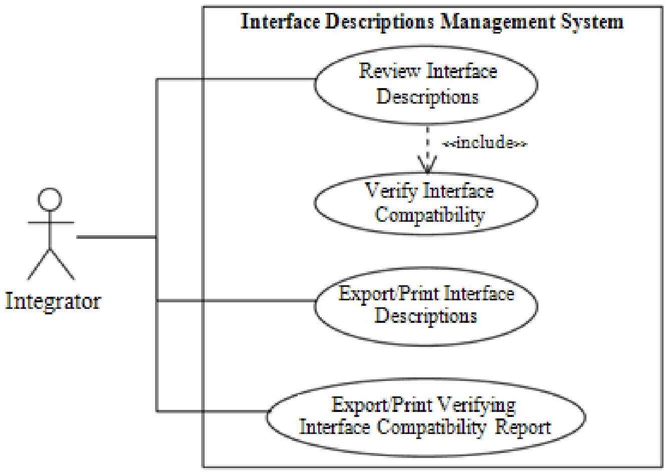

Interface Descriptions Management System
Review Interface Descriptions
<<include>>
Verify Interface Compatibility
Export/Print Interface Descriptions
Export/Print Verifying Interface Compatibility Report
Integrator

Fig. 2. Use-case diagram showing functionalities of IDMS

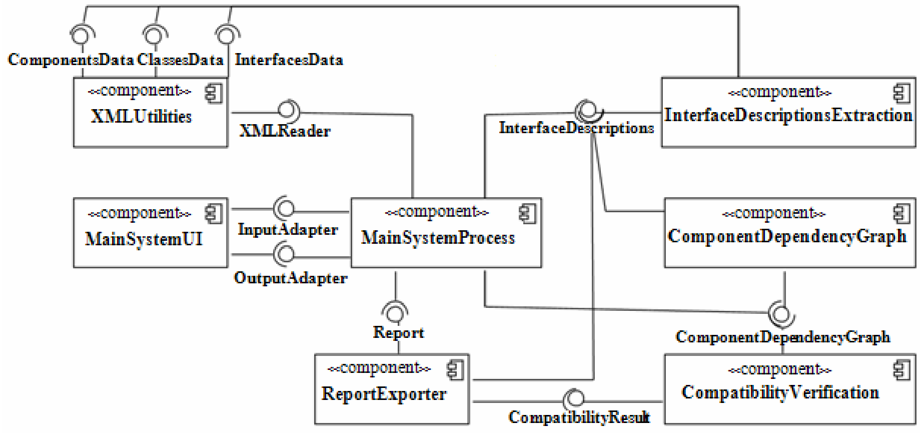

ComponentsData
ClassesData
InterfacesData
<<component>> XMLUtilities
XMLReader
InterfaceDescriptions
<<component>> InterfaceDescriptionsExtraction
<<component>> MainSystemUI
InputAdapter
OutputAdapter
<<component>> MainSystemProcess
<<component>> ComponentDependencyGraph
Report
ComponentDependencyGraph
<<component>> ReportExporter
CompatibilityResult
<<component>> CompatibilityVerification

Fig. 3. Component diagram of IDMS

The component diagram (Fig.3) illustrates the component architecture in "black box" view. It is composed of component, its interfaces and dependency relationships with other components. A component is represented by the rectangle symbol. The behavior of a component can be defined in terms of *required* interfaces and *provided* interfaces. The *provided* interfaces are depicted using "lollipop" notation, while the *required* interfaces use "socket" notation. A lollipop and a socket are represented by a circle and a semicircle respectively. Note that these interface notations indicate the dependency relationships between components by revealing the communication between components through their interfaces.

According to Fig. 3, IDMS consists of seven components namely CompatibilityVerification, ComponentDependencyGraph, InterfaceDescriptionsExtraction, MainSystemProcess, MainSystemUI, ReportExporter, and XMLUtilities. Each component is defined the dependency relationships through its interfaces in terms of *provided* and *required*. For example, the CompatibilityVerification component communicates with ReportExporter component via the provided *CompatibilityResult* interface, while interacts with ComponentDependencyGraph component via the required *ComponentDependencyGraph* interfaces.

Additional interface properties of each component are required as input to the IDMS. For example, interfaces require expanding its data members and operations as illustrated in Fig. 4.

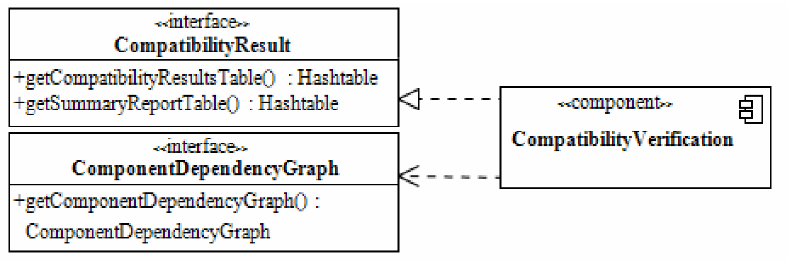
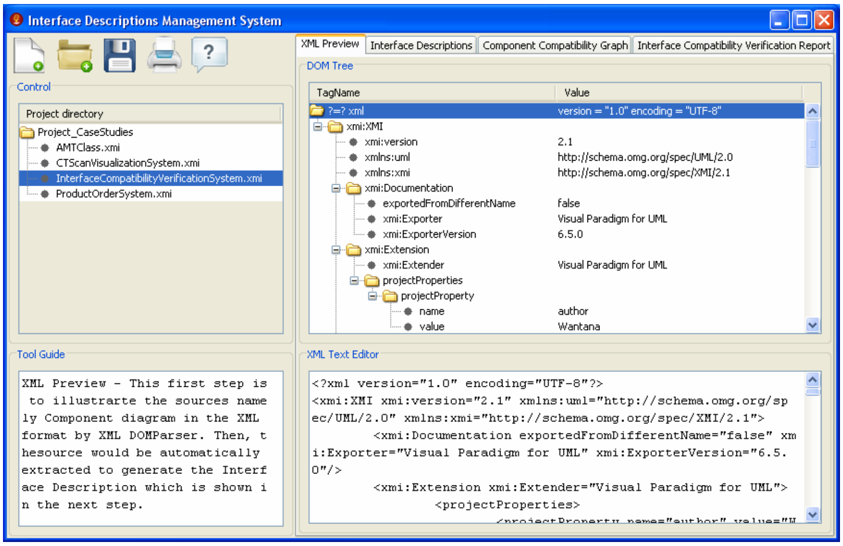

«interface»
CompatibilityResult
+getCompatibilityResultsTable() : Hashtable
+getSummaryReportTable() : Hashtable
«interface»
ComponentDependencyGraph
+getComponentDependencyGraph() : ComponentDependencyGraph
«component»
CompatibilityVerification

Fig. 4. Part of refined component diagram as input into IDMS

4.2 Implementation

IDMS is implemented in Java. The first build to extract all components' interface data from the input component diagram calls the library named DOMParser [11] to stream of XML text. Example screen demonstrating the feature of XML preview of the


Interface Descriptions Management System
Control
XML Preview
Interface Descriptions
Component Compatibility Graph
Interface Compatibility Verification Report
DOM Tree
Tool Guide
XML Text Editor

Project directory
- Project_CaseStudies
  - AMTClass.xmi
  - CTScanVisualizationSystem.xmi
  - InterfaceCompatibilityVerificationSystem.xmi
  - ProductOrderSystem.xmi

TagName	Value
?=? xml	version = "1.0" encoding = "UTF-8"
xmi:XMI	
xmi:version	2.1
xmlns:uml	http://schema.omg.org/spec/UML/2.0
xmlns:xmi	http://schema.omg.org/spec/XMI/2.1
xmi:Documentation	
exportedFromDifferentName	false
xmi:Exporter	Visual Paradigm for UML
xmi:ExporterVersion	6.5.0
xmi:Extension	
xmi:Extender	Visual Paradigm for UML
projectProperties	
projectProperty	
name	author
value	Wantana

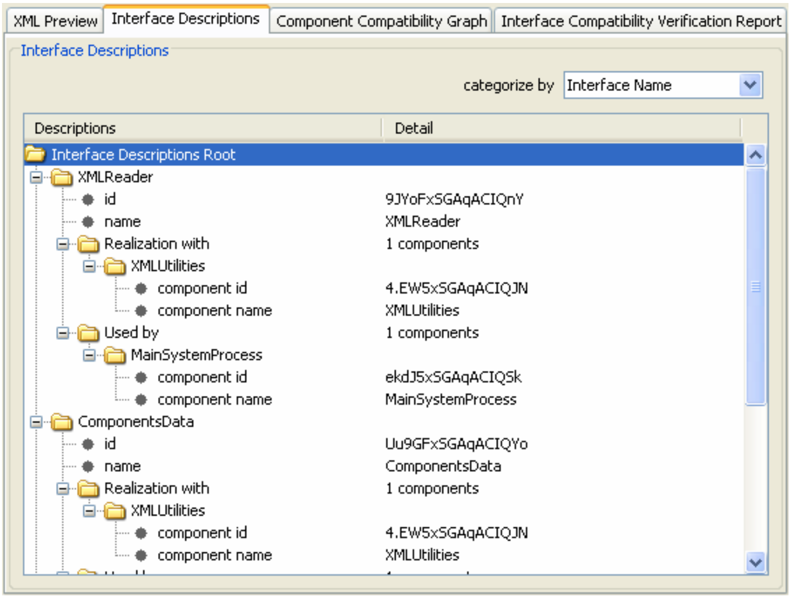
```
XML Preview - This first step is
 to illustrarte the sources name
ly Component diagram in the XML
format by XML DOMParser. Then, t
hesource would be automatically
extracted to generate the Interf
ace Description which is shown i
n the next step.
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.1" xmlns:uml="http://schema.omg.org/sp
ec/UML/2.0" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1">
        <xmi:Documentation exportedFromDifferentName="false" xm
i:Exporter="Visual Paradigm for UML" xmi:ExporterVersion="6.5.
0"/>
        <xmi:Extension xmi:Extender="Visual Paradigm for UML">
                <projectProperties>
```

Fig. 5. Example screen of the feature of XML preview of the component diagram

component diagram is depicted in Fig. 5, which contains the XML Preview tab at the right panel of Component Dependency Graph System. Once the user has imported a component diagram in XML file format, the system read this file as input, and then XML tags and values are contained in the DOM tree table window. The source of the XML file is shown in the XML Text Editor window.

Fig.6 displays an example of the interface descriptions extracted from the input component diagram (Fig.3), which has been converted into XML format.


XML Preview | Interface Descriptions | Component Compatibility Graph | Interface Compatibility Verification Report

Interface Descriptions

categorize by: Interface Name

Descriptions	Detail
Interface Descriptions Root	
XMLReader	
id	9JYoFxSGAqACIQnY
name	XMLReader
Realization with	1 components
XMLUtilities	
component id	4.EW5xSGAqACIQJN
component name	XMLUtilities
Used by	1 components
MainSystemProcess	
component id	ekdJ5xSGAqACIQSk
component name	MainSystemProcess
ComponentsData	
id	Uu9GFxSGAqACIQYo
name	ComponentsData
Realization with	1 components
XMLUtilities	
component id	4.EW5xSGAqACIQJN
component name	XMLUtilities

Fig. 6. Example screen of the extracted interface descriptions

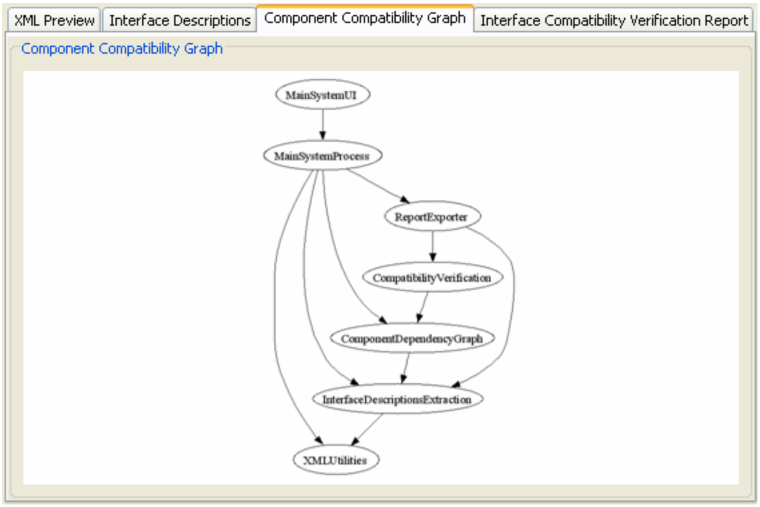
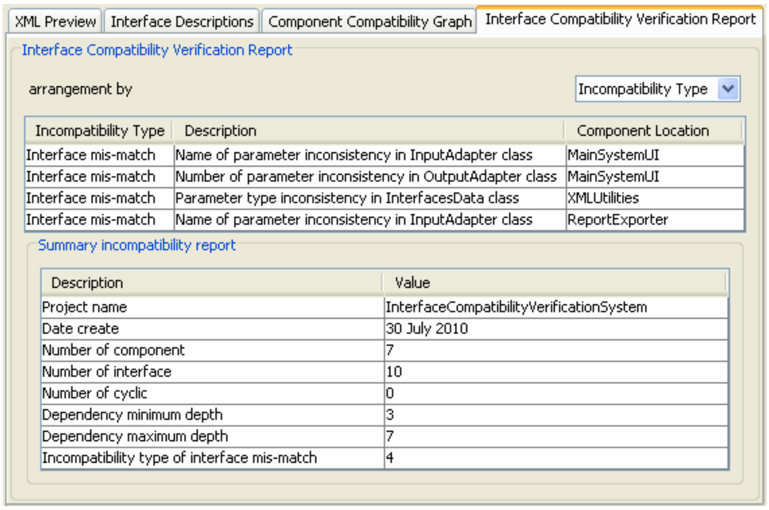

XML Preview
Interface Descriptions
Component Compatibility Graph
Interface Compatibility Verification Report
Component Compatibility Graph
MainSystemUI
MainSystemProcess
ReportExporter
CompatibilityVerification
ComponentDependencyGraph
InterfaceDescriptionsExtraction
XMLUtilities

Fig. 7. Example screen of Component Compatibility Graph

The construction of component dependency graph called CCG is implemented by calling the library named Grappa [12]. Example of CCG related to the component diagram (Fig.3) is shown in Fig.7. CCG helps verifying interface compatibility by inspecting the interface information contained in each node representing a component.

Example result of verifying component interface compatibility is reported in Fig.8.


XML Preview
Interface Descriptions
Component Compatibility Graph
Interface Compatibility Verification Report
Interface Compatibility Verification Report
arrangement by
Incompatibility Type
Summary incompatibility report

Incompatibility Type	Description	Component Location
Interface mis-match	Name of parameter inconsistency in InputAdapter class	MainSystemUI
Interface mis-match	Number of parameter inconsistency in OutputAdapter class	MainSystemUI
Interface mis-match	Parameter type inconsistency in InterfacesData class	XMLUtilities
Interface mis-match	Name of parameter inconsistency in InputAdapter class	ReportExporter

Description	Value
Project name	InterfaceCompatibilityVerificationSystem
Date create	30 July 2010
Number of component	7
Number of interface	10
Number of cyclic	0
Dependency minimum depth	3
Dependency maximum depth	7
Incompatibility type of interface mis-match	4

Fig. 8. Example screen as a result of verifying component interface compatibility

4.3 Preliminary Results

The evaluation of the IDMS implemented in this work is assessed in terms of the correctness and the usefulness. For the correctness, the system was inspected whether all features function properly. For example, the system can extract interface descriptions from the input component diagram correctly, construct the component dependency properly, and traverse the graph to generate valid results. We carried out the experiments on three case studies: IDMS itself, simple ordering product system, and Computed Tomography (CT) scan image visualization system. The preliminary results were satisfactory to the architectural designer by examining the outputs of the system. Regarding the aspect of usefulness, IDMS can be considered as the evidence of the implementation of practices recommended in the area of product integration in CMMI process improvement model. In particular, referring to the Process Implementation Indicator Descriptions (PIID) [13], the interface descriptions generated from IDMS can be considered as the Direct Artifact resulting from performing interface descriptions review for completeness (SP2.1). Regarding SP2.2, the Component Compatibility Graph can be considered as the mechanism to discover the interface mismatch among different product components, providing an evidence of partial implementation of interface management.

5 Conclusion

The belief of process management premise has encouraged the focus on improving the quality of process that will influence and result in the quality of product. Referring to many well-known process models and standards, the best practices of product integration recommend activities to ensure interface compatibility, for example, reviewing interface descriptions for completeness and managing interface throughout the product life cycle to reduce late discovery of mismatch [6]. Therefore, this research has implemented the Interface Descriptions Management System, or IDMS, to generate the interface descriptions of all components existing in the input UML component diagrams. Compared to reviewing the completeness of component interfaces using design documents, the use of the generated interface descriptions could accelerate and ease the task. In addition, IDMS is capable of inspecting interface compatibility and reporting the mismatches discovered. The Component Compatibility Graph, which is a kind of component dependency graph, is invented to support reviewing interface compatibility between associated product components. The correctness and usefulness of IDMS were subjectively evaluated on three case studies. The results were satisfactory.

Acknowledgments. This research has been funded by Thailand Graduate Institute of Science and Technology (grant number TG-01-51-084), National Science and Technology Development Agency (NSTDA) and National Electronics and Computer Technology Center (NECTEC).

References

1. Lau, K.-K.: Component-Based Software Development: Case Studies. World Scientific Publishing, Singapore (2004)
2. Stavridou, V.: Integration in software intensive systems. Journal of Systems and Software 48, 91–104 (1999)
3. Larsson, S., Myllyperkiö, P., Ekdahl, F., Crnkovic, I.: Software product integration: A case study-based synthesis of reference models. Information and Software Technology 51, 1066–1080 (2009)
4. Garg, A., Kazman, R., Chen, H.-M.: Interface descriptions for enterprise architecture. Science of Computer Programming 61, 4–15 (2006)
5. Object Management group: Unified Modeling Language: Superstructure version 2.0 (2005), `http://www.omg.org`
6. Chrissis, M.B., Konrad, M., Shrum, S.: CMMI® Second Edition Guidelines for Process Integration and Product Improvement. Addison-Wesley, Boston (2007)
7. Areeprayolkij, W., Limpiyakorn, Y., Gansawat, D.: An Approach to Verifying Interface Compatibility of Components with Component Dependency Graph. In: 2nd International Conference on Systems Engineering and Modeling (ICSEM), Bangkok, pp. 509–513 (2010)
8. Object Management group: MOF 2.0/XMI Mapping version 2.1.1 (2007), `http://www.omg.org`

9. Kanjilal, A., Sengupta, S., Bhattacharya, S.: CAG: A Component Architecture Graph. In: 2008 Technical Conference of IEEE Region 10 (TENCON 2008), Hyderabad, pp. 1–6 (2008)
10. Binder, V.R.: Testing Object-Oriented Systems: Models, Patterns, and Tools. World Addison-Wesley, Reading (2003)
11. The Apache Xerces Project, `http://xerces.apache.org/`
12. Grappa - A Java Graph Package, `http://www2.research.att.com/~john/Grappa/`
13. Ahern, D.M., Armstrong, J., Clouse, A., Ferguson, J.R., Hayes, W., Nidiffer, K.E.: CMMI® SCAMPI Distilled Appraisals for Process Improvement. Addison-Wesley, USA (2005)

Software Framework for Flexible User Defined Metaheuristic Hybridization

Suraya Masrom[1], Siti Zaleha Zainal Abidin[1],
Puteri Norhashimah Megat Abdul Rahman[1], and Abdullah Sani Abd. Rahman[2]

[1] Faculty of Computing and Mathematical Sciences
Universiti Teknologi MARA, Malaysia
{suray078,phashi655}@perak.uitm.edu.my
sitizaleha533@salam.uitm.edu.my
[2] Computer and Information Science Department
Universiti Teknologi PETRONAS, Malaysia
sanirahman@petronas.com.my

Abstract. Metaheuristic algorithms have been widely used for solving Combinatorial Optimization Problem (COP) since the last decade. The algorithms can produce amazing results in solving complex real life problems such as scheduling, time tabling, routing and tasks allocation. We believe that many researchers will find COP methods useful to solve problems in many different domains. However, there are some technical hurdles such as the steep learning curve, the abundance and complexity of the algorithms, programming skill requirement and the lack of user friendly platform to be used for algorithm development. As new algorithms are being developed, there are also those that come in the form of hybridization of multiple existing algorithms. We reckon that there is also a need for an easy, flexible and effective development platform for user defined metaheuristic hybridization. In this article, a comparative study has been performed on several metaheuristics software frameworks. The result shows that available software frameworks are not adequately designed to enable users to easily develop hybridization algorithms. At the end of the article, we propose a framework design that will help bridge the gap. We foresee the potential of scripting language as an important element that will help improve existing software framework with regards to the ease of use, rapid algorithm design and development. Thus, our efforts are now directed towards the study and development of a new scripting language suitable for enhancing the capabilities of existing metaheuristic software framework.

Keywords: optimization;metaheuristic;hybridization;scripting language; software framework.

1 Introduction

Starting from the last decade, researchers have been seriously looking into metaheuristic based techniques to solve COP based problems such as scheduling and time tabling[1]. They have been able to achieve remarkable results using the techniques. Some examples of metaheuristic algorithms are Genetic Algorithm

(GA)[2], Ant Colony Optimization (ACO)[3], Particle Swarm Optimization (PSO)[4], Tabu search[5] and Simulated Annealing[6].

Since COP relating to real life applications is categorized as a non deterministic polynomial time hard(NP-hard) problem, the process of finding the optimal solution is considered as exponential to the problem size. This translates into significant cost (time) to the performance of the algorithm. Depending on a single metaheuristic method can be very restrictive especially when applied to a real life and high complexity problems. This is the reason why hybridization has been investigated [7]. Some successful works have been achieved in metaheuristic and its variant hybridization[8-12].

This research is initiated with the focus on various aspects of the design and implementation of metaheuristics hybridization. The potential of using scripting language based application for its simple, flexible and efficient development will be explored in the hybridization design and implementation.

The remaining of the paper is organized as follows. Section 2 introduces general concepts on metaheuristic and its hybridization. Several software frameworks for the metaheuristic algorithm development is explained in section 3 followed by a comparative study on the software framework features in section 4. Before concluding, the proposed scripting language software framework designed is presented in section 5.

2 Metaheuristic Hybridization

Prior to metaheuristic, an exact heuristic method has been used in solving COP. However, as the problem size becomes larger and complex for real world cases, the method has been very time consuming and decrease in practicality[13]. So, metaheuristic has been introduced as an extension framework from the exact heuristic and local search methods.

In line with the goal, metaheuristic hybridization for algorithm improvement is becoming popular among the research community especially when dealing with real world and multi objectives Combinatorial Optimization Problem (COP). The metaheuristic research community has agreed that good metaheuristic should satisfy two equally important criteria; diversity and intensity[1, 14-16]. Diversity is associated with exploration while intensity is related to exploitation competence. It should be noted that one single metaheuristic is originally designed to act upon on one criterion only.

There are many works that have been successfully carried out regarding to metaheuristic hybridization. For example, ant colony optimization with tabu Search is introduced for solving the Open Vehicle Routing Problem [9]. Similarly, the tabu list concept in tabu search has been applied in particle swarm optimization search strategy [17]. There is also hybridization between genetic algorithm and particle swarm optimization [15, 18-20] for producing remarkable results compared to the performance of single particle swarm algorithm. In contrast, the hybridization that involves ant colony, simulated annealing and tabu search with exact heuristic techniques, often significantly more effective in terms of running time and solution quality [12].

3 Software Frameworks for Metaheuristic

Developing programming code for COP and its metaheuristic algorithm from scratch has been considered as complex [21, 22]. The process will have to go through huge works especially to those with no fundamental knowledge on computer programming concept and languages such as C, C++, C# and JAVA. Some of them are developed in object oriented programming paradigm, thus requiring additional knowledge on abstract and intricate programming concepts.

Recently, several metaheuristics software frameworks have been developed with the concern to reduce development complexity and phase cycle. Some of the available software frameworks are iOPT, HotFrame, EasyLocal^{++}, ParadisEO, HeuristicLab and JCLEC.

3.1 iOPT

The iOPT [23] software framework has been developed in JAVA. The software provides generic framework for developing scheduling applications and has a logic structure for heuristic search algorithms. The current supported metaheuristics are based on local search paradigm like simulated annealing, tabu search and Guided Local Search as well as evolutionary optimization.

The component GUI provided in iOPT allows easy and rapid application development. However, the component based interface is not originally designed for flexible user defined hybridization. As a result, the creation and addition of new metaheuristic and hybridization would involve very complicated task and process.

3.2 HotFrame

HotFrame[24] or Heuristic Optimization Framework is a software framework developed in C++ that provides both component and template based paradigm. It incorporates with several local search algorithms such as steepest descent, simulated annealing and tabu search. Generality and expendability are the major characteristics for the software design. These will allow general classes to be adaptable for new instantiation of metaheuristic and hybridization specific to users' need.

Unfortunately, HotFrame is not sufficient enough to create useful representations for many types of metaheuristic algorithms[25]. The generality of metaheuristic class is merely tailored to local search type metaheuristic. Moreover, amendment and inclusion of new metaheuristic and hybridization can be done in a complicated way that requires programming knowledge in editing the code skeleton provided.

3.3 EasyLocal++

EasyLocal++ provides a variety of hybrid local search techniques in a relatively "neat conceptual scheme" [26]. The design and analysis of local search algorithms use object oriented based software framework to accommodate the reusability aspect. The EasyLocal++'s algorithm is flexible in solving numerous scheduling problems, however, it has very complicated and difficult to follow constructions and descriptions. The metaheuristic libraries are only limited to local search strategies, which consist of hill climbing, tabu search and simulated annealing. Extensibility and hybridization in EasyLocal++ are also restricted to the above search strategies only.

3.4 ParadisEO

ParadisEO [27] is a white-box object oriented framework which provides a broad range of features including evolutionary algorithms, local search, hybridization, parallel and distribution models. The specific setting for metaheuristic and application can be easily configured from the component based interface. Code templates in the form of text editor will be generated regarding to the users setting which is free for further editing and customization. Working with ParadisEO requires users to have fundamental knowledge and understanding of the object oriented programming concept.

3.5 HeuristicLab

HeuristicLab[28] sofware design describes a very generic and extensible framework, flexible for new metaheuristic and problem definition. Fig.1 shows the general class abstraction in HeuristicLab.

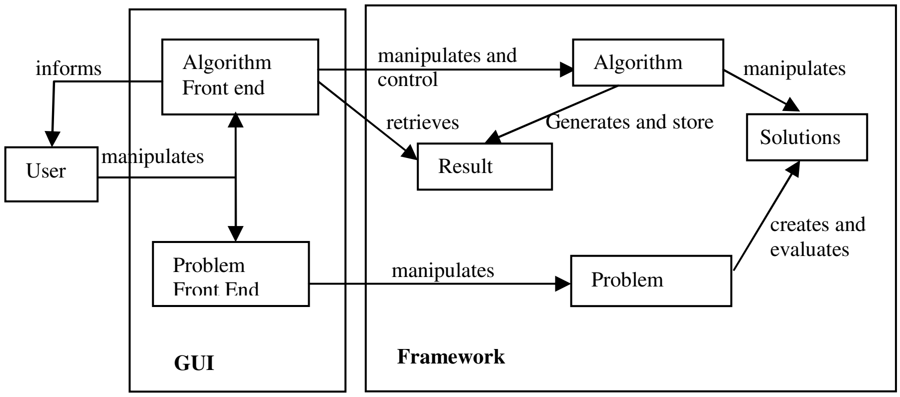


Fig. 1. Interaction between HeuristicLab classes [28]

The component based GUI facilitates users for new algorithm and problem description. Based on the setting, automated codes generation and compilation will be implemented at the back end.

HeuristicLab has also introduced new plug-in concept in the framework. Although users are given flexibility in choosing variety of plug-ins, they will find difficulties to combine (hybridize) different plug-ins from the internal structure. In actuality, the users need to create new plug-in for the hybridization purposes before execute it independently. Finally, the users would have to install it to the framework as a new plug-in.

3.6 JCLEC

JCLEC software framework has two main concerns; maximum reusability and minimum programming efforts[29]. The architecture has been divided into three; *system core*, *experiment runner*, and *GenLab*. The *system core* is where all the systems and algorithms definition are built in. The *experiment runner* facilitates Evolutionary

Algorithms (EAs) execution configuration. *GenLab* is the GUI for general user windows setting as well as results visualizations. The *experiment runner* can be seen as a simple EA scripting environment.

With respect to hybridization, JCLEC only allows combination of different EA algorithms which consist of genetic algorithm, genetic computing, evolutionary algorithm and evolutionary computing.

4 Limitations of the Available Software Frameworks

Most of the available software frameworks provide a variety of software library collections for some metaheuristics. However, they either have none or very limited hybridization capabilities. Some of them focus on either local search or evolutionary algorithms only. As a result, hybridization is restricted within the limited metaheuristics.

In most of the software frameworks, amendment or creation of new hybridization would require the programmer to write new code either from scratch or by customizing the provided template. In any case, it is imperative that the programmer have deep understanding of the class libraries. On the other hand, component based interface applied by most software framework provide a convenient way for users to define their problem and algorithm using the GUI. None of currently available frameworks provide user defined metaheuristics hybridization capability. This is due to the restriction in the component based GUI in giving detail representation of classes, functions flow and data exchange among metaheuristic classes [30]. Table 1 summarizes all the software framework features with their strengths and limitations.

Table 1. Features comparison among software frameworks

Name	**Paradigm**	**Usability**	**Extendibility**	**Dynamic configuration**	**User defined Hybridization**
iOPT[23]	Component	General	Low	Static	Not provided
Hotframe[24]	Template	Expert	Moderate	Static	Low
EasyLocal++[26]	Template	Expert	Moderate	Static	Low
ParadisEO [27]	Template	Expert	Moderate	Static	Low
HeuristicLab[28]	Component or Template	General or Expert	High	Static	Low
JCLEC[29]	Scripting language or component	General	Low	Dynamic	Not provided

Most software frameworks use similar programming paradigm that is either component based, template based, scripting language or command line. This paradigm refers to the method used in the programming interface including program setting and configuration. Component based paradigm offers Graphical User Interface (GUI) and drag and drop mechanisms. The template based paradigm require users to include and customize existing code skeleton according to their need.

The usability feature is the ease of use level for the software framework without user prior training. The levels are categorized as either *Expert* or *General* user. The *Expert* refers to someone who is having a high programming competency, while the *General* can include users from non-related information technology (IT) fields.

One important feature of software framework is extendibility. It refers to the possibility of having users to add non standard metaheuristic algorithms to the existing software libraries. The platform is deemed highly extendable if the users are not required to have extensive knowledge in computer programming and they are able to contribute to the software framework. In contrast, moderate level is associated with template based paradigm which requires some basic knowledge of computer programming for doing code modifications. Extending software framework by developing new program would be considered as low extendibility.

The software framework is considered as dynamically configured if a user is only asked to write a single configuration file to describe any particular problem. He/she will be able to test several algorithms at the same time using the same existing configuration file.

User defined hybridization is defined by the possibility of merging two or more metaheuristic algorithms together to create a new one. Each package is assessed by how easy it is for the user to create new hybrid algorithms without prior formal training on the software framework.

5 Scripting Language Paradigm

Scripting language is an alternative programming method for easy, effective and rapid system development. It has became popular because of the simple language structure and strong functionality[31]. Moreover, scripting language can be designed to have the capabilities for creating dynamic memory management, powerful data type and data structure comparable to other programming languages [32].

Scripting language has been effectively deployed in some successful applications [33-37]. In these applications, the language operates as an interface paradigm for C++ and JAVA libraries as well as Visual Basic and Cobra components. The scripting language is also effectiveness in defining the integration rules among agents in distributed systems [35].

5.1 Scripting Language for Metaheuristic

Among the available software frameworks for metaheuristic, JCLEC[29] is the only one that implements scripting language paradigm. Its design however, is merely directed towards EAs algorithms and dynamic configurations capability only. We propose a framework, equipped with a compiled scripting language capability along with two other capabilities. First, our framework will provide metaheuristic generalization capability which will make it highly extensible with respect to problems and algorithms. Second, the scripting language will be designed so that metaheuristics hybridization will become easy to users. We were inspired by the design of the HeuristicLab[28] with respect to the generic and extensible framework. However, we reckon that two extensions are necessary to make hybridization possible; hybridization class abstraction for enabling easy user defined hybridization and operator class abstraction for general operation definition. Fig. 2 shows the class interaction in the proposed framework.

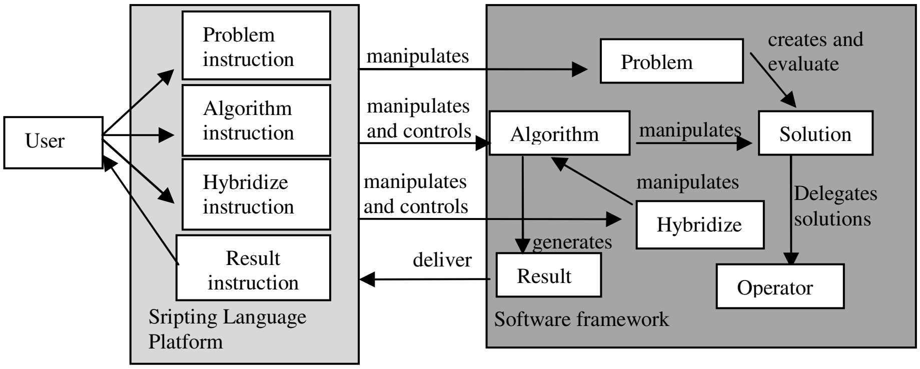


Fig. 2. The classes interaction in the proposed software framework

There are six general abstraction classes defined in the software frameworks as `Algorithm`, `Problem`, `Solution`, `Hybridize`, `Operator` and `Result`. Scripting language command can be used by users at the front end to inherit, manipulate and control the particular classes. Inherit process permits users to instantiate new class of algorithm, problem and hybridization process. Selection of metaheuristic for hybridization can be manipulated from the `Hybridize` class that communicates with `Algorithm`. There are many common metaheuristics that have been pre inherited from the `Algorithm` class such as particle swarm, ant colony, tabu search, simulated annealing and genetic algorithm. The implementation of the scripting language constructs will be embedded into the JACIE(Java Authoring for Collaborative Interaction Environment) [37, 38] software environment.

6 Conclusion

Over the last decade, metaheuristics have become a substantial part of the optimization software frameworks. Ready to use programs from software framework or class libraries related to metaheuristics are being developed increasingly. Until recently however, most of them are meant for expert users with advance knowledge in programming language. Component based GUI has been introduced to reduce difficulties but it still does not produce easy and flexible user defined hybridization capabilities. Single metaheuristic can sometimes be quite limited in providing good and fast solution for real case of COP. It is therefore very important to have a flexible metaheuristics hybridization development environment. Scripting language, defined as a simplified computer programming language, has the potential in providing the desired environments for metaheuristics development. It can offer the possibility for novice users to define metaheuristics hybridization themselves. The implementation process is easier than the template based method. Moreover, with the scripting language, dynamic application configuration for several metaheuristics can all be written in a single file. This will help create a more effective development platform for metaheuristics hybridization.

Acknowledgement

The authors would like to thank Universiti Teknologi MARA for its financial support to this project.

References

1. Affenzeller, M., Beham, A., Kofler, M., Kronberger, G., Wagner, S.A., Winkler2, S.: Metaheuristic Optimization. In: Buchberger, B., Affenzeller, M., Ferscha, A., Haller, M., Jebelean, T., Klement, E.P., Paule, P., Pomberger, G., Schreiner, W., Stubenrauch, R., Wagner, R., Weiß, G., Windsteiger, W. (eds.) Hagenberg Research, pp. 103–155. Springer, Heidelberg (2009)
2. Affenzeller, M., Winkler, S., Wagner, S., Beham, A.: Genetic Algorithms and Genetic Programming - Modern Concepts and Practical Applications. CRC Press, Boca Raton (2009)
3. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press Ltd., Cambridge (2004)
4. Clerc, M.: Particle Swarm Optimization. In: ISTE (2006)
5. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic, Dordrecht (1998)
6. Laarhoven, P.J.M.V., Aarts, E.H.L.: Simulated Annealing: Theory and Applications (Mathematics and Its Applications). Kluwer Academic Publishers Group, Dordrecht (1988)
7. Blum, C., Roli, A.: Hybrid Metaheuristics: An Introduction. In: Blum, C., Aguilera, M.J.e.B., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics, vol. 114, pp. 1–30. Springer, Heidelberg (2008)
8. Chiarandini, M., Birattari, M., Socha, K., Rossi-Doria, O.: An effective hybrid algorithm for university course timetabling. Journal of Scheduling 9(5), 403–432 (2006)
9. Li, X.-Y., Tian, P., Leung, S.: An ant colony optimizationmetaheuristic hybridized with tabu search for open vehicle routing problems. Journal of the Operational Research Society 60, 1012–1025 (2009)
10. Martens, A., Ardagna, D., Koziolek, H., Mirandola, R., Reussner, R.: A Hybrid Approach for Multi-attribute QoS Optimisation in Component Based Software Systems. In: Heineman, G.T., Kofron, J., Plasil, F. (eds.) Research into Practice – Reality and Gaps. LNCS, vol. 6093, pp. 84–101. Springer, Heidelberg (2010)
11. Adewumi, A.O., Sawyerr, B.A., Ali, M.M.: A heuristic solution to the university timetabling problem. Engineering Computations 26(7-8), 972–984 (2009)
12. Raidl, G.u.R., Puchinger2, J.: Combining (Integer) Linear Programming Techniques and Metaheuristics for Combinatorial Optimization. In: Blum, C., Aguilera, M.J.e.B., Roli, A., Sampels, M. (eds.) Hybrid Metaheuristics, vol. 114, pp. 31–62. Springer, Heidelberg (2008)

13. Talbi, E.-G.: Metaheuristics: From Design to Implementation. Wiley, Chichester (2009)
14. Wei-min, Z., Shao-jun, L., Feng, Q.: θ -PSO: a new strategy of particle swarm optimization. Journal of Zhejiang University - Science A 9(6), 786–790 (2008)
15. Tiew-On, T., Rao, M.V.C., Loo, C.K., Sze-San, N.: A new class of operators to accelerate particle swarm optimization. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 2404, pp. 2406–2410 (2003)
16. Shuang, B., Chen, J., Li, Z.: Study on hybrid PS-ACO algorithm. Applied Intelligence (2009)
17. Wen, W., Liu, G.: Swarm Double-Tabu Search. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 1231–1234. Springer, Heidelberg (2005)
18. Stacey, A., Jancic, M., Grundy, I.: Particle swarm optimization with mutation. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 1422, pp. 1425–1430 (2003)
19. Wei, X., Xingsheng, G.: A hybrid particle swarm optimization approach with prior crossover differential evolution. In: Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation. ACM, New York (2009)
20. Matthew, S., Terence, S.: Breeding swarms: a GA/PSO hybrid. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation. ACM, New York (2005)
21. Wagner, S., Winkler, S., Pitzer, E., Kronberger, G., Beham, A., Braune, R., Affenzeller, M.: Benefits of Plugin-Based Heuristic Optimization Software Systems. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2007. LNCS, vol. 4739, pp. 747–754. Springer, Heidelberg (2007)
22. Jos Garc, N.a., Enrique, A., Francisco, C.: Using metaheuristic algorithms remotely via ROS. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. ACM, New York
23. Voudouris, C., Dorne, R., Lesaint, D., Liret, A.: iOpt: A Software Toolkit for Heuristic Search Methods. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 716–729. Springer, Heidelberg (2001)
24. Fink, A., Voß, S.: Hotframe: A Heuristic Optimization Framework. In: Optimization Software Class Libraries. Operations Research/Computer Science Interfaces Series, vol. 18, pp. 81–154. Springer, US (2002)
25. Ciarleglio, M.I.: Modular Abstract Self-Learning Tabu Search (MASTS) Metaheuristic Search Theory and Practice. PhD, University of Texas, Austin (2009)
26. Gaspero, L.D., Schaerf, A.: EASYLOCAL++: an object-oriented framework for flexible design of local search algorithms. Software-Practice and Experience, 1–34 (2003)
27. Cahon, S., Melab, N., Talbi, E.G.: ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. Journal of Heuristics - Special Issue on New Advances on Parallel Meta-Heuristics for Complex Problems 10, 357–380 (2004)
28. Wagner, S., Affenzeller, M.: HeuristicLab: A Generic and Extensible Optimization Environment. In: Ribeiro, B., Albrecht, R.F., Dobnikar, A., Pearson, D.W., Steele, N.C. (eds.) Adaptive and Natural Computing Algorithms, pp. 538–541. Springer, Vienna (2005)
29. Ventura, S., Romero, C., Zafra, A., Delgado, J.A., Hervás, C.: JCLEC: a Java framework for evolutionary computation. In: Soft Computing - A Fusion of Foundations, Methodologies and Applications, Engineering, vol. 12. Springer, Heidelberg (2008)
30. Kramer, J., Magee, J.: Dynamic Configuration for Distributed Systems. IEEE Transactions on Software Engineering SE-11(4), 424–436 (1985)
31. Abidin, S.Z.Z.: Interaction and Interest Management in a Scripting Language. In: Computer Science, University of Wales, Swansea (2006)
32. Spinellis, D.: Java makes scripting languages irrelevant? Software 22(3), 70–71 (2005)

33. Robert, M.S., Denis, D., Kenneth, G.F., Amol, J.: Extending a scripting language for visual basic forms. SIGPLAN Not. 40(11), 37–40 (2005)
34. Tongming, W., Ruisheng, Z., Xianrong, S., Shilin, C., Lian, L.: GaussianScriptEditor: An Editor for Gaussian Scripting Language for Grid Environment. In: Eighth International Conference on Grid and Cooperative Computing, GCC 2009, pp. 39–44 (2009)
35. Hua, X., Qingshan, L., Yingqiang, W., Chenguang, Z., Shaojie, M., Guilin, Z.: A Scripting Language Used for Defining the Integration Rule in Agent System. In: IEEE International Conference on e-Business Engineering, ICEBE 2008, pp. 649–654 (2008)
36. Winroth, H.: A scripting language interface to C++ libraries. In: Technology of Object-Oriented Languages and Systems, TOOLS 23, Proceedings, pp. 247–259 (1997)
37. Abidin, S.Z.Z., Chen, M., Grant, P.W.: Managing interaction for multimedia collaboration - through the keyholde of noughts and crosses games. In: IEEE International Symposium on Multimedia Software Engineering, pp. 132–135
38. Haji-Ismail, A.S., Min, C., Grant, P.W., Kiddell, M.: JACIE-an authoring language for rapid prototyping net-centric, multimedia and collaborative applications. Multimedia Software Engineering, 385–392 (2000)

Program Visualization for Debugging Deadlocks in Multithreaded Programs[⋆]

Byung-Chul Kim and Yong-Kee Jun[⋆⋆]

Gyeongsang National University, Jinju, 660-701, South Korea
{bckim,jun}@gnu.ac.kr

Abstract. Debugging deadlocks in multithreaded programs is a notoriously difficult task. A key reason for this is to understand the high behavioral complexity resulting from the inherent nondeterminism of multithreaded programs. We present a novel visualization technique which abstracts the nested patterns of locks and represents the happens-before relation of the patterns. We implement the technique in a prototype tool for Java, and demonstrate its power using a number of multithreaded Java programs. The experimental result shows that this graph provides a simple yet powerful representation to reason about deadlocks in an execution instance.

Keywords: Multithreaded programs, deadlock debugging, visualization.

1 Introduction

In multithreaded programs with nested locks, a deadlock blocks a set of threads forever when a thread in the set is trying to acquire a lock already held by another thread in the set. Unfortunately, debugging deadlocks in multithreaded programs is a notoriously difficult task due to *non-repeatability* and *high behavioral complexity* of multithreaded programs. To address such non-repeatability problem, a number of efficient techniques [1,2,3] have been developed to detect various conditions which can leads to deadlocks in multithreaded programs. Despite advances on the non-repeatability problem, the other problem remains – a trace of an execution with bugs can be complicated and large.

Visualization [4,5] has played a critical role in helping to develop intuition about non-deterministic constructs and programming errors in multithreaded programs. Many approaches use traces to visualize a partial-order relation [6] of events for representing the behavior of multithreaded programs. Previous visualization, however, is hard to understand the nested patterns in which locks

⋆ This research was supported in part by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency), NIPA-2010-(C1090-1031-0007).

⋆⋆ Corresponding author: In Gyeongsang National University, he is also involved in the Research Institute of Computer and Information Communication (RICIC).

are acquired and released by each thread because they represent only a partial-order relation of locking operations.

We present a novel visualization technique which represents the nested patterns of locks on a causality graph, called *Lock-Pattern Causality Graph (LPCG).* Nodes in the graph represent the nested patterns of locks and each edge in the graph represents the happens-before relation between patterns. A nested pattern of locks is graphically represented by a set of rectangles with a layered structure, where each rectangle represents a lock in the pattern. Edges are depicted by a line with an arrow. We implement the technique in a prototype tool, and demonstrate its power using a number of multithreaded Java programs. The demonstration shows that this graph provides a simple yet powerful representation to reason about deadlocks in an execution instance.

This paper is organized as follows. Section 2 describes an execution model of multithreaded programs. Section 3 presents the LPCG presented by this paper. Section 4 demonstrates our visualization on some benchmark programs. Section 5 describes the related works. Finally, conclusion and future works on this paper are given in Section 5.

2 An Execution of Multithreaded Programs

An execution of a multithreaded program is represented by a trace σ which is a finite sequence $e_1, e_2, ..., e_n$ of events. An event $e \in \sigma$ is a tuple (t, action, s), where t is the identity of the thread generating this event, s is the line number where this event is generated, and *action* is one of $fork(u)$, $join(u)$, $lock(l)$, and $unlock(l)$. The action $fork(u)$ creates a new thread with identifier u. The action $join(u)$ blocks until the thread with identifier u terminates. The actions $lock(l)$ and $unlock(l)$ respectively acquire and release a lock with identifier l.

The events in a multithreaded execution can be ordered by happens-before relationship. If an event e_i happens before an event e_j, then e_i must occur before e_j in all feasible executions of the execution. Given a trace $\sigma = e_1, e_2, ..., e_n$, the happens-before relation $e_i \rightarrow e_j$ for σ is the smallest transitively-closed relation satisfying the following conditions:

- $Thread(e_i) = Thread(e_j)$ and $i < j$.
- e_i has action $fork(Thread(e_j))$.
- e_j has action $join(Thread(e_i))$.

The nested-by relation of locks is defined as an order in which locks are acquired and released by threads. If a lock m is nested within a lock l, denoted $\langle l \langle m \rangle \rangle$, a thread acquires lock l before acquiring lock m and releases lock m before releasing lock l. A root lock is a lock which is not nested by a lock, and a leaf lock does not have a lock within it. A lock pattern is a pattern in which a thread acquires and releases a root lock and locks nested within the root lock. Given a lock pattern P, the nested level of a lock $l \in P$, denoted by $Level(l, P)$, is

main thread

```
001: t1 = fork();
002: lock(l2);
003:   lock(l3);
004:   unlock(l3);
005:   lock(l4);
006:   unlock(l4);
007: unlock(l2);
008: t2 = fork();
009: join(t1);
010: lock(l3);
011:   lock(l2);
012:   unlock(l2);
013: unlock(l3);
```

t_1 thread

```
101: t3 = fork();
102: lock(l1);
103:   lock(l2);
104:     lock(l3);
105:     unlock(l3);
106:   unlock(l2);
107: unlock(l1);
108: join(t3);
```

t_2 thread

```
201: lock(l1);
202:   lock(l3);
203:     lock(l2);
204:     unlock(l2);
205:   unlock(l3);
206: unlock(l1);
207: lock(l1);
208:   lock(l4);
209:   unlock(l4);
210: unlock(l1);
```

t_3 thread

```
301: lock(l2);
302:   lock(l3);
303:   unlock(l3);
304: unlock(l2);
305: lock(l1);
306:   lock(l4);
307:   unlock(l4);
308: unlock(l1);
```

Fig. 1. A trace collected from a execution of a multithreaded program which consists of four threads

the number of locks which nest lock l, and the breadth of a lock $l \in P$, denoted by $Breadth(l, P)$ is the number of leaf rocks in lock l. The nested depth of P, denoted by $Depth(P)$, is the maximum number of $Level(l, P)$, where $l \in P$.

Figure 1 shows a trace collected from an execution of a multithreaded program with four threads and four locks. Thread *main* has two lock patterns such that are $\langle l_2\langle l_3\rangle\langle l_4\rangle\rangle$, and $\langle l_3\langle l_2\rangle\rangle$. Before obtaining the first lock pattern, *main* thread creates t_1 thread and before obtaining the second lock pattern, *main* thread creates t_2 and joins t_1. Thread t_1, created by *main* thread, has one lock pattern $\langle l_1\langle l_2\langle l_3\rangle\rangle\rangle$ after creating t_3 thread and before joining t_3 thread. After having a lock pattern $\langle l_1\langle l_3\langle l_2\rangle\rangle\rangle$, t_2 thread has another lock pattern $\langle l_1\langle l_4\rangle\rangle$. While having the lock pattern $\langle l_2\langle l_3\rangle\langle l_4\rangle\rangle$ obtained by *main* thread has the nested depth of one, the nested pattern $\langle l_1\langle l_2\langle l_3\rangle\rangle\rangle$ by t_1 thread has the nested depth of two.

3 Program Visualization

In this section, we describe the lock-pattern causality graph presented by this paper and explain the usefulness of the graph for debugging deadlocks in multithreaded program.

3.1 Lock-Pattern Causality Graph (LPCG)

We represent the behavior of multithreaded programs on a causality graph, called *Lock-Pattern Causality Graph (LPCG)*. A LPCG G is a pair (V, R) where V is a set of nodes and $E \subseteq \{(u, v)|u, v \in V \wedge u \neq v\}$ is a set of directed edges. Each node in V encodes a tuple (t, P, φ), where t is a thread, P is a lock pattern, and φ is one of the following types: *a) terminal*: a node starting and terminating the execution of a thread, *b) fork*: a node generating a child thread, *c) join*: a

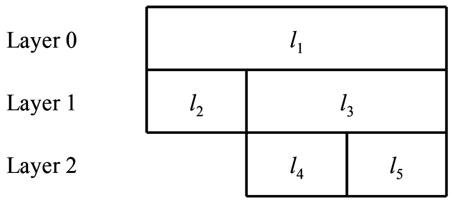


Fig. 2. A layered icicle diagram representing lock pattern $\langle l_1 \langle l_2 \rangle \langle l_3 \langle l_4 \rangle \langle l_5 \rangle \rangle \rangle$

node waiting for a thread to terminate, and *d*) *lock pattern*: a node representing a lock pattern. Each edge in E has a tuple (s, d, λ), where s is a source node, d is a destination node, and λ is one of the following types: *a*) *synchronization*: $s.t \neq d.t$, and *b*) *continuation*: $s.t = d.t$.

The lock pattern of a lock node is shown in the form of a layered icicle diagram. The diagram represents a lock with nested level i as a box labeled by the identifier of the lock in a layer with depth i. The depth of a layer is equal to the nested level of a lock in the layer. The nested relation of locks in a lock pattern is represented by the overlapped relation of boxes. If there two locks l and m in a lock pattern such that l is nested by m, then the box corresponding lock l is overlapped by the box corresponding lock m. For example, Figure 2 shows the layered icicle diagram which represents lock pattern $\langle l_1 \langle l_2 \rangle \langle l_3 \langle l_4 \rangle \langle l_5 \rangle \rangle \rangle$. Lock l_1 which is the root lock in the pattern is located on the top layer `0`, locks l_2 and l_3, whose nested level is 1, are located on layer `1` , and locks l_4 and l_5, whose nested level is 2, are placed on layer `2`. Lock l_2 and l_3 are nested by lock l_1, but since lock l_4 and l_5 are overlapped by only lock l_3, they are nested by lock l_3.

Nodes other than lock nodes are depicted as rounded rectangles, each labeled with the identifier of a thread. Each terminal node is labeled with the identifier of a thread of which starts and terminates the execution. Each fork node is labeled with the identifier of a thread which is created and each join node is labeled with the identifier of a thread which is joined back. Edges are represented by an arrow heading to its destination node from its source node. While each continuation edge is depicted by a solid line, each synchronization edge is depicted by a dotted line.

Figure 3 shows the LPCG that represents the execution trace of the Figure 1 program. Through the graph, we can capture the nested relation of locks as well as the happens-before relation of events in the execution. For example, lock l_2 is nested by locks l_1 on nodes `C` and `F` and l_3 on nodes `E` and `F`, and nests locks l_3 on nodes `B`, `C`, and `D` and l_4 on node `D`. The LPCG shows the happens-before relation between lock patterns by imposing a partial order on nodes. Given a LPCG $G = (V, E)$ and two nodes $n_i, n_j \in V$ corresponding to lock patterns P_i and P_j, respectively, there exists a path from n_i to n_j on the graph, if and only if $P_i \rightarrow P_j$. In Figure 3, node `A` happens before node `E` but does not node `F`.

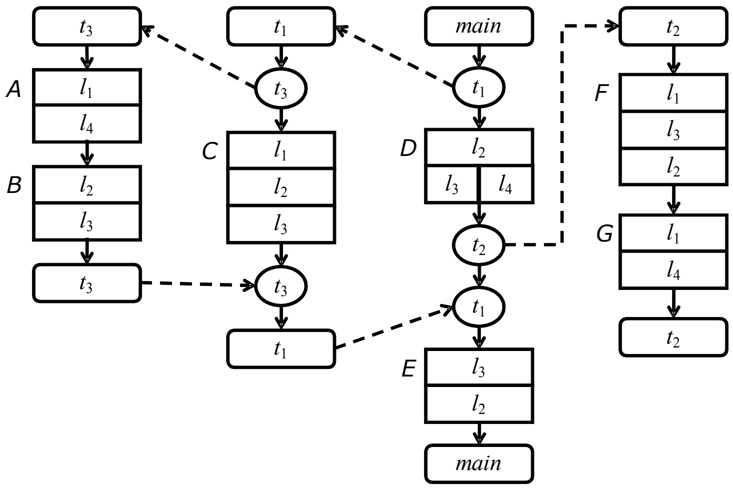


Fig. 3. A lock-pattern causality graph representing the execution in Figure 1

3.2 Debugging Deadlocks

Some practical detection techniques [7,3] detect a deadlock in the execution of Figure 1 when threads t_2 and t_3 try to acquire lock l_2 at `203` and lock l_3 at `302`, respectively. They report the detected deadlock with information including threads (t_2, t_3), locks (l_2, l_3), and line numbers (`203`, `302`) of source code. With the deadlock information, it is severely hard to correct the deadlock. One may change the nested relation of locks in thread t_2 from $\langle l_3 \langle l_2 \rangle\rangle$ to $\langle l_2 \langle l_3 \rangle\rangle$ for correcting the deadlock. The correction, however, causes a new deadlock because *main* thread acquires and releases the locks in reverse order, $\langle l_3 \langle l_2 \rangle\rangle$. Similarly, changing the nested relation of locks in thread t_3 produces a new deadlock.

The LPCG helps to understand how locks are utilized in a multithreaded program and, therefore, to help to correct deadlocks. In Figure 3, l_2 and l_3 form two nested patterns. One pattern is $\langle l_2 \langle l_3 \rangle\rangle$ in nodes `A`, `C`, and `D` of t_3, t_1, and *main* threads respectively, and the other pattern is $\langle l_3 \langle l_2 \rangle\rangle$ in node `E` in *main* thread and node `F` in t_2 thread. Node E does not involve a deadlock because the node happens before nodes `A`, `C`, and `D`. since node `F` happens before node `D`, the nodes can not involve a deadlock. In addition, since node `F` and node `C` have the same lock l_1 before acquiring locks l_2 and l_3, node `F` cannot form a deadlock with node `C`. Therefore, it is two nodes `A` and `F` that are involved in a deadlock, because there are no a path between two nodes nor a lock nesting the locks. There are several ways for correcting the deadlocks. However, the fact that can know from the graph is that the threads in the graph acquire lock l_1 before acquiring other locks. The fact implies that lock l_1 is used as a gate lock for preventing a deadlock. Therefore, the simple and best way to correct the deadlock is to add and force lock l_1 to nest the locks involved in the deadlock.

4 Demonstrations

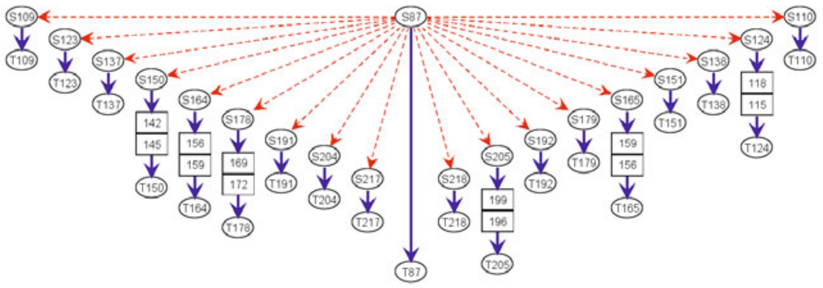
The technique presented in this paper is implemented in a prototype tool for multithreaded Java programs. The tool consists of three modules: a instrumentation module, detection module, and a visualization module. The instrumentation module instruments the bytecode class files of a compiled Java program by inserting callback functions before or after fork, join, lock, and unlock operations. When executed, the callback functions generate events, which are traced in thread-local files. The detection module applies existing Goodlock algorithm [8,1] to the generated events. The algorithm uses the events to construct a lock graph and then find the existence of a cycle on the graph by traversing. If a cycle is detected on the graph, the detection module reports that locks in the cycle are involved in a deadlock. The visualization module provides a deadlock view containing information of detected deadlocks and a graph view depicting the lock-pattern causality graph constructed from the traced events. The graph view highlights a deadlock, which is designated in deadlock view, on the graph. When constructing the graph, we filter out lock-patterns with nested depth of 0 because the patterns consists of only a single lock with no nested lock. The simple filtering policy causes the number of nodes in the graph to decrease significantly.

We have demonstrated the technique on Java Collections Frameworks. We have performed our experiments on Intel Core2 Duo 2.5Ghz with 2GB of RAM. Figure 4 shows the LPCGs constructed from the executions of programs using List and Map package in Java Collection Frame. We have found a single deadlock in each program. From the graph depicting the program using List package, we can see that while thread 164 has a locking pattern $\langle 156 \langle 159 \rangle \rangle$, thread 165 has a locking pattern $\langle 159 \langle 156 \rangle \rangle$. The graph depicting the program using Map package describes that while thread 468 has a locking pattern $\langle 452 \langle 455 \rangle \rangle$, thread 469 has a locking pattern $\langle 455 \langle 452 \rangle \rangle$. The graphs delineates that the locks involved in the deadlock are not utilized by other threads as well as the threads involved in the deadlock dose not have nested locks any more. Therefore, in order to correct the deadlock, we can focus on the locks and threads involved in the deadlock.

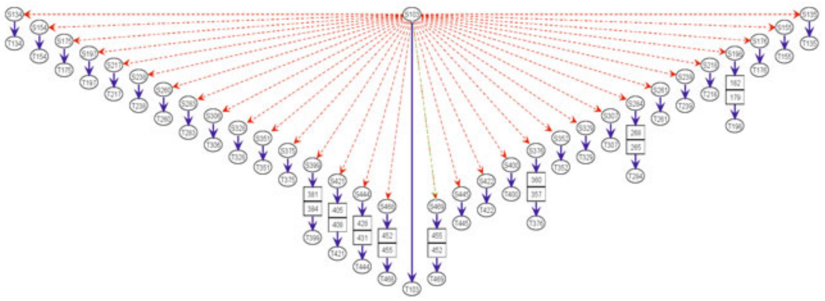
5 Related Work

Many approaches have employed visualization to understand the alternate feasible executions of multithreaded programs for debugging deadlocks in such programs.

Some of these approaches use the UML (Unified Modeling Language) paradigm which is the standard for visual modeling of object-oriented systems. Javis [9] is a visualizing and debugging tool for multithreaded Java programs. Javis visualizes only the actual deadlocks detected from traces on two diagrams, which are the sequence diagram and the collaboration diagram. Javis supports abstraction, showing only the objects directly involved in the deadlock. Jacot [10] is also a dynamic visualization tool for multithreaded Java programs. The tool has the UML sequence diagram and the thread state diagram to depict the interaction between objects and the interleaving of threads over the flow of time.



(a) A LPCG for a program using List package



(b) A LPCG for a program using Map package

Fig. 4. LPCGs which are constructed from the executions of programs using List and Map packages in Java Collection Frame

The standard sequential diagram of UML is insufficient for representing the happens-before relation between events in multithreaded programs [11]. An extension technique [11] of the sequential diagram is presented in order to address shortcomings of the diagram. The technique adds some notations for threads as executable tasks and the happens-before relation between events.

MutexView [12] has two size-varying circles to represent the relationship between threads and locks of the POSIX thread programming library on the KSR system. A thread is indicated with a small circle distinguished by color and the locks are represented with big circles. When a thread is trying to acquire the lock, the small circle of the thread appears somewhere around the circle corresponding to the lock. When a thread gets the lock, it moves into the circle and remains there until it releases the lock. Then it leaves the circle and disappears. MutexView animates such situations and represents the only deadlocks which actually occur at runtime.

The History Graph Window [13] shows the execution history of all threads. This tool shows each thread with a history bar running left to right. Each history bar has a color coded with green, blue, or red for running, joining or blocked by a synchronization primitive, respectively. The History Graph Window has several tags, each of which is related to a monitor, to visualize the behavior of locking operations on-the-fly.

The lock-causality graph [14] represents alternate orders over locking operations for helping to identify and debug potential deadlocks in multithreaded programs. The graph uses three types of symbols, which are $\sqcap$, $\sqcup$ and $\vee$ to depict lock acquire, lock release, and lock wait and notify, respectively. And the graph uses three kinds of arrow symbols, which are solid, dashed, and dotted arrows to depict the threading operations. The graph, however, does not provide the lock pattern abstraction of this paper.

6 Conclusion and Future Works

Debugging multithreaded programs continues to be an area of great interest as multithreaded runtime environments become more powerful, complex, and unpredictable. The work presented in this paper provides a powerful method of understanding the behavior of multithreaded program with nested locks and helping to debug deadlocks in the programs. A causality graph, called *Lock-Pattern Causality Graph (LPCG)*, is graphically represented to understand the happens-before relation and nested patterns of locks. The demonstration shows that this graph provides a simple yet powerful representation to reason about deadlocks in an execution instance.

This paper has some limitations. First, this paper attempts only to depict the executions caused by different schedules, not different inputs to the program. This limit is intrinsic to pure dynamic analysis, which look only at executions and not at the program itself. Second, since this paper considers only some explicit synchronization caused by thread fork/join and lock/unlock, alternate behaviors caused by implicit synchronization through accesses to shared memory and conditional synchronization like wait/notify in Java, are not represented.

We have some future works related to this work. The first is to extend the graph to represent various synchronization primitives such like semaphores. The second is to improve the graph for different concurrency bugs such as data races and atomicity violation.

References

1. Bensalem, S., Havelund, K.: Dynamic deadlock analysis of multi-threaded programs. In: Ur, S., Bin, E., Wolfsthal, Y. (eds.) HVC 2005. LNCS, vol. 3875, pp. 208–223. Springer, Heidelberg (2006)
2. Naik, M., Park, C.S., Sen, K., Gay, D.: Effective static deadlock detection. In: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering (ICSE), pp. 386–396. IEEE Computer Society, Los Alamitos (May 2009)
3. Joshi, P., Park, C.S., Sen, K., Naik, M.: A randomized dynamic program analysis technique for detecting real deadlocks. In: Proceedings of the 2009 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2009), pp. 110–120. ACM, New York (June 2009)

4. Kraemer, E.: Visualizing concurrent programs. In: Software Visualization: Programming as a Multimedia Experience, pp. 237–258 (January 1998)
5. Diehl, S.: Software Visualization: Visualizing the Structure, Behavior, and Evolve of Software. Springer, Heidelberg (May 2007)
6. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Communications of the ACM 21(7), 558–565 (1978)
7. Bensalem, S., Fernandez, J.C., Havelund, K., Mounier, L.: Confirmation of deadlock potentials detected by runtime analysis. In: Proceedings of the 2006 Workshop on Parallel and Distributed Systems: Testing and Debugging (PADTAD 2006), pp. 41–50. ACM, New York (2006)
8. Havelund, K.: Using runtime analysis to guide model checking of java programs. In: Havelund, K., Penix, J., Visser, W. (eds.) SPIN 2000. LNCS, vol. 1885, pp. 245–264. Springer, Heidelberg (2000)
9. Mehner, K.: Javis: A uml-based visualization and debugging environment for concurrent java programs. In: Diehl, S. (ed.) Dagstuhl Seminar 2001. LNCS, vol. 2269, pp. 163–175. Springer, Heidelberg (2002)
10. Leroux, H., Réquilé-Romanczuk, A., Mingins, C.: Jacot: a tool to dynamically visualise the execution of concurrent java programs. In: Proceedings of the 2nd International Conference on Principles and Practice of Programming in Java (PPPJ 2003), pp. 201–206. Computer Science Press, Inc., Rockville (2003)
11. Artho, C., Havelund, K., Honiden, S.: Visualization of concurrent program executions. In: Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), pp. 541–546. IEEE Computer Society, Los Alamitos (July 2007)
12. Zhao, Q.A., Stasko, J.T.: Visualizing the execution of threads-based parallel programs. Technical Report GIT-GVU-95-01, College of Computing, George Institute of Technology (January 1995)
13. Carr, S., Mayo, J., Shene, C.K.: Threadmentor: a pedagogical tool for multithreaded programming. J. Educ. Resour. Comput. 3(1), 1 (2003)
14. Kim, B.C., Jun, S.W., Hwang, D.J., Jun, Y.K.: Visualizing potential deadlocks in multithreaded programs. In: Malyshkin, V. (ed.) PaCT 2009. LNCS, vol. 5698, pp. 321–330. Springer, Heidelberg (2009)

A Fast PDE Algorithm Using Adaptive Scan and Search for Video Coding

Jong-Nam Kim

Dept. of IT Convergence and Application Engineering
Pukyong National University, Busan, 608-737 Korea
jongnam@pknu.ac.kr

Abstract. In this paper, we propose an algorithm that reduces unnecessary computations, while keeping the same prediction quality as that of the full search algorithm. In the proposed algorithm, we can reduce unnecessary computations efficiently by calculating initial matching error point from first 1/N partial errors. We can increase the probability that hits minimum error point as soon as possible. Our algorithm decreases the computational amount by about 20% of the conventional PDE algorithm without any degradation of prediction quality. Our algorithm would be useful in real-time video coding applications using MPEG-2/4 AVC standards.

1 Introduction

In video compression, full search (FS) algorithm based on block matching algorithm (BMA) finds them optimal motion vectors which minimize the matching difference between reference block and candidate block. It has been widely used in video coding applications because of its simple and easy hardware implementation. However, heavy computational load of the full search with very large search range can be a significant problem in real-time video coding application. Many fast motion estimation algorithms to reduce the computational load of the full search have been studied in the last decades.

We classify these fast motion estimation methods into two main groups. One is lossy motion estimation algorithm with degradation of predicted images and the other is lossless one without any degradation of predicted images compared with the conventional FS algorithm. The former includes following subgroups : unimodal error surface assumption (UESA) techniques, multi-resolution techniques, variable search range techniques with spatial/temporal correlation of the motion vectors, half-stop techniques using threshold of matching distortion, integral projection technique of matching block, low bit resolution techniques, sub-sampling techniques of matching block, and so on [1]. The latter as fast full search technique contains following several algorithms: successive elimination algorithm (SEA) using sum of reference block and candidate block and its modified algorithms, partial distortion elimination (PDE) method and its modified algorithms, and so on [2].

In lossless motion estimation algorithms, PDE is very efficient algorithm to reduce unnecessary computation for matching error calculation. To further reduce unnecessary computation in calculating matching error, J.N. Kim and et al [3]. proposed fast PDE algorithms based on adaptive matching scan, which requires additional computation to get matching scan order. But the additional computation for the matching scan order can be burden when cascading other fast motion estimation algorithm such as SEA.

In this paper, we propose a fast motion estimation algorithm to reduce computational amount of the PDE algorithm without any degradation of predicted images compared with the conventional PDE algorithm for motion estimation. We reduced only unnecessary computations which doesn't affect predicted images from the motion vector. To do that, we do block matching by 1/N partial SAD (sum of absolute difference). Unlike previous approaches, we try to find a good candidate with minimum error from initial partial errors. By doing that, we can find minimum error point faster than the conventional spiral search pattern. Our algorithm can easily control the trade-off between prediction quality and computational reduction for motion estimation with proper error criterion. Our algorithm reduces by about 20% of computations for block matching error compared with the conventional PDE algorithm without any degradation of prediction quality.

2 Proposed Algorithm

In fast lossless motion estimation, partial distortion elimination (PDE) algorithm reduces efficiently only unnecessary computation by stopping remaining calculation in a matching block. In PDE algorithm, if an intermediate sum of matching error is larger than the minimum value of matching error at that time, the remaining computation for matching error of the block is abandoned.

$$APSAD_k = \sum_{i=1}^{k} \sum_{j=1}^{N} |f_t(i,j) - f_{t-1}(i+x, j+y)| \quad \text{where } k = 1, 2, \cdots, N \tag{1}$$

$$APSAD_k \leq \mathrm{SAD}_{\min} \quad \text{where } k = 1, 2, \cdots, N \tag{2}$$

Eq. (1) and (2) show the partial sum of absolute difference (SAD) matching criterion of each row in the conventional PDE. Eq. (1), $f_t(i,j)$ denotes the pixel value for matching block at position (i,j) at time t, $f_{t-1}(i+x, j+y)$ denotes the pixel value of matching block at position $(i+x, j+y)$ at time $t-1$.

As described previously, conventional fast PDE algorithms focused on getting large matching error in a matching block via adaptive matching. In this paper, we try to get minimum candidate with some clues. Unlike conventional PDE methods, we get initial 1/N block matching error, and then find a minimum error and its position from initial matching errors. Here, N is the size of a matching block. When we get initial 1/N matching error, we use not a sub-block, but sub-sampling pattern to get more even error distribution. After that, we calculate a SAD for the minimum error point of the previous step. According to the SAD, we eliminate impossible candidate of remaining initial checking points. Then, we apply PDE algorithm to remaining candidates. We can summarize our algorithm as shown in Fig. 1.

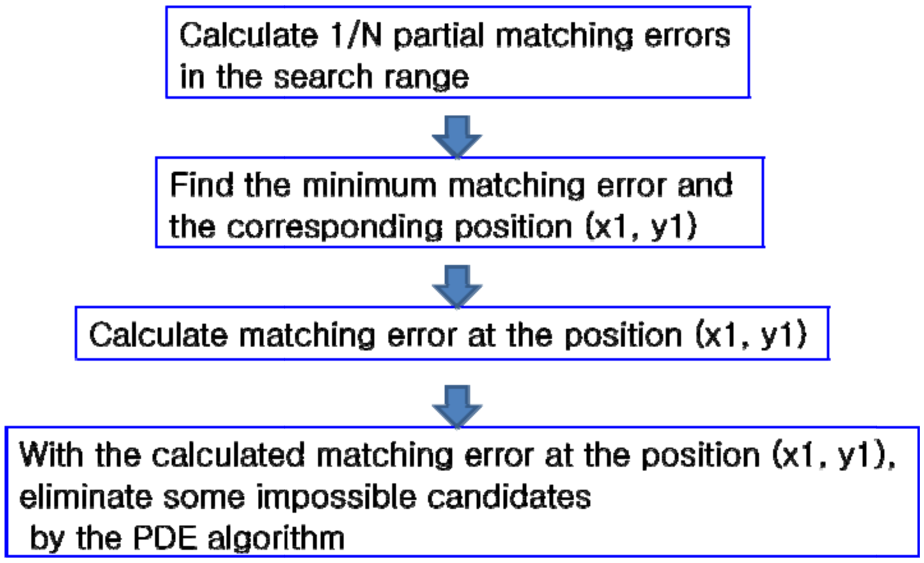

Calculate 1/N partial matching errors in the search range
Find the minimum matching error and the corresponding position (x1, y1)
Calculate matching error at the position (x1, y1)
With the calculated matching error at the position (x1, y1), eliminate some impossible candidates by the PDE algorithm

Fig. 1. Procedure of the proposed algorithm

3 Experimental Results

To compare the performance of the proposed algorithm with the conventional algorithms, we use 95 frames of “bus”, ”bally”, ”bicycle”, “flower garden” and “football” video sequences. Matching block size is 16x16 pixels and the search window is ±15 and ±7 pixels. Image format is SD(720*480) for each sequence and only forward prediction is used. The experimental results are shown in terms of computation reduction rate with average checking rows and prediction quality with peak-signal-to-noise ratio (PSNR). We compared the proposed algorithm with the conventional full search algorithm, conventional PDE [4], and Normalized PDE algorithm [5]. We didn’t compare the results of complexity based PDE algorithm [3] with ours because it can be cascaded to ours.

Fig. 2 shows the reduced average computation computed for various algorithms in “Bus” sequences. The horizontal axis represents the number of frames and the vertical axis represents reduced computation. As you can see, the proposed method has smaller computations than the conventional PDE. Fig. 3 shows prediction quality with PSNR measure in “Bus” sequences. The vertical axis represents PSNR with the unit of dB. As you can see, the proposed method has the same prediction quality as that of than the conventional PDE and full search algorithms. Even though conventional NPDE algorithm has smaller computational amount than that of the proposed algorithm, it has poorer prediction quality than ours for all frames.

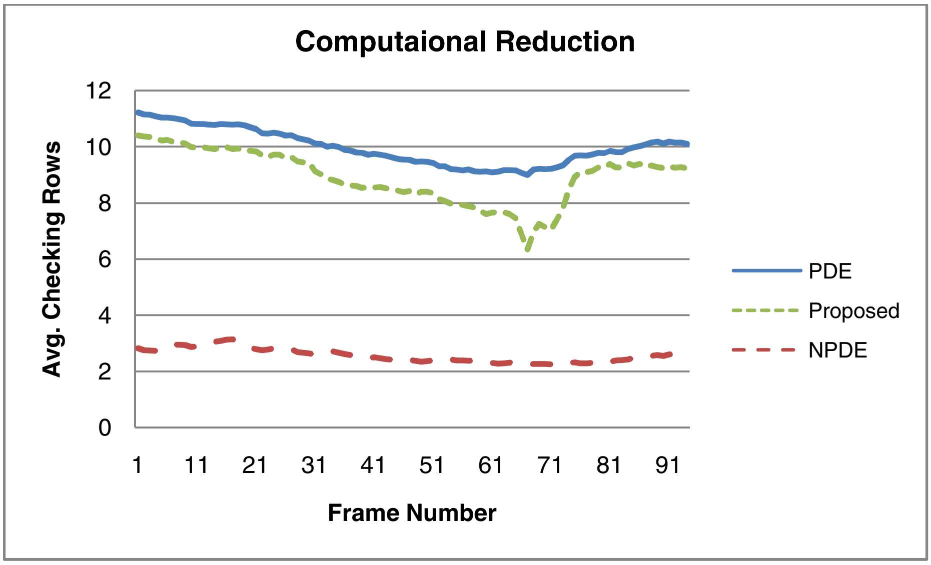


Fig. 2. Computation reduction for each frame of "Bus" sequences

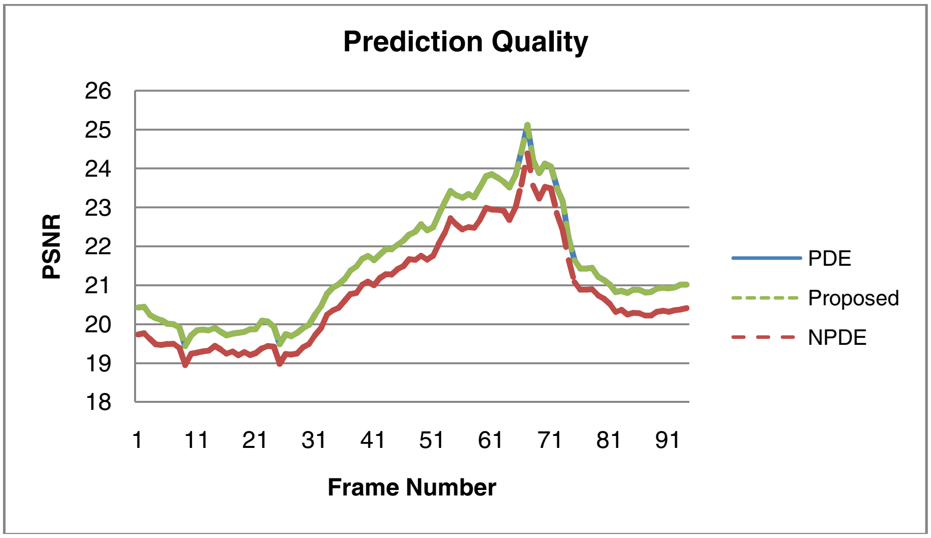


Fig. 3. Prediction quality for each frame of "Bus" sequences

Table 1 and Table 2 summarize the average computations and prediction qualities computed for various algorithms in all sequences. From Table 1, we can see that computational amount of the proposed algorithm is smaller than that of PDE and FS algorithm. Despite of reduced computation, Table 2 shows that the prediction quality of the proposed algorithm is almost same compared with full search algorithm.

Table 1. Average number of rows computed for all sequences of 30Hz

Algorithms	bally	bus	bicycle	flower garden	foot-ball
FS	16	16	16	16	16
PDE[4]	7.5	9.9	9.5	6.6	7.5
NPDE[5]	1.7	2.5	2.3	1.6	1.8
Propposed	7.4	8.9	9.4	5.5	7.3

Table 2. PSNR for all sequences of 30Hz

Algorithms	bally	bus	bicyclel	Flower garden	football
FS	30.57	21.45	22.59	26.86	23.29
PDE[4]	30.57	21.45	22.59	26.86	23.29
NPDE[5]	29.99	20.82	21.90	26.25	22.56
Propposed	30.57	21.44	22.59	26.86	23.29

4 Conclusions

In this paper, we propose an algorithm that reduces unnecessary computations and finds likely candidates as soon as possible, while keeping the same prediction quality as that of the full search. In the proposed algorithm, we set initial matching error with the result of first 1/16 partial error in the search range. By checking initial SAD from first 1/16 partial error, we can increase the probability of finding minimum error points. We can decrease computational amount for motion estimation without any degradation of prediction quality. Our algorithm will be useful in real-time video coding applications using MPEG-2/4 AVC standards.

Acknowledgement

This work was supported from LEADER by RIS Project by KOTEF.

References

[1] Dufaus, F., Moscheni, F.: Motion estimation techniques for digital TV: A review and a new contribution. Proceedings of IEEE 83, 858–876 (1995)

[2] Kim, J.N.: A study on fast block matching algorithm of motion estimation for video compression, Ph.D. Thesis of GIST (2001)

[3] Kim, J.N., Byun, S.C., Kim, Y.H., Ahn, B.H.: Fast full search motion estimation algorithm using early detection of impossible candidate vectors. IEEE Trans. Signal Processing 50, 2355–2365 (2002)

[4] `http://iphome.hhi.de/suehring/tml/download/old_jm/`

[5] Cheung, C.K., Po, L.M.: Normalized partial distortion search algorithm for block motion estimation. IEEE Trans. Circuits Syst. Video Technol. 10, 417–422 (2000)

Evolvability Characterization in the Context of SOA

Jose L. Arciniegas H. and Juan C. Dueñas L.

[1] Universidad del Cauca, Calle 5 # 4-70 Popayán, Colombia
jlarci@unicauca.edu.co
[2] Universidad Politécnica de Madrid, Ciudad Universitaria (s/n), CP: 28040, Madrid, Spain
jcduenas@dit.upm.es

Abstract. Service-Oriented Architecture (SOA) is an architectural style which promotes reuse of self-contained services. These self-contained services allow a better consideration of software quality characteristics as they can be independently analyzed. In our work, the evolvability quality characteristic has been considered, due to its impact in the stages of Maintenance and Evolution (M&E) for the software enterprises. Three goals are underlined in this paper: first, the relationship between SOA and quality characteristics focusing on a precise definition of evolvability of a software product from the SOA perspective, second a M&E model for SOA, and finally, some experiences are presented in order to assess evolvability in real software products. Two case studies have been executed: the first one analyzing the evolvability of the OSGi framework. And in the second case, the model is used in local Small and Medium Enterprises (SMEs), where an improvement process has been executed[1].

Keywords: Service-Oriented architecture, evolvability, software lifecycle, quality characteristics, maintenance and evolution.

1 Introduction

There is a clear tendency among software development organizations aiming at the evolution of formerly monolithic software architectures to convert them into services. A service is a self-contained well-defined function which does not depend on the context or state of other services. Then the usage of services to describe a software architecture (Services Oriented Architecture-SOA) compose a domain-specific architectural style, whose goal is to achieve loose coupling between providers and consumers of services, thus offering the possibility for adaptation and changes in a short time. This is especially attractive for quality attributes because quality characteristics can be improved in just one or some few services, almost without affecting other parts of the system. In this paper, the evolvability characteristic has been considered, for its large incidence during Maintenance and Evolution (M&E) phase of

[1] The authors wish to thank the Spanish company Telvent and Parquesoft-Popayán in Colombia for their support in this work. The work has been partially granted by the ITEA OSAMI-Commons project and the Spanish Ministerio de Industria, Turismo y Comercio and the REVVIS project of CYTED.

the system, as it means a continuous process improving the system, adapting it to new requirements and enhancing the quality of services.

2 Relationship between SOA and Quality Characteristics

Services have a close relationship with qualities; a service usually improves certain quality in a product. The ISO 25010 standard [1] defines the most representative quality requirements. However, certain quality characteristics are more important than others depending on the context; for SOA the most critical qualities were presented in [2].

Each system has special conditions of quality, for example a real-time system requires high conditions of performance and maybe low conditions of reusability or portability. Quality characteristics can have more or less incidence depending on whether they are considered as lifecycle characteristics, execution characteristics or quality in use. Figure 1 identifies three axes, taking into account the previous classification. The key problem appears when more than one quality characteristic in different axes are essential for the system. In the Information and Communication Technologies (ICT) domain it is a quite common situation: an improvement in one of them affects the others. For example when security is improved it affects usability, reusability and others. Finding the balance between them is of paramount importance for the success of the final solution.

Consequently, the qualities with incidence on the three dimensions are more critical for the systems (functional suitability, reliability, operability and evolvability). In this paper Evolvabilty is the point of attention, meaning that systems must be flexible to be adapted, changed or improved to new requirements or environments.

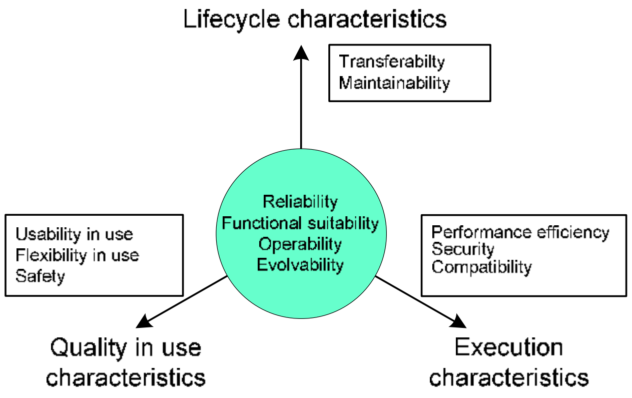


Fig. 1. Quality characteristic dimensions

Evolvability plays an important role during maintenance phases in two senses: reduction of the total cost during maintenance and extension of useful lifetime of software. However, evolvability has no clear definition because it affects several quality and functional characteristics, i.e. improvements in evolvability means improvement in other qualities. In this paper we try to clarify the evolvability attributes, specifically finding the relationship between them in the context of SOA.

Evolvability is a quality very close to maintainability but with some differences: some authors consider that maintainability is for fine-grained changes while evolvability is for coarse-grained (structural changes). In [3] considers also evolvability as a more general quality than maintainability. In addition, some metrics for measurement of evolvability are presented. In this sense evolvability is more important for the people who changes software while maintainability is more relevant for the ones who use it. On the other hand, evolvability is also related with other characteristics such as: adaptability, modifiability, replaceability, portability, flexibility, integrability, reusability, extensibility, traceability, variability and tailorability. Figure 2 represents the influence of evolvability based on the classification of Figure 1.

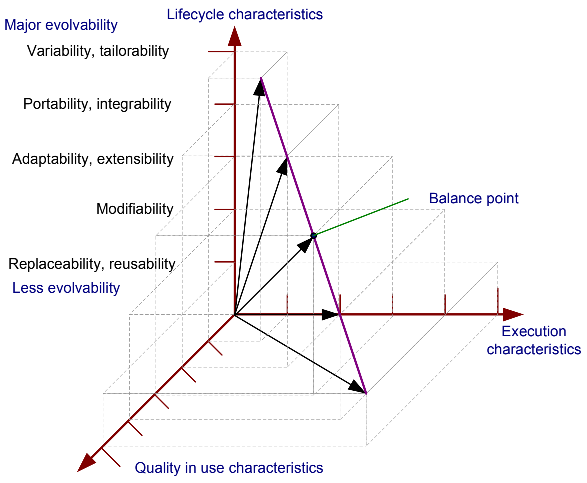

Major evolvability
Lifecycle characteristics
Variability, tailorability
Portability, integrability
Adaptability, extensibility
Modifiability
Replaceability, reusability
Less evolvability
Balance point
Execution characteristics
Quality in use characteristics

Fig. 2. Evolvability effects on other qualities

Figures 1 and 2 show that evolvability has a direct effect in overall axes, but their major effect is over the lifecycle characteristics. After an empirical analysis about the evolution of several systems, we conclude that, when evolvability is increased, the lifecycle characteristics are also increased but other qualities are decremented (quality in use and execution). In Figure 2, a hypothetical line and some points are represented; the line shows the inverse relation among quality attributes. In consequence, the lifecycle characteristics concerning with evolvability, where replaceability and reusability have less incidence, and variability and tailorability are considered as the major incidence over the system evolvability. Figure 2 shows that extreme conditions are not good to the overall system, so a balance between evolvability and the others is required. In an ideal situation the balance point could be located where the values of each axe are equal. Taking into account the balance point, we have defined evolvability in a most specific way:

Evolvability can be described as the ability to anticipate the locations of transformations in a system and its capacity to be adapted or modified to get up a balance among cost and resources, restrictions and local or global effects.

On the other hand, the evolvabiliy of the systems is directly associated to Software Architecture (SA). The SA discipline allows filling the gap during the evolution, because the architecture eases the transformation of the system avoiding tedious processes. Figure 3 shows the relationship between evolvability and architectural development paradigms, where service-oriented development presents the best attributes for the evolvability quality. However, in opposition to Figure 2 where evolvability raises while some other qualities are decreased, then service-oriented development loses quality in use and execution qualities [4].

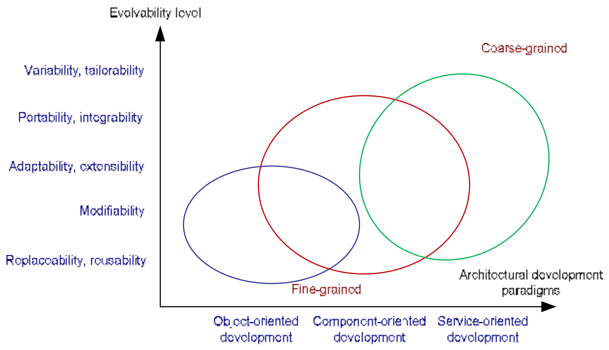

Evolvability level
Variability, tailorability
Portability, integrability
Adaptability, extensibility
Modifiability
Replaceability, reusability
Coarse-grained
Fine-grained
Architectural development paradigms
Object-oriented development
Component-oriented development
Service-oriented development

Fig. 3. Relationship between evolvability and software architecture

3 Maintenance and Evolution (M&E) Model for SOA

Several authors have studied this topic since more than 30 years ago. For example, Manny Lehman [5] states that the product line approach solves part of this problem using common concepts, common assets and locating variation points [6]; the variability can be handled by using a large number of strategies, ranging from controlling the system configuration, to the support of dynamic updating procedures. However, the solutions provided by the product line engineering are not enough, since they solve the "variability in space", but not the "variability in time" problem (evolution). So, the available methods, techniques and tools must be adapted in order to support the extended system lifecycle that includes software evolution.

M&E conceptual model

The proposed conceptual model bases on Dart's model [7]. The core of the M&E conceptual model relies on transformations (see Figure 4). In this model any transformation of the system is considered as part of the M&E process. The proposed conceptual model has the following considerations: the architecture as the structure of the system (static and dynamic views), architectural assets and services as the concept of components as well as the team concept is extended to stakeholders in order to include all staff involved during software development (not only developers). However, the major difference is configuration process; as Darts considers three basic configuration processes (Auditing, Accounting and Controlling) by mixing configuration and change management activities.

The stakeholders decide the changes driven by a particular focus, for example after an architectural assessment or an architectural recovery process. This decision is usually taken by some of them (project manager, engineers, testers, quality manager or customer), and these decisions are always influenced by the user needs and market strategies. The objective and focus are quality-driven, i.e. the stakeholders define one quality to improve, then define the objectives and focus accordingly. The objectives and focus have been usually identified as a result of the architectural assessment process, when limitations, gaps or errors have been found. In this case the focus is the evolvability in SOAs.

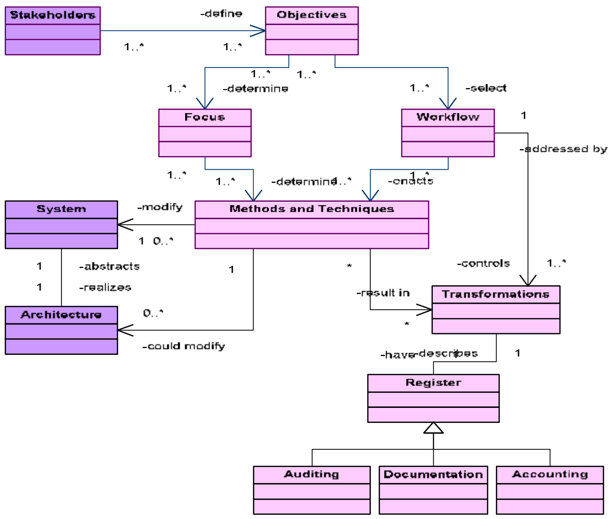


Fig. 4. M&E conceptual model

The M&E workflow manages system versions and changes throughout its lifecycle. In [8] the workflow controls how, when and where transformations (changes) are made. The methods and techniques comprise a set of strategies for transformation, such as: refactoring, composition, replacing, updating, etc. The methods and techniques are applied to assets, services or to the full system. In some cases these transformations affect the original system architecture, although it is not a desirable situation. When the system architecture has been changed, we consider that the system has evolved to another system.

M&E transformations

Transformations have different sources, such as for example changes in the hardware, business, organization or software. Transformations can be classified from the classical concepts of maintenance as: "*ForwardTransformation*" and "*BackwardTransformation*". Transformations are associated with a register where the information of configuration and changes is stored. Auditing, Accounting and Documentation are part of the register. Auditing keeps an audit trail of the system and its processes. Accounting gathers statistics about the system and its processes. Documentation synchronizes the information of the systems with respect to the changes.

The *ForwardTranformation* controls the transformation in order to improve some functional or non-functional characteristics; we consider:

- *FineEvolution*, set of changes that could be made to asset code: bugs, moves, additions, deletions, modifications, comments and cleaning.
- *MergerEvolution*, is a type of transformation when a system becomes part of another. In this type of transformation, the system can be considered as an asset/service (black box) where their interfaces and data interchanges should be well defined. Before the merge, the system interfaces or (input and output) data formats can be modified. In addition, other neighbor assets/services can be affected, so they could also be modified.
- *ConfigurationEvolution* is related to configuration changes, not software modifications, such as changes in parameters, values, defect conditions and connections are included.
- *ForkEvolution* occurs when the original system is divided into two or more derived systems. In some cases derived systems become in services.

Not all *ForwardTransformations* are successful; in fact, conflicts, limitations, dependencies and other problems are very common during the evolution. We consider the next types of *BackwardTransformations*:

- *SteppingBack*, in some cases, the changed introduces into new versions are catastrophic, in these cases the best option is to come back to a previous stable version.
- *AssetRecovery* extracts operative assets from implemented systems (the best case occurs when service are recovered). However, the extracted asset should be often adapted in order to be reused [8].
- *ConfigurationRecovery* is the reverse transformation of *ConfigurationEvolution*. In some cases, the setting information should be recovered from previous versions.

M&E workflow

The workflow manages the configuration and evolution of a system. It uses some methods and techniques in order to control the transformations (forward and backward). However, workflow does not have an explicit sequence to be performed; for example *FineEvolution* or *SteppingBack* can be executed after a detection of errors, and *ConfigurationEvolution* or *ConfigurationRecovery* can occur when something in the context has changed. These transformations are unpredictable and should be carried out as soon as possible. Some other transformation could be planned (looking ahead changes) for example *MergerEvolution*, *ForkEvolution* or *AssetRecovery*. The last three activities should be executed periodically.

Some special activities for configuration during M&E were identified: customization, configuration, versioning and composing frameworks (capacity of an asset to join with one or several assets and create a new product, application or service).

M&E methods and techniques

The following techniques are considered as the most relevant for *ForwardTransformations*:

- *Refactoring*: is a technique for restructuring an existing body of code. Refactoring is the process of rebuilding from previous versions by adding, correcting, deleting, cleaning or improving some parts of the systems [10].
- *Factoring*: is a technique that builds new assets or services by adaptation or improvement of the functional or non-functional characteristics from the original version. In any case, no code of the original asset/service is reused.
- *Rearchitecting*: is a technique of rebuild from previous versions by adding, correcting, deleting, cleaning, changing the architecture configuration, composing or improving some architecture assets or services (high level of abstraction).

On the other hand, *BackwardTranformations* techniques are very common in systems under test, because certain changes have been introduced but their effects are still unknown. Most of errors, conflicts or limitations are discovered in this phase.

- *SteppingBack*, is a "big undo" when an unsuccessful *ForwardTransformations* has occurred. Others complementary techniques for support of the SteppingBack are proposed in [10], such as: Changeset support, line-wise history, release tagging, collaboration style, branching and merging.
- For *ConfigurationRecovery*, the configuration information depends to large extent on the application (user profiles, some setting environment variables, context, etc), for this reason it is usually passed to the application or operating system responsibility.
- An *AssetRecovery*, it is a part of reverse engineering where only assets or services are recovered from implemented systems. A complete list of techniques is presented in [9].

4 Experiences

The evolution of the OSGi specification (Open Services Gateway initiative) [11] represents a good academic study to evaluate evolability of a SOA framework. OSGi is an excellent framework that supports a great variety of service-oriented applications (embedded systems, soft real-time, consumer electronics, mobile, etc). OSGi was also designed to be used in a large number of devices and requires little memory for operation.

In the case study, the evolution from OSGi R3 to OSGi R4 [11] was studied. This evolution was analyzed with respect to lifecycle characteristics. OSGi R4 performs a big effort in organization of the OSGi framework. Some new ideas are introduced and some concepts clarified about the work of the framework. The transformation from OSGi R3 to OSGi R4 is a good example of rearchitecting, which is evident in its logical structure. OSGi R3 is a simple model organized into layers where the framework is supported on a specific execution environment (Java virtual machine) and both are supported on an operating system and hardware. The applications or bundles can interact with the framework or directly with the other layers. OSGi R4 preserves the same structure in layers but it divides the framework in additional functional layers: services, service register life cycle and modules. In addition, a transversal layer is added for security. These elements are not new in OSGi R4, some of them were found in OSGi R3. For example, security appears spread along all the OSGi R3 specification and some recommendations were done. In OSGi R4 it is specifically defined the security as a transversal layer. The separation of functionalities affects OSGi M&E aspects as adaptability, maintainability, modifiability, replaceability or others. In the OSGi evolution, new assets (services or utilities) have been included, others have been retired and some changes of the original configuration have been carried out. The essential structure was conserved in order to guarantee compatibility between versions.

The main change is inside the core framework, because it is enriched with some services and utilities that in previous versions were an external part of the framework. In addition, the service and utilities roles are defined into several layers: security, module life cycle and services. Big and small transformations were made from OSGi R3 to R4 (FineEvolution and MergerEvolution). Perhaps, OSGi R4 will be a transition version because the old services and utilities conceptually moved to the framework core, such as: packageadmin, starlevel, url and others, should be physically (in the same folder) moved as part of the framework core. In addition, possibly more services will be integrated. Other important transformation from OSGi R3 to R4 is the conception of each asset (service, utility or layer) as an independent part; consequently, they could also evolve into independent directions.

In addition, evolvability has a direct relation with other intrinsic properties of OSGi; for example composability, encapsulability, managementability, security and deploymentability are properties of special interest in OSGi domain. However, other criteria from OSGi are in close relation with the evolution [12] (flexibility, testability, integrability, etc), the table 1 shows the relation among evolvability attributes and criteria, weakness or advantages of these criteria allow the prediction of system transformations.

Table 1. Relation of evolvability attributes with respect to some criteria of OSGi specification

Evolvability Attributes / Criteria	Replaceability	Modifiability	Adaptability	Maintainability
Composability		+	+	+
Encapsulability		+	+	+
Managementability			+	+
Security				+
Deploymentability			+	+
Flexibility	+	+	+	+
Testability				+
Integrability	+	+	+	+
Reusability	+		+	+
Extensibility		+	+	+
Portability	+		+	+
Variability			+	+
Tailorability			+	+
Monitorability				+
Traceability				+

An inquiry was used to obtain the perception and experience in the usage of the OSGi framework. The enquirers were a group of developers and architects from University (Universidad Politécnica de Madrid) and Industry (Telvent and Telefonica I+D companies from Spain). The inquiry contained questions about the usability of each asset (element, service or utility) defined in the specification. In addition, non-used elements were detected. These results can be used to make estimations in the architecture evolution. For example, it is expected the consolidation of some services (Log and Http). For the other assets it is expected a soon evolution in order to increase their utilization and better use.

Table 2. Evolvability of OSGi R3 assets

Evolvability levels / OSGi Assets	Reusability Replaceability	Modifiability	Adaptability Extensibility	Portability Integrability	Variability Tailorability
Framework API, Log Service and Http Service	M				
StartLevel service, Permission Admin service and Configuration Admin service	H	M			
URL	H	M	M		
Device Access, User Admin service, IO Connector, Preferences Service, Wire Admin service, Metatype, UPnP API, Provisioning Service, XML Parser service, Service Tracker, Package admin, Measurement, Position, and Jini.	H	H	M	M	

Table 3. Evolvability of OSGi R4 assets

Evolvability levels / OSGi Assets	Reusability Replaceability	Modifiability	Adaptability Extensibility	Portability Integrability	Variability Tailorability
Framework API, Log Service, Http Service, Life cycle layer, Service layer and Service registry layer	H				
StartLevel service, Permission Admin service, Configuration Admin service and Modules layer	H	H			
Admin service, URL Stream and Content Handlers API, and Security layer	H	H	M		
Conditional Permission Admin, Device Access, User Admin service, IO Connector, Preferences Service, Wire Admin service, Metatype, Service Component, UPnP API, Provisioning Service, Event Admin, XML Parser service, Package admin, Service Tracker, Measurement and Position	H	H	H	M	

Table 2 and 3 summarize the qualitative evaluation obtained from the inquiry in OSGi R3 and R4 respectively, The assets from R3 and R4 have been organized by its incidence in evolvability characteristics. A simple scale was used by each assets, so H means High, M means medium and L means low compliance with one characteristic. For example, Security layer in OSGi R4 is considered as an asset that its reusability, replaceability and modifiability is high, but its adaptability and extensibility could be improved. The results indicate that the evolvability of OSGi R4 is increased compared with OSGi R3, for the following reason: reusability and replaceability level is increased because new layers were introduced (Life cycle, service and service register). Modifiability level is also increased because the Modules layer is added. Adaptability and extensibility are increased because the admin service is added and security layer is restructured. Finally, portability and integrability are increased because several assets were introduced (conditional permission, service component, event admin), others have been improved (metatype, upnp, provisioning and service tracker).

In addition, the model has been used in commercial systems. Industrial studies have been carried out in Parquesoft-Popayán (Colombia), which is an software development industrial incubator for short and medium size organizations. Parquesoft is distributed in six Colombian cities being Popayán one of them. The model was applied in order to improve the quality of the products in more than 10 small organizations (less than 10 people involved). However, the first problem was the poor knowledge about the architecture and quality characteristics; obviously the same situation was identified about SOA knowledge. In consequence, the first evolution in these organizations was not the improvement of evolvability in their products, but the migration from previous paradigms (see Figure 3) to SOA. The Figure 6 illustrates the current situation of selected organizations, most of them use object-oriented development; however, they are in change process being SOA the target paradigm. The current assets developed by Parquesoft-Popayán have some evolvablity characteristics; however there are several considerations that should be improved.

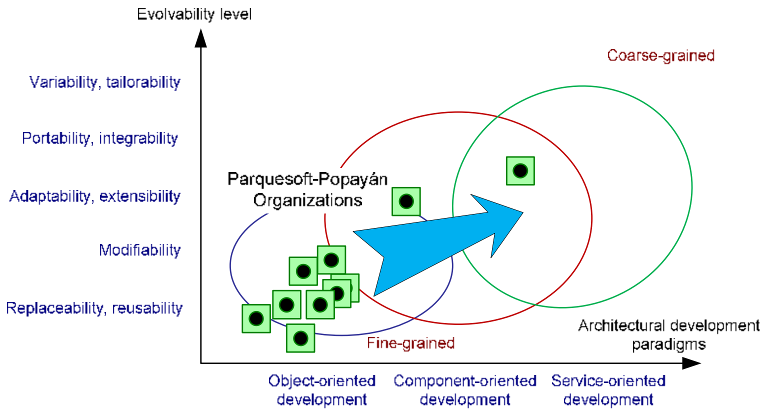

Evolvability level
Variability, tailorability
Portability, integrability
Adaptability, extensibility
Modifiability
Replaceability, reusability
Coarse-grained
Parquesoft-Popayán Organizations
Fine-grained
Architectural development paradigms
Object-oriented development
Component-oriented development
Service-oriented development

Fig. 6. Location of Parquesoft-Popayán organizations

5 Conclusions

M&E underlines the role of the architecture as the cornerstone for maintenance and evolution of the software. The transformations (changes and configuration variations) during maintenance and evolution can be controlled and applied to the architecture in order to extend the lifecycle of the software. The effort in transformation is reduced when SOA is used. In addition, quality improvements are the most frequent changes during maintenance and evolution time. The main contribution of this paper is the quality-driven methodological support for maintenance and evolution; specially, the conceptual model for M&E focused on the transformation (forward and backward). In addition, some extra activities carried out during this period are illustrated, such as: customization, configuration, versioning and composing frameworks as well as methods and techniques that support forward and backward transformations.

The academic case study presents criteria to measure and discover tendencies of usage of the OSGi framework to support services. An inquiry was performed in order to obtain experiences from other groups (continuous learning) that have used the OSGi specification as a framework for their applications. The results of the inquiry show that several aspects can be improved in future versions, such as: the documentations of some assets; support for the correction of gaps or errors, security, management, etc. In addition, some prediction and tendencies could be established, for example, the most used assets represent a measure of the maturity of the framework, while less used elements means potential limitations or possible problems that should be corrected.

The industrial case studies also contribute to the continuous process of refinement of the model. Some important results have been found, the main contribution for the current Parquesoft-Popayán organization is the start of the improvement process to increase the quality of their products.

References

[1] ISO/IEC 25010-CD – JTC1/SC7. Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE). Internal ISO/IEC JTC1/SC7 Document, Currently at Commission Draft Stage, International Standardization Organization (2007)

[2] O'Brien, L., Bass, L., Merson, P.: Quality Attributes and Service-Oriented Architectures. The Software Engineering Institute and Carnegie Mellon University (September 2005)

[3] Cook, S., He, J., Harrison, R.: Dynamic and static views of software evolution. In: Proceedings of IEEE International Conference on Software Maintenance, November 7-9, pp. 592–601 (2001)

[4] Brown, A., Johnston, S., Kelly, K.: Using service-oriented architecture and component-based development to build Web service applications. A Rational Software White Paper (2002)

[5] Lehman, M.: Programs, Life Cycles and Laws of Software Evolution. Proceeding of the IEEE, Special Issue on Software Engineering 68(9), 1060–1076 (1980)

[6] van der Linden, F., Bosch, J., Kamsties, E., Känsälä, K., Obbink, H.: Software Product Family Evaluation. In: Nord, R. (ed.) SPLC 2004. LNCS, vol. 3154, pp. 110–129. Springer, Heidelberg (2004)

[7] Dart, S.: Concepts in Configuration Management Systems. In: Proceeding of the Third Int'l Software Configuration Management Workshop, pp. 1–18 (June 1991)

[8] Opdyke, W.F.: Refactoring Object-Oriented Frameworks. PhD thesis. University of Illinois at Urbana Champaign (1992)

[9] Knodel, J., John, I., Ganesan, D., Pinzger, M., Usero, F., Arciniegas, J., Riva, C.: Asset Recovery and Incorporation into Product Lines. In: The 12th Working Conference on Reverse Engineering, WCRE 2005, Pittsburgh, Pennsylvania, USA (Carnegie Mellon), November 8-11 (2005)

[10] Robles, G., Amor, J., Gonzalez-Barahona, J., Herraiz, I.: Evolution and growth in large libre software projects. In: Eighth International Workshop on Principles of Software Evolution, September 5-6, pp. 165–174 (2005)

[11] OSGi. Open Services Gateway Initiative. OSGI Service Platform, Specifications, `http://www.osgi.org/Specifications/HomePage` (last visited date at, 14/09/10)

[12] Matinlassi, M., Niemelä, E.: The impact of maintainability on component-based software systems. In: Proceedings of 29th Euromicro Conference, September 1-6, pp. 25–32 (2003)

Design and Implementation of an Enterprise Internet of Things

Jing Sun[1], Huiqun Zhao[1], Ka Wang[1], Houyong Zhang[1], and Gongzhu Hu[2]

[1] Department of Computer Science
North China University of Technology
Beijing, 100144, China
zhaohq6625@sina.com

[2] Department of Computer Science
Central Michigan University
Mount Pleasant, Michigan, 48859, USA
hu1g@cmich.edu

Abstract. Since the notion of "Internet of Things" (IoT) introduced about 10 years ago, most IoT research has focused on higher level issues, such as strategies, architectures, standardization, and enabling technologies, but studies of real cases of IoT are still lacking. In this paper, a real case of Internet of Things called ZB IoT is introduced. It combines the Service Oriented Architecture (SOA) with EPC global standards in the system design, and focuses on the security and extensibility of IoT in its implementation.

1 Introduction

The term *Internet of Things* (IoT) was first introduced in the EPC Global standards in 1999 [1]. It is intended to extend the Internet from a network of computers to a network of things (or objects). The primary technologies to make Internet of Things possible include Electronic Product Code [2], Radio Frequency Identification [3], and EPC networks [4].

As a global coding standard, Electronic Product Code (EPC) is similar to barcode for identification of manufacturers and products, but it has an advantage over barcode by including coding digits segment for individual item of products (rather than just product categories) so the item can be identified and tracked during its life cycle. Successful use of EPC depends on the Radio Frequency Identification (RFID) and EPC Networks. RFID is an automatic identification technology that can store and process information, modulate and demodulate radio-frequency signal, and transmit and receive signals. A RFID tag, that is an EPC imprinted with a tiny RFID chip, can transmit the EPC information to be read by RFID readers. An EPC network consists of EPC Information Services (EPCIS) that are individual companies' services, Object Naming Service (ONS) [5] that is a directory of EPCIS, and EPC Discovery Services that are registry services interacting with the EPCIS. With these technologies, goods and products with RFID tags can be "linked" through EPC networks to enable businesses and their trading partners to automatically track individual product items throughout the supply chain. This scenario illustrates what would make the Internet of Things possible.

EPC Global has published a series of standards [1], [4], [5] to streamline and regulate the developments of IoT. However, there are only few real IoT cases available [6]. The academic community has made great efforts conducting research to overcome some of these obstacles towards making IoT but few examples applicable to real world enterprises have ever been presented. We believe that an example of a typical case would be very helpful and provide a scientific research platform for IoT experiments. In this paper we present a case study of the design and implementation of IoT for a real world enterprise called Zhong Bai Mall (ZB).

2 A Prototype System and Design of IoT

In this section, we first briefly describe the business model of Zhong Bai Mall, and then give a design of ZB IoT.

2.1 A Brief Introduction of ZB Mall

Zhong Bai Mall is large company in China doing business in retail and wholesale. It has many manufacturing and trading partners. Its business is mainly composed of two parts: purchasing and retail. The company has experienced a significant increase in trading quantity and scope in recent years. The management of the company realized that the original Management Information System (MIS) is no longer adequate to satisfy the increasing trading requirements. They are facing a new challenge to meet the demands of processing huge volumes of data on a timely manner. To overcome this problem, they have adjusted their business model to allow direct sales by some of the manufacturing partners, and keep the accounting operations within ZB. The new business model has not only benefited ZB but also simplified and accelerated the trading cycle. Under the new model, however, the manufacturers do not have direct access to the sales data and have difficulties keeping track of the sales quantity, simply because the MIS only delivers sales data to the partners on a monthly basis. To solve this problem, ZB and its partner decided to built an Internet of Things for their community and we are charged to collaborate with ZB to develop a prototype of its IoT.

2.2 Design of ZB IoT

The approach we took is to design a logical structure of the ZB IoT based on the standards of EPC global and the Service Oriented Architecture (SOA) with a new framework of Web Service. Fig. 1 shows a logical structure of ZB IoT that combine SOA into the original EPCIS architecture [4].

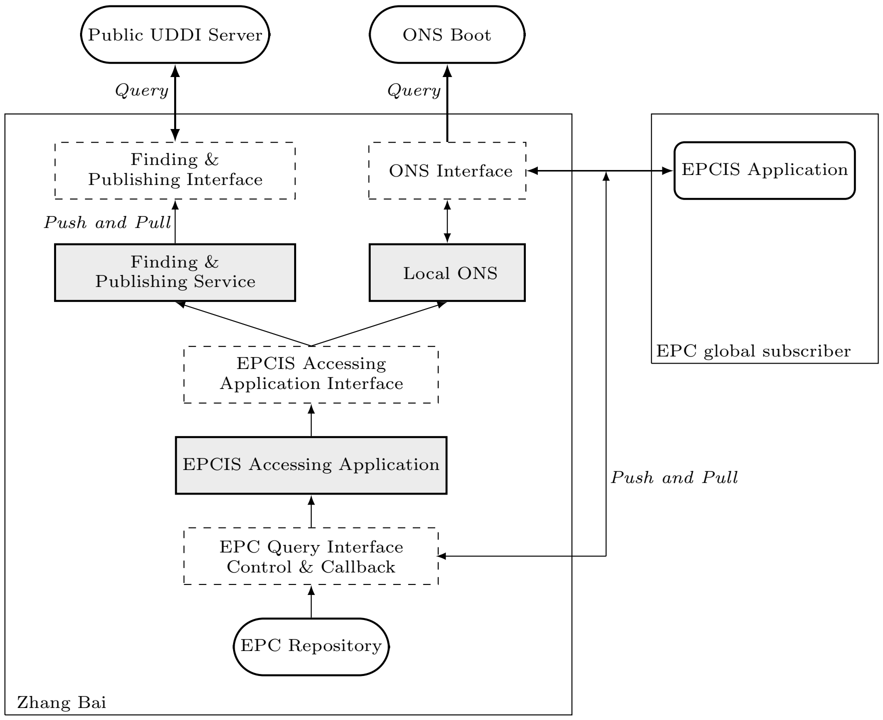

Public UDDI Server
ONS Boot
Query
Query
Finding & Publishing Interface
ONS Interface
EPCIS Application
Push and Pull
Finding & Publishing Service
Local ONS
EPC global subscriber
EPCIS Accessing Application Interface
EPCIS Accessing Application
Push and Pull
EPC Query Interface Control & Callback
EPC Repository
Zhang Bai

Fig. 1. A logical chart of IoT

We add a JUDDI, an open source Java implementation of the Universal Description, Discovery, and Integration (UDDI) specification for Web Services, as the registration space of the public service. The partner's EPCIS subscribes the service through the interface of local ONS. For this IoT case, we designed a new security strategy using a simplified Public Key Infrastructure (SPKI) [7]. The encoding and decoding methods for security used in our implementation are based on the original ONS IP encoding and decoding.

Fig. 2 is the detailed logical structure of ZB IoT. The upper portion of the chart directly reflects the business process. The middle portion of the chart is composed of three components: a subsystem of the IoT called Intranet and two databases. One of the databases stores the product data and the other stores sales data obtained from the EPC collection process. The lower portion of the chart illustrates the EPC data collection process where EPC readers receive information directly from EPC/barcodes or from EPC/barcode applications through filtering and store the data in the EPC repositories. The last component is called *EPCIS Capturing Application* that is responsible for the operations of the EPC elements.

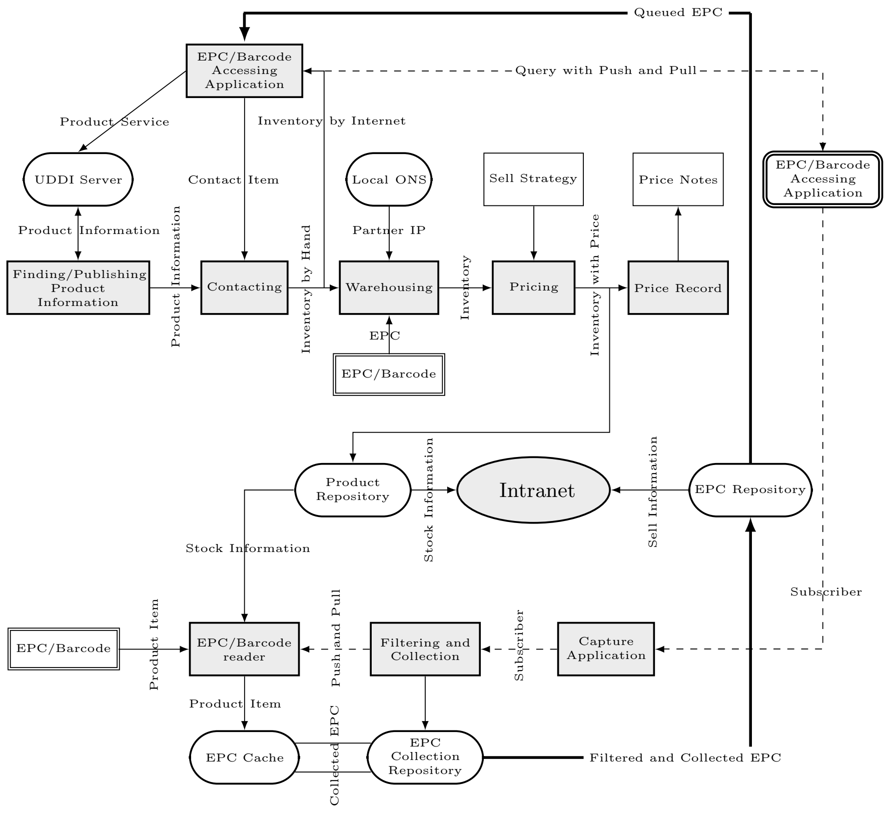


Fig. 2. Logical structure of ZB's IoT

3 Implementation of ZB's IoT

We shall describe a few of the key modules in the implementation of the ZB IoT in this section. We focus on the implementation of the security property of IoT and the subscription to the EPC service. We also explore the implementation strategy for the IoT's extensibility.

3.1 Security

We adopt Public Key Infrastructure (PKI) [8] in our implementation of the ZB IoT with some simplification. Our new PKI is called Simplified PKI (SPKI). In SPKI, the EPC tag and the EPCIS are the same as in the original EPC network, but the ONS encoding and decoding processes are replaced by our new algorithms.

We follow the general IoT architecture for the ZB IoT using Object Naming Services. The EPC tag, EPCglobal, and EPCIS are the same as specified in the standards, but the ONS encoding and decoding processes are quite different with original ONS strategies.

In the new ONS architecture the ONS encoding and the decoding components can use any strategies. The particular strategy we use in our Simplified PKI aims at reducing the system cost and improve the system performance. The algorithms used in the strategy are given below. Algorithm 1 is the new encoding algorithm with SPKI encryption.

Algorithm 1. Simplified Encoding Algorithm

```
Input: An EAN.UCC GTIN-14 consisting of digits $d_1 d_2 \dots d_{14}$ and the
       length $L$ of the company prefix portion of the GTIN.
Input: A Serial Number $S$ where $S < 2^{25}$.
Input: A Filter Value $F$ where $F < 8$.
Output: The encoding of the GTIN code.
1 Extract the EAN.UCC Company Prefix $d_2 d_3 \cdots d_{L+1}$ by doing a reverse
  lookup of Company Number in the Company Prefix Translation Table to
  obtain the corresponding Company Number Index, $C$. If the Company
  Prefix was not found in the Company Prefix Translation Table, stop: this
  GTIN cannot be encoded in the SGTIN-64 encoding schema.
2 Constructs and encrypts a digital $d_1 d_{L+2} d_{L+3} \dots d_{13}$ as the Item
  Reference and considering the result to be a decimal integer, $I$. If $I > 2^{20}$,
  stop: this GTIN cannot be encoded.
3 Construct the final encoding by concatenating the following bit fields,
  from most significant to least significant: Header 10 (2 bits), Filter Value
  $F$ (3 bits), Company Prefix Index $C$ from Step 1 (14 bits), Item Reference
  from Step 2 (20 bits), Serial Number $S$ (25 bits).
```

Algorithm 2 outlines the new decoding algorithm with SPKI decryption.

3.2 Extensibility of ZB's IoT

As mentioned before that EPC has not been widely used due to costs and other reasons, barcode is still the primary product coding approach. Due to the fact that the EPC is an extension of barcode, we decided to use barcode in the ZB IoT implementation for now that can be extended to EPC at a later time without much difficulty. To use barcode with EPC networks according to the EPCglobal standards, we proposed a new General Identification Number (GIN) [9] where the information in the barcode can be represented, and use the mapping strategy between GIN and the Universal Resource Format (URF) that is required by the EPC network. The mapping strategy is represented as the cryptographic encoding and decoding processes given in Algorithms 1 and 2.

GIN is similar to the General Identification (GID), an EPC encoding standard. The purpose of defining the new coding scheme GIN is to design a general structure for presenting independent identification of any concrete specification of EPC or barcode.

Algorithm 2. Simplified Decoding Algorithm

```
Input: An SGTIN-64 as a 64-bit bit string $10b_{61}b_{60}\cdots b_0$ (where the first
       two bits 10 are the header).
Input: A Serial Number.
Output: EAN.UCC GTIN-14.
1 Bits $10b_{61}b_{60}b_{59}$, considered as an unsigned integer, are the Filter Value.
2 Extracts the Company Prefix Index $C$ by considering $b_{58}b_{57}\ldots b_{45}$ as an
  unsigned integer. Look up the Company Prefix Index $C$ in the Company
  Prefix Translation Table to obtain the Company Number $p_1p_2\cdots p_L$
  consisting of $L$ decimal digits (the value of $L$ is also obtained from the
  table).
3 Considers $b_{44}b_{43}\ldots b_{25}$ as an unsigned integer. If this integer is greater
  than or equal to $10^{13-L}$, stop: the input bit string is not a legal
  SGTIN-64 encoding. Otherwise, convert this integer to a $(13-L)$-digit
  decimal number $j_1j_2\cdots j_{13-L}$, adding leading zeros as necessary to make
  $13-L$ digits.
4 Decrypt $j_1j_2\cdots j_{13-L}$ as $i_1i_2\cdots i_{13-L}$.
5 Constructs a 13 digit number $d_1d_2\cdots d_{13}$ where $d_1 = i_1$ from Step 4,
  $d_2d_3\cdots d_{L+1} = p_1p_2\cdots p_L = C$ from Step 2, and
  $d_{L+2}d_{L+3}\cdots d_{13} = i_2i_3\cdots i_{13-L}$ from Step 4.
6 Calculate the check sum.
7 The EAN.UCC GTIN-14 is the concatenation of digits from Steps 5 and
  6: $d_1d_2\cdots d_{14}$. Bits $b_{24}b_{23}\cdots b_0$, considered as an unsigned integer, are the
  Serial Number.
```

GIN is composed of four fields: Header, General Company Number, Object Class and Serial Number. The Header field is a classified type that is a 8-bit binary value.

The General Company Number identifies an organizational entity that is responsible for maintaining the numbers in subsequent fields – Object Class and Serial Number. The designer must ensure that each General Company Number is a unique decimal number. The Object Class is used by its entity to identify a category or “type” of things. These object class numbers, of course, must be unique within each General Company Number domain. It is also a decimal number. Finally, the Serial Number Code, or serial number, is a unique decimal number within the object class. In other words, the management of the company is responsible for assigning a unique serial number to every instance within the object class.

In order to connect Internet, we propose a parsing algorithm, given in Algorithm 3, that takes advantage of the ONS strategy of EPCglobal [5].

4 System Test

We conducted two different kinds of experiments. One is for demonstrating the efficiency of the encoding and decoding algorithms that use the Simplified PKI, and another is for the extensibility of barcode.

Algorithm 3. Parsing IP address

```
Input: A GIN code.
Output: An IP address involved in the GIN.
1 Map GIN code to URF using Algorithm 1.
2 Remove the serial number field.
3 Invert the order of the remaining fields.
4 Append '.onsepc.com'.
5 Do the client application's DNS resolver query for DNS Type Code 35 295
  (NAPTR) records [10].
```

4.1 Experiment of the Simplified Encrypt and Decrypt Algorithms

Several experiments were conducted to test the performance and security of Algorithms 1 and 2. Only the field Item Reference is encrypted in the algorithms. We sampled the SGTIN-64 Item Reference as a 13-digit plain code (`1467359243048`) and use the commonly used RSA cryptography algorithm [11] as the encryption strategy. We ran the algorithms on 1000 EPCs with three different lengths for private keys (512 bits, 1024 bits, and 2048 bits) to compare the run-time performance of the algorithms. The results of the experiment is shown in Table 1(a). Table 1(b) shows the results of applying the RSA algorithm for 1000 EPCs with a 16-digit plain text code `14CB2FECA24A3B8A`.

Table 1. Results of RSA

(a) 13-digit Item Reference for plain text code `1467359243048`

Public key	128	128	128
Private key	512	1024	2048
EPCs	1000	1000	1000
Encrypt time (ms)	235	245	253
Decrypt time (ms)	259	469	953
EPC encode/decode time (ms)	29/62	29/62	29/62

(b) 16-digit Item Reference for plain text code `14CB2FECA24A3B8A`

Public key	128	128	128
Private key	512	1024	2048
EPCs	1000	1000	1000
Encrypt time (ms)	258	284	297
Decrypt time (ms)	984	1796	3313
EPC encode/decode time (ms)	29/62	29/62	29/62

The results show that the encryption time increases only slightly as the private key length increases from the 1024 to 2048. It indicates that the 2048-bit private key is appropriate for our experimental environment. Comparing to the time cost between the different lengths of the plain text, it is not only that 13-digit plain text needs less time than 16-digit text but also its increase rate become higher. Therefore, the SPKI's performance and encrypting cost is better than PKI.

4.2 Test Case of Parsing IP from Barcode

The EAN.UCC Global Trade Identifier (GTIN) has a small family, including GTIN-14, GTIN-13, GTIN-12 and GTIN-8. We use GTIN-13 in our study.

The GTIN-13 is a 13-digit number composed of an Company Prefix, an Item reference, and a Check Digit. The Company Prefix is a unique variable-length number assigned to a company by EAN.UCC. Item Reference is a variable-length number assigned by the holder of the Company Prefix to uniquely identify the class or type of item. Check Digit a calculated one-digit number used to ensure data integrity.

An example of this input is `1234567890123` where if the company prefix length were specified as 6, then the company prefix would be the first 6 digits `123456`, the item reference would be the next 6 digits `789012` and the check digit would be the final digit `3`. According to Algorithm 2, the input code `1234567890123` has its URF format as: `id:GTIN-13.123456.789012.3`. Carry out Algorithm 2 to parse the IP address as following steps:

1. `id:gtin:6901.789012.3`
2. `789012.6901.id:gtin`
3. `789012.6901.id:gtin.onsbarcode.com`
4. `http://epc-is.example.com/`
   `epc-wsdl.xml`

Meanwhile, the number `6901` is the index of company in the translation table. The barcode.com is a server of inquiring the IP address for each company.

5 Related Work

EPC and RFID technologies have been around for some time, and are getting more and more attention in the last decade as the needs for using EPC networks increases in the business world towards the developments of Internet of Things. A comprehensive report about RFID-based IoT was presented in [12].

Adelmann et al. developed a toolkit [13] for bar code recognition. Leong et al. [14] provided an overall concept of EPC networks and explored the data flow and item tracking applications within the EPC network. Harrison [10] gave an overview of the Java web service prototypes of EPC Information System (EPCIS) and the EPCIS Discovery Service at the MIT Auto-ID Lab. Rieback et al. [15] explored the concept of malware for RFID systems, including RFID exploits, worms, and viruses; and presented RFID malware design principles. The paper [16] proposed a secure web services framework in an environment designed to integrate RFID system into the EPC global Network. Fabian et al [17] analyzed the privacy and security implications of ONS deployment, and also proposed a distributed hash tables (DHT) to enhance the performance and robustness of the domain name systems (DNS). They investigated the effectiveness of a decentralized alternative to ONS based on DHT [18].

A research team at the University of Washington developed a RFID ecosystem as an IoT example. The ecosystem is an infrastructure that creates a microcosm for the IoT, along with a suite of Web-based, user-level tools and applications for the IoT [19].

6 Conclusion

In this paper, we presented a real case of Internet of Things. In this case we focus on the design of an IoT using SOA and EPC Global standards, and enhance the security of the IoT by SPKI. We provide a method and technology for using the barcode as the identification coding in the IoT, that is very helpful for transferring barcode-based information system into IoT for a business.

In order to demonstrate the feasibility of our design and implementation, we used a series of test cases for the ZB IoT according to the ZB trading data. All the outputs of ZB IoT were consistence with the real trading data. We hope that can help readers understand how a business can develop an IoT and how an IoT can play a new role in its information system. Although the preliminary results are promising, we are currently conducting more experiments and performing comparisons of ZB IoT and other existing IoTs.

Acknowlegement

The work presented in this paper was partly supported by the China National Science Foundation grant #61070030.

References

1. EPCglobal: Architecture framework final version (2005), `http://www.epcglobalinc.org/standards/architecture/architecture_1_0-framework-20050701.pdf`
2. Brock, D.L.: The Electronic Product Code (EPC) - a naming scheme for physical object. White paper, Auto-ID Labs, Massachusetts Institute of Technology (2001)
3. Landt, J.: Shrouds of time: The history of RFID. An AIM publication, AIM Inc. (2001)
4. EPCglobal: EPC information services (EPCIS) version 1.0.1 specification (2007), `http://www.epcglobalinc.org/standards/epcis/epcis_1_0_1-standard-20070921.pdf`
5. EPCglobal: EPCglobal object name service (ONS) 1.0.1. Ratified standard specification, EPCglobal Inc. (2008)
6. Sun Microsystems: The Sun EPC network architecture, a technical white paper (2004)

7. Sun, J., Zhao, H., Xiao, H., Hu, G.: Lightweight Public Key Infrastructure and service relation model for designing a trustworthy ONS. In: Proceedings of 8th IEEE/ACIS International Conference on Computer and Information Science, pp. 295–300. IEEE Computer Society, Los Alamitos (2009)
8. Weise, J.: Public Key Infrastructure overview. Blueprints, Sun Microsystems, Inc. (2001)
9. Sun, J., Zhao, H., Hu, G.: Study of the key techniques for implementing barcode network. In: Computer and Information Science. Studies in Computational Intelligence, vol. 317, pp. 47–57. Springer, Heidelberg (2010)
10. Harrison, M.: EPC information service (EPCIS). In: Auto-ID Labs Research Workshop (2004)
11. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21, 120–126 (1978)
12. Buckley, J.: From RFID to the Internet of Things: Pervasive networked systems. In: Report on the Conference Organized by DG Information Society and Media, Networks and Communication Technologies Directorate, Brussels (2006)
13. Adelmann, R., Langheinrich, M., Flörkemeier, C.: Toolkit for bar code recognition and resolving on camera phones c jump starting the Internet of Things. In: Informatik 2006 Workshop on Mobile and Embedded Interactive Systems (2006)
14. Leong, K.S., Ng, M.L., Engels, D.W.: EPC network architecture. In: Auto-ID Labs Research Workshop (2004)
15. Riebacka, M.R., Simpsona, P.N., Crispoa, B., Tanenbauma, A.S.: RFID malware: Design principles and examples. Pervasive and Mobile Computing 2, 405–426 (2006)
16. Shih, D.H., Sun, P.L., Lin, B.: Securing industry-wide EPCglobal network with WS-security. Industrial Management & Data Systems 105, 972–996 (2005)
17. Fabian, B., Günther, O., Spiekermann, S.: Security analysis of the object name service. In: 1st International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, pp. 525–529. IEEE Computer Society, Los Alamitos (2005)
18. Fabian, B., Günther, O.: Distributed ONS and its impact on privacy. In: IEEE International Conference on Communications, pp. 1223–1228. IEEE Computer Society, Los Alamitos (2007)
19. Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska, M., Borriello, G.: Building the Internet of Things using RFID: The RFID ecosystem experience. IEEE Internet Computing 13, 48–56 (2009)

Aggregating Expert-Driven Causal Maps for Web Effort Estimation

Simon Baker and Emilia Mendes

Computer Science Department, The University of Auckland,
Private Bag 92019, Auckland, New Zealand
sbak030@aucklanduni.ac.nz, emilia@cs.auckland.ac.nz

Abstract. Reliable Web effort estimation is one of the cornerstones of good Web project management. Hence the need to fully understand which factors affect a project's outcome and their causal relationships. The aim of this paper is to provide a wider understanding towards the fundamental factors affecting Web effort estimation and their causal relationships via combining six different Web effort estimation causal maps from six independent local Web companies, representing the knowledge elicited from several domain experts. The methodology used to combine these maps extended previous work by adding a mapping scheme to handle complex domains (e.g. effort estimation), and the use of an aggregation process that preserves all the causal relations in the original maps. The resultant map contains 67 factors, and also commonalities amongst Web companies relating to factors and causal relations, thus providing the means to better understand which factors have a causal effect upon Web effort estimation.

Keywords: Web effort estimation, causal maps, Web effort prediction, map aggregation.

1 Introduction

A cornerstone of Web project management is effort estimation, the process by which effort is forecasted and used as basis to predict project costs and allocate resources effectively, so enabling projects to be delivered on time and within budget [1]. Effort estimation is a very complex domain where the relationship between factors is non-deterministic and has an inherently uncertain nature.

There have been numerous previous attempts to model effort estimation of Web projects, but none yielded a complete causal model incorporating all the necessary component parts. Mendes and Counsell [3] were the first to investigate this field by building a model using machine-learning techniques with data from student-based Web projects, and size measures harvested late in the project's life cycle. Mendes and collaborators also carried out a series of consecutive studies (e.g. [4]-[20]) where models were built using multivariate regression and machine-learning techniques and used data on industrial Web projects. Later they proposed and validated size measures harvested early in a project's life cycle, therefore better suited to effort estimation [1]' when compared to other Web effort predictors previously proposed [21]. Reifer [22] proposed an extension of an existing software engineering resource model, and a single size measure harvested late in the project's life cycle. None were validated empirically. This size measure was later used by Ruhe et al. [23], who further extended a software engineering hybrid estimation technique to Web projects, using a small data set of industrial projects, with this technique mixing expert judgement and multivariate regression. Baresi et al. [24][25] and Mangia et al. [26] investigated effort estimation models and size measures for Web projects based on a specific Web development method. More recently there have been a number of studies describing causal maps for Web effort estimation [27]-[30], where their causal relationships were identified by a domain expert, using only the set of factors that are part of the Tukutuku database [32]. Other more recent studies compared Web effort prediction techniques, based on existing datasets [33]-[36].

There are issues with all previous studies in that none, when identifying important factors for Web effort estimation, focused solely on factors that presented a cause & effect relationship with effort, i.e., they included any factors correlated with effort. In addition, when surveying companies to identify suitable effort predictors, those studies did not assess how good these companies were at estimating effort for their Web projects.

As part of a NZ government-funded research, Mendes elicited several company-specific expert-driven Web effort estimation causal maps from NZ Web companies [26]. The elicitation process employed is detailed in [26]. Each map was part of a larger model, named a Bayesian Network model (detailed in Section 2), and provided a representation of the Web effort estimation domain from the perspective of the single Web company from which that model had been elicited. All participating companies were consulting companies that developed different types of Web applications (e.g. static application, applications that used a content management system, database-driven Web applications).

Experience from eliciting such maps showed that companies found it much easier and more effective to use an initial set of factors as basis to elicit their causal maps, rather than to build one from scratch [29][30]. In addition, anecdotal evidence obtained throughout the elicitation process with these companies revealed that the use of an aggregated causal map representing the expert knowledge of different Web companies with regard to factors and relationships relevant for Web effort estimation would be extremely useful to help with the elicitation of company-specific causal maps. Some Web companies would like to use such large causal map at the start of the elicitation process; others believed that such maps would be extremely useful to provide them with a sort of "checklist" against which to compare their own, once it had been elicited.

In addition, we believe that such an aggregated map is important for practitioners for the following reasons:

- It depicts a blueprint of the most common factors and causal relationships important for companies that develop and manage Web projects as part of their core businesses. Such knowledge can be very useful to Web projects managers to help them revisit the factors they currently take into account during an effort estimation process.
- Provides companies with a starting point to building a single-company causal map, which can later be used to create a single-company Bayesian Network model. Anecdotal evidence shows that when eliciting factors, causal relationships

and even probabilities using as its basis tacit knowledge, companies find it much easier and more effective to customise an existing causal map to their needs, rather than to build one from scratch. This is also an approach suggested in [29][30].

- It can be used as a benchmark model (i.e. 'measuring stick') to compare different single-company Web effort estimation causal maps.

The contribution of this paper is therefore to provide a wider understanding on the fundamental factors affecting Web effort estimation and their causal relationships via combining six different Web effort estimation causal maps, elicited from six independent local Web companies, representing the knowledge elicited from several domain experts.

The remainder of this paper is structured as follows: an introduction to Bayesian Networks is given in Section 2, for those unfamiliar with this model, followed by a discussion relating to the aggregation of different causal maps (Section 3), and a summary of prior work (Section 4). Then we detail our proposed solution (Section 5) and methodology (Section 6) for aggregating causal maps, followed by a discussion of our results in Section 7. Finally the threats to the validity of our approach and our conclusions are given in Sections 8 and 9, respectively.

2 Bayesian Networks

A Bayesian Network (BN) is a probabilistic model that allows for reasoning under uncertainty. A BN is composed of two components [1]. The first is a graphical causal map, depicted by a Directed Acyclic Graph (DAG) (see Figure 1). The DAG's nodes represent the relevant variables (factors) in the domain being modelled, which can be of different types (e.g. observable or latent, categorical). The DAG's arcs represent the causal relationships between variables, where relationships are quantified probabilistically. These graphs may be simple (as in the example in Figure 1), or very complex in terms of nodes and relations.

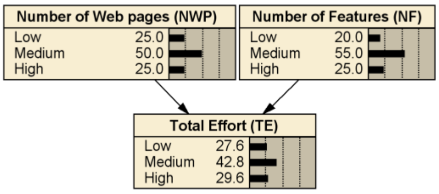


Fig. 1. A small Bayesian Network model and its CPTs

The second component of a BN is the quantitative part: a Conditional Probability Table (CPT) associated to each node in the network. A parent node's CPT describes the relative probability of each state (value); a child node's CPT describes the relative probability of each state conditional on every combination of states of its parents. Each row in a CPT represents a conditional probability distribution and therefore its values sum up to one [1]. A detailed description of the process employed to build BNs is detailed in [38]. The work presented herein is only concerned with the first component of a Bayesian Network, i.e. the causal map graph.

3 Problems Relating to the Aggregation of Causal Maps

It is often recommended that such causal maps be constructed through elicitation from different domain experts in order to derive a comprehensive and accurate causal map [39][40][41][45]. However, it is difficult to combine the beliefs of different experts in a coherent and impartial manner.

In order to arrive at a comprehensive causal map we would need to consult domain experts, many of whom working for different and perhaps competing companies, and thus likely to have a different prospective about the Web development domain. Therefore, the difficulty in combining expert-based causal maps increases for the following reasons:

- Identifying Common Variables: Different experts might represent semantically equivalent concepts in their causal maps using different variable names (e.g. '*Number of Developers*' vs. '*Project Human Resources*'). Furthermore, experts might use a different number of variables to represent the same concept.
- Conflicting Causal Relations: Variables might have contradictory causal relations according to different experts. Two kinds of causal relation conflicts can occur: the first when there is a causal influence between two variables according to an expert's belief, which is strictly prohibited by another expert's belief. The other type of conflict is the occurrence of cycles (which is ruled out within the context of this work in order to keep the resulting aggregated causal map consistent with all the six individual maps being used as input, which were all Directed Acyclic Graphs (DAGs).
- Collaboration Constraints: One feasible way to construct a generic causal map for Web effort estimation is to elicit a single map from a group of domain experts from a representative sample of Web development companies. This would need to be done in stages, and such approach might work well with small groups of domain experts but will likely to be impractical when additional models are included in the unified model. However, within the context of this research, any form of collaboration between domain experts is not feasible because all of the participating companies compete in the same market. This means that, by collaborating with other experts, they would be forced to share sensitive business information that they are not willing to disclose.

Therefore, it is vital to apply a methodology for combining different expert-elicited causal maps that solves the difficulties abovementioned. In this paper we propose a method, detailed in Sections 5 and 6, which solves many of the affiliated challenges in combining expert-elicited BNs that have not been sufficiently addressed in prior work.

Note that although all the causal maps that are used as input to our aggregated causal map are part of larger models - BNs, i.e., all represent the qualitative parts of single-company BNs that have been previously built for Web effort estimation, it is not our aim herein to build a cross-company Bayesian Network model for Web effort estimation, but rather to focus solely on the merging of causal maps.

4 Related Work

There are well established techniques that attempt to find consensus between expert opinions such as the Delphi method [37] and many others [41]. However, these methods generally require collaboration and information sharing, and are time consuming. As detailed in Section 3, such requirements are not suitable within the context of this work.

Sagrado and Moral [42] have proposed a qualitative method that combines separate BN Structures as either a union or intersection of DAGs. They consider the initial BN Structures to be I-Maps; I-Maps can be defined as BNs that contain probability parameters consistent with all Markov assumptions implied by its causal structure [43] . The Markov assumption states that a given node is independent of any non-descendent nodes in the network given its parent nodes. Figure 2 shows an example of the Markov assumption at node *X*. The highlighted nodes, (*D, C*, and *G*) are not influenced by *X* (and likewise, cannot influence *X*) given the probabilities of its parent nodes *A* and *B*. An I-Map is a BN possessing probability parameters that are consistent with the Markov assumption at every node.

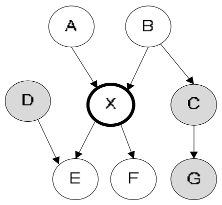


Fig. 2. Markov Assumption at node X

Sagrado and Moral based their approach on a proposition established by Castillo, Gutiérrez, and Hadi [44] , which states: Given two I-Maps defined over the same set of variables, if there is a common ancestral ordering in both I-maps, then the DAG obtained from their intersection is a directed minimal I-Map. In other words, it is possible to derive an intersection structure of two independent causal maps that share the same variables if there is at least one common prior node. If so, then the resultant intersection structure will be a minimal I-Map, which is an I-Map that cannot be simplified any further. As a consequence, the intersection structure will conserve all independencies that exist in the original causal structures.

The main constraint in the proposition by Castillo et al. is that both structures must share exactly the same variables. Sagrado et al.'s approach bypasses this limitation by extending the notion of intersection to "extended intersection", which is essentially a union of the two causal maps in reference to the starting prior node; i.e. all nodes reachable from the starting prior node in both causal structures. This solution suits our situation because of the automation it provides, but more importantly, it is non-intrusive and allows us to combine causal maps without the need to ask unwilling domain experts to disclose business process information to other domain experts. Nevertheless, this approach has never been empirically verified, and the authors present only a theoretical account of this method without any real world example. In terms of our research, the drawbacks to this method are as follows:

It assumes that individual models will share at least one prior node in order to perform the union/intersection operation. Although this might seem very probable for our case, given that all the participating companies share the same problem domain, in reality this is not the case. All the company models have defined variables that are very specific to their business process; even though there exist similar nodes shared between the models, their exact definition and context makes them different.

Sagrado and Moral's approach does not preserve, unlike the original proposal by Castillo et al. [44], independence relationships; in fact, it can result in the contradiction of the topological rules of BN causal structures, such as the introduction of cycles. However, they show that whenever there are no head-to-head (converging) edges, the resultant causal map is a minimal I-Map.

We believe that aggregating structures provides one with much more relevant information than to a simple union or intersection of structures, as we can distinguish between more common factors and causal relations in the domain, whereas a simple union or intersection looses this information in the process [45].

5 Proposed Solution

We propose a qualitative methodology that pragmatically addresses the shortcomings of Sagrado et al.'s approach by:

1. Introducing a mapping scheme, i.e., a way to identify similar existing variables in the participating companies' BN models.
2. Instead of using a simple union/intersection, which can only include a common node or edge exactly once, we attempt to aggregate the causal structures. By aggregation, we imply that all edges and nodes in the original map are preserved. As more causal maps are aggregated, the most common variables and causal links emerge, thereby simulating in our view a form of consensus between the different companies' maps.

We termed the resultant aggregated graph as a Causal Structure Aggregation Model (CSAM). Strictly speaking, it is not a unified causal map, but it is a tool for discovering a consensual causal maps. A CSAM is a graph that represents the cumulative union of individual causal maps according to a node mapping scheme. The aim of a structure aggregation model is to identify causal commonalties between independently developed maps that share the same domain. Consider the three causal maps presented in Figure 3. All are used to estimate the total effort required to develop a Web application. Since they all share the same domain, it is possible to assume that the nodes in two different models portray the same factor. For example, nodes A1, B1, and C1 all model the same factor - *the number of developers required to develop a Web application*, and therefore, it is possible to map those three nodes into a single factor.

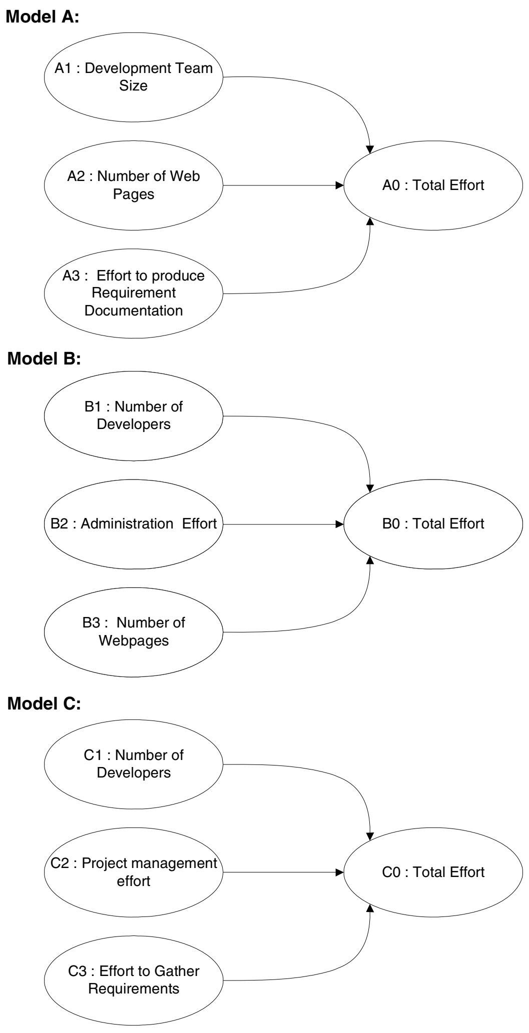

Model A:
A1 : Development Team Size
A2 : Number of Web Pages
A3 : Effort to produce Requirement Documentation
A0 : Total Effort
Model B:
B1 : Number of Developers
B2 : Administration Effort
B3 : Number of Webpages
B0 : Total Effort
Model C:
C1 : Number of Developers
C2 : Project management effort
C3 : Effort to Gather Requirements
C0 : Total Effort

Fig. 3. Three Basic Examples of Causal Maps

Some nodes are more subjective in their definition, e.g., nodes B2 and C2 both attempt to model the effort required to develop a Web application, but the exact details of how to measure this effort might vary between the two companies. However, because both models share the same domain, both B2 and C2 are likely to portray the same underlying concept. By performing this type of mapping between the three models, we can produce the CSAM presented in Figure 4.

The left partition of a CSAM's node represents a factor of interest, while the right partition contains a list of nodes from the original models that map to this factor. All the causal links from the original models are preserved in the CSAM, i.e., if there is a link between two nodes in one of the original models (for example from A2 to A0), then in the CSAM there must be an edge from every node that contains A2 in its mapping to every node that contains A0 in its mapping. The numbers attached to the edges in the CSAM represent the cardinality of their mapping. For example, the edge from node (1) to node (0) has a cardinality of three; this is because there are three original edges that map to it: A1 to A0, B1 to B0, and C1 to C0. The cardinality of a node is the number of 'original' nodes that it maps to (i.e. the number of nodes listed in its right partition). The example in Figure 4 has a simple one-to-one mapping between the CSAM factors and the nodes from the example causal maps. It is possible to have a many-to-many mapping to resolve more ambiguous situations.

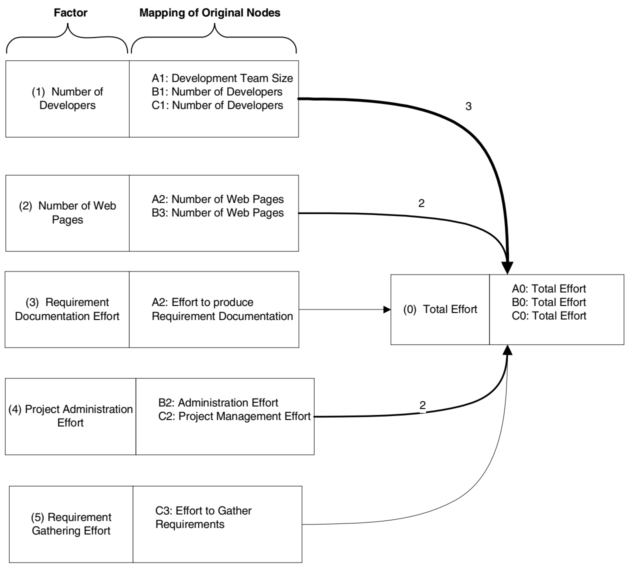

Factor
Mapping of Original Nodes
(1) Number of Developers
A1: Development Team Size
B1: Number of Developers
C1: Number of Developers
3
(2) Number of Web Pages
A2: Number of Web Pages
B3: Number of Web Pages
2
(3) Requirement Documentation Effort
A2: Effort to produce Requirement Documentation
(0) Total Effort
A0: Total Effort
B0: Total Effort
C0: Total Effort
(4) Project Administration Effort
B2: Administration Effort
C2: Project Management Effort
2
(5) Requirement Gathering Effort
C3: Effort to Gather Requirements

Fig. 4. Causal Structure Aggregation Model (CSAM) of the causal maps in Fig. 3

6 Methodology

The goal of this research was to build a CSAM by aggregating six different expert-driven Web effort estimation causal maps. The methodology used to combine these maps comprised a six-step process (detailed below) combining both linear and iterative approaches (see Figure 5).

1) Formatting of the companies' causal maps

The companies' maps were first formatted so they could be handled by the aggregation algorithm (step 4). The formatting consisted of the following steps:

Each node in every map was given a unique Identifier. The identifiers chosen for our research represented a concatenation between a company's causal map identifier and a unique natural number (a number only valid within the context of a single causal map). Each causal map was represented in a parseable format, where the format chosen herein was CSV (Comma Separated Values).

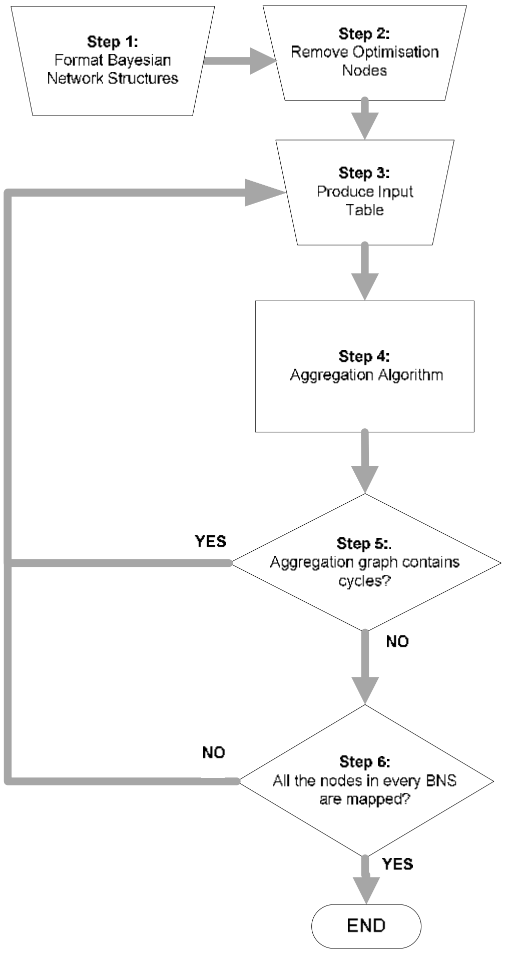

Step 1:
Format Bayesian Network Structures
Step 2:
Remove Optimisation Nodes
Step 3:
Produce Input Table
Step 4:
Aggregation Algorithm
Step 5:.
Aggregation graph contains cycles?
YES
NO
Step 6:
All the nodes in every BNS are mapped?
NO
YES
END

Fig. 5. Process Flow Diagram for Producing a CSAM

The choice relating to the identifiers' representation and parsing format to use was informed by the tool implemented to help with this aggregation process.

2) Removal of Optimisation Nodes

Optimisation nodes are intermediate nodes that were inserted into a causal structure to partition large CPTs in order to reduce their probability elicitation effort. In general, such nodes are not part of the original map elicited with the domain experts; rather, they are suggested by the Knowledge Engineer, and approved by the experts. The purpose of our CSAM was to only aggregate the factors and causal relationships originally modelled by the experts, and as such, the inclusion of optimisation nodes was deemed inappropriate.

Optimisation nodes were first identified from the documentation available for each of the companies' maps. To remove an optimisation node we connected all of its incoming edges (coming from its parent nodes) directly to all of its child nodes, followed by the removal of this optimisation node and all of its outgoing edges (see Fig. 6):

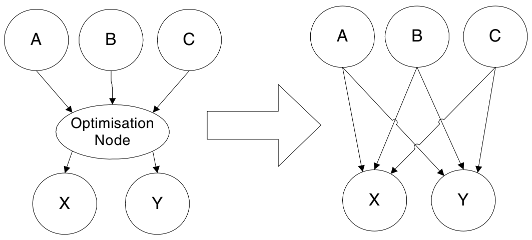


Fig. 6. Removing an Optimization node

During this operation BNs' existing graph rules must always hold (note that each casual map used was part of a larger model – a BN model). For example, only a single edge could have the same source and destination nodes; therefore, if the removal of an optimisation node resulted in adding an edge between two nodes that were already directly linked, then the resultant edge had to be discarded.

3) Creation of an Input Table

Each node in our CSAM corresponded to a semantically equivalent node originating from one of more of the causal maps used as input. Sometimes different causal maps would contain the same node however named differently; when carrying out the mapping (as detailed below) we checked for the semantic equivalence between nodes across causal maps. These mappings were documented using a Table, where each row was used to map a CSAM node to all the other semantically equivalent nodes originating from causal maps. The table's first column represented a CSAM node (factor), identified by a unique ID; the remaining columns contained node identifiers associated with the nodes contained in the input causal maps.

Given a company's input causal map, the first node to be mapped was the most-posterior node, which within our context always happened to be the *Total Effort*. We chose this node because it was part of all the participating companies' causal maps, and therefore we believed it to be the easiest node to identify and map. Once *Total Effort* was mapped, the remaining nodes were mapped according to the following steps:

1. Selection of a node (factor) from a company's causal map that had not yet been mapped.
2. Identification of the contextual meaning of the factor selected in (1), which usually involved interpreting the underlying concept that the DE employed when that factor was elicited. We first identified the units and quantification used to measure the factor, followed by looking at the supporting documentation from the elicitation sessions, which contained examples and additional commentary about the DEs' beliefs. In the rare cases where a factor's contextual meaning was still ambiguous, the DEs were contacted for clarification.

3. Attempt to map the factor identified in (1) (f) to a factor, or set of factors, already present in our CSAM. Whenever there was no corresponding factor(s) clearly mapping to f, we created a new factor(s) within our CSAM to match that given factor f.

There were no strict rules as to whether an original node was mapped to one or more factors within our CSAM; however, we always aimed to keep as much of the original context as possible through the mapping. Thus the reason why our methodology is iterative and not linear is because mappings often change as new factors are created and old ones are revised.

In order to minimise the effort of constantly changing the mappings as the aggregation map was populated, we decided to map the original nodes in different iterations rather than mapping all nodes at once. This gave us the opportunity to run the aggregation algorithm (see step 4 below) and generate the CSAM several times, containing incomplete aggregation maps, and then to look for faults and inconsistencies (e.g. cycles). The first iteration involved mapping every prior node from all the companies' causal maps. The second iteration involved mapping all the nodes from all the companies' maps that were directly pointed to by all prior nodes, and so on until the most posterior (the *Total Effort* Node) was reached.

4) Aggregation Algorithm

The Table prepared in step 3 was used as input to an aggregation algorithm that produced a graphical representation of the CSAM. The algorithm worked by first merging the prior nodes according to the mapping specified in the Table, and continuing until all nodes in all the companies' maps were processed [39].

Whenever the Table from Step 3 did not include mappings for some of the nodes in the inputted causal maps, then these nodes were represented in the CSAM by *placeholder* nodes. The purpose of the placeholder node was so that we were aware of which nodes still required mapping in the next iteration of this process (see Step 6).

5) Check if the Aggregation Graph Contains Cycles

The aggregation algorithm allowed for the occurrence of cycles since it simply followed what was documented in the Table used as input. Therefore, when the generated CSAM graph contained cycles, the input Table needed to be modified so that all of the documented cycles were broken. Cycles could be broken by changing the mapping of one or more nodes that made up the cycle, which could be achieved by either removing or adding factors to the input Table. However, because all the companies' maps were independent of one another and yet shared the same domain, it is theoretically possible, in theory, to have cycles occurring that may not be resolved. This would occur whenever nodes in their original causal maps did not form cycles, but ended up contributing to a cycle in the CSAM due to conflicting contexts.

6) Check if All Nodes are Mapped

The final step in the process was to check whether every node (except for optimisation nodes) in all the companies' causal maps had been mapped in the CSAM. For this we looked for the existence of placeholder nodes in the CSAM outputted by the algorithm. If found, we mapped the map's nodes identified by the placeholder nodes by referring back to Step 3; conversely, if there were no placeholder nodes, we considered that the CSAM was complete according to our mapping.

7 Results

In this section, we present our results from aggregating six independently elicited singly-company causal maps. The elicited models varied in their sizes; as summarised in the following table:

Table 1. List of Causal maps and their sizes

Map	Number of Nodes	Number of Edges
Model A	30	32
Model B	16	20
Model C	15	14
Model D	26	27
Model E	26	29
Model F	19	18

The CSAM[1] resulting from our 6-step methodology (presented in Section 5) enabled us to identify common factors and causal relations shared amongst the six independent single-company BNs. This CSAM presented 67 nodes in total, encompassing all the factors identified by all six participating companies via their BNs. This combined list of factors brought us one step closer to determining all the factors in our target domain (Web development effort estimation), and therefore closer to a unified BN for Web effort estimation. Table 2 lists the Factors[2] in our CSAM and their cardinality, which corresponds to the number of input causal maps that contained that factor. Therefore, a factor's cardinality is an indication of how common this factor was as a predictor amongst the six participating companies.

The most common factor in our CSAM, presenting the highest cardinality on the list, was the '*Number of New Web Pages*'. This may perhaps be an expected result, as the number of Web pages is often likely to be used to determine project scope [2]. This result provides even stronger supporting evidence that there is a causal relationship between new Web pages developed and total development effort; however, given that the sample used was not random we cannot generalise this trend to all remaining Web companies. Another observation is that factors that perhaps may have higher importance in conventional software development did not seem to be common amongst the participating companies; for example, factors related to requirements engineering and documentation. This is perhaps indicative of more agile software development methodologies being employed by the participating companies.

[1] The resultant CSAM is available here:
`http://www.cs.auckland.ac.nz/~emilia/ASEA/CSAM.pdf`

[2] A description of all CSAM factors is given here:
`http://www.cs.auckland.ac.nz/~emilia/ASEA/factors.pdf`

Table 2. List of CSAM Factors and their cardinality

Factor Description	Cardinality	Factor Description	Cardinality
Number of new web pages	6	Effort producing animations using software	1
Number of reused web pages	5	Effort programming animations	1
Number of features off the shelf	4	Is development process documented?	1
Project management effort	4	Means of supplying multimedia	1
Adaptation effort of features off the shelf	3	Number of features off the shelf adapted	1
Average project team experience with technology	3	Number of images (new and reused)	1
Development effort of new features	3	Number of images per page	1
Client difficulty	2	Number of images requiring high effort to manipulate	1
Development team size	2	Number of images requiring low effort to manipulate	1
Number of features requiring high effort to create	2	Number of images requiring medium effort to manipulate	1
Number of features requiring high effort to modify/adapt	2	Number of supplied multimedia	1
Number of features requiring low effort to create	2	Pre-development documentation	1
Number of features requiring low effort to modify/adapt	2	Requirements scope	1
Number of images reused	2	Server side language/framework	1
Number of new images	2	Subcontractor development team size	1
Number of web page templates	2	Type images preparation	1
Requirements clarity	2	Type of application	1
Template structure	2	Effort required to train clients	1
Type of project	2	Amount of text per application	1
Client web literacy	2	Quality of third party deliverables	1
Effort template look & feel	2	Number of key client's people	1
Effort to produce web pages	2	Web company's hosting control	1
Effort to produce template mock-up	2	Effort to reuse features off-the-shelf	1
Amount of text per page	1	Effort to adapt features off-the-shelf	1
Are metrics used throughout the project?	1	Effort of graphical design	1
Are the look & feel requirements provided by the client?	1	Effort research third party features	1
Average project team experience (excluding technology)	1	Effort of search engine optimization	1
Average subcontractor development team experience	1	Effort to produce requirements documentation	1
Client location	1	Effort to produce development documentation	1
Client professionalism	1	Effort to develop user interface	1
Technology (client side) language	1	Requirements complexity	1
Data persistence type	1	Effort to program features	1
Development process model	1	Effort to implement the web application	1
Effort spent on images manipulation	1		

Figure 7 shows the proportion of factors according to their equivalent nodal cardinality. We can see that approximately 66% of all factors appeared in only a single company's map, and 34% of factors were common to at least two maps. The percentage of nodes decreased as the cardinality increased, suggesting that the total number of factors available in the target domain significantly outnumbered the factors being considered by individual companies. Likewise, the percentage of causal edges also rapidly decreases with respect to edge cardinality, which suggests that there are many causal relationships not considered by individual companies.

There were 101 causal edges our CSAM. We were able to determine the most common causal relations by selecting all the matched causal relations in the CSAM, that is, all causal edges with a cardinality of two or more. Our results showed that 16% of all causal relations were shared between at least two maps. The most prevalent causal relationship was between factors '*Project Management Effort*' and '*Total Effort*' whereby 66.6% of the participating companies included such relationships in their causal maps. Other common causal relationships identified were:

- Relationship from *Number of New Web Pages*' directly influencing *'Total Effort'*.
- Relationship from *'Number of Reused Web Pages*' directly influencing *'Total Effort'*.
- Relationship from *'Average Project Team Experience with Technology'* directly influencing *'Total Effort'*.

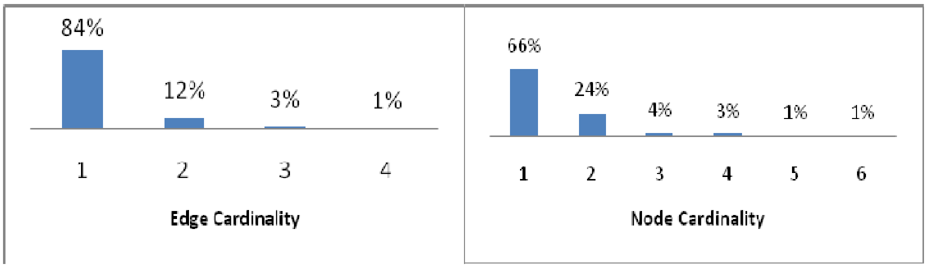


Fig. 7. Distribution of node and edge cardinalities in our CSAM

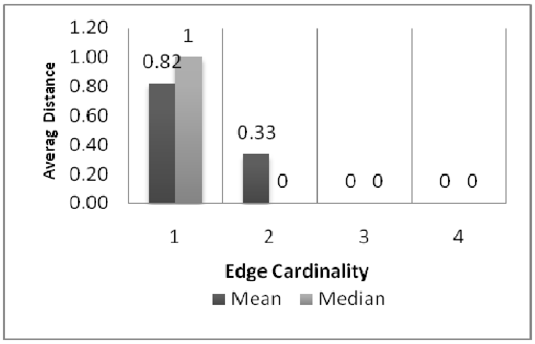


Fig. 8. Average Distance by edge cardinality to Total Effort node

Each of the three abovementioned causal relations appeared in 50% of the companies' maps. Edges with higher cardinality tended to be closer to the most posterior node (*'Total Effort'*). Figure 8 shows a falling trend in the mean and median average distances to the Total Effort node. An average mean distance = 0.82 for edges with cardinality of 1, and mean distance = 0.33 for edges with cardinality of 2. This is in our view an important outcome because 'effort' is what all the causal maps used in this research aim to predict; it is therefore advantageous to know which factors were likely to have a direct effect upon effort, since this would be the focal point of any future consensus-based causal map.

Figure 9 shows a sub-graph of the resultant CSAM with all edges having cardinality greater than two. Factors were grouped into higher level categories (grey boxes) in order to aid readers understand it. This figure can be described as an aggregated intersection of all the causal edges in the inputted causal maps. The higher the weight value of an edge the more common the causal relation is. This figure is therefore very useful as it indicates likely relationships that exist between factors within the Web development domain. We believe that as we further aggregate causal maps to our resultant CSAM, a more informative and decisive consensus will emerge, thus also strengthening the external validity of this model. In other words, a CSAM is a maturing model, providing further certainty as further causal maps are aggregated.

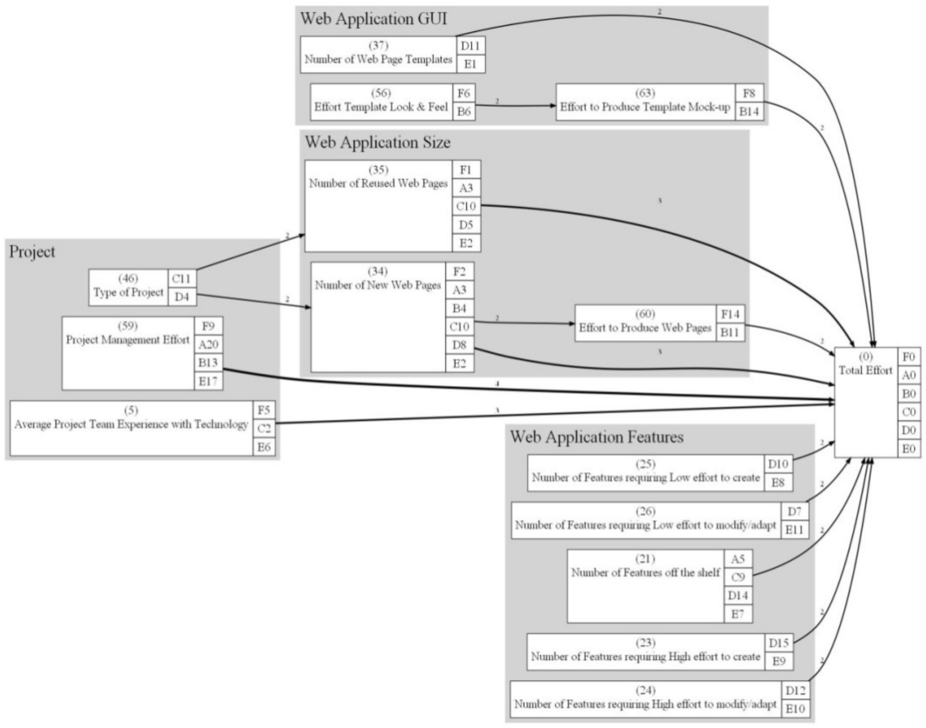

Web Application GUI
(37) Number of Web Page Templates
D11
E1
(56) Effort Template Look & Feel
F6
B6
(63) Effort to Produce Template Mock-up
F8
B14
Web Application Size
(35) Number of Reused Web Pages
F1
A3
C10
D5
E2
(34) Number of New Web Pages
F2
A3
B4
C10
D8
E2
(60) Effort to Produce Web Pages
F14
B11
Project
(46) Type of Project
C11
D4
(59) Project Management Effort
F9
A20
B13
E17
(5) Average Project Team Experience with Technology
F5
C2
E6
(0) Total Effort
F0
A0
B0
C0
D0
E0
Web Application Features
(25) Number of Features requiring Low effort to create
D10
E8
(26) Number of Features requiring Low effort to modify/adapt
D7
E11
(21) Number of Features off the shelf
A5
C9
D14
E7
(23) Number of Features requiring High effort to create
D15
E9
(24) Number of Features requiring High effort to modify/adapt
D12
E10

Fig. 9. A sub-graph of the resultant CSAM with all edges having cardinality greater than two (grouped by category)

8 Threats to Validity

There are a few threats to the validity of our work. One is the mapping of original nodes (i.e. creating the aggregation map in third step in our methodology from Section 5). The mapping was performed by the researchers (i.e. knowledge engineers), not the domain experts; therefore, there is always the possibility of bias being introduced. However, it is important to note that many steps were undertaken to mitigate this risk. All mappings were based on documentation provided by the experts, and for cases where there was ambiguity, the experts were contacted directly for further clarification.

Another threat is that our methodology does not in any way guarantee that the final CSAM is free of cycles. Although for the six company maps, all potential cycles were resolved by further investigation and remapping; this might not always be the case. It is always possible to have intrinsically contradictory causal maps, rendering it impossible to resolve cycles unless at least one edge is omitted from the CSAM.

Finally, for the CSAM to be fully comprehensive in terms of domain factors, it is necessary to aggregate a large number of maps. For our case, the aggregation of six maps is not enough to represent all factors and causal relations that impact effort estimation in the Web development domain. However, we note that the resultant CSAM is a maturing model, and we plan to aggregate further causal maps as part of our future work.

9 Conclusions

The aim of this paper was to investigate further the important factors for Web effort estimation and their cause and effect relationships by aggregated six single-company Web effort estimation causal maps. To build such an aggregated model presents numerous challenges, namely identifying common variables, resolving causal relation conflicts, and company collaboration constraints.

We believe that one can overcome some of these challenges by applying an aggregation process that can yield the most common patterns shared between single-company causal maps. Our proposal for building an aggregated map was based on an earlier proposition by Sagrado et al. [42], which attempted to combine BNs' causal maps using intersection/union of DAGs forming a consensus causal structure. Our proposal improved upon this proposition in two ways: first by introducing a mapping mechanism for grouping related variables from different single-company maps, and secondly by using an aggregation of nodes and edges instead of a simple union/intersection, thus preserving all edges and nodes from the original maps. We termed the aggregated causal map as a Causal Structure Aggregation Model (CSAM), and its chief rationale was to identify structural commonalities (common factors and causal relations) found in the original causal maps.

We have constructed a CSAM using six expert-driven single-company causal maps (part of single-company BNs), all of which elicited from local Web development companies in Auckland, NZ. This CSAM contained 67 factors and 101 causal edges. The resultant CSAM revealed the following patterns: i) 34% of the CSAM factors were shared between at least 2 single-company maps; ii) The most common factor was *'Number of New Web Pages'*; a size measure of Web applications, which supports the findings from earlier studies [2]; iii) The proportion of nodes rapidly decreased as cardinality increased, implying that the total number of factors relevant in the Web effort estimation domain significantly outnumbers the number of factors being considered by individual companies; iv) 16% of all causal relations found in the CSAM were shared between at least two single-company maps; v) The most common causal relationship in our CSAM was between factors '*Project Management Effort*' and '*Total Effort*', included in 66.6% of the single-company maps; vi) Three other common causal relationships which were evident were: '*Number of New Web Pages*', '*Number of Reused Web Pages*', and '*Average Project Team Experience with Technology''*, all of which directly influenced '*Total Effort*'; vii) Edges with higher cardinality tended to be closer to the most posterior node, suggesting that most factors influenced total effort directly.

The abovementioned points show that even with a small number of companies we can already see reasonable commonality in terms of factors and causality. The CSAM is a maturing model, which means that as more causal maps are aggregated; the more common factors and causal links will emerge, hence providing an improved consensus. The aggregation process presented herein can be used to aggregate other causal maps. In addition, to our knowledge this is the first time that a study in either Web or Software Engineering describes the creation of a large causal map for effort estimation via the aggregation of several single-company causal maps. Our future work involves the aggregation of other expert-driven single-company causal maps from companies in NZ, and also overseas.

Acknowledgements

We thank Dr. Mosley[*] and Professor Jorgensen for their comments on an earlier version of this paper, and also all the participating companies in this research. This work was sponsored by the Royal Society of New Zealand (Marsden research grant 06-UOA-201).

References

1. Jensen, F.V., Nielsen, T.D.: Bayesian networks and decision graphs. Springer, Heidelberg (2007)
2. Mendes, E., Mosley, N., Counsell, S.: Investigating Web Size Metrics for Early Web Cost Estimation. Journal of Systems and Software 77, 157–172 (2005)
3. Mendes, E., Counsell, S.: Web Development Effort Estimation using Analogy. In: Proc. 2000 Australian Software Engineering Conference, pp. 203–212 (2000)
4. Fewster, R., Mendes, E.: Empirical Evaluation and Prediction of Web Applications' Development Effort. In: Proc. EASE 2000 (2000)
5. Fewster, R., Mendes, E.: Measurement, Prediction and Risk Analysis for Web Applications. In: Proceedings of IEEE Metrics Symposium, pp. 338–348 (2001)
6. Mendes, E., Kitchenham, B.A.: Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. In: Proc. IEEE Metrics, pp. 348–357 (2004)

[*] `http://www.metriq.biz`

7. Mendes, E., Mosley, N.: Does the Linear Size Adjustment to Estimated Effort Improve Web Applications Effort Estimation Accuracy? Special Issue of the Journal of Computational Methods in Science and Engineering 5(1), 171–184 (2005)
8. Mendes, E., Mosley, N.: Web Cost Estimation: principles and applications. In: Khosrow-Pour, M., Travers, J. (eds.) Web Engineering – Principles and Techniques, pp. 182–202. Idea Group, Inc., USA (2005)
9. Mendes, E., Mosley, N.: Further Investigation into the Use of CBR and Stepwise Regression to Predict Development Effort for Web Hypermedia Applications. In: Proc. ACM/IEEE ISESE, Nara, Japan, pp. 79–90 (2002)
10. Mendes, E., Counsell, S., Mosley, N.: Web Hypermedia Cost Estimation: further assessment and comparison of cost estimation modelling techniques. NRHM 8, 199–229 (2002)
11. Mendes, E., Counsell, S., Mosley, N.: Towards the Prediction of Development Effort for Hypermedia Applications. In: Proc. Hypertext 2001, pp. 249–258 (2001)
12. Mendes, E., Mosley, N., Counsell, S.: Exploring case-based reasoning for Web hypermedia project cost estimation. IJWET 2(1), 117–143 (2005)
13. Mendes, E., Mosley, N., Counsell, S.: A Replicated Assessment of the Use of Adaptation Rules to Improve Web Cost Estimation. In: Proc. ISESE, pp. 100–109 (2003)
14. Mendes, E., Mosley, N., Counsell, S.: Early Web Size Measures and Effort Prediction for Web Costimation. In: Proceedings of the IEEE Metrics Symposium, pp. 18–29 (2003)
15. Mendes, E., Mosley, N., Counsell, S.: Comparison of Length, complexity and functionality as size measures for predicting Web design and authoring effort. IEE Proc. Software 149(3), 86–92 (2002)
16. Mendes, E., Mosley, N., Counsell, S.: The Application of Case-Based Reasoning to Early Web Project Cost Estimation. In: Proc. Compsac 2002, pp. 393–398 (2002)
17. Mendes, E., Mosley, N., Counsell, S.: Web metrics - Metrics for estimating effort to design and author Web applications. IEEE MultiMedia, 50–57 (January-March 2001)
18. Mendes, E., Mosley, N., Counsell, S.: Using an Engineering Approach to Understanding and Predicting Web authoring and Design. In: Proc. HICSC (2001)
19. Mendes, E., Mosley, N., Watson, I.: A Comparison of Case-Based reasoning Approaches to Web Hypermedia Project Cost Estimation. In: Proc. WWW 2002 (2002)
20. Mendes, E., Watson, I., Triggs, C., Mosley, N., Counsell, S.: A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. ESE 8(2), 163–196 (2003)
21. Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E.: Comparing Size Measures for Predicting Web Application Development Effort: A Case Study. In: Proceedings ESEM 2007 (2007)
22. Reifer, D.J.: Web Development: Estimating Quick-to-Market Software. IEEE Software, 57–64 (November-December 2000)
23. Ruhe, M., Jeffery, R., Wieczorek, I.: Cost estimation for Web applications. In: Proceedings ICSE 2003, pp. 285–294 (2003)
24. Baresi, L., Morasca, S., Paolini, P.: An empirical study on the design effort for Web applications. In: Proceedings of WISE 2002, pp. 345–354 (2002)
25. Baresi, L., Morasca, S., Paolini, P.: Estimating the design effort for Web applications. In: Proceedings of Metrics 2003, pp. 62–72 (2003)
26. Mangia, L., Paiano, R.: MMWA: A Software Sizing Model for Web Applications. In: Proc. Fourth International Conference on Web Information Systems Engineering, pp. 53–63 (2003)
27. Mendes, E.: Predicting Web Development Effort Using a Bayesian Network. In: Proceedings of EASE 2007, pp. 83–93 (2007)

28. Mendes, E.: The Use of a Bayesian Network for Web Effort Estimation. In: Baresi, L., Fraternali, P., Houben, G.-J. (eds.) ICWE 2007. LNCS, vol. 4607, pp. 90–104. Springer, Heidelberg (2007)
29. Mendes, E., Polino, C., Mosley, N.: Building an Expert-based Web Effort Estimation Model using Bayesian Networks. In: 13th International Conference on Evaluation & Assessment in Software Engineering (2009)
30. Rajabally, E., Sen, P., Whittle, S., Dalton, J.: Aids to Bayesian belief network construction. In: Proceedings of 2004 2nd International IEEE Conference on Intelligent Systems, vol. 2, pp. 457–461 (2004)
31. Mendes, E., Mosley, N.: Bayesian Network Models for Web Effort Prediction: A Comparative Study. IEEE Trans. on Soft. Engineering 34(6), 723–737 (2008)
32. Mendes, E., Mosley, N., Counsell, S.: Investigating Web Size Metrics for Early Web Cost Estimation. Jour. of Systems and Software 77(2), 157–172 (2005)
33. Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E.: Applying Support Vector Regression for Web Effort Estimation using a Cross-Company Dataset. In: Proceedings of the ACM/IEEE Symposium on Empirical Software Measurement and Metrics, pp. 191–202 (2009)
34. Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E.: Using Support Vector Regression for Web Development Effort Estimation. In: Abran, A., Braungarten, R., Dumke, R.R., Cuadrado-Gallego, J.J., Brunekreef, J. (eds.) IWSM 2009. LNCS, vol. 5891, pp. 255–271. Springer, Heidelberg (2009)
35. Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., Mendes, E.: Investigating Tabu Search for Web Effort Estimation. In: Proceedings of Euromicro SEAA 2010 Conference (2010)
36. Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E.: How the Choice of the Fitness Function Impacts on the Use of Genetic Programming for Software Development Effort Estimation? In: Proceedings of PROMISE 2010 (2010) (best paper award)
37. Brown, B.B.: Delphi process: A methodology used for the elicitation of opinions of experts, Santa Monica, CA, Rand Corporation (1968)
38. Woodberry, O., Nicholson, A., Korb, K., Pollino, C.: Parameterising Bayesian Networks. In: Australian Conference on Artificial Intelligence, pp. 1101–1107 (2004)
39. Baker, S.: Towards the Construction of Large Bayesian Networks for Web Cost Estimation. In: Department of Computer Science Auckland: University of Auckland (2009)
40. Montironi, R., Whimster, W.F., Collan, Y., Hamilton, P.W., Thompson, D., Bartels, P.H.: How to develop and use a Bayesian Belief Network. Journal of Clinical Pathology 49, 194 (1996); Mendes, E.: A Comparison of Techniques for Web Effort Estimation. In: First International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, pp. 334–343 (2007)
41. Fink, A., Kosecoff, J., Chassin, M., Brook, R.H.: Consensus methods: characteristics and guidelines for use. American Journal of Public Health 74, 979 (1984)
42. Sagrado, J.D., Moral, S.: Qualitative combination of bayesian networks. International Journal of Intelligent Systems, 237–249 (2003)
43. Flesch, I., Lucas, P., Gamez, J.A., Salmeron, A.: Markov Equivalence in Bayesian Networks (2007)
44. Castillo, E., Gutiérrez, J.M., Hadi, A.S.: Combining multiple directed graphical representations into a single probabilistic model. In: Actas de la Séptima Conferencia Espanola para la Inteligencia Artificial, CAEPIA, pp. 645–652 (1997)
45. Hu, X.-x., Wang, H., Wang, S.: Using Expert's Knowledge to Build Bayesian Networks. In: Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops, pp. 220–223 (2007)

Bug Forecast: A Method for Automatic Bug Prediction

Rudolf Ferenc

University of Szeged, Department of Software Engineering
H-6720 Szeged, Árpád tér 2, Hungary
`ferenc@inf.u-szeged.hu`
`http://www.inf.u-szeged.hu/~ferenc`

Abstract. In this paper we present an approach and a toolset for automatic bug prediction during software development and maintenance. The toolset extends the Columbus source code quality framework, which is able to integrate into the regular builds, analyze the source code, calculate different quality attributes like product metrics and bad code smells; and monitor the changes of these attributes. The new bug forecast toolset connects to the bug tracking and version control systems and assigns the reported and fixed bugs to the source code classes from the past. It then applies machine learning methods to learn which values of which quality attributes typically characterized buggy classes. Based on this information it is able to predict bugs in current and future versions of the classes.

The toolset was evaluated on an industrial software system developed by a large software company called evosoft. We studied the behavior of the toolset through a 1,5 year development period during which 128 snapshots of the software were analyzed. The toolset reached an average bug prediction precision of 72%, reaching many times 100%. We concentrated on high precision, as the primary purpose of the toolset is to aid software developers and testers in pointing out the classes which contain bugs with a high probability and keep the number of false positives relatively low.

Keywords: bug prediction, machine learning, software product metrics, bad code smells.

1 Introduction

The aim of this project was to provide short term bug prediction on *evosoft*'s, a large software company's commercial software system by using the results of static source code analysis and assigning bug information from the past to classes and code analysis results. The developed method and toolset forecast the faulty classes and the tools make the monitoring of them easy and user friendly. The motivation of the bug prediction is to make a toolset available for the programmers by which they are able to point out possibly problematic classes, which can contain bugs. By pointing out critical classes the testing and code-reviewing phases could be planned easier, therefore the resource management of programmers and testers gets more efficient. By using the described bug prediction methods the cost and time planning of the maintenance and the development of new features are getting more accurate.

For the research and evaluation purposes evosoft handed over its source code and bug database. This dataset is a collection of a continuous industrial development of a commercial software and it was collected for more than 1.5 years. More than 128 former snapshots of the software code were analyzed. The snapshots changed regularly on three to seven days periods. As the result of the static code analysis, product metrics (e.g. LOC – Lines of Code, CBO – Coupling Between Object Classes) [4] and the number of different bad code smells (e.g. DC – Data Class, FE – Feature Envy) [8] were calculated for all the classes. Combining the metrics with the bug number information, which was available for all the classes, a correlation among product metrics and faultiness of classes could be set up.

During the research project a bug forecast toolset was developed. This toolset extends the Columbus source code quality framework [7], which is able to integrate into the regular builds, analyze the source code, and calculate different quality attributes like product metrics and bad code smells. The toolset extracts the bug information (e.g. number of bugs, the date of opening and closing) by connecting to the bug and version tracking systems. The information gets assigned to the particular source code class, which makes the tracking of the bugs' life cycle available. The information is stored in the platform independent source code quality assurance tool, called SourceInventory [2], which queries, monitors and tracks system development life cycle attributes (e.g. product metrics). Machine learning methods are performed on the dataset by the toolset.

As the toolset was developed only recently, the efficiency of the toolset was evaluated by performing the following simulation: if the developers used the toolset from the beginning of the development for 1.5 years, how accurate bug prediction would they get.

Several machine learning algorithms were studied, and it turned out that the tree-based models performed best. The best choice was the best-first decision tree algorithm, which provided the best results for accuracy, precision and recall (indicators). Experiments showed that to achieve the best indicator values the learning dataset had to be constructed to hold 6 weeks of data before the chosen date.

The machine learning process was done by learning on binary bug data, which means that the classes were classified as having at least one bug or not having bugs. Recall means the number of correctly identified classes having bug(s) divided by the total number of classes having bug(s). On the other hand, the dataset containing the number of bugs per a class was also available. Therefore the completeness values were also calculated, which means the number of bugs in the classes identified as buggy divided by the total number of bugs in the current snapshot. The above results indicate that the true positive (found real buggy classes) rate is high and the false positive (not buggy class identified as buggy) rate is low. The goal of the research project was to help developers in finding buggy classes. Having a high precision and a small false positive rate provides valuable bug prediction data.

Related Work. For the purpose of bug prediction in software systems researchers already used several approaches. Diverse machine learning algorithms, and source code, version, and bug attributes were used to learn on one or more versions of a software system covering small to large development time frames. Bug prediction can successfully be done by extracting source code metrics and assigning them to bug information [3]. The detection of design problems based on metrics and data linking of the version control system was done by Marco D'Ambros [5]. The bugs can be predicted for packages, files [11], classes [9] and for procedures. Bug prediction methods could differ in large-scale open-source software systems [6] covering long-term developments and small projects covering short-term development time. Effective bug prediction could also be done by the evaluation of the bug tracking and version control system without analyzing the source code [1]. Combining the bug tracking system's information with the source code change patterns can also lead to high precision results [10].

The following section describes the tools for extracting the metrics and assigning the bug numbers to the classes. Section 3 describes our experiment and shows the detailed results on the predictions about the expected bug occurrences in the classes. Finally, Section 4 concludes our paper and outlines some directions for future work.

2 The Bug Forecast Toolset

The *Columbus* source code quality framework [7] was developed to accomplish source code analysis. The tools in the framework are capable to analyze source code written in programming languages such as C/C++, C#, Java, PL/SQL, Transact SQL, Magic and Pyton. The analysis results contain data about source code product metrics, coding rule violations, bad code smells [8], code duplications, etc. A system development life cycle attribute monitoring, querying and analyzing software named *SourceInventory* [2] is also part of the framework. SourceInventory provides statistics and tendencies in many forms of visualization possibilities (e.g. bar chart, time line, histogram) about the source code's properties and changes. In the remaining part of this article the tools of the Columbus framework, except SourceInventory, will be referred to as *source code analyzer tools.*

The bug forecast toolset, containing three new tools (*BugMiner*, *BugLearner* and *BugPredictor*), was developed for the purpose of extending the existing framework with bug prediction capabilities (see Figure 1). The extension was done in a two step process, as a result two new metrics – the *BugNumber* (the number of fixed bugs for classes and methods) and the *BugPrediction* (the probability of containing bugs for classes) – were added to the old ones.

As a first step, a tool called *BugMiner* was suited into the process. This tool can assign the number of bugs to each source code element that contains them (i.e. to classes, methods and functions). The execution of this new tool is taking place right after the source code analyzer tools have successfully ran and their results get uploaded into the database of SourceInventory. By processing these results the new metric called the *BugNumber* was gained.

During the second step, the other two new tools were added to the framework, which are responsible for performing the bug prediction. *BugLearner* is a tool for training a bug predictor model based on past metric values and BugNumber-s of the classes. The *BugPredictor* tool's task is to predict faulty classes by using the created learning model and the recently calculated metric values (predictors). The result of the prediction for all the classes is the new metric called *BugPrediction*. Its value resembles the information about a class' faultiness based on the prediction. Both tools are using the *Weka* data mining and machine learning tool, which has a good reputation in this research area.

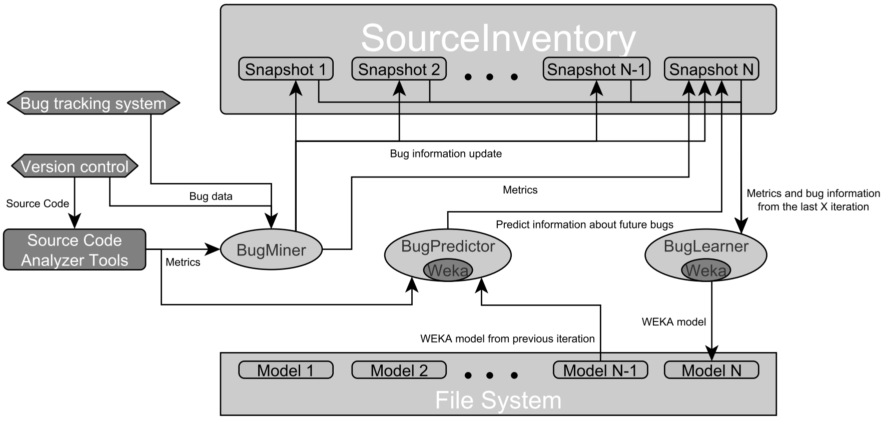


Fig. 1. N^{th} analysis iteration of the Columbus framework

The analysis done by the Columbus framework extended with the bug toolset is performed on a regular basis, typically during the nightly builds of the analyzed and monitored software. The N^{th} execution iteration of the analysis is presented in Figure 1. Except the first execution (when learning data is not yet available due to the lack of a previous analysis) the process requires the learning model provided by the previous analysis iteration.

In the following, a detailed description is given step-by-step about the process. The steps follow the way of the data-flow during one analysis iteration. The first step of the process covers the extraction of the bug data and the execution of the analysis with the help of the source code analyzer tools. The BugMiner tool directly connects to the bug tracking and version control system of the analyzed software (currently it is compatible with the well-known bug tracking systems BugZilla and IssueZilla). In this work, BugMiner was modified to work also with the evosoft company's own proprietary bug tracking system. The data was available in a textual file format, which contained the bug identifiers, bug opening and closing dates, the patches correcting the bugs: patch release dates and the changes made by the patches. These are the information needed for the proper execution of the bug toolset.

The input of BugMiner is the bug data and the data generated by the source code analyzer tools, which contains detailed position information of all the language elements (i.e. file name with full path, starting and ending lines and columns in the file). The data from the previous analysis iterations is also used. After all the needed information is gathered and loaded, BugMiner examines all the bugs, which had new patches since the last analysis. The bugs having new patches and the source code affected by the patches are examined one-by-one. It can be easily done, because the patch files contain the information about the affected files and lines (deleted, inserted, replaced lines). By using this file and line information the affected class(es) are identified and the bug IDs are assigned to the classes and methods of the currently analyzed snapshot of the software.

As the bug reports are typically older than the bug fixing patches, the bug IDs are assigned to the affected classes in all of the affected previous snapshots as well as illustrated in Figure 2. It was assumed that the bug occurred in the snapshot just before it was reported (unfortunately it was not known since when the bug already existed in the source code without being noticed) and that it existed in all snapshots right until it got fixed.

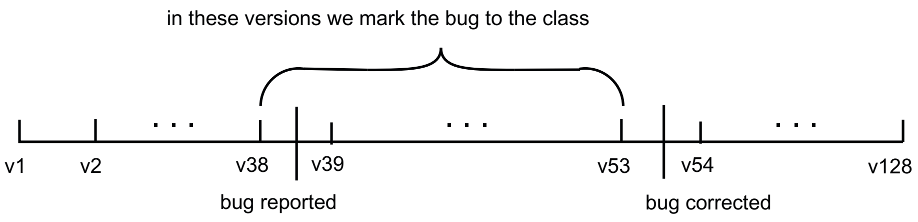


Fig. 2. Affected snapshots by a concrete bug

The BugMiner tool creates two types of outputs. The first output is basically the same as the output of the source code analyzer tools, except that some non-relevant technical information is added to it. The second type of output contains the data which is needed for updating the BugNumber metric for all the affected previously stored analysis iteration results in the database of SourceInventory because of the newly fixed bugs. This data upload information is called *BugUpdate*. This update is necessary because a bug's life can be very long. That means a bug turns out to be present in numerous previous analyses only delayed (after it gets fixed) so the BugNumber-s of the affected classes have to be increased in all of the previous snapshots covering the bug's life cycle. Before the BugUpdate output is created, handling of the possible overlaps between bugs and patches must be done. E.g. if a class is marked twice because it was affected by two patches, then the BugNumber for this class is 2 if the patches belonged to different bugs. If the BugNumber is 1, then the patches belonged to the same bug.

The next step is the uploading. During this step the first output of BugMiner is taken and is being uploaded into the database of SourceInventory. After the uploading is done all the BugNumber metrics in the database are getting updated using the BugUpdate output of BugMiner.

After the analysis, the bug learning and predicting phase begins. The second new tool called *BugPredictor* is invoked, which is a wrapper around Weka. As a first step it transforms the metric data created by the source code analyzer tools to a Weka compatible file format called *arff*. The transformed temporary file becomes the input for the Weka tool, which uses the previous iteration's bug predictor model to calculate the BugPrediction metrical values for all of the classes in the currently analyzed snapshot of the software. The BugPrediction metrics are uploaded into the SourceInventory database, so the N^{th} snapshot of the analysis results will be completed.

The last step of the process is to gather predictor information from SourceInventory (based on the last K weeks' analysis results, see Section 3) to create a new bug predictor model including the results of the current analysis iteration (this information is also saved into an arff file). As this dataset contains quite a large amount of repeated lines, the non-faulty and unchanged classes through all the involved snapshots are filtered out. This filtered dataset will be the input for the third new tool, called *BugLearner*, which is also a wrapper for Weka. The BugLearner creates a new bug predictor model based on the filtered dataset and the chosen learning algorithm (see Section 3). This model is going to be used in the next analysis iteration.

3 Case Study

The most important part of the validation of the extended framework was to see how well it can predict bugs in an evolving system with the continuously collected data. The goal was to tailor the framework to be used every day by the developers, testers and QA managers. During the development, while the source code is changing, they have to be able analyze the new snapshots and predict the buggy classes based on the experience from the past. In other words, a bug prediction on an N^{th} snapshot of a software is needed, which is based on learning results from the last K weeks. This aspect is different than most of the other studies in this field, which used ten-fold cross validation for measuring the efficiency of the prediction model which are built based on collected data from typically larger development intervals. Compared to this, our experiment used several snapshots of the analyzed software that usually changed just a little bit between two snapshots.

3.1 The Simulation

Evosoft Ltd. uses the Columbus framework in the everyday development activities, so all the analysis data were available from the software's 1.5 year development period. Additionally the database of their bug tracking system was also available. The data used in our experiment included source code product metrics, bad code smell metrics and bug reports, so all the information was available to know which classes were buggy and when (see Section 2). In the first 3 months of the software development there were no usable bug reports, so the collected data from this time interval could not be used. The same case was with the last 10 snapshots of the software, since no bug reports were available for that time frame. Hence, the available and usable learning dataset covered 128 former snapshots. Because the bug toolset was developed only recently, it was decided to test its efficiency by performing a simulation which showed the accuracy of the bug prediction in the case if the toolset was used on a daily basis during the examined 1.5 years.

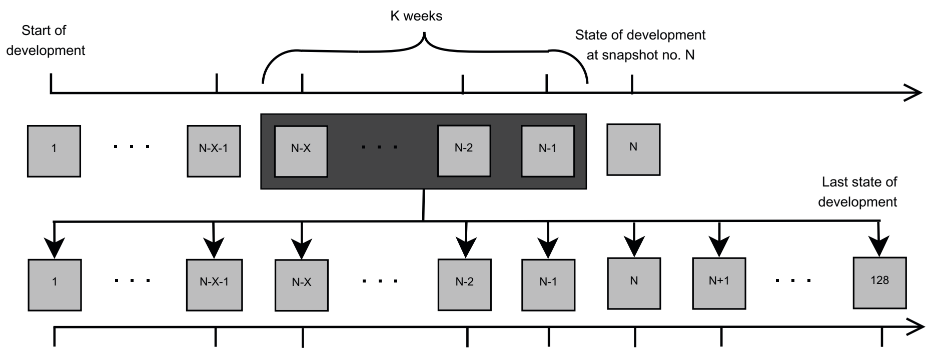

Start of development
K weeks
State of development at snapshot no. N
1
N-X-1
N-X
N-2
N-1
N
Last state of development
N+1
128

Fig. 3. Simulation time line showing the N^{th} iteration

For the machine learning tasks the well known Weka system was applied, since it implements most of the important learning algorithms. The first step was a ten-fold cross validation to find a good machine learning algorithm for the next step. The second step was the real validation, the simulation of the toolset's functionality that is illustrated in Figure 3. The upper time line shows the real-life dataset at a given moment in time (the N^{th} snapshot). This dataset contained only the bug information that was known at the given moment. The lower time line shows the testing dataset, which contained all the bug information that was known at the end of the 1.5 year period.

The figure shows that there were 128 snapshots of the analyzed software. The validation process created a learning dataset for every N^{th} snapshot ($N \in \{1 \dots 128\}$) and ran a machine learning algorithm to create a model for bug prediction. Then the simulation made a prediction for every snapshot of the test dataset (lower time line). During the process, analysis and bug results from K weeks before the N^{th} date containing the snapshots from N-1 to N-X (see the boxes with dark grey background) were taken and joined into a learning dataset. The important case was to predict bugs precisely in the N^{th} snapshot and in the following few snapshots, but for research reasons the predictions were made for all of the snapshots (including the past snapshots as well).

An important question arose: What is the best value for the K time interval? To decide this, the simulation was ran with different K values and the best one was chosen (see Section 3.4).

3.2 Definitions

For measuring the efficiency of the bug predictor models, the accuracy, precision, recall and completeness values were calculated. The accuracy, precision and recall are standard statistic quantities.

It is important to note that if there were no buggy class examples in a test dataset then precision and recall was defined as 0.0 apart from some other studies which define it as 1.0 (hit 0 bugs from 0 bugs). It is also explicable to define it as 1.0, but if there would be several test datasets where bug numbers are 0 then the results would be overrated.

The models were trained to provide binary predictions, which means that they predict if a class is prone to be faulty or not (a class contains at least one bug or not). But the developed bug toolset extracts exact bug numbers from the bug tracking systems and this information is stored as well. By using this information the completeness was also calculated. Completeness measures what percentage of bugs the model reveals from all of the bugs in a current snapshot.

A predicator model's best result is a balanced high value of both precision and recall. However, this is generally not the case. In practice, developers and testers use the bug prediction to reduce the time spent for testing. The less code reviews and tests brings up the bugs, the better. Hence, the developed bug predictor toolset has to be useful in practice, so the most important thing is to reach a high precision value (resulting in few false positives).

3.3 Ten-Fold Cross Validation

The first experiment was to find a good machine learning algorithm for the validation. The ten-fold cross validation was used to choose between the algorithms. It was running on all of the analyzed data. This is the dataset shown on the lower time line in Figure 3 joined together. As can be seen on Figure 4, this validation gives a much better result opposed to the simulation. The statistics were

Used Algorithm	Accuracy	Precision	Recall
Bayes Net	0.735	0.667	0.603
Naive Bayes	0.713	0.656	0.52
Logistic Regression	0.744	0.747	0.493
Voted Perceptron	0.591	0.470	0.596
Decision Tree	0.832	0.802	0.742
Conjunctive Rule	0.695	0.623	0.502
J48	0.879	0.876	0.795
Best-First Dec. Tree	0.875	0.856	0.809

Fig. 4. Ten-fold cross validation with some machine learning methods

very good, but this may not be true on the forecast tests. In this test several algorithms were used which are implemented in Weka, like rule based, bayes, function based methods and others. The best models were generally created by the tree based algorithms. We chose the best-first decision tree algorithm as the algorithm for predicting. This gave a high accuracy value of 87.5% and 85.6% of precision with a 80.9% recall.

3.4 The Validation

As the first step of the validation, for every N^{th} snapshot the learning data included all the analysis results starting from the 1^{st} snapshot. As the first attempts showed, this method was not usable to effectively predict the bug occurrences. If major changes were made during a software's development (these are called drifts), it influences the prediction efficiency. The number of bugs during the development is generally decreasing and apart from significant refactoring phases in the source code, a class will not change much after a while. So in later stages of the development, a group of some metric values, which previously marked the class as faulty are not likely to correctly mark the classes faulty again. As a summary it could be said that joining old and new learning data could create an inconsistency in the predictions.

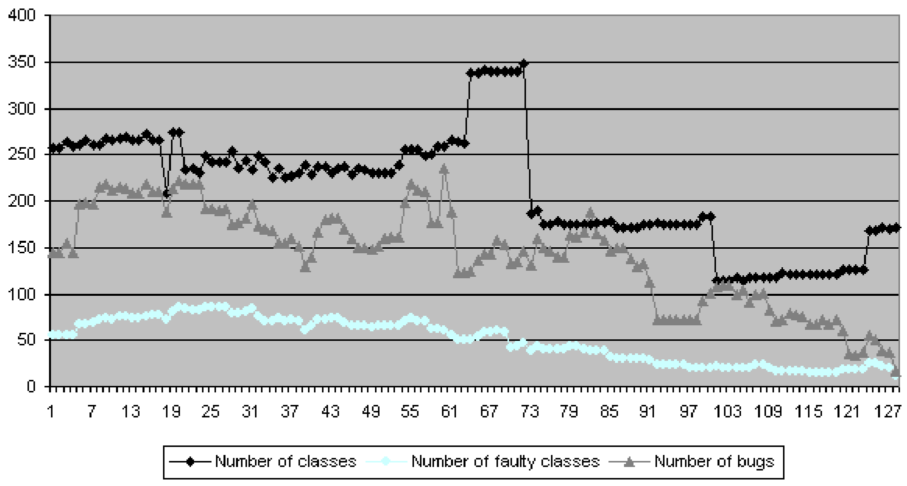


Fig. 5. Details about the measured software system

Figure 5 shows some details of the snapshots of the examined software: the number of classes, the amount of classes containing bug(s) and the number of bugs in the software. As can be seen in the figure, the software was probably going through some refactoring phases and/or some major changes on some dates. These changes are much likely to influence the quality of the prediction.

As for the next step the learning tasks were performed on K time intervals. In this case the bug information data are those that were known till the selected dates, not all the data available till the last development snapshot. With this kind of learning data more actual rules for prediction could be obtained by not letting the elder ones to influence the model. This is important since adequate predictions for the future could not be made by using a full retrospective dataset. The chosen time intervals for K were 4, 8, 12, 16 and 20 weeks in the first round. After the experiments were done the tests pointed out that the interval having the bests results is between 4 and 12 weeks. So the predictions were ran with the dataset which was built from the results of the past 5, 6, 7, 9, 10 and 11 weeks (including 4, 8 and 12 from the previous run). To find the best precision, recall and completeness values various statistical calculations were made. These statistics were calculated for the next 4 snapshots from the actually selected date as the prediction was planned to cover only the near future. These statistics showed that the best value for K was 6 weeks.

Figure 6 shows the average, median, standard deviation, minimum and maximum values for the precision, recall, and completeness. E.g. an average precision means that a learning task with 6-week retrospective data was made on every snapshot available (128 pieces) and an average was calculated from the resulting precision values. These values were calculated for 1, 2, 3 and 4 weeks forward prediction.

As can be seen, the average and median precision were above 70% and the deviation was around 25%. Unfortunately, the recall values could be considered low. These values are presenting the percentage of the revealed buggy classes by the prediction, compared to the number of all buggy classes. The observations showed that as the learning time interval was raised, the recall values were

Precision	1	2	3	4
Average	0.7181	0.6963	0.6892	0.7022
Median	0.7417	0.7143	0.7000	0.7029
Deviation	0.2535	0.2671	0.2601	0.2499
Max.	1.0000	1.0000	1.0000	1.0000
Min.	0.0000	0.0000	0.0000	0.0000
Recall	**1**	**2**	**3**	**4**
Average	0.1204	0.1213	0.1231	0.1257
Median	0.1013	0.1013	0.1032	0.1053
Deviation	0.0833	0.0833	0.0841	0.0850
Max.	0.4200	0.4118	0.3889	0.3966
Min.	0.0000	0.0000	0.0000	0.0000
Completeness	**1**	**2**	**3**	**4**
Average	0.2331	0.2354	0.2331	0.2353
Median	0.2250	0.2250	0.2200	0.2100
Deviation	0.1400	0.1417	0.1402	0.1380
Max.	0.5800	0.5800	0.5900	0.5900
Min.	0.0000	0.0000	0.0000	0.0000

Fig. 6. 4-snapshot forward prediction statistics based on the 6-week interval learning data

getting better but the precision values were falling at the same time. However, a maximum recall value of 42% was reached. The average was around 12%.

The completeness values were calculated as well to check the rate of the found bugs in a snapshot compared to all of the bugs in it. If a higher value could be reached than the recall, it would mean that faultier classes were found. The results showed that the completeness was higher than the recall with an average of 9.5% and the difference was about 20-30% many times. The average completeness was 23-24%.

As can be noticed, the minimum values of the precision, recall and completeness were 0.0. The cause of these values were exceptional cases, where the system was not able to predict, as the 6-weeks learning data did not contain enough samples of buggy classes to perform a successful machine learning task. (In the case of larger time intervals for learning, these cases were gone, but the precision values were much lower.) There were many cases where the precision reached a maximum of 100% (1.0 value) for all of the 4 predicted snapshots. Figure 7 shows a histogram about the first, second, third and fourth snapshot's prediction's precision results. As it is presented, the 1.0 precision result was the most frequent case among the predictions.

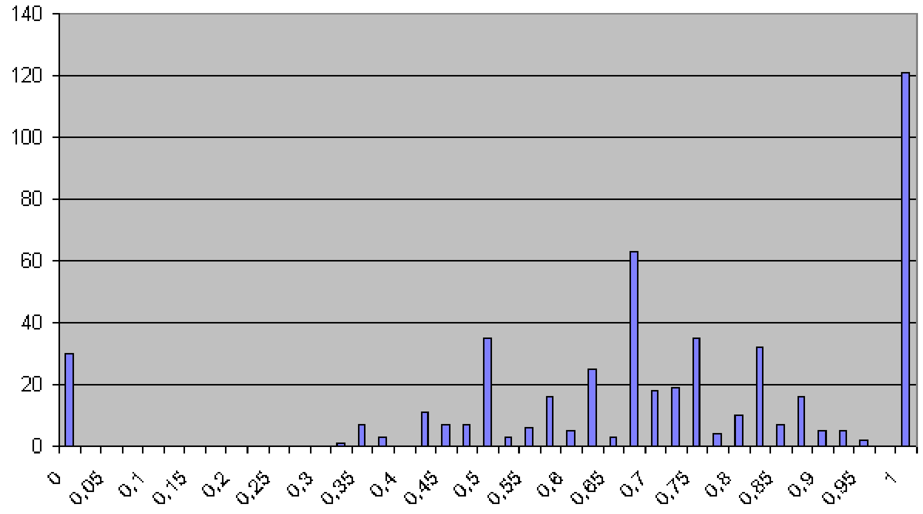


Fig. 7. The 6-weeks interval's learning precision histogram in the case of 4-snapshot prediction

4 Conclusion

The Columbus source code quality framework was extended with a bug prediction functionality, the bug forecast toolset. The toolset connects to the bug tracking and version control system and assigns the reported and fixed bugs to source code elements (classes, methods, functions). The Columbus framework extracts among others product and bad code smell metrics from the source code by using static code analysis methods. The toolset is capable to integrate into the regular builds. Combining the metrics and bug information, a relation among them could be set up, which serves as a basis for bug prediction. By the usage of machine learning methods a trustful bug prediction became available predicting for a future of 4 snapshots development time, with an average precision of 72%, reaching 100% many times.

The toolset was evaluated in an industrial environment. Evosoft's software was analyzed for a 1.5 year development time, which covered 128 snapshots. The research's focus was to reach high precision values to aid developers and testers with valid bug information by listing them the classes, which contain bugs with a high probability. The bug prediction helps to focus resources on possibly problematic code parts, therefore it helps increasing the software's quality and makes the resource management of developers and testers easier.

Acknowledgements

This research was supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and the Hungarian national grants TECH 08-A2/2-2008-0089, GOP-1.1.2/1-2008-0007, and OTKA K-73688.

References

1. Ayari, K., Meshkinfam, P., Antoniol, G., Di Penta, M.: Threats on Building Models From CVS and Bugzilla Repositories: the Mozilla Case Study. In: Proceedings of the 2007 Conference of the Center for Advanced Studies on Collaborative Research, CASCON 2007, pp. 215–228 (2007)
2. Bakota, T., Beszédes, Á., Ferenc, R., Gyimóthy, T.: Continuous Software Quality Supervision Using SourceInventory and Columbus. In: Research Demonstrations of 30th International Conference on Software Engineering (ICSE 2008), pp. 931–932 (May 2008)
3. Basili, V.R., Briand, L.C., Melo, W.L.: A Validation of Object-Oriented Design Metrics as Quality Indicators. IEEE Transactions on Software Engineering 22, 751–761 (1996)
4. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
5. D'Ambros, M.: Supporting Software Evolution Analysis with Historical Dependencies and Defect Information. In: IEEE International Conference on Software Maintenance, pp. 412–415 (2008)
6. Ekanayake, J., Tappolet, J., Gall, H.C., Bernstein, A.: Tracking concept drift of software projects using defect prediction quality. In: 6th IEEE International Working Conference on Mining Software Repositories, pp. 51–60 (2009)
7. Ferenc, R., Beszédes, Á., Tarkiainen, M., Gyimóthy, T.: Columbus – Reverse Engineering Tool and Schema for C++. In: Proceedings of the 18th International Conference on Software Maintenance (ICSM 2002), pp. 172–181. IEEE Computer Society, Los Alamitos (October 2002)

8. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, Reading (1999)
9. Gyimóthy, T., Ferenc, R., Siket, I.: Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. IEEE Transactions on Software Engineering 31, 897–910 (2005)
10. Kim, S.: Adaptive bug prediction by analyzing project history. PhD thesis, Advisor-E. James Whitehead, Jr. (2006)
11. Zimmermann, T., Premraj, R., Zeller, A.: Predicting Defects for Eclipse. In: Third International Workshop on Predictor Models in Software Engineering (2007)

TCD: A Text-Based UML Class Diagram Notation and Its Model Converters

Hironori Washizaki, Masayoshi Akimoto, Atsushi Hasebe, Atsuto Kubo, and Yoshiaki Fukazawa

Department of Computer Science and Engineering,
School of Fundamental Science and Engineering, Waseda University,
3-4-1 Okubo, Shinjuku-ku, Tokyo 1698555, Japan
washizaki@waseda.jp,
{aki12,a-hasebe,a.kubo}@fuka.info.waseda.ac.jp,
fukazawa@waseda.jp
http://www.washi.cs.waseda.ac.jp
http://www.fuka.info.waseda.ac.jp

Abstract. Among several diagrams defined in UML, the class diagram is particularly useful through entire software development process, from early domain analysis stages to later maintenance stages. However conventional UML environments are often inappropriate for collaborative modeling in physically remote locations, such as exchanging models on a public mailing list via email. To overcome this issue, we propose a new diagram notation, called "TCD" (Text-based uml Class Diagram), for describing UML class diagrams using ASCII text. Since text files can be easily created, modified and exchanged in anywhere by any computing platforms, TCD facilitates the collaborative modeling with a number of unspecified people. Moreover, we implemented model converters for converting in both directions between UML class diagrams described in the XMI form and those in the TCD form. By using the converters, the reusability of models can be significantly improved because many of UML modeling tools support the XMI for importing and exporting modeling data.

1 Introduction

Unified Modeling Language (UML[1]) is a standardized diagram-based modeling language in the field of software/system engineering, especially object-oriented software development. Among several diagram specifications defined in UML, the class diagram is particularly useful through entire development process, from early domain analysis stages to later deployment/maintenance stages. UML class diagrams are used for expressing the static structure of some targets, such as problem domains, systems, and software, by describing the internal structure (attributes and operations) of classes and the interrelationships (e.g. generalization and association) between each class.

The main description methods used for UML class diagrams are descriptions made using UML modeling tools and handwritten notes. Modeling tools such as astah*[2] and Rational Rose[3] enable intuitive and advanced model editing using GUI, and provide advanced functions such as validation of descriptions and source code generation. On the other hand, handwriting is a simple method of description regardless of the situation.

However, there are some situations in which the above two description methods are difficult to use. One such situation is collaborative modeling with a number of unspecified people in physically remote locations, such as exchanging models on a public mailing list via email. Figure 1 shows an example of an actual class diagram[1] presented at a mailing list[6]. This class diagram is expressed using characters like "+" and "-," and discussions are held by attaching or describing class diagrams in e-mail messages; however this diagram has been described in an ad-hoc manner.

```
+--------+        +-------+     +-------+
| Person |--------|  ATM  |<>---| State |
+--------+        +-------+     +-------+
     A                             A
     |                             |
     |                             |
     |                             |
  +-------+                +-------+-------+
  |       |                |       |       |
+----++--------+       +------++------++-------+
|Crew||Customer|       |Active|Stopped||Failed|
+----++--------+       +------++------++-------+
```

Fig. 1. Actual class diagram taken from a discussion mailing list[6]

To describe, share, edit and reuse class diagrams in the discussions held on mailing lists, a common format with a clear grammar is necessary for exchanging class diagrams to enable those to be interpreted exactly in the same way under any reader's environment. Although most of modeling tools can output images in common formats such as JPEG, the images cannot be imported or edited by another user's modeling tool.

Many modeling tools support the XML Metadata Interchange (XMI[4]) as a common specification based on text format for saving data. XMI is a standard specification for the exchange of models conforming to the MetaObject Facility specification (MOF[5]) in the form of XML documents. XMI data is described in the form of text, but it is described in a way that allows it to be understood easily by computers (i.e. machine-readable); it is not easy for humans to understand text in XMI (i.e. NOT human-readable), so users who do not have modeling tools cannot utilize such modeling data. Thus, XMI is not well suited to work environments where many people need to view and modify the data. Models

[1] Texts in the diagram are originally described in Japanese.

described in handwritten form are also inappropriate for collaborative modeling because it is difficult to deliver and share models in this form.

To overcome the above-mentioned problems in conventional description methods, formats and specifications, we propose a new diagram notation, called "TCD," for expressing UML class diagrams using ASCII text format. Moreover, we implemented model converters for converting in both directions between UML class diagrams described in the XMI form and those in the TCD form.

The remainder of this paper is organized as follows. Next section introduces the concept and feature of TCD. Section 3 describes our model converters. Section 4 describes several related works. In the last section, we draw a conclusion and state future works.

2 Text-Based UML Class Diagram

We defined TCD based on the following concepts to make it truly useful for above-mentioned collaborative modeling situations.

– Conformity to UML class diagram specification: TCD conforms to most of features defined in UML class diagram by using ASCII text format. In TCD, each class definition is described as an independent text element. The specification of the class description is defined as an extension of a conventional text-based notation called "U–Language"[7]; we have added several important features that are not supported in U–Language such as `static`/`final` members of a class, and an `abstract` class.
– Balancing machine-readable and human-readable: TCD supports the description of associations among classes by using mainly two characters: "`|`" and "`-`". Thus TCD can handle not only horizontal lines in association definitions, but also vertical lines, which enables association definitions to be description in a way that makes overall structure easy to understand.
– Built-in conversion between TCD and XMI formats: we developed tools for TCD to XMI data conversion, and for XMI to TCD conversion, to enable the migration between TCD and modeling tools. By using the tools, users can continue modeling activities started in a different format.

For example, the UML class diagram of the Abstract Factory design pattern[10] shown in Figure 2 can be described by the combination of the class definitions in the left side of Figure 3 and the association definitions in the right side of Figure 3 in TCD.

By using the example, details of TCD descriptions are explained in below.

(1) Class Definition

The class definitions describe the details of the classes that appear in the class diagram. For example, the left side of Figure 3 shows definitions of two classes: `AbstractFactory` and `ConcreteFactory1` taken from the Abstract Factory

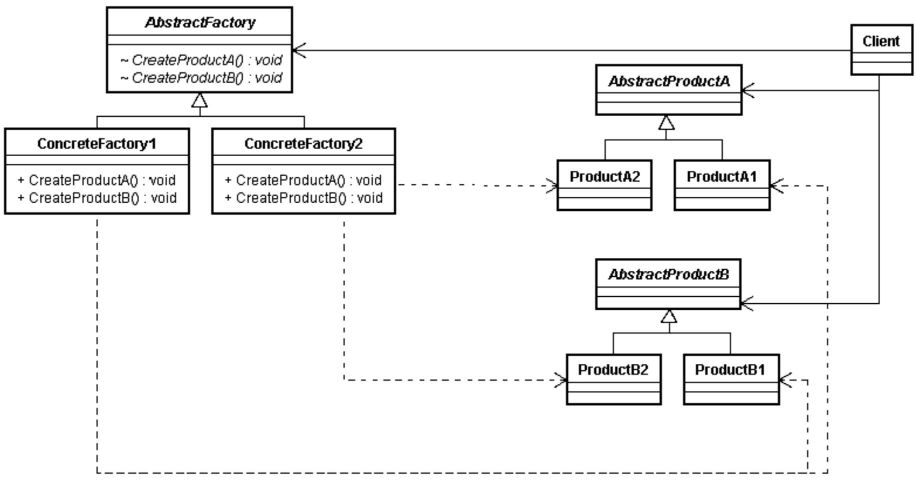

AbstractFactory
~ CreateProductA() : void
~ CreateProductB() : void
ConcreteFactory1
+ CreateProductA() : void
+ CreateProductB() : void
ConcreteFactory2
+ CreateProductA() : void
+ CreateProductB() : void
Client
AbstractProductA
ProductA2
ProductA1
AbstractProductB
ProductB2
ProductB1

Fig. 2. UML Class diagram describing the AbstractFactory pattern

pattern[10]. In the figure, two methods with precise signature definitions are specified for each class.

As shown in Figure 3, each class is specified by writing it between upper and lower boundary lines formed using "=" characters. Furthermore, two lines formed using "-" characters are inserted between the boundary lines, to create three compartments: class name, attributes and methods.

Since `AbstractFactory` is an abstract class, "`&`" is written before the class name. And since it does not have any attributes, nothing is written in the second compartment from the top. In the lowest compartment, two methods are specified. To declare the method, "`()`" is added after the method name. Return value type is specified by writing "`:`" after the method name and writing the type name.

(2) Association Definition

Associations are expressed by specifying the two classes that are associated with each other, and drawing lines with arrows to indicate the type of each association between the classes.

For example, the right side of Figure 3 shows 10 classes and 13 associations. Among them, the line emerging from the lower side of `AbstractFactory` is described using the characters "`^`", "`-`", and "`|`", indicating that the class generalizes the classes listed at the opposite end of the line; the next character is "`+`", and here the generalization line emerging from `AbstractFactory` branches. "`*`" and "`#`" indicate the association's rotation point and intersection point, respectively.

```
// Class definitions                 // Association definitions

============================         Client--->AbstractFactory
    & AbstractFactory                |             ^
-----------------------------        |             -
-----------------------------        |             |
 + CreateProductA() : void           |         *---+--------------*
 + CreateProductB() : void           |         |                  |
=============================        |       ConcreteFactory1   ConcreteFactory2
                                     |            ^                    ^
                                     |            :                    :
=============================        |            :                    :
    ConcreteFactory1                 |         *..+..............*    :
-----------------------------        |         :                  :    :
-----------------------------        |         :         *........#....+......*
 + CreateProductA() : void           |         :         :        :           :
 + CreateProductB() : void           |  ProductA1  ProductA2    ProductB1  ProductB2
=============================        |         |         |        |           |
                                     |         |         |        |           |
                                     |         -         -        -           -
=============================        |         v         v        v           v
    ConcreteFactory2                 |       AbstractProductA   AbstractProductB
-----------------------------        |         ^                  ^
-----------------------------        |         |                  |
 + CreateProductA() : void           *---------+------------------*
 + CreateProductB() : void
=============================
```

Fig. 3. TCD description of the AbstractFactory pattern (left side: classes, right side: associations)

Moreover, users can describe associations by simply listing up each association such as `AbstractFactory <--- Client` and `AbstractFactory <|--- ConcreteFactory1`. TCD enables users to select whether to emphasize ease of description (like the above-mentioned listing up) or ease of understanding (like Figure 3).

Multiplicities and roles (association ends) regarding associations can be specified for those described horizontally by using brackets ("`( multiplicity )`") and square brackets ("`[ role ]`") for each end. For example, `ClassA (1) ---> (*) ClassB` denotes one-to-many relationship from `ClassA` to `ClassB`.

3 Model Converters

We developed tools for TCD to XMI data conversion, and for XMI to TCD conversion, to enable the migration between TCD and modeling tools. By using the converters, users can continue modeling activities started in a different format.

We implemented converters by using Java and JavaCC[11], shown in Figure 4. Since these converters can be used on any system that can run Java programs, it can be used very widely. Figure 2 shows an UML class diagram by inputting

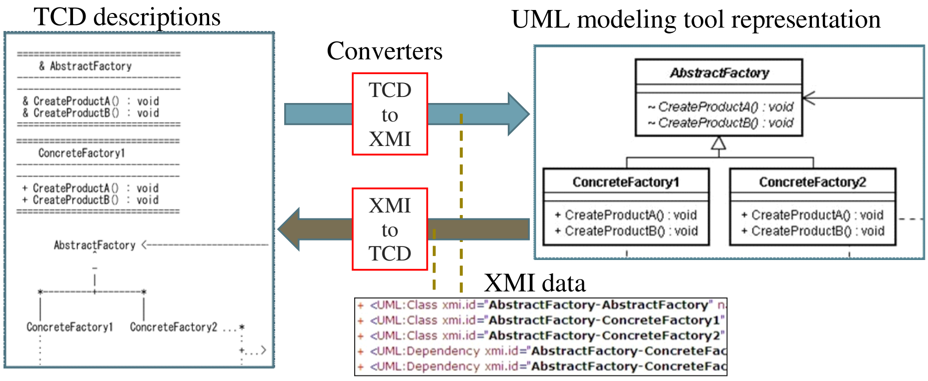

TCD descriptions
& AbstractFactory
& CreateProductA() : void
& CreateProductB() : void
ConcreteFactory1
+ CreateProductA() : void
+ CreateProductB() : void
AbstractFactory
ConcreteFactory1
ConcreteFactory2
Converters
TCD to XMI
XMI to TCD
UML modeling tool representation
AbstractFactory
~ CreateProductA() : void
~ CreateProductB() : void
ConcreteFactory1
+ CreateProductA() : void
+ CreateProductB() : void
ConcreteFactory2
+ CreateProductA() : void
+ CreateProductB() : void
XMI data
+ <UML:Class xmi.id="AbstractFactory-AbstractFactory"
+ <UML:Class xmi.id="AbstractFactory-ConcreteFactory1"
+ <UML:Class xmi.id="AbstractFactory-ConcreteFactory2"
+ <UML:Dependency xmi.id="AbstractFactory-ConcreteFac
+ <UML:Dependency xmi.id="AbstractFactory-ConcreteFac

Fig. 4. TCD converters and related formats

TCD descriptions in Figure 3 into the converter, and importing the XMI output of the converter into a modeling tool `astah*`.

It is found that all of contents described in TCD are kept in standard UML notation. The converters allow for easy collaboration between users of TCD and modeling tools. For example, these converters allow work on a model that uses class diagrams initially written in TCD to be continued using a modeling tool. They also enable conversion of class diagrams created by a modeling tool into TCD format for exchange in email-based discussions.

4 Related Work

Although not in mainstream use, there are several methods for describing UML class diagrams based on text, such as U–Language[7] and Silvertejp[9]. These methods enable user write class diagrams into text forms such as e-mail and Wikis, where conventional description methods are difficult to apply.

However, existing description formats are equipped with on-way converters to enable reutilization of described models; these converters are only capable of converting from each text description format to other formats, or from other formats to the text description form. For example, there is a converter that outputs Java source code from descriptions in U–Language[8]; however there is no support for converting Java source code into U–Language descriptions. Furthermore, there is no compatibility between different description formats and it is also difficult to make use of other converters, so the reusability of models is low.

In addition, existing formats for text description-based class diagrams tend to feature either very good ease of description or very good ease of understanding (not both); it is hard to use one format adaptively for different situations such as a case in which ease of description is most important or another case in which ease of understanding is most important.

5 Conclusion and Future Work

We formulated a new kind of text description-based method to express UML class diagrams – Text-based Class Diagram (TCD) – to overcome the problems with conventional description methods, and we developed converters to convert between TCD and XMI formats.

TCD conforms in part to the description specifications of the existing description formats that offer good ease of description, and enables the expression of vertical associations, which allows for layouts that are easy to understand. These features enable users to select between description that emphasizes ease of description and description that emphasizes ease of understanding, according to the application. In addition, the converters make collaboration with users of modeling tools easy and highly reliable, thereby improving model reusability.

As our future work, we have a plan to conduct real experiments for confirming the usefulness of TCD and its converters compared with conventional environments. Moreover, since many large-scale class diagrams make use of package-related notation, it is necessary to support package-related notation in order to expand the range of class diagrams that can be handled using TCD.

Acknowledgements

This research has been partially supported by the Mizuho Foundation for the Promotion of Sciences (2009-2010) and the JGC-S SCHOLARSHIP FOUNDATION (2010-2011).

References

1. Object Management Group: Unified Modeling Language (UML), http://www.uml.org
2. Change Vision, Inc.: astah* - UML and Mind Mapping Integrated Modeling Tool, http://astah.change-vision.com/en/
3. IBM: Rational Rose, http://www-306.ibm.com/software/awdtools/developer/rose/
4. Object Management Group: MOF 2.0/XMI Mapping Specification, http://www.omg.org/technology/documents/formal/xmi.htm
5. Object Management Group: MetaObject Facility, http://www.omg.org/mof/
6. Ogis-RI: Object Square, http://www.ogis-ri.co.jp/otc/otc2/oosquare-ml/
7. Hiranabe, K.: U–Language – Human and machine readable UML text format, ObjectClub (in Japanese), http://www.objectclub.jp/technicaldoc/uml/u_lang/
8. Hexagonta: U Language Parser, http://sourceforge.jp/projects/ulparser/
9. Wettin, K.: Silvertejp, http://silvertejp.tigris.org/
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1994)
11. Viswanadha, S.: Java Compiler Compiler (JavaCC), https://javacc.dev.java.net/

SQL-Based Compound Object Comparators: A Case Study of Images Stored in ICE

Dominik Ślęzak[1,2] and Łukasz Sosnowski[3,4]

[1] Institute of Mathematics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
[2] Infobright Inc.
Krzywickiego 34 lok. 219, 02-078 Warsaw, Poland
[3] Systems Research Institute, Polish Academy of Sciences
Newelska 6, 01-447 Warsaw, Poland
[4] Dituel Sp. z o.o.
Wąwolnicka 4 lok. 22, 04-023 Warsaw, Poland
`slezak@infobright.com, l.sosnowski@dituel.pl`

Abstract. We introduce the framework for storing and comparing compound objects. The implemented system is based on the RDBMS model, which – unlike other approaches in this area – enables to access the most detailed data about considered objects. It also contains ROLAP cubes designed for specific object classes and appropriately abstracted modules that compute object similarities, referred as comparators. In this paper, we focus on the case study related to images. We show specific examples of fuzzy logic comparators, together with their corresponding SQL statements executed at the level of pixels. We examine several open source database engines by means of their capabilities of storing and querying large amounts of such represented image data. We conclude that the performance of some of them is comparable to standard techniques of image storage and processing, with far better flexibility in defining new similarity criteria and analyzing larger image collections.

Keywords: Compound Objects, Similarity, Comparators, Image Analysis, Fuzzy Logic, RDBMS Engines, Infobright Community Edition (ICE).

1 Introduction

There is a growing demand for systems that can retrieve compound objects based on their mutual similarities, membership to certain groups, or satisfaction of some criteria. Such systems need to identify objects quickly and accurately, based on their comparison against some patterns or exclusion according to some forbidden features. In some applications, such systems work mainly with various types of objects' indexes and metadata. In other applications, they may also involve the objects' storage, which raises additional challenges but, on the other hand, opens new possibilities for the retrieval process improvements.

In this paper, we outline a framework that is able to retrieve objects basing on similarities, including their classification and interpretation. Similarities can be defined by a number of criteria and measures, varying with respect to the object classes and application types [13,18]. However, from the system's architecture standpoint, their implementation can be kept within a universal structure, referred here as a comparator. Actually, comparators occur in the literature in various contexts, usually for the purposes of image analysis and processing [4,10]. In our approach, however, the comparator is an abstracted module responsible for comparing collections of input objects (not necessarily images) and reporting outputs in the form of, e.g., parameters of the most similar objects. The comparator may be thus regarded as analogous to a mathematical function, whose values can be applied at further stages of analysis of objects.

An object is an entity that we want to measure, describe, classify or compare with other objects. It may correspond to a physical phenomenon, situation, state, process, signal, etc. It may have some features that are useful in classification or similarity analysis. Our understanding of an object is close to the concept of entity in relational databases or an object in information retrieval [7,15]. In this paper, we refer to objects at a possibly abstracted level, although it is useful to distinguish some specific classes of objects, such as images, texts or sequences. Variety of possible object classes is not in contradiction with universality of the proposed system's architecture. It is important because analogous solutions usually focus on implementation of algorithms dedicated to some particular types of objects that are not so easily transferrable to other cases.

The systems aimed at compound object retrieval usually assume a sharp distinction between the layers of storage and analysis. This means that the analytical algorithms have direct access only to precalculated features, often stored within an RDBMS framework, while the objects themselves are encoded as BLOBs or stored in an independent repository [5,9]. Initial phases of processing and segmentation in the image retrieval systems may be coupled with computation of the values of a pre-defined set of attributes based on histogram computation, edge detection, shape recognition, texture analysis, etc. Such an approach is quite convenient and, actually, it satisfies our above-formulated universality assumptions. However, it does not provide opportunity to efficiently refine and extend the set of features while applying new algorithms, as there is no direct interface to quickly manipulate the detailed data.

We propose an alternative approach to representing compound objects in relational data model. Namely, we put decoded information about objects into ROLAP cubes. For example, we suggest basing the cubes for images on the data table, where each pixel of each image is represented as a separate row. This way, the whole system takes the form of an integrated data warehouse. It enables a convenient access to arbitrary fragments of objects or collections of objects, as well as their analysis using standard database operations. Furthermore, it guarantees easiness of completing or modifying object descriptions in an arbitrary moment, not only at the stage of supplying objects to the system. Surely, the cubes and the underlying data need to be designed separately for different object classes. On the other hand, the usage of ROLAP operations by other system's components can be similar for all types of compound objects.

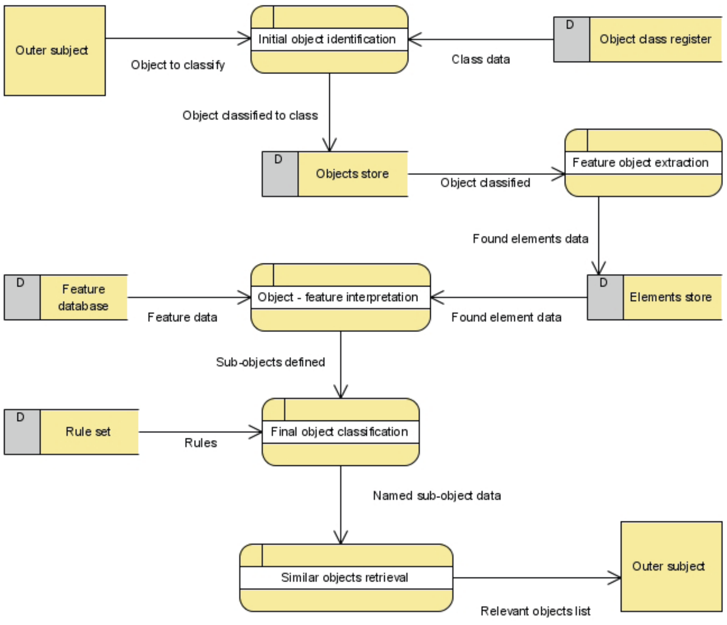


Fig. 1. DFD for compound object comparison

The remainder of the paper is organized as follows: In Section 2, we describe the proposed architecture with a special emphasis on the comparator aspects. In Section 3, we discuss the database framework aspects – compound object representation, the choice of appropriate technology and the underlying data schemas. In Section 4, we introduce a more specific case study of image comparator based on histogram analysis and fuzzy logic. In Section 5, we run some performance tests and analyze their results. In Section 6, we conclude the paper.

The presented approach is a continuation of our research in [19,20], extended here by a more complete study of the RDBMS layer. Thanks to pixel-based image representation, histogram computations and image comparisons, such as those in Sections 3 and 4, can be conducted using standard SQL syntax. Certainly, one might look at the idea of representing compound objects in such a detailed way as unrealistic, given the expected size of, e.g., large image databases. However, the results reported in Section 5 prove that modern analytic database engines provide fully satisfactory data compression and query performance.

2 Algorithmic Outline

In order to clearly present the proposed framework for compound object comparisons at a possibly universal level, we carefully follow the structural design and modeling standards [2,14]. Figure 1 presents a general Data Flow Diagram (DFD) for our solution. Table 1 provides more detailed information.

Table 1. Detailed description of components illustrated in Figure 1

Component name	Description
Outer subject	A user or a system that initializes and triggers identification/classification of an object by its comparison to the existing reference stores
Object class register	Detailed information about the types of object classes
Initial object identification	Recognizing the class of an object (e.g.: image, video, sound, text) in order to choose an appropriate set of comparison features and rules
Object store	Temporary storage of the investigated object
Feature object extraction	Using feature extraction techniques available for a given object class (e.g.: for images, it includes the edge detection, the histogram extraction, etc.)
Elements store	Temporary storage of the extracted elements (elements do not have a status of features yet)
Feature database	Types of features – specialized functions used to measure their values (e.g.: red color histogram)
Object-feature interpretation	Interpreting membership of the investigated object to the sets of elements with given major features (it can be conducted by using, e.g., fuzzy rule sets)
Rule set	Rules defining final classification of objects
Final object classification	Analyzing similarities between the identified elements of the investigated object and features in database; it enables to reject fake features and better interpret the remaining ones; final classification is designed to be conducted by an ensemble of comparators
Similar objects retrieval	Finding objects that are most similar to the investigated one, basing on its classification

Figure 2 presents the main stages of comparator's work. Comparing to Figure 1, this is a more atomic level of the proposed solution. The algorithm verifies similarity of an investigated object to the reference objects, including their elements too. There is also a mechanism of similarity exception handling. For each reference object b, we register its so called forbidden features. If the investigated object a turns out to have one of such features, then b cannot be reported as similar to a. Identification of forbidden features is based on the fuzzy classifier [6,12]. This way, we can take an advantage of domain knowledge and we obtain a convenient framework for the parameter tuning.

If b is not forbidden with respect to a, we compute the degree of similarity of a to b. As in [19,20], we use fuzzy logic apparatus also at this stage. We compute fuzzy membership of (a, b) to the similarity relation defined on compound objects. Definition of membership can be adjusted to reflect a general similarity within a given object class or, e.g., similarity of some specific aspects of objects. Membership can be represented as a function $\mu : R \times R \rightarrow [0, 1]$, where 0 and 1 mean total dissimilarity and similarity, respectively.

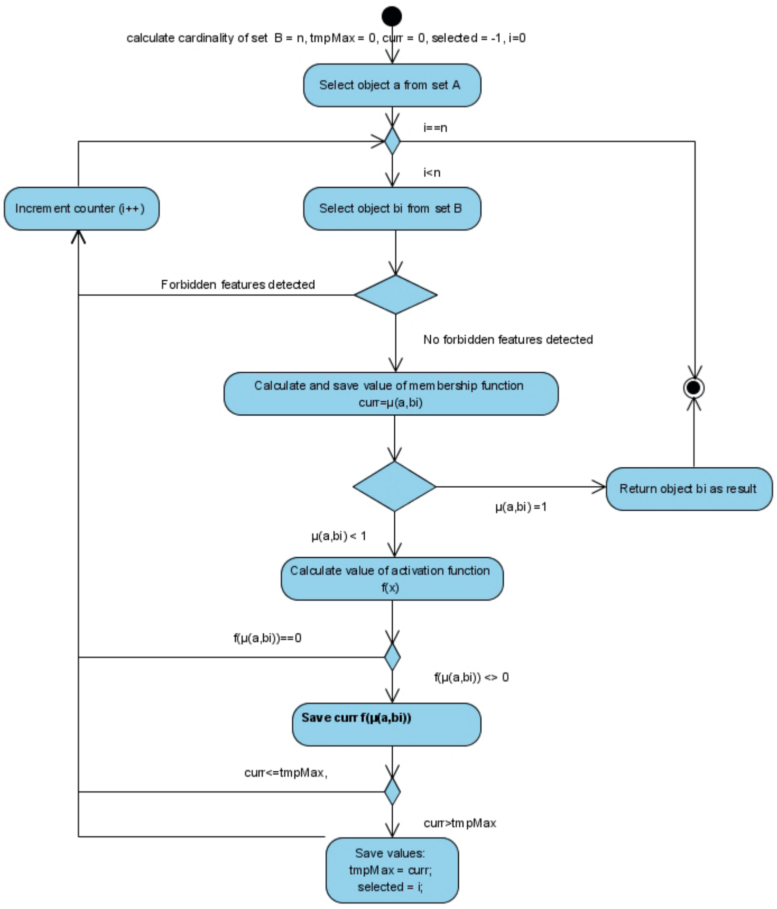

calculate cardinality of set B = n, tmpMax = 0, curr = 0, selected = -1, i=0
Select object a from set A
i==n
i<n
Increment counter (i++)
Select object bi from set B
Forbidden features detected
No forbidden features detected
Calculate and save value of membership function curr=μ(a,bi)
Return object bi as result
μ(a,bi) =1
μ(a,bi) < 1
Calculate value of activation function f(x)
f(μ(a,bi))==0
f(μ(a,bi)) <> 0
Save curr f(μ(a,bi))
curr<=tmpMax,
curr>tmpMax
Save values: tmpMax = curr; selected = i;

Fig. 2. Activity diagram of comparator

The degree of similarity can be further treated as an input to an activation function $f : [0, 1] \to \{0, 1\}$, which assigns 1 to values greater than a threshold $p \in [0, 1]$ and 0 otherwise. One may adjust p according to the expert knowledge or, e.g., as a result of evolutionary optimization process. Remember that various ways of computing function μ are just approximations of an actual notion of similarity between compound objects. Thus, one may search for p as the lowest possible threshold yielding results expected by the users.

In the last phase, we check whether the value of μ is higher than those computed so far. If it is, we memorize the corresponding reference object. Finally, we can get: a) no output (because of forbidden features or not exceeding threshold p), b) exactly one output, c) multiple outputs (if there are multiple objects b with the maximum value of $\mu(a, b)$). If we are interested only in a single output, we can stop after finding the first b satisfying $\mu(a, b) > p$.

3 RDBMS Framework

3.1 Compound Object Representation

The proposed design is based on defining different classes of compound objects that are equipped with comparable functionality. Usually, objects are available in various specific formats that are useful for storage but inconvenient for more advanced processing, when direct access to decoded data is required. In our solution, we want to enable the users to extract information from objects on ongoing basis, in order to provide better continuity of gathered knowledge, including methods of its gathering themselves. Knowledge and rule bases related to object classes should be allowed for evolving along with the users' needs. Hence, we decided to represent and store compound objects in their fully decoded form, in a relational database. Let us emphasize that we do not use such data types as BLOB, IMAGE, BYTE, as they do not address the above-mentioned issues. Instead, we operate with data schemas at the semantically richer level of atomic components of the compound objects.

For example, for collections of images, we suggest constructing data tables with rows corresponding to pixels. Each pixel is assigned with its coordinates within an image, as well as with its image identifier. Certainly, this means that even for low-resolution images the corresponding data table will grow very fast. In order to address this potential issue, we need to carefully select an underlying relational database engine technology.

3.2 Infobright Community Edition (ICE)

ICE[1] is an open source RDBMS engine integrated with MySQL. It may be applied in data warehouse or analytical database scenarios [16,17]. From the user perspective, it provides standard means for relational data modeling and querying. Internally, it decomposes data both vertically and horizontally, onto so called data packs, each of them storing values of up to 2^{16} rows for one of attributes. Data packs are compressed separately from each other. During load, besides compression, the content of each data pack is automatically annotated with its basic statistics that are stored in so called database knowledge grid. The acquired statistics are used in various ways in query execution, with the main goal of minimizing and optimizing an access to data packs.

ICE provides high compression ratio (reported as 10:1 on average) and high speed of analytical SQL statements, for which the gathered statistics are especially useful. ICE can easily handle tens of terabytes of data on a single PC machine. Also, it does not require maintenance of any additional database indexes, on top of database knowledge grid which is relatively small (reported as 1% of the compressed data size on average) and generated transparently to the users. Thus, given the challenges outlined in Subsection 3.1, we regard ICE as potentially applicable as the RDBMS layer of our solution.

[1] `en.wikipedia.org/wiki/Infobright`

3.3 Data Layout

Our solution is split onto two major parts: 1) OLTP (transactional layer) and 2) ROLAP (cubes dedicated to store information about objects). The OLTP part contains objects' metadata, their basic features, class membership, etc. It simplifies object management but has no direct impact on analytical capabilities. The ROLAP part resembles a data warehouse model. Cubes are created for specific data within object classes. Cubes may have partially common dimensions. This leads towards constellation schemas [1,7], which are convenient for more advanced analytics. Furthermore, we assume that cubes can be built for decoded objects, their specific elements or fragments, as well as for various types of pre-aggregates or statistics. Our solution enables to create such cubes in an arbitrary moment, depending on the users' requirements.

3.4 Relationships

Let us focus on simple examples of the above-mentioned components. The OLTP layer is displayed in Figure 3, by means of a standard Entity-Relationship Diagram (ERD). Table 2 describes particular entities. The data is stored in the third normal form in order to assure easiness of extensions and modifications. In the ERD diagrams, we use typical notation for foreign keys (FK) in order to emphasize joins that we expect to occur in SQL statements. However, there are no constraints / indexes assumed to be maintained.

In the ROLAP layer, each object class has its own star schema. Let us concentrate on the example of images, as illustrated by Figure 4 and Table 3. One can see that pixels are stored as rows in the fact table. Dimensions refer to coordinates X and Y, as well as to object identifiers. Such representation enables to express a number of image processing operations in SQL. For other useful types of operations we design specific stored procedures.

Figure 5 and Table 4 correspond to image histograms [3,11]. Histogram-based comparators can be quite efficient, as reported in Section 5. Histogram cube provides information about histograms of objects or their fragments. It can be automatically appended for each image being loaded into the database or computed at once by using, e.g., the following:

```
SELECT FK_OBJECTS_ID, 1 AS CHANNEL, RED AS
BRIGHTNESS, 1 AS IMG_AREA, COUNT(*) AS VALUE FROM
FACT_IMAGES GROUP BY FK_OBJECTS_ID, RED UNION ALL
SELECT FK_OBJECTS_ID, 2 AS CHANNEL, GREEN AS
BRIGHTNESS, 1 AS IMG_AREA, COUNT(*) AS VALUE FROM
FACT_IMAGES GROUP BY FK_OBJECTS_ID, GREEN UNION ALL
SELECT FK_OBJECTS_ID, 3 AS CHANNEL, BLUE AS
BRIGHTNESS, 1 AS IMG_AREA, COUNT(*) AS VALUE FROM
FACT_IMAGES GROUP BY FK_OBJECTS_ID, BLUE UNION ALL
SELECT FK_OBJECTS_ID, 4 AS CHANNEL, ALPHA AS
BRIGHTNESS, 1 AS IMG_AREA, COUNT(*) AS VALUE FROM
FACT_IMAGES GROUP BY FK_OBJECTS_ID, ALPHA
```

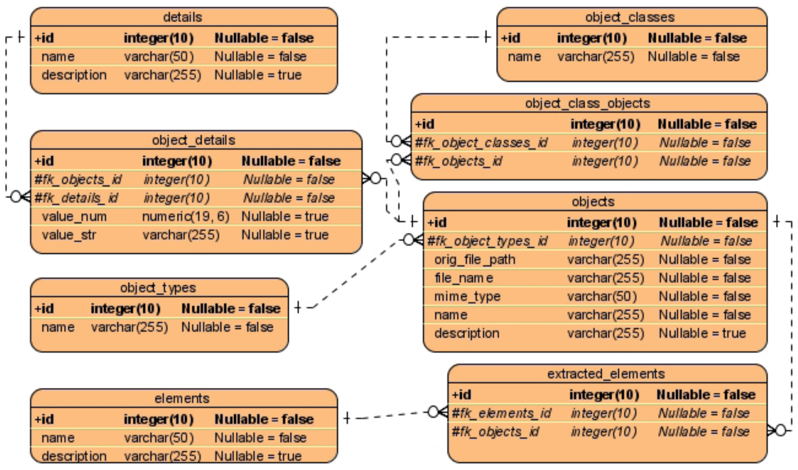

details
+id integer(10) Nullable = false
name varchar(50) Nullable = false
description varchar(255) Nullable = true
object_classes
+id integer(10) Nullable = false
name varchar(255) Nullable = false
object_class_objects
+id integer(10) Nullable = false
#fk_object_classes_id integer(10) Nullable = false
#fk_objects_id integer(10) Nullable = false
object_details
+id integer(10) Nullable = false
#fk_objects_id integer(10) Nullable = false
#fk_details_id integer(10) Nullable = false
value_num numeric(19, 6) Nullable = true
value_str varchar(255) Nullable = true
objects
+id integer(10) Nullable = false
#fk_object_types_id integer(10) Nullable = false
orig_file_path varchar(255) Nullable = false
file_name varchar(255) Nullable = false
mime_type varchar(50) Nullable = false
name varchar(255) Nullable = false
description varchar(255) Nullable = true
object_types
+id integer(10) Nullable = false
name varchar(255) Nullable = false
extracted_elements
+id integer(10) Nullable = false
#fk_elements_id integer(10) Nullable = false
#fk_objects_id integer(10) Nullable = false
elements
+id integer(10) Nullable = false
name varchar(50) Nullable = false
description varchar(255) Nullable = true

Fig. 3. ERD for OLTP layer

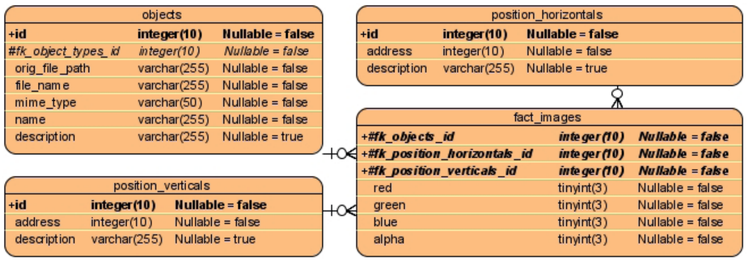

objects
+id integer(10) Nullable = false
#fk_object_types_id integer(10) Nullable = false
orig_file_path varchar(255) Nullable = false
file_name varchar(255) Nullable = false
mime_type varchar(50) Nullable = false
name varchar(255) Nullable = false
description varchar(255) Nullable = true
position_horizontals
+id integer(10) Nullable = false
address integer(10) Nullable = false
description varchar(255) Nullable = true
fact_images
+#fk_objects_id integer(10) Nullable = false
+#fk_position_horizontals_id integer(10) Nullable = false
+#fk_position_verticals_id integer(10) Nullable = false
red tinyint(3) Nullable = false
green tinyint(3) Nullable = false
blue tinyint(3) Nullable = false
alpha tinyint(3) Nullable = false
position_verticals
+id integer(10) Nullable = false
address integer(10) Nullable = false
description varchar(255) Nullable = true

Fig. 4. ERD for image data

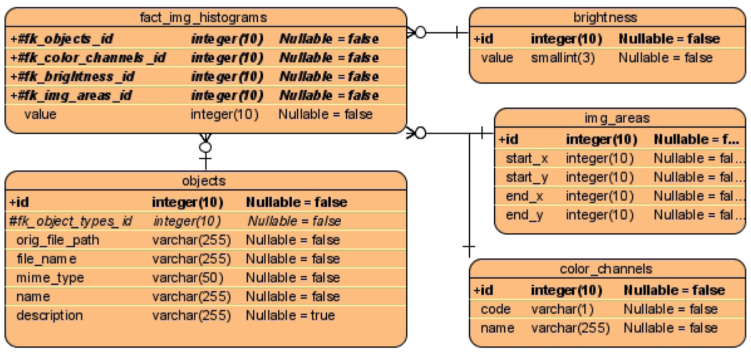

fact_img_histograms
+#fk_objects_id integer(10) Nullable = false
+#fk_color_channels_id integer(10) Nullable = false
+#fk_brightness_id integer(10) Nullable = false
+#fk_img_areas_id integer(10) Nullable = false
value integer(10) Nullable = false
brightness
+id integer(10) Nullable = false
value smallint(3) Nullable = false
img_areas
+id integer(10) Nullable = f...
start_x integer(10) Nullable = fal...
start_y integer(10) Nullable = fal...
end_x integer(10) Nullable = fal...
end_y integer(10) Nullable = fal...
objects
+id integer(10) Nullable = false
#fk_object_types_id integer(10) Nullable = false
orig_file_path varchar(255) Nullable = false
file_name varchar(255) Nullable = false
mime_type varchar(50) Nullable = false
name varchar(255) Nullable = false
description varchar(255) Nullable = true
color_channels
+id integer(10) Nullable = false
code varchar(1) Nullable = false
name varchar(255) Nullable = false

Fig. 5. ERD for image histograms

Table 2. Description of entities visible in Figure 3

Entity name	Description
object_classes	Related to the meaning or context, such as people, architecture, holiday photos etc.; independent from categories of types or formats, such as text, image etc.
elements	Dictionary of elements/features extracted from objects
object_class_objects	Assignment of objects to (possibly multiple) classes
objects	It includes information about objects' types; on the other hand, it serves as dimension for ROLAP cubes
object_types	Dictionary of object types
extracted_elements	Assignment of elements to objects
object_details	Features that describe objects' details
details	Dictionary of possible objects' details

Table 3. Description of entities visible in Figure 4

Entity name	Description
position_horizontals	Horizontal coordinates of pixels in their images
position_verticals	Vertical coordinates of pixels in their images
objects	Refers to "objects" in Table 2
fact_images	Fact table – rows correspond to images' pixels

Table 4. Description of entities visible in Figure 5

Entity name	Description
color_channels	Channel dimension (red R, green G, blue B, alpha A)
img_areas	Dimension defining areas that the histogram refers to (the whole image or, e.g., an object visible at an image)
brightness	Dimension defining pixels' brightness levels
objects	Refers to "objects" in Table 2
fact_img_histograms	Fact table for histogram-based ROLAP cube

The histogram cube contains up to 256×4 rows per image (256 brightness levels, 4 channels). Histograms can be further quantized by rounding their values to multiples of n. For instance, for $n = 10$, quantization can look as follows:

```
SELECT FIH.FK_OBJECTS_ID, CC.CODE, B.VALUE - (B.VALUE mod 10),
SUM(FIH.VALUE) FROM FACT_IMG_HISTOGRAMS FIH INNER JOIN BRIGHTNESS
B ON FIH.FK_BRIGHTNESS_ID = B.ID INNER JOIN COLOR_CHANNELS CC
ON FIH.FK_COLOR_CHANNELS_ID = CC.ID WHERE FIH.FK_IMG_AREAS_ID = 1
GROUP BY FIH.FK_OBJECTS_ID, CC.CODE, B.VALUE - (B.VALUE mod 10)
```

Quantization simplifies further steps of image comparison. If the histogram cube is already in place, the above SQL runs in milliseconds. On the other hand, we noticed that $n = 10$ does not lead to a significant decrease of accuracy.

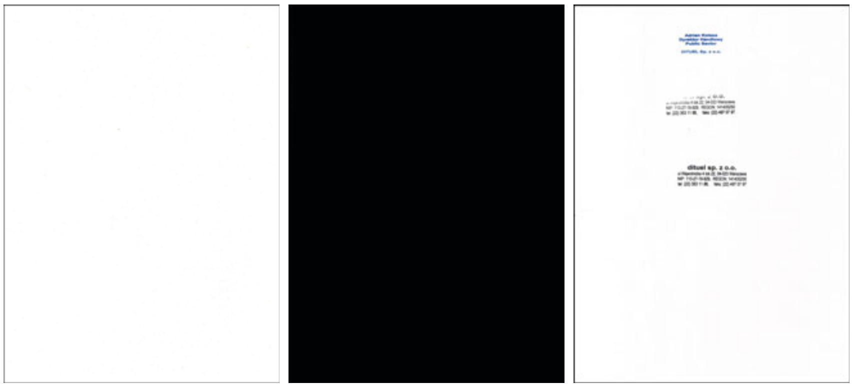


Fig. 6. Images #1, #2, #3

4 Case Study

Consider images #1,#2,#3 presented in Figure 6. They have resolution 1024×1408, which means 1441792 rows per image in the database. Table 5 shows quantization results for $n = 10$. Each image yields at most 26 rows. Images #1 and #2 are almost white and black, respectively. This explains why the first row for #1 and the last row for #2 have both very large values.

Consider comparators K_R, K_G and K_B corresponding to the color channels Red, Green and Blue, respectively.[2] Each K computes similarity of an investigated image to a reference image. Let us use the following function:

$$\mu(a, b) = 1 - \frac{\sum_{j=1}^{26} |a[j] - b[j]|}{2} \qquad (1)$$

where $a[j]$ ($b[j]$) denotes the j-th index of the quantized and normalized histogram vector for the investigated (reference) image. The values of $\mu(a, b)$ are then treated as degrees of membership in fuzzy rules applied to final interpretation of similarity. We use conjunctions of memberships related to RGB.

Assume that image #3 is the investigated object. Let us compare it with the reference images #1 and #2 by following the approach outlined in Section 2. For simplicity, assume that there are no forbidden features. Set up $p = 0.999999$. By putting information from Table 5 into equation (1), for K_R, we obtain:

$$\mu(\#3, \#1) = 0.980630 \quad \mu(\#3, \#2) = 0.000006$$

Although image #3 is far more similar to #1 than to #2, the output of the algorithm displayed in Figure 2 is empty, because the result of K_R still needs to be combined within a conjunction with those of K_G and K_B, with no chance to exceed the threshold of 0.999999. One more time, we refer to [19,20] for details on how to adjust parameters of the proposed comparator methodology.

[2] Alpha can be omitted for jpg files. We plan to analyze other formats in the future.

Table 5. Quantized histograms for images #1 (top left), #2 (middle left) and #3

#1	R	G	B
40	1	1	1
70	1	1	1
160	3	3	3
190	0	0	2
200	2	2	5
210	5	5	9
220	14	16	14
230	25	26	34
240	332	331	387
250	1441409	1441407	1441336

#2	R	G	B
0	1441264	1441263	1441260
10	524	524	520
20	2	3	10
40	1	1	1
50	1	1	1

#3	R	G	B
10	4	1	1
20	34	1	1
30	337	39	24
40	1006	251	166
50	1497	851	610
60	1800	1456	1208
70	1835	1804	1680
80	1292	1727	1344
90	934	1577	864
100	767	1114	655
110	702	829	625
120	699	746	567
130	705	753	639
140	606	627	714
150	672	650	865
160	664	643	1050
170	678	694	1148
180	765	775	1108
190	834	837	1018
200	800	777	911
210	972	948	925
220	1180	1097	1128
230	1772	1392	1394
240	7756	6060	7805
250	1413481	1416143	1415342

5 Performance Tests

We report the results obtained on a standard laptop, Intel Core Duo T9600, 8GB RAM, 500GB 5400 RPM, Windows 7 64 bit.

The first part of our tests refers to images in Figure 6. Table 6 shows the size of their jpg files and the size of their corresponding sets of pixels stored in ICE. Compression ratios in ICE are worse than in case of dedicated jpg format but the difference is less than one might expect, especially for compression algorithms that are by definition lossless and lossy, respectively.

The speed of decoding a jpg file into a csv file and then loading it into ICE is on average 3,702 milliseconds. It may be improved in the future by avoiding creation of intermediate files. We repeated the tests twice: 1) for initially empty database and 2) for database containing initially around 300,000,000 rows, which corresponds to around 2,000 images of the considered resolution. The ICE load speed was approximately the same in both scenarios.

The last two columns in Table 6 reflect the speed of creating and quantizing histograms in Java [11,19] and ICE (using SQL similar to those in Subsection 3.4 but referring to each image separately). ICE speed is reported for 300,000,000$^+$-row data. In case of Java, it means finding a required jpg among 2,000 of other files and opening it to extract a quantized histogram.

Table 6. Image sizes (3 cases) and histogram creation + quantization speed (2 cases)

#	Size of jpg	Size of csv	Size in ICE	Java Speed	ICE Speed
1	13 KB	38,196 KB	91 KB	201 ms	1,035 ms
2	10 KB	29,750 KB	34 KB	214 ms	850 ms
3	78 KB	38,174 KB	770 KB	200 ms	1,170 ms

Table 7. Comparison of RDBMS engines for 3 × 1441792-row data. Results averaged over images #1,#2,#3. Last column refers to histogram creation + quantization speed.

	Load Speed	Database Size	Execution Speed
ICE 3.3	3,702 ms	298 KB	910 ms
MySQL 5.0	2,803 ms	40,832 KB	7,732 ms
PostgreSQL 8.4	17,803 ms	83,616 KB	20,755 ms

Table 8. Finding 10 out of 100/500/1,000 images. – Example of the search criterion.

Amount of Images	Search Speed in Java	Search Speed in ICE
100	19,310 ms	101 ms
500	89,349 ms	127 ms
1,000	181,474 ms	142 ms

Table 7 shows why we recommend ICE. Here, we consider only three images, as we had problems with 300,000,000 rows in other engines. Recall that ICE does not need indexes, as they are replaced by much lighter statistics that are, actually, very helpful when executing the discussed queries. In case of MySQL[3] and PostgreSQL[4], for better performance, one might use indexes. However, they would cause further increase of size and maintenance effort.

Going back to Table 6, one may claim that a standard approach is still faster than the RDBMS-based methodology. However, it turns out quite opposite in case of typical search processes – the subject of the second part of our tests. Table 8 illustrates the performance of our overall solution when the task is to find 10 out of 100, 500, or 1000 images that match to the highest degree some pre-defined conditions. Precisely, we search for images with a large number of pixels with the brightness = 250 for the blue channel, that is:

```
SELECT FK_OBJECTS_ID FROM FACT_IMG_HISTOGRAMS WHERE
FK_COLOR_CHANNELS_ID = 3 AND FK_BRIGHTNESS_ID = 250
ORDER BY VALUE DESC LIMIT 10
```

Surely, it is not as complicated as queries that we may encounter for advanced comparators. However, it illustrates how to use SQL efficiently.

[3] `dev.mysql.com/doc/refman/5.0/en/index.html`
[4] `www.postgresql.org/docs/8.4/static/index.html`

6 Conclusions and Discussion

We proposed a novel approach to storing and analyzing compound objects, where the object processing and comparison algorithms are able to run over complete, atomic data in a relational database model. We discussed high-level ideas, as well as technological details such as, e.g., the choice of Infobright Community Edition (ICE) as the underlying RDBMS solution. Comparing to our previous research [19,20], we focused on overall performance, paying less attention to analytical accuracy. The main goal of this paper was to provide a data representation framework that is flexible enough to tune advanced algorithms working efficiently with large collections of objects.

The results presented in Section 5 may be insufficient to fully convince everyone that our approach is worth considering. For example, given Tables 6 and 8, one might take an advantage of both standard and RDBMS-based solution by keeping images in their specific format and, as a complement, storing their histogram information in a relational data schema. However, as discussed in the earlier sections, the major reason for storing data about compound objects in the decoded form is to achieve better flexibility in feature extraction and comparison strategies. The already-mentioned results are supposed to prove that our solution should not be disqualified because of unacceptable compression or execution of the most typical operations. On the other hand, it clearly leads towards functionality that is beyond other methods.

In the nearest future, we will investigate opportunities that our framework provides at the level of processing multiple compound objects. Consider an example of images. Practically all the existing technologies assume that images are processed separately. Surely, it enables to parallelize the most time-consuming index and feature extraction operations. However, all further stages of, e.g., image comparisons need to be based on the extracted information instead of complete data. Among applications that may suffer from such limitation, one may look at, e.g., video analysis or 3D MRI brain segmentation [3,8]. More generally, the data layout proposed in Section 3 enables to run arbitrary SQL statements over atomic data related to an arbitrary subset of objects.

The knowledge about compound objects may be also employed to improve the database engine efficiency and functionality. In this paper, we use compound object hierarchies explicitly, at the data schema level. Alternatively, such knowledge can be expressed internally, transparently to the users and modules communicating with a database via SQL. Both strategies should be taken into account depending on practical needs. They may lead to better domain-specific compression, domain-specific statistics and also new ways of understanding SQL (see e.g. [6,16] for further discussion and references).

Acknowledgment. The first author was partially supported by the grants N N516 368334 and N N516 077837 from the Ministry of Science and Higher Education of the Republic of Poland.

References

1. Agosta, L.: The Essential Guide to Data Warehousing. Prentice Hall PTR, Englewood Cliffs (2000)
2. Booch, G., Rumbaugh, J., Jacobson, I.: Unified Modeling Language User Guide, 2nd edn. Addison-Wesley Professional, Reading (2005)
3. Bovik, A.C. (ed.): Handbook of Image and Video Processing, 2nd edn. Academic Press, London (2005)
4. Cantu-Paz, E., Cheung, S.S., Kamath, C.: Retrieval of Similar Objects in Simulation Data Using Machine Learning Techniques. In: Proc. of Image Processing: Algorithms and Systems III, SPIE, vol. 5298, pp. 251–258 (2004)
5. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image Retrieval: Ideas, Influences, and Trends of the New Age. ACM Comput. Surv. 40(2), 1–60 (2008)
6. Galindo, J. (ed.): Handbook of Research on Fuzzy Information Processing in Databases. Information Science Reference (2008)
7. Garcia-Molina, H., Ullman, J.D., Widom, J.: Database Systems: The Complete Book, 2nd edn. Prentice Hall PTR, Englewood Cliffs (2008)
8. Khotanlou, H., Colliot, O., Atif, J., Bloch, I.: 3D brain tumor segmentation in MRI using fuzzy classification, symmetry analysis and spatially constrained deformable models. Fuzzy Sets Syst. 160(10), 1457–1473 (2009)
9. Lew, M.S., Sebe, N., Djeraba, C., Jain, R.: Content-based Multimedia Information Retrieval: State of the Art and Challenges. ACM Trans. Multimedia Comput. Commun. Appl. 2(1), 1–19 (2006)
10. Lorenz, A., Blüm, M., Ermert, H., Senge, T.: Comparison of Different Neuro-Fuzzy Classification Systems for the Detection of Prostate Cancer in Ultrasonic Images. In: Proc. of Ultrasonics Symp., pp. 1201–1204. IEEE, Los Alamitos (1997)
11. Lyon, D.A.: Image Processing in Java. Prentice Hall PTR, Englewood Cliffs (1999)
12. Melin, P., Kacprzyk, J., Pedrycz, W. (eds.): Bio-Inspired Hybrid Intelligent Systems for Image Analysis and Pattern Recognition. Springer, Heidelberg (2010)
13. Pękalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition: Foundations and Applications. World Scientific, Singapore (2005)
14. Rajan, S.D.: Introduction to Structural Analysis & Design. Wiley, Chichester (2001)
15. Sarawagi, S.: Information Extraction. Foundations and Trends in Databases 1(3), 261–377 (2008)
16. Ślęzak, D.: Compound Analytics of Compound Data within RDBMS Framework – Infobright's Perspective. In: Proc. of FGIT. LNCS, vol. 6485, pp. 39–40. Springer, Heidelberg (2010)
17. Ślęzak, D., Eastwood, V.: Data Warehouse Technology by Infobright. In: Proc. of SIGMOD, pp. 841–845. ACM, New York (2009)
18. Smyth, B., Keane, M.T.: Adaptation-guided Retrieval: Questioning the Similarity Assumption in Reasoning. Artif. Intell. 102(2), 249–293 (1998)
19. Sosnowski, Ł.: Intelligent Data Adjustment using Fuzzy Logic in Data Processing Systems (in Polish). In: Hołubiec, J. (ed.) Systems Analysis in Finances and Management, vol. 11, pp. 214–218 (2009)
20. Sosnowski, Ł.: Constructing Systems for Compound Object Comparisons (in Polish). In: Hołubiec, J. (ed.) Systems Analysis in Finances and Management, vol. 12, pp. 144–162 (2010)

Intermediate Structure Reduction Algorithms for Stack Based Query Languages

Marta Burzańska[1], Krzysztof Stencel[1,2], and Piotr Wiśniewski[1]

[1] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University, Toruń, Poland
[2] Institute of Informatics, University of Warsaw, Warsaw, Poland

Abstract. Data processing often results in generation of a lot of temporary structures. They cause an increase in processing time and resources consumption. This especially concerns databases since their temporary data are huge and often they must be dumped to secondary storage. This situation has a serious impact on the query engine. An interesting technique of program fusion has been proposed for functional programming languages. Its goal is to reduce the size or entirely eliminate intermediate structures. In this paper we show how this technique can be used to generate robust execution plans of aggregate and recursive queries of query languages based on Stack Based Approach. We will use SBQL as an exemplary language.

Keywords: Intermediate structures, optimization, rewriting algorithms, SBQL, program fusion.

1 Introduction

Many implementations of data processing algorithms require creating temporary structures, which are totally irrelevant from users' point of view. When dealing with huge amounts of data, as with database query languages, such intermediate structures might consume much of systems resources since often they have to be swapped to slow secondary storage. When dealing with query languages operating on distributed databases, nowadays gaining more and more popularity, there is also the issue of amount of data being transferred through the network.

In 1990 P. Wadler [1] presented an algorithm of elimination of such structures in functional languages which he called the deforestation algorithm. This method became also known as "program fusion" because the basic idea behind it is to "fuse" together two functions of which one consumes an intermediate structure generated by the other. Nowadays there are many variations of deforestation algorithms. One of those version is known as the *foldr-build* rule (a.k.a. cheap deforestation, shortcut fusion) [2,3]. It has been successfully implemented in Glasgow Haskell Compiler [4] becoming the most universal and the simplest of fusion algorithms. It will be described briefly in Section 2 (*Cheap Deforestation*).

Due to declarative nature of query languages the optimization techniques for functional languages often became inspiration for query optimization. There are also papers on translation of object query language into a functional notation with a strong emphasis on the usage of foldr/build form [5,6].

Our research focus centers around the Stack Based Query Language (SBQL), which is a query language for object database systems [7]. Its main ideas are presented in Section 4 (*SBQL*) while Section 3 (*Stack Based Approach*) presents the main ideas behind the stack based approach to query languages. The following paper presents three techniques of intermediate structures elimination designed for languages based on SBA. Section 5 (*Simple SBQL Query Deforestation*) shows how can we adapt cheap deforestation for simple SBQL queries. Section 6 (*Optimising Recursion*) presents a new deforestation techniques for SBQL recursive queries. Section 7 (*Distributivity of algebraic functions over dot operator*) shows alternative technique of intermediate structures elimination designed especially for SBQL. Section 8 (*Conclusions*) concludes.

2 Cheap Deforestation

Cheap deforestation algorithm was introduced in [2]. This algorithm is based on an usage of a collection generating function (*build*) and a rule known as *foldr/build rule*. The notations in this section are written in the Haskell language, for which the algorithm was originally addressed. The meaning of the following rules is that the left side can be rewritten into the right side. Also, in Haskell language the notation $g\ f\ n$ means that g is a function call with two arguments: f and n. Let us start by defining the *build*, *foldr* and *foldl* functions:

```
build g = g : []
foldr f z [] = z
foldr f z (x:xs) = f x (foldr f z xs)
foldr f z [] = z
foldl f z (x:xs) = foldl f (f z x) xs
```

where colon is a list concatenation operator and [ ] is an empty list. The basic functionality of the *foldr* (*fold-right*) function is presented by Example 1 while Example 2 shows the usage of *foldl* (*fold-left*).

Example 1. `foldr (/) 2 [18,12,24]` results in: (18 / (12 / (24 / 2))) =18

Example 2. `foldl (-) 2 [18,12,24]` results in: (((18 - 12) - 24) - 2) = -20

Definition 1 (The foldr/build rule)

```
foldr f n (build g) = g f n
```

The assumption for functional languages is to translate all operators and functions into their equivalent compositions of *foldr* and *build* functions. Having such form one needs to apply interchangeably foldr/build rule with β-reduction (application of arguments). When none of those transformations can be applied, the outer *foldr* function should be rewritten using its definition and a proper collection constructor. Resulting function instead of applying subsequent operations to collections of intermediate data, would perform all operations sequentially on individual elements of the initial collection.

Many papers dealing with various languages and problems have shown that the usage of the cheap deforestation has a positive effect on the speed of program processing and the reduction of system resources consumption. The work [8] proves the equivalence of the *foldr/build* and *foldl/build* rules. This equivalence will be the basis of the deforestation algorithm described in Section 5.

3 Stack Based Approach

Stack Based Approach Stack based approach has been introduced by K. Subieta in [9]. It is a general approach to construction of query languages for object and semi-structural databases. SBA relies on the three basic elements: data model, environment stack and non-algebraic operators. Subieta proposed a set of store models that could be used in the Object DBMS. The basic model is called the *Abstract Data Model* or the *M0 Model*.

In this model an object is a triple $< i, n, v >$, where i is an identifier, n – the object's name, and v is its value. The value of i should be unique. Depending on a type of value of v we distinct three types of objects: atomic (when v is an atomic value), pointer (when v is an identifier of another object from the storage) and complex objects (when v is a collection of objects).

The M0 Model is a set of objects and a special set of identifiers of root objects. A more detailed description of this model can be found in [7,10].

Example 3 (A simple M0 model database in a graphical form).

```
<i1, Emp,{ <i2, fname, "John">, <i3, sname, "Smith">,
        <i4, dept, i20>, <i5, salary, 2000> }>
<i6, Emp,{ <i7, fname, "Bob">, <i8, sname, "Gordon">}
<i20, Dept,{ <i21, name, "IT">, <i22, employee, i1>,
        <i23, employee, i2>, <i24, boss, i6 > }>
R=[i1,i6,i20]
```

The *Environment Stack* (ENVS) is a structure responsible for correct binding of names with programming entities. The element that *bindes* a name with such entity is a pair (n, v) called a *binder.* In such pair n is a name by which a given programming entity exists in a given context, and v is a given programming entity. Binders as grouped in multisets called *sections.* Sections form a stack structure – the *Environment Stack.* An important feature that distincts ENVS from programming languages' environment stack (aka. call stack) is the lack of uniqueness of binders name within a single section.

The non-algebraic operators are a special group of binary operators for data manipulation. Their evaluation is highly dependent on the ENVS. Among the non-algebraic operators are: selection, projection\navigation, dependent joins, quantifiers and transitive closures. During evaluation of a non-algebraic operator two special functions are called: the *nested* and the *pop* functions. Both operate on the Environment Stack. The *nested* function creates a new section on top of the ENVS with binders to the contents of an object passed as an argument. The *pop* function deletes the top section from the ENVS. The ENVS itself is used to establish the context of a query. The detailed description of the evaluation process of the non-algebraic operators and the full description of the nested function may be found in [7].

Research on SBA has influenced the development of other scientific projects such as PySBQL [11] and AOQL (Aspect Object Query Language) that has been considered by OMG group as a proposal of standardization for object query languages [10].

4 SBQL

SBQL language is an object query language based on SBA approach [7] and a model language for all other projects influenced by SBA. The basic idea of SBQL is to combine querying and programming capabilities in one language that eliminates impedance mismatch. SBQL's query semantics is based upon recursive evaluation of the syntax tree and binding of names using environment stack. Compositionality and clarity of the language's definition resulting from those assumptions are a good place to begin work on optimization, especially on all kinds of query rewriting algorithms [12]. The key non-algebraic operators for SBQL are the selection operator *where* and the *dot* operator used for navigation and projection. Examples 4 and 5 show a sample usage of both operator.

Example 4 (For each employee working in the "IT" department return their personal data and salary).

```
(Emp where deptname=='IT').(name+' '+surname, salary)
```

Example 5 (For each department return its name and the average salary paid for its employees).

```
Dept.(name, avg(worksIn.Emp.salary) )
```

The compositionality of SBQL's structure allows also for easy and precise introduction of transitive closure. The main recursive operator is close by [13], unavailable in any form in OQL nor other popular object query language. The Section 6 of the following work centers around this group of operators.

5 Simple SBQL Query Deforestation

During the execution of the SBQL queries a lot of intermediate structures is being created. To reduce their sizes we propose a new algorithm that works on a level of execution plans. The main idea is inspired by a similar work for OQL ([5]), but it also addresses the problems of SBA. The execution plans in the following sections are written using lambda expressions from Python language notation. Although the same execution plans may be written using Haskell language notation the authors of this paper believe that the Python language would be more accessible for the broader audience because of its current popularity. The equivalent of the Haskell's foldl function in Python is the reduce function so in the following sections we will use the name "reduce/build" in place of "foldr/build".

Rule 1 (The reduce/build rule). *Let us assume that during the analysis of an execution plan the algorithm encounters two* nested *operations having the same argument and between their calls no other* nested *nor* pop *function is called. The inner* nested *call with corresponding* pop *function call and* slice *operation may be removed.*

Application of shortcut fusion to SBQL requires three steps. The first is to create a proper definition of the *build* function without violation of the main concept. The second is to create execution plan in the reduce/build notation for each operator. While doing it we must include the operations on the Environment Stack. The last step takes place during the creation of an execution plan for a composite query. It consists of interchangeable application of reduce/build rule with λ-calculus conversions until no more transformation can be used. Also, additional rule is needed regarding the operations on the Environment Stack:

Rule 2 (Nested/pop elimination). *Every occurrence of the function call:* `reduce( f,(build g), n)` *may be replaced with* `g(f, n)`

Just before running the execution plan one last step should be performed – the replacement of the *build* function with its definition and final simplification of the plan. Let us start by preparing a new set of execution plans:

Definition 2 (The execution plans for the SBQL's operators)

```
def build( f ): return f(struct.__add__, struct() )
where = lambda q1,q2:build( lambda c,n:reduce((
      lambda ys,y: (nested(y), (q2 and c(ys,y) or ys),
      pop())[1]),q1,n))
dot = lambda q1,q2:build( lambda c,n:reduce((
      lambda ys,y: (nested(y), reduce(c,q2,ys),
      pop())[1]), q1,n))
join = lambda q1,q2:build( lambda c,n:reduce((
      lambda ys,y: (nested(y), reduce(lambda e,es:
         c(es,struct(y,e)),q2, ys), pop())[1]), q1,n))
all = lambda q1,q2:build( lambda c,n:reduce((
      lambda ys,y: y and (nested(y), q2, pop())[1]), q1,True))
sum = lambda q1: reduce((lambda ys,y: __add__(ys,y)),q1,0)
```

The *all* operator can be used for expressing functions like *forall*, *exists*. Also, the *sum* function may be used as a prototype for functions like *count*, *min*, etc. Having the above definitions while modifying the execution plans for the composition of operators the following steps should be undertaken: first the replacement of the operators with their execution plans. Secondly we should search the query text to find a place for application of the reduce/build rule. Next we should simplify the plan's text using β-reduction and nested/pop elimination. Those procedures should be repeated until no further modification is possible. Finally we apply the build definition and execute the plan.

In order to explain how the algorithm operates let us consider an example query:

```
(Emp where sname = "Smith").dept                                    (1)
```

For the sake of shorter and clearer notation we will write P instead of ($sname =$ `"Smith"`). Evaluation of this predicate is irrelevant to the deforestation technique.

The first step of the algorithm is to translate the query (1) into the composition of the basic execution plans:

```
build( lambda c,n: reduce((lambda ys,y: (nested(y), reduce(c,
   evaluate('dept'),ys),pop())[1]), build( lambda c2,n2:
      reduce((lambda zs,z: (nested(z), (evaluate(P) and
         c2(zs,z) or zs),pop())[1]), evaluate('Emp'),n2)),n))
```

For the inner *reduce* and *build* functions we apply the reduce/build rule:

```
build( lambda c,n:(lambda c2,n2: reduce((lambda zs,z:
   (nested(z), (evaluate(P) and c2(zs,z) or zs), pop())[1]),
   evaluate('Emp'),n2)) ((lambda ys,y: (nested(y),
      reduce(c, evaluate('dept'),ys), pop())[1]),n))
```

After multiple application of the β-reduction and the nested/pop elimination rule:

```
build( lambda c,n:reduce((lambda zs,z: (nested(z),
   (evaluate(P) and reduce(c,evaluate('dept'),zs) or zs,
   pop())[1]), evaluate('Emp'),n))
```

Because we cannot apply neither reduce/build transformation, nested/pop elimination nor β-reduction we now have to apply the definition of the build function. After one more β-reduction we acquire:

```
reduce((lambda zs,z: (nested(z), (evaluate(P) and
   reduce(struct.__add__,evaluate('dept'),zs) or zs,
      pop())[1]), evaluate('Emp'), struct()))
```

During the evaluation of the not-optimized input plan one intermediate list would be created - a list of employees fulfilling the predicate P. In the deforested version this intermediate structure is not being created. The output plan has the following meaning: during its execution for each employee check if the surname condition is met, and if so, add their department reference to the result collection. Each employee is considered only once, what reduces resources consumption.

Another benefit of this method is that the evaluation of the output program is at least as fast as evaluation of the input program in the worst case scenario, and in a better one - can speed up the process. Additionally, after a new plan for a specific composition of two operators has been generated it can be stored for future usage.

6 Optimising Recursion

The previous section describes a proposition of optimization technique for SBQL queries. However, this technique targets only non-recursive queries. This section proposes a rewriting algorithm for elimination of intermediate structures occurring during evaluation of a composition of the *close by* operator and an aggregate function. The construction of this algorithm has been inspired by lightweight fusion technique for functional languages described in [14].

Before describing the mentioned algorithm, we first need to introduce an execution plan for the *close by* operator written using Python language notation.

Definition 3. *The execution plan for the close by operator is represented with the following recursive function definition and call:*

```
   def closeby (dotFunction, queryRes):
      if isEmpty(queryRes): return bag()
      else: return bag.__add__(queryRes,
         closeby (dotFunction, dotFunction(queryRes))
closeby(makeDotF(leftQuery),eval(rightQuery))
```

Let Q be a close by query and A be an aggregate function that takes Q as an argument. Our algorithm is composed out of three steps: firstly inline the A function's call into both return clauses of the close by's execution plan function; secondly simplify all calculation that can be computed without searching through the database section; next generate a new execution plan function representing the composition of the analyzed operators and replace the A(Q) call with this execution plan. The newly generated execution plan function may be stored for the commonly used compositions. Let us analyze this algorithm on a composition of the *count* function with the *close by* operator.

```
   count( Q1 close by Q2)                                        (2)
```

Where Q1 and Q2 are general queries matching the limitations of the *close by* operator. To optimize this composition we first should inline the *count* function into the definition of the *close by*'s execution plan function:

```
   if isEmpty(queryRes): return count(bag())
   else:  return count(bag.__add__(queryRes,
      closeby (dotFunction,dotFunction(queryRes)))
```

After basic simplification, the last step is to generate a new execution plan function for the query (2):

```
   def count_closeby (dotFunction, queryRes):
      if isEmpty(queryRes): return 0
      else: return count(queryRes) +
         count_closeby (dotFunction, dotFunction(queryRes))
count_closeby(makeDotF(Q2),eval(Q1))                           (2*)
```

This new function will calculate the same result as the initial query, but it does not use an intermediate structure containing all of the database elements that are retrieved during the evaluation. Instead it counts those elements at each iteration of the recursion. For multilevel hierarchy it significantly reduces the size of intermediate structures, because the maximum size of such structure is equal to the sum of objects acquired in a given iteration. This technique can be efficiently combined with *reduce/build* rules described in Section 5.

7 Distributivity of Algebraic Functions over Dot Operator

This section presents alternative rewriting algorithm that may be used in SBQL implementations not using a functional meta-language [11,15]. This algorithm targets mostly distributed databases, however it may be used in a standalone server especially one equipped with parallel query evaluation engine. Let us consider a simple query:

```
    sum(Dept.employs.Emp.salary)                                    (3)
```

Using two techniques previously described this query may be folded into a simple function that traverses the tree of employment and adds up every encountered salary. However, in a distributed database such approach may reduce the network traffic, however based on the unique property of the dot operator we have developed a new technique of deforestation that significantly reduces the amount of data sent through the network. This algorithm is based on adaptation of distributivity of linear algebraic functions over the dot operator. Our algorithm takes as an input a simple algebraic function like *sum, min, max* that has a dot expression as an argument. On output it generates a query in which after each occurrence of dot operator it inserted the initial function. The result of processing query (3) with this algorithm is:

```
   sum(Dept.sum(employs.sum(Emp.sum(salary))))                      (4)
```

Now let us assume that in our hypothetical company we have 100 departments, each one employing at least 1000 employees. Without any optimization we have to store more than 100 000 *salary* objects in an intermediate structure and transfer most of them through the network. For the optimized query the size of the biggest intermediate structure is reduced 100 times. Also, the distributed servers with Emp objects stored on different servers may perform partial evaluation of this query hugely decreasing the network traffic – instead of sending 100 *salary* objects only one number would be sent to the main server.

The functions that can be distributed over the dot operator include *sum, min, max.* The *count* function might seem troublesome for the use of this technique. But when we translate *count(query)* into its equivalent *sum(query.1)* it becomes apparent that the *count* function is also susceptible to distributiveness. One other operator that is often used in database queries is *avg* (arithmetic average) operator. This operator can also be translated into a composition of few functions (we omit here the details of implementation of execution plans):

```
avg_p(x) = ( sum(x), count(x) )
avg_p_sum(plist) = ( sum(first(plist)), sum(second(plist)) )
avg_div(x,y) = if y!=0: x/y else: 0
```

The `avg_p_sum` function simultaneously increases the aggregate and the count variable. The `avg_p_sum` function takes a list of pairs of numbers, and returns a pair of numbers that represent the sum of respectively the first and the second elements of pairs. This function is susceptible to distributiveness over the dot operator Having those three functions we now can present a new definition of the *avg* function:

```
avg x = avg_div(avg_p_sum(avg_p(x)))
```

On flat collections this transformation creates intermediate structures and it worsens the speed of evaluation. But it is meant to deal with complex path (dot) queries, and for them it has the advantage of reducing the intermediate structures and increasing the speed of evaluation. Let us consider a query:

```
avg((Emp where position == "Manager").subordinate.Emp.salary)
```

Let us assume that each of the managers has 1000 subordinates, and there are 100 managers. An intermediate structure of 100 000 database objects would be created in order to calculate the result. Now let us consider the alternative definition already in a distributed form:

```
avg_div(avg_p_sum((Emp where position == "Manager").
      avg_p_sum(subordinate.avg_p_sum(Emp.avg_p(salary)))
```

During the evaluation of this query the outermost *avg_p_sum* reads from the database those employees that match the filtering condition. For each one of them it evaluates the inner *avg_p_sum* that would bind the name *subordinate* within the context of a current employee, and so on. This way the biggest intermediate structure will consist of 100 pair of numbers, which is a considerable storage saving.

8 Conclusions

In this paper we have shown three optimization techniques for stack based query languages. Their goal is to reduce the size of intermediate structures that are created during the evaluation of queries. The application of those techniques has been presented using SBQL language as an example. Initial tests have shown that without a big memory expenditure on the optimization algorithm, all three methods significantly reduce the amount of memory resources needed to deal with intermediate structures. One more advantage of the reduce/build notation is that it integrates efficiently with imperative operators (like if then else). Also, nearly all operators used to query a database can be defined using this notation. This allows us to use the functional language notation as a form of meta-language for a general query translation.

Deforestation techniques that translate SBQL into the functional notation are stronger methods than the distributivity over the dot operator, and when used together, distributivity will be overwritten. But it is not predetermined which method is better. Depending on the context and the cost model one maybe preferable or more efficient than the other.

References

1. Wadler, P.: Deforestation: Transforming programs to eliminate trees. Theor. Comput. Sci. 73(2), 231–248 (1990)
2. Gill, A.J., Launchbury, J., Jones, S.L.P.: A short cut to deforestation. In: FPCA, pp. 223–232 (1993)
3. Johann, P.: Short cut fusion: Proved and improved. In: Taha, W. (ed.) SAIG 2001. LNCS, vol. 2196, pp. 47–71. Springer, Heidelberg (2001)
4. Jones, S.P., Tolmach, A., Hoare, T.: Playing by the rules: rewriting as a practical optimisation technique in GHC. In: Haskell Workshop, ACM SIGPLAN, pp. 203–233 (2001)
5. Grust, T., Scholl, M.H.: Query deforestation. Technical report, Faculty of Mathematics and Computer Science, Database Research Group, University of Konstanz (1998)
6. Trigoni, A.: Semantic optimization of OQL queries. Technical Report UCAM-CL-TR-547, University of Cambridge, Computer Laboratory (2002)
7. Subieta, K.: Theory and Construction of Object Query Languages [in Polish]. Publishers of the Polish-Japanese Institute of Information Technology (2004)
8. Gill, A.: Cheap deforestation for non-strict functional languages. PhD thesis, The University of Glasgow (1996)
9. Subieta, K., Beeri, C., Matthes, F., Schmidt, J.W.: A stack-based approach to query languages. In: East/West Database Workshop, pp. 159–180 (1994)
10. OMG Object Database Technology Working Group: Next-generation object database standarization. White Paper (2007)
11. Rogińska, M., Wiśniewski, P.: Pysbql - python-like query language constructed using stack base approach. In: Annales UMCS, Informatica, pp. 143–151 (2007)
12. Płodzień, J.: Optimization Methods in Object Query Languages. PhD thesis, Institute of Computer Science, Polish Academy of Science, Warsaw, Poland (2000)
13. Pieciukiewicz, T., Stencel, K., Subieta, K.: Recursive Query Processing in SBQL. In: Proceedings of the First International Conference on Object Databases, pp. 56–76 (2008)
14. Ohori, A., Sasano, I.: Lightweight fusion by fixed point promotion. In: Hofmann, M., Felleisen, M. (eds.) POPL, pp. 143–154. ACM, New York (2007)
15. Stencel, K., et al.: LoXiM database project (2010), `http://loxim.sourceforge.net/`

Service-Oriented Software Framework for Network Management

Dongcheul Lee[1] and Byungjoo Park[2,*]

[1] Mobile R&D Laboratory, KT
17, Woomyeon-dong, Seoul, Korea
jackdclee@kt.com
[2] Department of Multimedia Engineering, Hannam University
133 Ojeong-dong, Daeduk-gu, Daejeon, Korea
bjpark@hnu.kr

Abstract. A network management is important in managing network elements. While hundreds of network-related new services have been created, the network management functions should have been re-coded to adapt new business rules. In addition, redundant information could be transferred repeatedly to the inter-operating systems because they request similar information. This environment brings unnecessary system development cost, and increases redundancy in the inter-operating functions. To reduce the cost and the redundancy, we propose the service-oriented software framework for network management. In order to do so, we identified common services for network management, made service specifications and service flows, and conducted service realization. Also, we present four types of services as the case studies: authentication service, SMS service, previous alarm inquiry service, and current alarm management service. Furthermore, we implemented the framework in KT to manage IP backbone networks. After adopting the framework, the system's performance and flexibility were improved, and duplicated functionalities of the systems were reduced.

Keywords: Software Framework, Network Management, Service-oriented Architecture.

1 Introduction

Over the last few years, Service-oriented Architecture (SOA) [1] has got much attention, bringing a new era of software development and business agility. It provides the ability to make loosely coupled links between business functions and specific applications by isolating service definition and usage from each system's service implementation. An Enterprise Service Bus (ESB) is a core component of a SOA and implements a SOA through middleware that offers connection and integration management of an organization's IT infrastructure across many differing systems.

* Corresponding author.

Even though there had been many attempts to solve the enterprise's IT problems with a SOA, it is very hard to find the successful deploying cases of a SOA. Because many IT decision makers try to solve the problems only by adopting an ESB product without the full understanding of a SOA, leaving the previous functions and logics unchanged. Meanwhile, IBM suggested best practices [2] for deploying a successful SOA but it didn't address concrete method for each specific business field. S. Glen and J. Andexer [3] decomposed eTOM [4] operation level two and deduced services for each operation. M. Brandner, M. Craes, F. Oellermann and O. Zimmermann [5] featured the lessons learned during the implementation of SOA in the finance industry.

However, we have not found the studies which addressed service-oriented network management field in spite that the field is worth to challenge. A Network Management System (NMS) plays an import role in managing and monitoring a network. Other systems request the information which was produced by the NMS for analyzing network faults, reporting customer's traffic analysis, testing customers' equipments, and network engineering, etc. The NMS should be re-coded to adapt new business rules, logic, and data access method while hundreds of network-related new businesses are created. In addition, redundant information must be transferred between these systems because, generally, they request similar information. This environment brings increased system development cost, and increases redundancy in system development process. Therefore, we present the service-oriented software framework for network management to reduce the development cost and to eliminate the redundancy in the NMS development process by deploying a SOA using an ESB.

The next section defines the problem of network management and suggests to-be architecture which is a SOA. Section 3.1 explains how we identified network management related services and gives some examples of them. In section 3.2, we introduce document templates for specifying and realizing the services so that system designers and software developers can communicate effectively through the document. Section 4 presents best case studies which were conducted while we were deploying a SOA to our framework. In section 5, we describe the process and the result of the implementation and discuss the lesson learned. The paper concludes with a short discussion.

2 Problem Definition

KT's NMS and other interoperating systems have been used for monitoring and managing various network elements. Each NMS has its own roll and other systems which need the information collected from a NMS request a service for requiring the information. Therefore, many complex interoperation functions had existed for exchanging the information. These peer-to-peer connections made systems hard to manage. Also, each system had communicated synchronously which made systems tightly coupled. Therefore, if a system was turned down, other systems that had communicated with the system made errors.

Fig. 1 shows that these complex peer-to-peer connections can be eliminated by using an ESB. Because all the connections between systems are established through an ESB, the interoperation structure becomes simple. Also, because an ESB has a message multicasting functionality, a service-providing system does not need to develop similar services for each business process if the processes work similarly. Furthermore, an ESB makes systems loosely coupled because if a service provider does not reply for a request for some reasons, a service consumer would receive a reply message which is the result of an ESB's error handling. Also, if the Store And Forward (SAF) functionality of an ESB is used, the data sender would not need to care whether the data were sent successfully or not. This is because an ESB stores the data in its own database and transmits the data repeatedly until the data will be sent successfully.

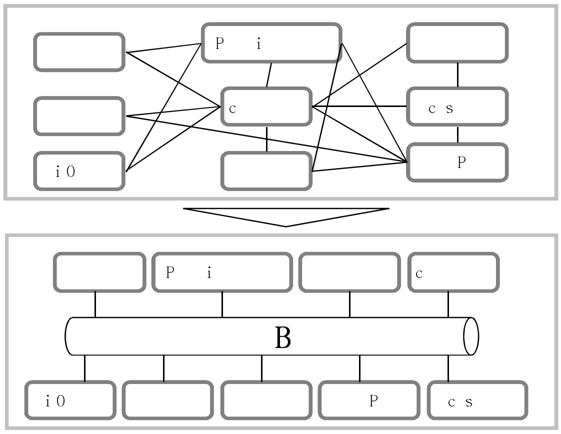


Fig. 1. After adopting an ESB, peer to peer connections are eliminated

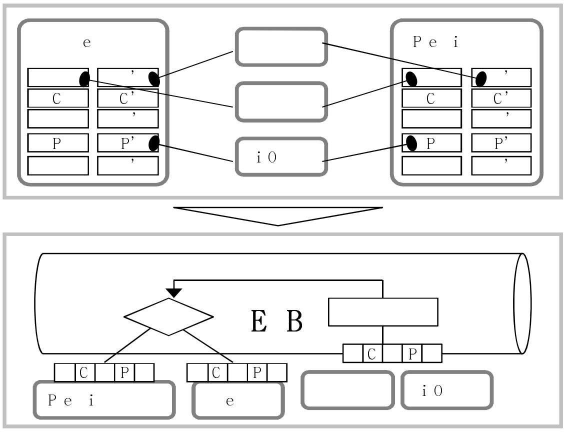


Fig. 2. After adopting an ESB, redundant functions are reduced

Another problem of the NMS is their architecture. Each system had been developed in 'silo' type so that the NMS had to develop its own fault, configuration, accounting, performance, and security (FCAPS) [6] functionalities. Therefore, redundant functionalities were developed between the NMS. Also, the business rules were hard-coded in each system so that if the rule had changed, the system had to be re-coded in spite that understanding the legacy code which should be modified is time-consuming and difficult job.

Fig. 2 shows that this ‘silo’ type system development method can be overcome by adopting a SOA to the systems using an ESB. Once FCAPS services were published on an ESB, each system which wants to use the service does not have to implement the service on its own. Also, a service consumer does not need to know where a service provider is located so that a location transparent service can be provided. Furthermore, handling business process change is easy because the flows in the service can be re-routed or modified without re-coding of an application. In addition, an ESB provides various types of adapters for web services, transaction processing (TP), TCP/IP, HTTP, FTP and DBMS. They interconnect an ESB with in-coming requests so that diverse applications having different protocols can use the services of an ESB.

3 Service Analysis and Design

To design the service-oriented NMS, we adopted IBM’s process of service-oriented modeling and architecture [7]. As shown in Fig. 3, the process consists of three steps: identification, specification and realization of services, components, and flows.

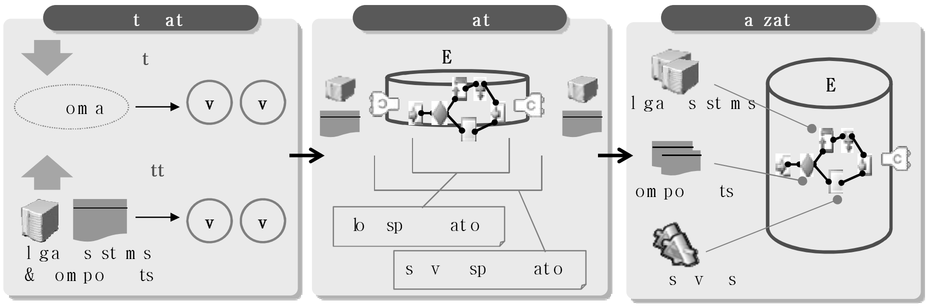


Fig. 3. The process of service-oriented modeling is shown

3.1 Identification

In the identification step, we identified the group of services that the NMS should provide commonly, and classified them into several categories. To identify the group, we used a cross-sectional approach that combines the top-down and the bottom-up approaches. The top-down approach analyzes business domains and decomposes these domains into business processes, sub-processes and use cases. Whereas the bottom-up approach analyzes legacy systems and components to realize the services defined in the business domains and discover services that had not found in the top-down approach.

In the top-down approach, we classified the services into three levels as shown in Table 1. In the first level, we identified general business processes of the NMS which were network fault management, network resource collection management, network resource performance management, network resource inventory management, operation environment management, etc. In the second level, the business processes defined in the first level were subdivided into specific categories. For example, the network fault management process was subdivided into current network fault management and previous network fault management, because operators usually monitor current network's faults in real time whereas they search previous network's faults when they need to analyze the cause of faults or to write a monthly network fault report. In the final level, services were identified by decomposing the second level categories and we affirmed uncovered services by using the bottom-up approach. For example, the current network fault management category deduced reporting current alarm service, identifying alarm service, and releasing alarm service.

As a result of the identification step, we identified 110 services which can be commonly used at any NMS.

Table 1. Services were identified by using a cross-sectional approach

1st Level	2nd Level	3rd Level: Service
network fault management	current network fault	report current alarm
		identify alarm
		release alarm
	previous network fault	inquiry previous alarm
operation environment management	send messages	request SMS message sending
etc.	authentication	request authentication

3.2 Specification and Realization

For each service, which was defined in the identification step, we should specify the details of the components that will implement the services in the specification step. The definition of the messages and the events were made at this step. After the specification step, we should decide whether the software should be re-used or re-built. To do this, we made standard specification templates which address the flow specification and the service specification. For each identified service, we made out the documents. Both documents play an important role as a communication method between service designers and developers.

The service specification document deals with end-to-end service specification between the service provider and the service consumer. The specification has two types depending on the service adapter type: web service type and TP type. The web service type specification requires service ID, web service URL, service description, operation name, WSDL, operation in/out elements, element type, element size, mandatory or optional constraint, and description. Because most KT's NMS work on the TP-monitor, we also made the TP type specification. The specification requires service ID, service name, service description, and operation in/out elements.

The flow specification document deals with the flows in the service, or adapter rules, and the mapping between the adapter rule and the message or the service in an ESB. The service rule part describes in-coming messages and reply messages and flows between the services. It requires service flow ID, service flow diagram, and flow description. Fig. 4 (a) is the example of a service flow diagram. The adapter rule part describes the rules applied to the inbound adapter. It requires adapter rule ID, type, SAF flag, context path, operation name, message ID, operation in/out elements, element type, element size, mandatory constraint, and description. The type of adapter rule can be either web service type or TP type. The message ID, which can be found in the service flow diagram, and the operation name have one to one relation. The SAF flag is set when a request should be handled in SAF mechanism. The element type can be string, int, long, short, float, or double. The element size field is needed only if the type is a string. The mandatory constraint field is set when the element must exist. The mapping part requires adapter rule ID, mapping diagram, message or service ID, in/out elements, message or service field, and their mapping rule. Fig. 4 (b) shows the example of a mapping diagram. If the mapping rule has one to one relation, which is a direct flow, the mapping part do not need to be described.

4 Best Case Studies

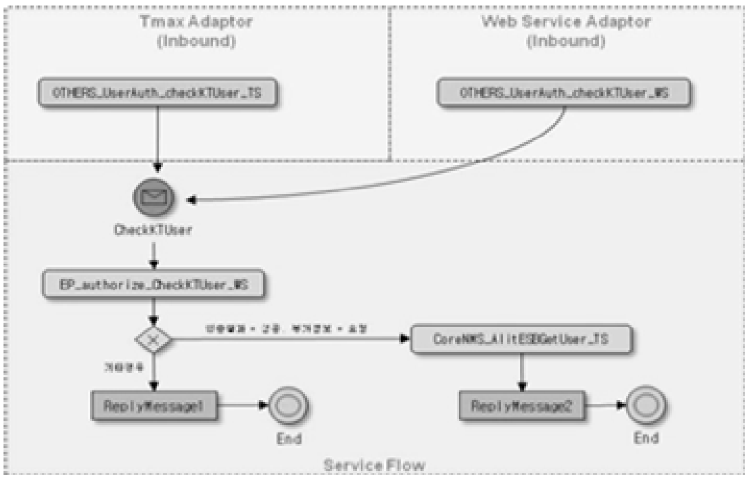
Among the services which were defined in the previous section, we present four services as best case studies as shown in Fig. 5. They include authentication service, SMS service, previous alarm inquiry service, and current alarm management service.

The EP system is KT's enterprise authentication server and the HRM system manages KT's personnel information. Other systems had requested an authentication service to the EP system and had inquired personnel information to the HRM system directly. Other systems had combined information to manage their user's accounts after receiving information. After adopting an ESB, we made the authentication service so that other systems do not need to combine authentication information and personnel information on their own. Also, we made two kinds of adapters which operate identically with different protocols.

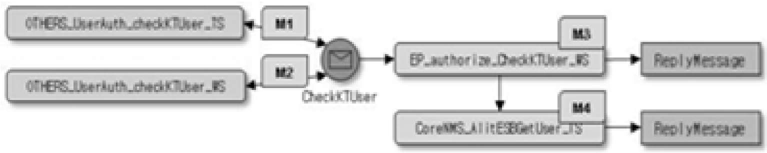
The enterprise SMS server, RTSMS, provides a SMS web service. The service has been published on an ESB with two kinds of adaptors so that other systems can use a SMS service more efficiently.

The bNMS is a backbone network management system. It had provided network fault related information to other systems directly with different services. After an ESB has provided the alarm inquiry service, other systems only have to call one integrated service because an ESB re-directs the requests to appropriate services for each system. Therefore, even though the systems call the same service, they can receive different domain data.

Many systems want to receive the current alarm status from the NMS. The current alarm status is changed when following events occurred: detecting network faults, identifying an alarm by an operator, and releasing network faults. Before adopting an


Tmax Adaptor
(Inbound)
Web Service Adaptor
(Inbound)
OTHERS_UserAuth_checkKTUser_TS
OTHERS_UserAuth_checkKTUser_WS
CheckKTUser
EP_authorize_CheckKTUser_WS
CoreNMS_AlltESBGetUser_TS
ReplyMessage1
ReplyMessage2
End
End
Service Flow

(a)


OTHERS_UserAuth_checkKTUser_TS
M1
OTHERS_UserAuth_checkKTUser_WS
M2
CheckKTUser
M3
EP_authorize_CheckKTUser_WS
ReplyMessage
M4
CoreNMS_AlltESBGetUser_TS
ReplyMessage

(b)

Fig. 4. The example of a service flow diagram (a) and a mapping diagram (b) is shown

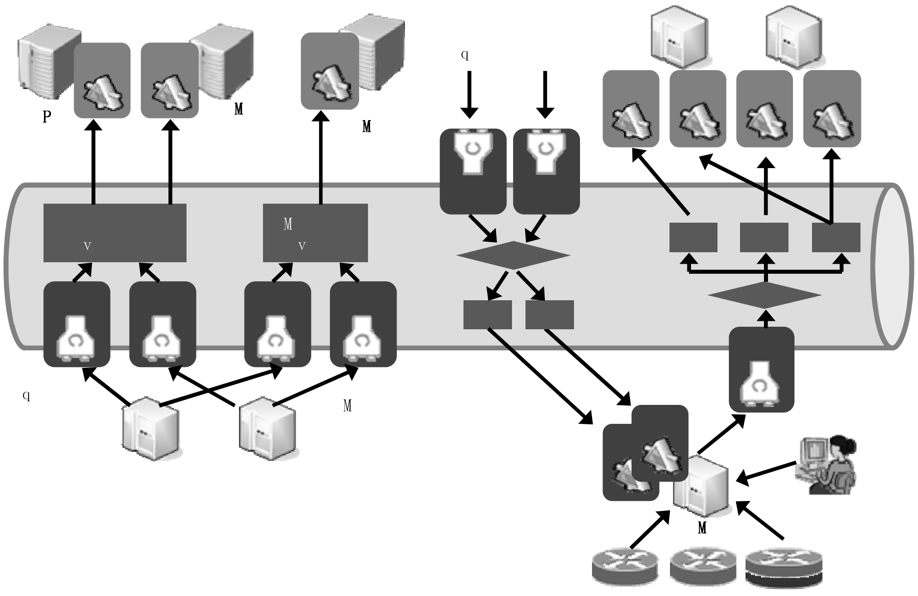


Fig. 5. Four services were derived as a result of service-oriented modeling

ESB, each event should have been sent to other systems repeatedly having different contexts: a system may only want the network link event, whereas another system may want total network events. Because an ESB can multicast the events to other systems while the contents of the events can be diverse, bNMS does not need to send the events repeatedly. Therefore, the performance of bNMS is increased by using the multicast functionality of an ESB.

5 Implementation

The service-oriented NMS has been implemented in KT's bNMS with TMAX ProBus [8] ESB. 11 NMS has changed their inter-operation target from the bNMS to an ESB. As we mentioned in the previous section, the non-NMS services has been also published in an ESB.

As a result, a redundant function or service development is reduced because published services can be re-used by other systems. Also, the messages exchanged between systems can flow across different transport protocols so that systems with different platforms can communicate efficiently. Furthermore, system designers do not need to analyze complex code to understand business process because an ESB offers the service flow diagram.

However, we also have encountered some challenges after adopting an ESB. First, because an ESB is a standalone server as other NMS, the system administrator should manage more objects. Also, if an error occurred while using a service, it is hard to identify the cause of the error because an ESB itself can be an additional error point. Furthermore, an ESB engineer is needed whenever services are created or modified because most system developers cannot handle ProBus studio which is a development tool for an ESB.

The performance of an ESB server, especially average CPU usage, has not been raised above seven percent while handling five hundred thousand transactions per day. Among these transactions, three hundred thousand transactions were web service type and two hundred thousand transactions were TP type. Also, the average CPU usage of bNMS application server is lowered by two percent because repeated calls were converted into multicast calls.

Since it is the initial stage of spreading the usage of an ESB across the NMS and other legacy systems, gradual transition to the new architecture is needed. To do this, we used the direct flow for previously existed services to minimize the re-coding of legacy system's interoperating functions. However, at least destination IP address should have been changed from bNMS's to the EBS's in the case of an in-coming request to bNMS. That is, other systems do not have to change anything for the out-going request from bNMS. However, the dynamic message routing will be used for newly created services. Furthermore, the legacy code and the service have been co-existed since various legacy systems have different agenda to transit to the new architecture.

However, the business process change has not been occurred often in the field of network management as the field of operation support so that the flow coordination functionality has not been used frequently. In spite of this, the service-oriented NMS is still attractive because the interoperation structure becomes simplified and the services can be re-used.

6 Conclusions and Future Works

We proposed and implemented a service-oriented network management system using an ESB, especially to address the integration and agility problems of KT's NMS. To do that, we identified common services of the NMS and created document templates for the specification and the realization of the services. Also, we presented best case studies which were deduced while the implementation. As a future works, we will spread a SOA to other systems and will apply dynamic message routing to legacy services which would be implemented in the direct flow.

Acknowledgments. This work has been supported by the 2010 Hannam University Research Fund.

References

1. Arsanjani, A.: Service-oriented modeling and architecture (2004), `http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/`
2. IBM Global Services: Five best practices for deploying a successful service-oriented architecture (2008)
3. Glen, S., Andexer, J.: A practical application of SOA (2007), `http://www.ibm.com/developerworks/webservices/library/ws-soa-practical/`
4. Kelly, M.B.: Report: The TeleManagement Forum's Enhanced Telecom Operations Map (eTOM). Journal of Network and Systems Management 11(1), 109–119 (2003)
5. Brandner, M., Craes, M., Oellermann, F., Zimmermann, O.: Web services-oriented architecture in production in the finance industry. Informatik Spektrum 26, 136–145 (2004)
6. Audin, G., Lodge, F.: FCAPS: a Model for VOIP/IPT Management. Business Communications Review (2006)
7. Arsanjani, A.: Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA (2004), `http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/`
8. ProBus, `http://us.tmaxsoft.com/jsp/product/detailcontents.jsp?psCd=00PD13&menuCd=0PD` SOPB

Authors

Dongcheul Lee received the B.S. and M.S. degrees in Computer Science and Engineering from Pohang University of Science and Technology, Pohang, Korea in 2002 and 2004, respectively. He is currently working towards a Ph.D. degree with the Department of Information and Communication Engineering of Hanyang University, Seoul, Rep. of Korea. He has been a senior researcher in the KT Network Technology Laboratory, Korea since 2004. His research interest includes algorithm and application of mobile communications, workforce scheduling issues for network service provisioning, and task scheduling algorithms for GRID environment.

Byungjoo Park received the B.S. degree in electronics engineering from Yonsei University, Seoul, Rep. of Korea in 2002, and the M.S. and Ph.D. degrees (First-Class Honors) in electrical and computer engineering from University of Florida, Gainesville, USA, in 2004 and 2007, respectively. From June 1, 2007 to February 28, 2009, he was a senior researcher with the IP Network Research Department, KT Network Technology Laboratory, Rep. of Korea. Since March 1, 2009, he has been a Professor in the Department of Multimedia Engineering at Hannam University, Daejeon, Korea. He is a member of the IEEE, IEICE, IEEK, KICS, and KIISE. His primary research interests include theory and application of mobile computing, including protocol design and performance analysis in next generation wireless/mobile networks. He is an honor society member of Tau Beta Pi and Eta Kappa Nu, USA. His email address is vero0625@hotmail.com, bjpark@hnu.kr

Author Index