

Frontiers
in
Artificial
Intelligence
and
Applications

KNOWLEDGE-BASED SOFTWARE ENGINEERING

Proceedings of the Seventh Joint Conference
on Knowledge-Based Software Engineering

Edited by
Enn Tyugu
Takahira Yamaguchi

IOS
Press

KNOWLEDGE-BASED SOFTWARE ENGINEERING

Frontiers in Artificial Intelligence and Applications

FAIA covers all aspects of theoretical and applied artificial intelligence research in the form of monographs, doctoral dissertations, textbooks, handbooks and proceedings volumes. The FAIA series contains several sub-series, including "Information Modelling and Knowledge Bases" and "Knowledge-Based Intelligent Engineering Systems". It also includes the biannual ECAI, the European Conference on Artificial Intelligence, proceedings volumes, and other ECCAI – the European Coordinating Committee on Artificial Intelligence – sponsored publications. An editorial panel of internationally well-known scholars is appointed to provide a high quality selection.

Series Editors:

J. Breuker, R. Dieng, N. Guarino, J.N. Kok, J. Liu, R. López de Mántaras,
R. Mizoguchi, M. Musen and N. Zhong

Volume 140

Recently published in this series

- Vol. 139. A. Bundy and S. Wilson (Eds.), Rob Milne: A Tribute to a Pioneering AI Scientist, Entrepreneur and Mountaineer
- Vol. 138. Y. Li et al. (Eds.), Advances in Intelligent IT – Active Media Technology 2006
- Vol. 137. P. Hassanaly et al. (Eds.), Cooperative Systems Design – Seamless Integration of Artifacts and Conversations – Enhanced Concepts of Infrastructure for Communication
- Vol. 136. Y. Kiyoki et al. (Eds.), Information Modelling and Knowledge Bases XVII
- Vol. 135. H. Czap et al. (Eds.), Self-Organization and Autonomic Informatics (I)
- Vol. 134. M.-F. Moens and P. Spyns (Eds.), Legal Knowledge and Information Systems – JURIX 2005: The Eighteenth Annual Conference
- Vol. 133. C.-K. Looi et al. (Eds.), Towards Sustainable and Scalable Educational Innovations Informed by the Learning Sciences – Sharing Good Practices of Research, Experimentation and Innovation
- Vol. 132. K. Nakamatsu and J.M. Abe (Eds.), Advances in Logic Based Intelligent Systems – Selected Papers of LAPTEC 2005
- Vol. 131. B. López et al. (Eds.), Artificial Intelligence Research and Development
- Vol. 130. K. Zieliński and T. Szumuc (Eds.), Software Engineering: Evolution and Emerging Technologies
- Vol. 129. H. Fujita and M. Mejri (Eds.), New Trends in Software Methodologies, Tools and Techniques – Proceedings of the fourth SoMet_W05
- Vol. 128. J. Zhou et al. (Eds.), Applied Public Key Infrastructure – 4th International Workshop: IWAP 2005

Knowledge-Based Software Engineering

Proceedings of the Seventh Joint Conference on Knowledge-Based
Software Engineering

Edited by

Enn Tyugu

Institute of Cybernetics, Tallinn University of Technology, Estonia

and

Takahira Yamaguchi

Keio University, Japan

IOS
Press

Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

© 2006 The authors.

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without prior written permission from the publisher.

ISBN 1-58603-640-8

Library of Congress Control Number: 2006929180

Publisher

IOS Press

Nieuwe Hemweg 6B

1013 BG Amsterdam

Netherlands

fax: +31 20 687 0019

e-mail: order@iospress.nl

Distributor in the UK and Ireland

Gazelle Books Services Ltd.

White Cross Mills

Hightown

Lancaster LA1 4XS

United Kingdom

fax: +44 1524 63232

e-mail: sales@gazellebooks.co.uk

Distributor in the USA and Canada

IOS Press, Inc.

4502 Rachael Manor Drive

Fairfax, VA 22032

USA

fax: +1 703 323 3668

e-mail: iosbooks@iospress.com

LEGAL NOTICE

The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

Foreword

It is our pleasure to introduce this volume of proceedings of the Joint Conference on Knowledge-Based Software Engineering 2006 (JCKBSE'06) held in August 28–31, 2006 in Tallinn, Estonia. The JCKBSE is a traditional biannual conference focused at applications of artificial intelligence in software theory and practice. This conference has become an international event with talks submitted from sixteen countries this year. After careful reviewing, 37 talks from thirteen countries have been accepted. Traditionally for JCKBSE, most of talks are from Japan, the second place belongs to Russia. The talks address the research in theoretical foundations, practical techniques, software tools, applications and/or practical experiences in knowledge-based software engineering. The topic of the present conference includes a new field: research in web services and semantic web. This is a rapidly developing research area promising to give excellent practical outcome, and interesting for theoretically minded as well as for practically minded people. The largest part of talks belongs to a traditional area of applications of artificial intelligence methods to various software engineering problems. Another traditional section of the conference is application of intelligent agents in software engineering. A separate section is devoted to interesting applications and special techniques related in one or another way to the topic of the conference.

We would like to thank, first of all, the authors of submitted papers. Their research is a justification for a conference in this field of science. We express our gratitude to members of the Program Committee of the conference for their thorough work in reviewing the submissions. The work of program committee and its chairs has been greatly facilitated by EasyChair that is an excellent tool developed and managed by Andrei Voronkov, to whom we wish to express our gratitude for his kind support.

Enn Tyugu and Takahira Yamaguchi
JCKBSE'06 program co-chairs

Conference Organization

Steering Committee

- Christo Dichev, Winston-Salem State University, USA
- Pavol Navrat, Slovak University of Technology, Slovakia
- Vadim L. Stefanuk, IITP, Russian Academy of Sciences, Russia
- Shuichiro Yamamoto, NTT Data, Japan
- Seiichi Komiya, Shibaura Institute of Technology, Japan
- Ivan Rozman, University of Maribor, Slovenia

Program Committee Co-chairs

Enn Tyugu, Institute of Cybernetics, Tallinn University of Technology, Estonia
Takahira Yamaguchi, Keio University, Japan

Program Committee

- Tomohiro Aduma, NTT Data, Japan
- Christo Dichev, Winston-Salem State University, USA
- Darina Dicheva, Winston-Salem State University, USA
- Yoshiaki Fukazawa, Waseda University, Japan
- Naoki Fukuta, Shizuoka University, Japan
- Viktor Gladun, Institute for Cybernetics, Ukraine
- Hele-Mai Haav, Institute of Cybernetics at TUT, Estonia
- Masa-aki Hashimoto, Kyushu Institute of Technology, Japan
- Toyohiko Hirota, Kyushu Sangyo University
- Tadashi Iijima, Keio University, Japan
- Kenji Kaijiri, Shinshu University, Japan
- Yoshitsugu Kakemoto, The Japan Research Institute
- Shigeo Kaneda, Doshisha University, Japan
- Vladimir Khoroshevski, Computing Centre, Russian Academy of Sciences, Russia
- Mare Koit, University of Tartu, Estonia
- Seiichi Komiya, Shibaura Institute of Technology, Japan
- Teruo Koyama, National Institute of Informatics, Japan
- Mihail Matskin, KTH, Sweden
- Masaru Nakagawa, Wakayama University, Japan
- Pavol Navrat, Slovak University of Technology, Slovakia
- Toshio Okamoto, University of Electro-Communications, Japan

- Gennadii Osipov, Russian Academy of Sciences, Russia
- Ivan Rozman, University of Maribor, Slovenia
- Vadim L. Stefanuk, IITP, Russian Academy of Sciences, Russia
- Valery Tarasov, Bauman State Technical Institute, Russia
- Kuldar Taveter, University of Melbourne, Australia
- Haruki Ueno, National Institute of Informatics, Japan
- Vadim Vagin, Moscow Energy Institute, Russia
- Hiroyuki Yamada, Ehime University, Japan
- Shuichiro Yamamoto, NTT Data, Japan

Local Arrangements

- Vahur Kotkas (local arrangements chair)
- Riina Maigre
- Monika Perkmann

Sponsors

Estonian Academy of Sciences

AS Cell Network

This page intentionally left blank

Contents

Foreword <i>Enn Tyugu and Takahira Yamaguchi</i>	v
Conference Organization	vi
1. Software Engineering	
Automatic Generation Method of Relation Matrix for ISM <i>Nobuaki Kanazawa, Toshiaki Sasahira, Shigeo Kaneda and Hirohide Haga</i>	3
Intelligent Help for Managing and Training UML Software Engineering Teams <i>Maria Virvou and Kalliopi Tourtoglou</i>	11
SEEC – A Software Error Estimation Method for Multi-Component Software Projects <i>Timo Kuusela and Tuomas Mäkilä</i>	21
Support Method for Source Code Modification in Changing GUI Widgets <i>Junko Shirogane, Kazuhiro Fukaya and Yoshiaki Fukazawa</i>	31
A Method for Extracting Unexpected Scenarios of Embedded Systems <i>Toshiro Mise, Yasufumi Shinyashiki, Takako Nakatani, Naoyasu Ubayashi, Keiichi Katamine and Masaaki Hashimoto</i>	41
Automatic Creation of a Countermeasure Plan Against Process Delay Based on Fast-Tracking <i>Rihito Yaegashi, Daisuke Kinoshita, Yuichiro Hayashi, Keiichi Nakamura, Hiroaki Hashiura and Seiichi Komiya</i>	51
Detecting Defects in Object Oriented Designs Using Design Metrics <i>Munkhnasan Choinzon and Yoshikazu Ueda</i>	61
Exploring an Open Source Data Mining Environment for Software Product Quality Decision Making <i>Houria Yazid and Hakim Lounis</i>	73
Deep Semantics of Visual Languages <i>Pavel Grigorenko and Enn Tyugu</i>	83
Integrated System Analysis Environment for the Continuous Consistency and Completeness Checking <i>Erki Eessaar</i>	96
Defect Rate Profile in Large Software-Systems <i>Elena-Ramona Modroiu and Ina Schieferdecker</i>	106

Information Flow Diagram and Analysis Method for Unexpected Obstacle Specification of Embedded Software <i>Hidehiro Kametani, Yasufumi Shinyashiki, Toshiro Mise, Masa-aki Hashimoto, Naoyasu Ubayashi, Keiichi Katamine and Takako Nakatani</i>	115
A Generation Method of Exceptional Scenarios from a Normal Scenario <i>Atsushi Ohnishi</i>	125
An Effect of Comment Statements on Source Code Corrective Maintenance <i>Hirohisa Aman, Hirokazu Okazaki and Hiroyuki Yamada</i>	135
Catalogue of Design Patterns <i>L'ubomír Majtás</i>	139
Reuse of Patterns' Source Code <i>Jaroslav Jakubík and Pavol Návrat</i>	143
Goal Algebra for IT Alignment <i>Shuichiro Yamamoto</i>	147
Business Process Analysis Using Reference Model <i>Shinobu Saito and Shuichiro Yamamoto</i>	151
2. Semantic Web	
Re-Engineering OntoSem Ontology Towards OWL DL Compliance <i>Guntis Barzdins, Normunds Gružītis and Renārs Kudīns</i>	157
Searching over Public Administration Legal Documents Using Ontologies <i>Diego Berrueta, Jose Emilio Labra and Luis Polo</i>	167
Semantics Driven Development of Software Systems Based on Business Ontologies <i>Keiichi Kondo, Shogo Hoshii, Takeshi Morita, Takahira Yamaguchi, Noriaki Izumi and Kōiti Hasida</i>	176
Description of Temporal Constraints Using Semantic Web in Role-Based Access Control <i>Kazushi Tanihira, Yusuke Sakamoto and Hiromi Kobayashi</i>	186
Development and Analysis of a Problem Domain Knowledge Base Oriented to PLA Specifications <i>Henrikas Pranėvicius and Germanas Budnikas</i>	196
Extracting Opinions Relating to Consumer Electronic Goods from Web Pages <i>Taichi Nakamura and Hiroshi Maruyama</i>	206
3. Intelligent Agents	
A Correlation-Driven Reactive Layer for Situation-Aware BDI Agent Architecture <i>Gabriel Jakobson, John Buford and Lundy Lewis</i>	213

Building Agent-Based Appliances with Complementary Methodologies <i>Leon Sterling, Kuldar Taveter and The Daedalus Team</i>	223
AOEX: An Agent-Based Exception Handling Framework for Building Reliable, Distributed, Open Software Systems <i>Susan Entwistle, Seng Loke, Shonali Krishnaswamy and Elizabeth Kendall</i>	233
Agent-Based Modeling and Simulation of Network Softbots' Competition <i>Igor Kotenko and Alexander Ulanov</i>	243
Flexible and Efficient Use of Robot Resources Using Higher-Order Mobile Agents <i>Mayu Mizuno, Masato Kurio, Munehiro Takimoto and Yasushi Kambayashi</i>	253
System Support of Diabetic Self-Treatments Using Mobile Phones <i>Takuya Yoshihiro, Kazuhiro Ikemoto, Kumiko Mori, Sachiko Kagawa, Yasuhisa Yamamoto and Masaru Nakagawa</i>	263
4. Special Techniques and Applications	
Parallel Preprocessing for Classification Problems in Knowledge Discovery Systems <i>N.R. Akchurina and V.N. Vagin</i>	275
Categories for Description of Dynamic Production Systems <i>Vadim Stefanuk and Alexander Zhozhikashvili</i>	285
Linguistic Knowledge for Search Relevance Improvement <i>Gennady Osipov, Ivan Smirnov, Ilya Tikhomirov, Olga Vybornova and Olga Zavalova</i>	294
Reasoning by Structural Analogy in Intelligent Decision Support Systems <i>A.P. Eremeev and P.R. Varshavsky</i>	303
The Heuristic Methods of Obtaining the Effective Measurement in Diagnostic Systems <i>V.N. Vagin and P.V. Oskin</i>	307
An Advancement-Record System Using Knowledge Adaptable to Children's Developmental Stages <i>Kimio Shintani, Aki Kono, Masaya Asano, Hirohide Haga and Shigeo Kaneda</i>	317
A Database System of Buddhist Canons <i>Takehiko Tanaka, Yohei Nino, Rong Zhang, Martin Rolland, Masaru Nakagawa, Susumu Aoki, Keigo Utsunomiya and Toshinori Ochiai</i>	327
Author Index	337

This page intentionally left blank

1. Software Engineering

This page intentionally left blank

Automatic generation method of relation matrix for ISM

Nobuaki Kanazawa¹⁾, Toshiaki Sasahira²⁾, Shigeo Kaneda¹⁾²⁾, and Hirohide Haga¹⁾

¹⁾ Graduate School of Engineering, Doshisha University

²⁾ Graduate School of Policy and Management, Doshisha University

Abstract: This article proposes an automatic generation method of a relation matrix for Interpretive Structural Modelling (ISM). In ISM, keywords related to the problem are selected. Then the relation between two arbitrary keyword pairs is examined. When there is a relation between these two keywords, a non-zero value such as 1 is assigned to this relation. After examining the relations of all pairs of keywords, an $n \times n$ relation matrix is generated where n stands for the number of keywords. Then a reachable matrix is calculated from the relation matrix. By using the reachable matrix, the hierarchy structure of the problem is constructed. Building a relation matrix requires much time and labour. This article proposes an automatic generation method of relation matrix based on the co-occurrence probability of keywords in primary materials. As the proposed method is fully automatic, less labour is necessary than in the original ISM method. This method is evaluated by using newspaper articles.

Keywords. ISM, problem structuring, visualisation, automatic generation, co-occurrence probability, full text search

1. Introduction

This article proposes an automatic generation method of relation matrix for Interpretive Structural Modelling (ISM), which is a popular tool widely used for visualising problem structure, problem structure analysis, and decision making. In ISM, problems are divided into sub problems. Then hierarchy relations are introduced into the set of sub problems. The hierarchy structure is represented by a directed graph (ISM diagram). Users can clearly understand the hidden problem structures by browsing the ISM diagram.

Drawing an ISM diagram is time-consuming. First, keywords related to the problem are selected, and then the relation between an arbitrary pair of two keywords is identified. If there is a relation between the selected two keywords, the existence of the relation is confirmed. After examining all pairs of any two keywords, all relations are represented as an $n \times n$ matrix called a *relation matrix*. Next a reachable matrix is calculated from the relation matrix. By using the reachable matrix, a hierarchy structure is drawn. After fixing the relation matrix, all calculation will automatically be performed. But fixing the relation matrix, especially identifying the relation between any two keywords, requires time and labour.

In this article, we propose an automatic generation method of the relation between two keywords. First keywords are selected. Then original materials such as newspaper sources are scanned to extract articles that include at least one keyword. Then the

number of articles that include any two keywords is counted. Next the co-occurrence probability is calculated, and all probabilities are arranged in a matrix form. Then the average value of co-occurrence probability is calculated for each keyword, and the relation is evaluated based on comparisons of average probability and co-occurrence probability. It is assumed that if co-occurrence probability is larger than average probability, these two keywords have a relatively strong relation.

By using the above criteria, the relation of any pair of two keywords is evaluated, and the result becomes a relation matrix from which an ISM diagram will automatically be generated. As our proposed method is fully automatic, an ISM diagram is automatically generated from the original materials. In section 2, a brief introduction and two ISM problems are described. Section 3 shows the details of the proposed algorithm, and example usage and evaluation of the proposed method are described in section 4.

2. Brief description of ISM and its problems

2.1 Brief description of ISM

Interpretive Structural Modelling (ISM) is a method for visualising the hierarchy structure of elements. It was first introduced by J. Warfield [1] for modelling the relationship between many elements involved in a complex situation. In ISM, all elements are arranged hierarchically in a structure constructed from the dependency of two elements. ISM was first introduced for the visualisation of the summarisation of brainstorming. Now, it is widely used for the visualisation of the complex structure of problems in any field.

In ISM, elements (usually represented as keywords) are extracted through brainstorming or other methods. Then the dependency between two elements is evaluated. When dependency is approved, a non-zero value such as 1 is assigned to the relation. By arranging all elements in a matrix form, a relation matrix can be generated in which the topmost row and the leftmost column include all elements. 0 or 1 is assigned to all cells in accordance with the existence of dependency. After the identification of a relation matrix, a reachable matrix is calculated in the following manner:

- (1) Let A be a relation matrix. Then $n \times n$ unit matrix I is added to A . Let P be $A+I$.
- (2) Then the production operation is iterated until $P^{n+1}=P^n$ is satisfied where P^n denotes the product of matrix P n -times.
- (3) Then we can get reachable matrix $R=P^n$.

By using reachable matrix R , a hierarchy diagram is drawn.

2.2 Problems of ISM diagram generation

After determining relation matrix A , reachable matrix R is generated algorithmically, and then the diagram is drawn automatically. However, the definition of dependency between two elements consumes time and labour. As the number of elements increases, time and labour costs also increase in an exponential order. Furthermore, objectivity is another problem for identifying dependency. Since only a few users participate in identifying the dependency, its evaluation is heavily influenced by participants.

To resolve these two problems, we propose an automatic generation method of a relation matrix in which a computer performs these tasks.

3. Automatic generation method of a relation matrix

3.1 Overview

The automatic generation of a relation matrix is performed as follows:

1. Participants identify keywords that represent the problem.
2. Primary materials are fully text scanned, and materials that include at least one keyword are extracted from the material set.
3. Extracted materials are scanned again, and materials that include any pair of two keywords are extracted.
4. Based on the number of materials, the occurrence probability of all keywords and all pairs of two keywords are calculated.
5. Based on occurrence probability, the relation of any two keywords is determined.

As the above steps are performed by computer after identifying keywords, objectivity is secured. Furthermore, the problem of time and labour cost is also resolved. In our experiment, newspaper archives are used to evaluate this method.

3.2 Details of the automatic generation of a relation matrix

Let w_1, w_2, \dots, w_n be the determined keywords of a certain problem. First, all newspaper articles are lexically analysed and divided into a set of morphemes. Then a full text scanning operation finds articles that include at least one keyword w_i . Let n_i be the number of articles that include keywords w_i .

Then select other keywords w_j and find articles that include both w_i and w_j . After that, the number of articles that include both w_i and w_j are counted. Let n_{ij} be the number of articles that include both w_i and w_j . After that, following co-occurrence probability p_{ij} among two keywords is calculated:

$$p_{ij} = n_{ij} / (n_i + n_j - n_{ij}).$$

Next all probabilities are arranged as a table, as shown in Figure 1.

	w_1	w_2	w_n	Average
w_1		p_{12}		p_{1n}	pr_1
w_2	p_{21}			p_{2n}	Pr_2
:
w_n	p_{n1}	p_{n2}			pr_n
Average	pc_1	pc_2	pc_n	

Figure 1. Co-occurrence probability matrix

Then the average probability of each row and column is calculated. Let pr_i be the average probability of row i and pc_j be the average probability of column j . Then dependency can be identified, as in the following criterion:

If $p_{ij} > pr_i$, then it is assumed that there is a strong relation between w_i and w_j . Therefore, the value of 1 will be assigned to cell (i, j) . Otherwise, the value of cell (i, j) is 0.

This is because word w_j is more frequently used with word w_i than other words w_1, \dots, w_{i-1} and w_{i+1}, \dots, w_n . It means there is a relatively stronger relation between w_i and w_j than other words. Then we can get the 0/1 value matrix. But this matrix does not represent the dependency direction. To draw an ISM hierarchy diagram, we have to determine the dependency direction of any pair of keywords. Next we evaluate the dependency direction of w_i and w_j .

Dependency will be evaluated as follows:

- If $p_{ij} > pr_i$ and $p_{ji} < pc_j$, then there is a dependency where w_i depends on w_j , and we assign value 1 to cell (i, j) and 0 to cell (j, i) .
- If $p_{ij} > pr_i$ and $p_{ji} > pc_j$, then there is a dual dependency between w_i and w_j . Therefore, value 1 is assigned to cell (i, j) and cell (j, i) .

This is because w_i is used in the article where w_j is used more than in the article where another word such as w_k is used. On the other hand, w_j uses other words more than w_i . In the article, w_j refers to other words more than w_i . Word w_j does not depend on w_i . But w_i depends on w_j . Therefore, we consider w_j a source and w_i a destination of this relation. In other words, w_j is a cause, and w_i is a result of this relation.

By using the above two criteria, we can get the relation matrix. Once the relation matrix is obtained, an ISM hierarchy diagram can be drawn automatically. As this procedure only requires small labour and time costs, we can apply it to a wide range of problem areas, including diplomatic, technical, and everyday life fields. In the next section, we describe one example of our experiment from a societal information development system.

4. Experimental results of the proposed algorithm

4.1 Example of proposed algorithm

The experiment was conducted for “Information Systems in Society” using¹ the following 15 keywords: efficiency, cost, online, real-time, service, personal computer, word processor, network, internet, mobile phone, communication, personal information, safety, and reliability.

After applying the proposed algorithm, we drew an ISM diagram with which we evaluated the proposed algorithm’s validity.

4.2 Evaluation of algorithm

To evaluate the validity of the proposed algorithm, the long-range changes of ISM diagrams about “Information Systems in Society” were examined. As the proposed algorithm needs fewer time and labour costs, we generated nine years of changes (from 1996 to 2004) of ISM diagrams about “Information Systems in Society.” In our experiment, articles from the Mainichi newspaper [4] were used as primary materials. The morphological analysis software “ChaSen” [2] was used for lexical analysis. The hierarchy diagram was drawn by “Graphviz” [3]. Both ChaSen and Graphviz are free software. Figure 2 is a result of the diagram of 1996. Figures 3, 4, and 5 show the results of 1999, 2002, and 2004, respectively.

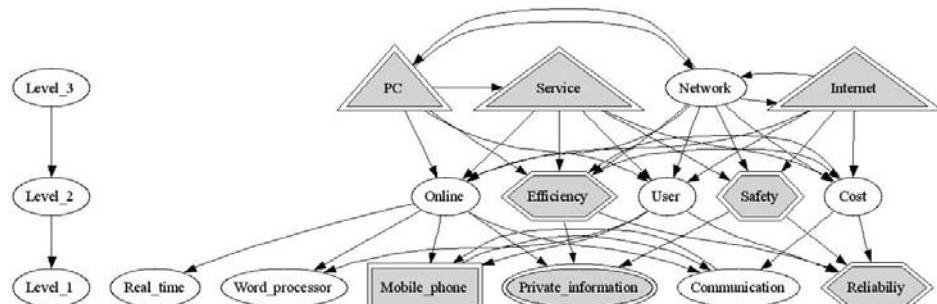


Figure 2. ISM diagram about Information Systems in 1996

¹ Since Japanese newspaper articles were used for the experiment, all keywords were represented in Japanese.

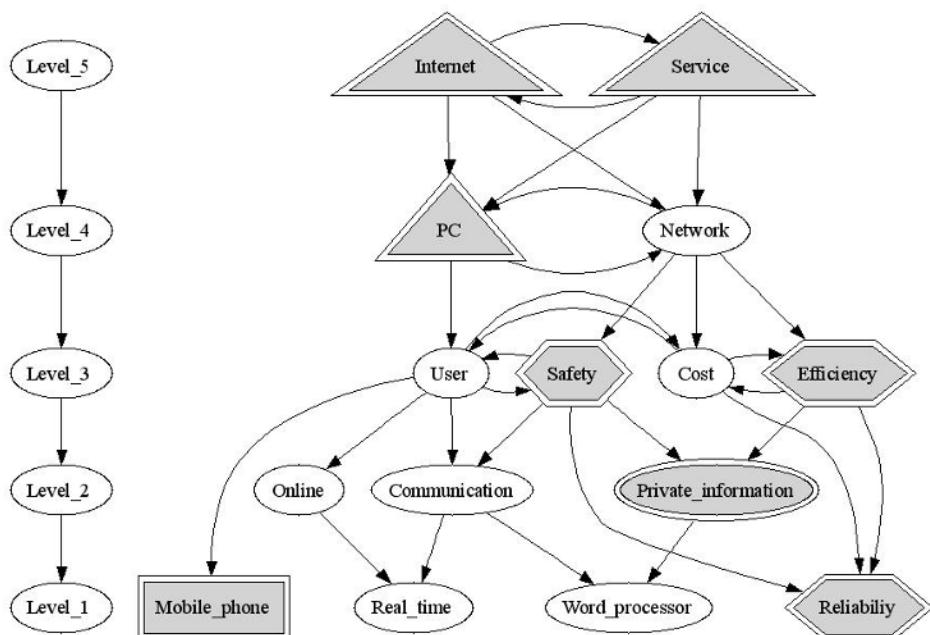


Figure 3. ISM diagram about Information Systems in 1999

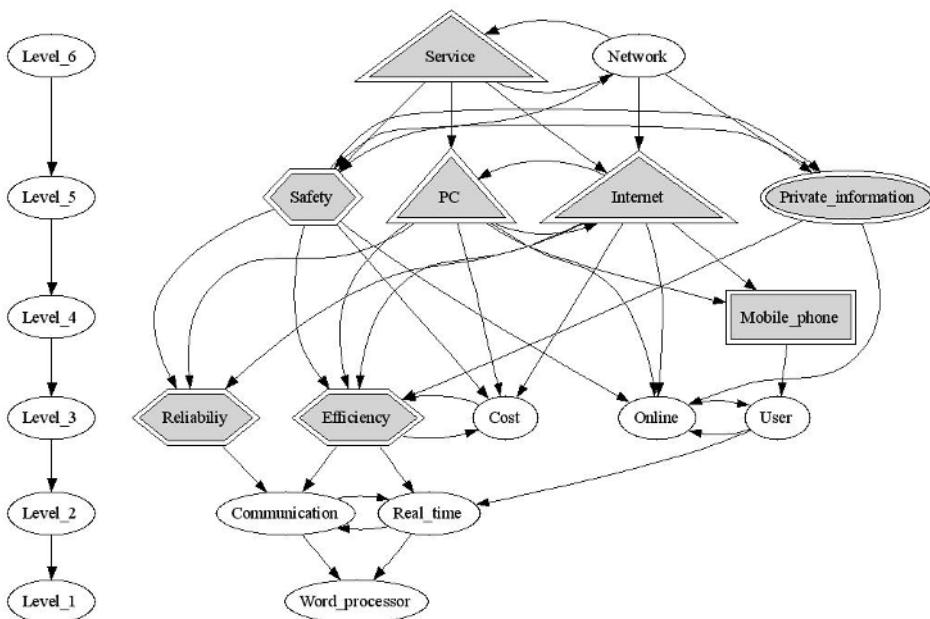


Figure 4. ISM diagram about Information Systems in 2002

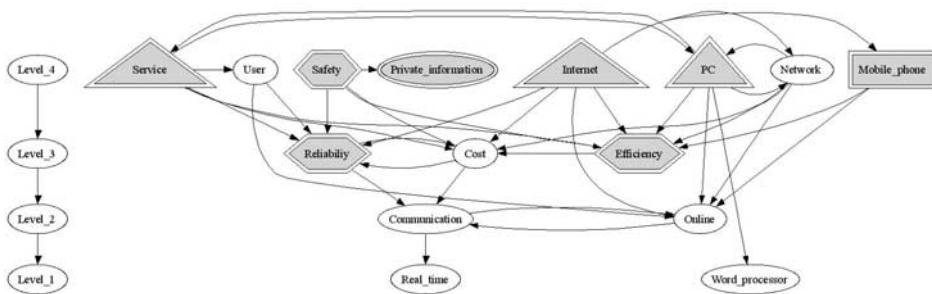


Figure 5. ISM diagram about Information Systems in 2004

From the changes in the diagrams, we can “read” the following trends:

- In 1996, the problem was less structured, as read from a diagram that has shallow hierarchy and where almost all keywords are connected to each other. This means that the structure is not well defined. Any keyword is used with no strong relation with other keywords.
- In 1999, the problem became more structured. The hierarchy has five levels, and clearer dependency can be seen between keywords. The number of links is fewer than 1996.
- In 2002, keyword “private information,” represented as a double line oval in all figures, rose up the diagram hierarchy, meaning that the importance of the word “private information” increased.
- The same trend can be seen for the word “mobile phone” represented as a double line rectangle. In 1996, “mobile phone” was located at the bottom of the diagram. In 2004, however, “mobile phone” is located at the top of the hierarchy. Mobile phone is one key component of Information Systems.
- Three keywords, “personal computer,” “internet,” and “service,” which are represented as double line triangles, are shown in all diagrams. These three components are the essential components of “Information Systems.”
- The change of the locations of “efficiency,” “reliability,” and “safety,” represented by a double line hexagon, is interesting. From 1996 to 1998, the location of “efficiency” was higher than “reliability” and “safety.” But in 1999, “safety” was located at the same level as “efficiency,” and from 2002, the location of “safety” was higher than “efficiency.” The location of “reliability” is also higher than previous years. It means that in society the important role of “Information System” changed from efficiency to safety and reliability.

All trends read from the results of the long-range changes of ISM diagrams seem to fit actual societal changes, reflecting the high reliability of the automatic generation algorithm proposed in this article.

5. Conclusion

In this article, an automatic generation algorithm of the relation matrix for ISM diagram drawing is proposed. This algorithm is based on the co-occurrence probability of two keywords. The proposed algorithm was evaluated by the long-range changes of

problem structure. The field of “Information Systems in Society” was selected for evaluation. An ISM diagram was generated from newspaper articles. Nine years of ISM diagram changes seem to reflect the changes of the actual world, which reflects the validity of the proposed method.

Acknowledgement: This work is supported by the “Synthetic Studies on Technology, Enterprise and Competitiveness” project, carried out under the 21st Century COE (Center of Excellence) Program of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

- [1] Warfield, J. N. (1973). Binary matrices in system modeling, *IEEE Trans. SMC*, Vol3, No.5, pp.441-449
- [2] Matsumoto, Y., et al (2000). Morphological Analysis System ChaSen version 2.2.1 Manual, Nara Institute of Science and Technology, <http://chasen.aist-nara.ac.jp/chasen/doc/chasen-2.2.1.pdf>
- [3] Graphviz - Graph Visualization Software, <http://www.graphviz.org/>
- [4] Mainichi-shinbun <http://www.mainichi.co.jp/>

Intelligent Help for Managing and Training UML Software Engineering Teams

Maria VIRVOU and Kalliopi TOURTOGLOU

Department of Informatics, University of Piraeus,
80 Karaoli & Dimitriou St. Piraeus 18534, Greece
{mvirvou, ktourtog}@unipi.gr

Abstract. In this paper, we present an intelligent help system for novice software engineers who use UML. This intelligent help is targeted to two kinds of user: the individual software engineer and the manager of the system, who coordinates the teams of software engineers. The system is called ISEA and can be used in real productive time even by the novice software engineers of a Software House. The intelligent help is generated based on a user-modelling module that records the actions of software engineers and evaluates them in terms of personality characteristics, their UML knowledge level and their collaboration with other software engineers. At the same time, ISEA supports a Group-Role Module that contributes to the automatic selection of the most effective collaboration schemes.

Keywords. UML, collaboration, cooperation, group, team, role, stereotype, student modelling, user modelling, software engineering, adaptivity, intelligent help.

1. Introduction

The Unified Modeling Language (UML) provides system architects working on object analysis and design with one consistent language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling [1]. The Unified Modeling Language (UML) is the de facto industrial standard of an object-oriented modeling language [2].

Software engineering is an inherently collaborative, team-based process [3]. That is why Software Houses tend to organize their personnel into groups. The tasks are assigned to these groups rather than to individual software engineers. It is rare to find an important task being performed by an individual without any sort of team and organizational coordination [4].

Taking into consideration this vital requirement of software industry, we have developed ISEA (Intelligent Software Engineering Advisor). ISEA is software for

constructing UML diagrams. These UML diagrams can be used as input to generate XML files, database structures and Delphi applications. Additionally, ISEA offers adaptive help to the software engineers and advice for the best cooperation between them.

There are already very effective and powerful professional software tools in the market that allow software engineers to build advanced UML diagrams and export reliable databases and full-functional applications. Our intention was not to implement software that would compete them. The main objective of ISEA concerns the managing of groups with respect to the most effective collaboration between individual software engineers as well as between teams of software engineers.

As UML is a quite new language, it is likely that even experienced software engineers have not yet had the opportunity to learn UML profoundly. On the other hand, the time that a software house can afford to train its employees is often limited. The question of how employees can be efficiently trained is considered important [5]. Therefore, a training and simultaneously productive software system for UML could be useful to software houses that intend to use UML.

There are some efficient preliminary approaches to support individual and collaborative learning of UML class diagrams. For example, COLLECT-UML [6] is an Intelligent Tutoring System that supports individual and collaborative learning of UML class diagrams. COLLECT-UML bases its inferences on evaluating the submitted solutions to specific exercises-problems. In contrast, ISEA is constantly monitoring the software engineers while they work on partial solutions using UML and collects evidence on their characteristics. Furthermore, there has already been introduced collaborative learning software for other purposes (not UML). For instance, FLE 3 [7] is a distributed collaborative learning environment. The collaboration model that it uses is based mainly on information about the properties of forum-messages posted by the students. On the other hand, the inferences of ISEA are based on information gathered throughout the use, which means every action, of the software engineers during their interaction with the system.

Practically, ISEA is an adaptive hypermedia system. Adaptive hypermedia systems build a model of the goals, preferences and knowledge of each individual user, and use this model throughout the interaction with the user, in order to adapt to the needs of that user [8]. Education is one of the most promising areas for adaptive hypermedia. Hypermedia is used in educational systems for providing student-driven exploration of educational material [9]. In particular, ISEA is an adaptive educational hypermedia, as it is mainly focused on the learning process of the software engineers that use it. The learning process in Adaptive Educational Hypermedia (AEH) environments is complex and may be influenced by aspects of the student, including prior knowledge, learning styles, experience and preferences [10].

In order to trace the learning process of the users, ISEA incorporates a user modelling component that is called User-Modeller. A technique for building user models is the use of stereotype-based reasoning. The implemented user model of ISEA is based on a double stereotype about the level of knowledge and the performance type of users. In addition, the user model takes into account personality features of user. A stereotype represents a collection of attributes that often co-occur in people [11].

2. Overview of the System

2.1. Operation of ISEA

As already mentioned ISEA is software for constructing UML diagrams, which can be used to export XML files, a database structure and a Delphi application. The main form of ISEA is illustrated in Figure 1. In the main form of ISEA, there are three toolbars: the standard, the project and the component toolbar. The standard toolbar contains the buttons for creating a new UML diagram, opening an existing one, saving and clearing the current UML diagram. The project toolbar has three buttons, which respectively have the functionality to export an XML file, a Delphi application and a database structure. Finally, the component toolbar includes buttons for the construction of the UML diagrams. These correspond to adding a UML icon, such as a UML class, a note, a composition etc. In Figure 1, under the toolbars, there are a horizontal and a vertical ruler perimetric the workspace. The workspace is the main area for the software engineer to design a UML diagram. The lowest part of the main form is a status bar, where additional and explicative information are shown to the user, such as the cursor position, the log data and some of the help messages of ISEA.

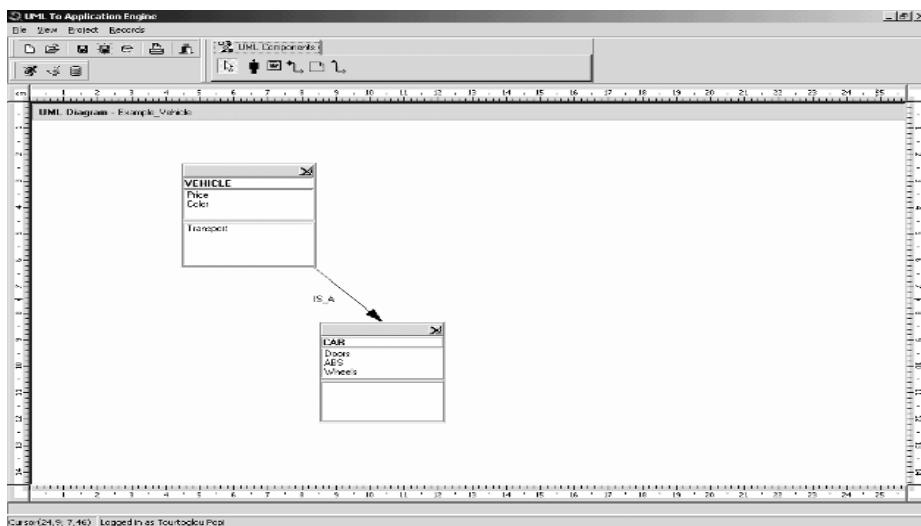


Figure 1. Main form of ISEA

ISEA is an educational and simultaneously productive software system. It is educational as one of its main aims is to guide software engineers to acquire knowledge through individual adaptive help and generalized adaptive advice on the collaboration patterns. These collaboration patterns may concern individuals or teams. On the other hand, ISEA could be considered as productive since its inferences are not based on results in specific tests/problems. The inference mechanism of ISEA is relied on the constant monitoring of the users. This means that software engineers are let to be practically in the productive line while they are being trained.

There are mainly two kinds of users of ISEA: the manager and the software engineer. The software engineers are the employees of a company that are supposed to be trained in constructing UML diagrams. The manager is the person who is assigned to guide the software engineers while they gain new knowledge and assist them to be productive for the company. This productivity is also associated with the effective cooperation and collaboration of the software engineers with each other. A successful collaboration is of vital importance for a company as it can bear a harmonious and thus productive environment. This harmony is related to the strengths and weaknesses of the employees' knowledge and personality characteristics.

The functionality of ISEA is illustrated in a use-case diagram in Figure 2. Every kind of action of the software engineer triggers the User Modeller. An action could be a click on a button, the opening or the closing of a form. The User Modeller is a component that traces a collection of traits of the user. These traits concern the user's UML knowledge, performance type and personality, which constitute the stereotypes used by ISEA and will be explained later. After being triggered, the User Modeller defines the possible values of attributes mentioned above. Then, it requires feedback from the user. This feedback is related to the conclusions that the User Modeller has just drawn. The user's opinion is stored in historical records for further inferences about his/her stereotype. Next, ISEA provides, if necessary, the software engineer with help. This help concerns the software engineer's progress on UML. The offered help is user-adaptive and is given to the users by means of messages, which can be either informative or interactive.

At the same time, the Group-Role Module is triggered. ISEA supports group-based work. The manager is entrusted with certain tasks associated with this group-based structure, which are performed with the contribution of the Group-Role Module. In particular, the manager should define roles, assign roles to the users, appoint types of groups in relation with roles and, finally, arrange specific teams. These teams are groups of software engineers based on correspondent types of groups. The Group-Role Module evaluates the User Modeller inferences with the aim to offer advice to the manager. This advice is related to the most effective collaboration arrangements that can be achieved between the individual software engineers and their teams. Advice can also concern an evaluation outcome of the structure of the defined by the manager teams. Another task of Group-Role Module is the continuous inspection of whether the users fit to the UML knowledge standards they are assigned through their role attribute.

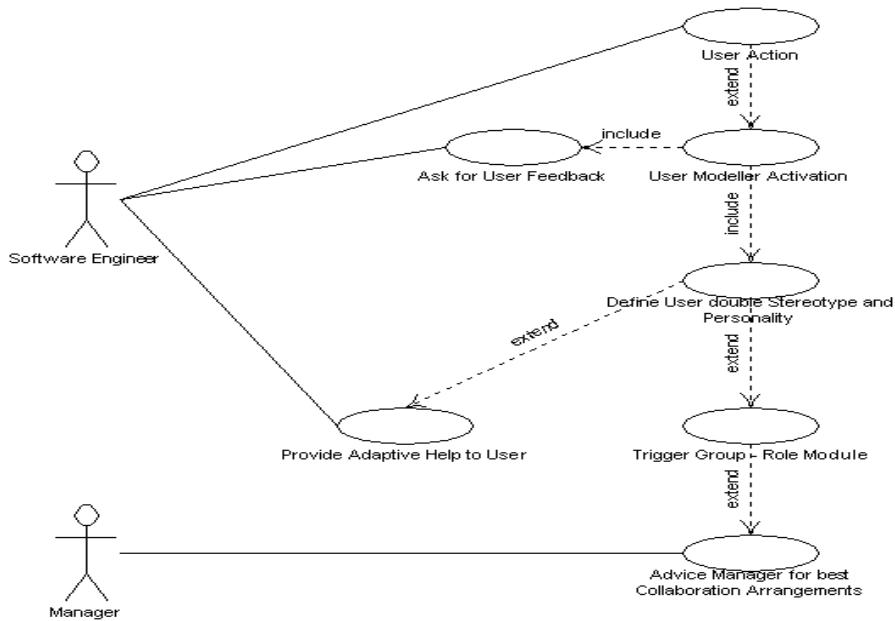


Figure 2. Use-case diagram of ISEA

2.2. Architecture of ISEA

The Architecture of ISEA, which is illustrated in Figure 3, is based on the User Modeller, the Help, the Group-Role and the Collaboration Modules. The main actors are the user and the manager.

ISEA includes a logging subsystem, which is responsible for fully recording the loggings of users. This recording includes user and environment data, such as the user name and the date-time stamp of logging. It should be pointed out that not only the active logging of each user is kept in ISEA database, but all of them in the system's lifetime. Keeping and processing historical records of users' loggings is not a trivial task for the system. These records are of valuable usefulness for the User Modeller.

Every action of the user triggers the User Modeller. Usually, the start point of the elaborations in user modelling systems is the submissions of a solution to given problems/exercises [6]. Some typical activities that ISEA considers as actions are the mouse and keyboard events, the activation/deactivation of forms, the drawing operation on the workspace and even the idleness of the user. The system includes a silent chronometer mechanism that records the time passed between actual user actions. This is used by the User Modeller to draw conclusions about the performance type and personality. Thus, even the idleness of the user could be regarded as an action and, therefore under certain circumstances, it may trigger the User Modeller. Moreover, every action (except from idleness) is possible to have an incorrect outcome. These mistakes are recorded in ISEA Database for the User Modeller to evaluate.

The triggered User Modeller defines the double stereotype and the personality of the software engineer. The double stereotype consists of the level of expertise and the performance type. In brief, the level of expertise refers to the UML knowledge the software engineer has and the performance type states the typical manner that s/he has

throughout the usage of the program. Personality refers to some typical characteristics of the users that concern mainly the process of acquisition and sharing of knowledge. All of these attributes are explained in the next section.

After requesting and collecting user feedback, the User Modeller activates the Help and the Group-Role Modules. Then, the Group-Role Module activates the Collaboration Module. The Help Module is applied directly to the software engineers and the Collaboration Module to the manager. Then, the manager can develop the results and share them with the software engineers.

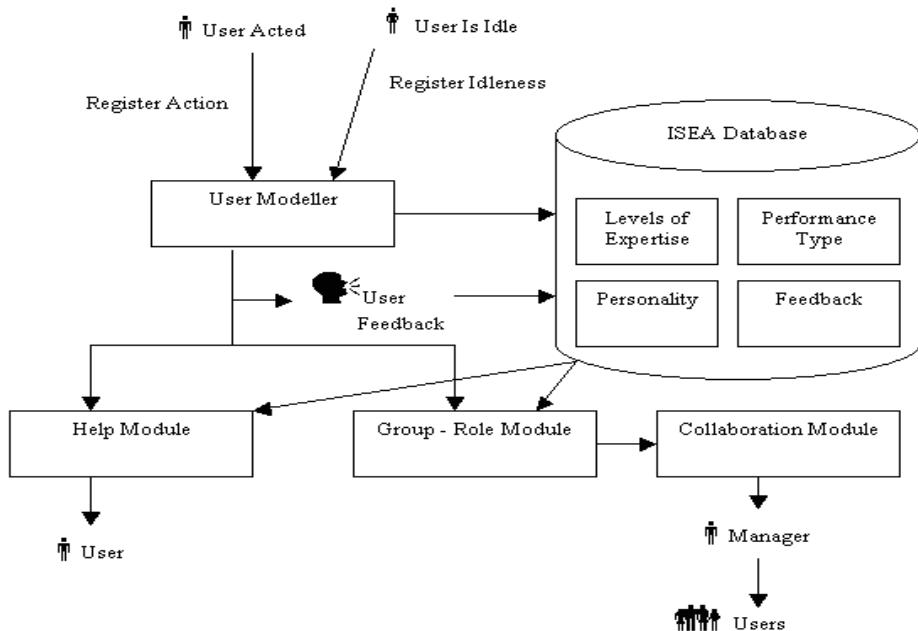


Figure 3. Architecture of ISEA

3. User Modelling Stereotypes

The implementation of the User Modeller is stereotype based. There are no specific stereotypes, but combinations between a two-dimensional stereotype and a set of personality traits. The two-dimensional stereotype includes the levels of expertise and the performance types.

A level of expertise represents the extent of the software engineer's knowledge on UML and ability to construct a correct UML diagram. The UML knowledge is stored in ISEA database, which means that the manager can update it. This reflects flexibility in the maintenance of such a crucial element. The UML knowledge is stored in a hierarchical structure of UML concepts. The UML concepts are the principles of constructing UML diagrams, such as Class Definition, Attributes Definition and Inheritance. The levels of expertise used by the system are No Knowledge, Basics, Junior, Senior and Expert. Each level of expertise is linked to some specific UML concepts. This link represents the prerequisites on UML knowledge for a software engineer of level of expertise classifications. Additionally, there is a degree of

knowledge upon each UML concept. For example, the Basics Level of Expertise can be linked to the Attributes Definition with degree 1. The fact that the structure of UML concepts is hierarchical contributes to the inference rules of ISEA stereotype-based user modelling. An example of inference about the UML knowledge and thus the level of expertise stereotype is: knows (Attributes) → knows (UML Class), which means that if a software engineer knows how to define attributes, ISEA infers that s/he knows how to define a UML class.

Performance types are the main typical manners of software engineers when using a program. The predefined performance types are: Sceptical, Hurried, Slip-prone, Unconcentrated and Efficient. Sceptical refers to the user who frequently lets the program idle for a noticeable amount of time, in his/her effort to complete a task. Hurried refers to a user who completes tasks quite quickly but not absolutely successfully. A user is characterized as Slip-prone when although s/he has the essential UML knowledge to complete a task successfully, s/he makes mistakes due to carelessness or imprudence. Unconcentrated is a user who seems to be sceptical (delays to act), but actually cannot be productive in UML probably because of lack of ability to concentrate. Efficient is the software engineer who seems to accomplish successfully a task at a reasonable speed. ISEA calculates specific parameters in order to conclude on the performance type value of a software engineer. These parameters are the duration until a task is completed, the UML knowledge of the user and the experience of ISEA about his/her behavior. It is not always clear if a software engineer really should be associated with a specific performance type. Hence, the performance type values are not boolean (true/false). They can get three kinds of values: no, yes, could be. Furthermore, there are some restrictions about performance types. For example, if the user is “Sceptical”, then s/he can be neither “Hurried” nor “Unconcentrated”, but can be “Effective” as well.

Personality refers to some typical characteristics of the software engineers that influence their behavior in relation with the acquisition, the usage and the sharing of knowledge. The personality features used by the system are the level of self-confidence, the leading potential, the diligence and the willingness to help others. The level of self-confidence is what the user believes about him/herself as compared with the reality that ISEA has assumed. For example, a user appears to have low confidence but has been classified by the system as an efficient user. The leading potential refers to the ability of the user to manage a group. For example, a person who has leading potential should be able to be influential within the group and, at the same time, preserve a harmonious relationship between them. The diligence measures the amount of work that the user is willing to take. The willingness to help others is the eagerness of the software engineer to cooperate with others and help them even if they do not belong in the same team.

Every stereotype should have a set of trigger conditions and a set of retraction conditions. Trigger conditions are the prerequisites of a user to belong to a stereotype. Retraction conditions are the prerequisites of a user to be withdrawn from a stereotype. For example, a trigger condition of a user to belong to the level of expertise stereotype “junior”, could be the possession of knowledge on attributes definition and the non-possession of any other knowledge. A retracting condition of a user not to belong any more to the same stereotype (“junior”) could be the possession of absolutely no knowledge or the possession of knowledge on attributes, methods and inheritance definition.

4. Group-Role Module

The Group-Role Module supports the potential collaboration of software engineers. The Group-Role Module does not use group modelling. Instead, it uses as input the inferences of the User Modeller, the user stereotypes and some specific definitions of the manager concerning the structure of teams and groups. The previously discussed attribute of personality is the basis for inferences of the Group-Role Module.

For this module to function, the manager should make some definitions. The manager's group-role tasks are illustrated in Figure 4 as a use-case diagram. First of all, the manager defines the roles of users. A role is an attribute of the software engineer, which describes the assignment s/he has in the system. For example, some roles could be: project manager, junior, senior and expert developer. The concept of a role is associated with the knowledge on UML. This means that the manager has to define what parts of knowledge and in what degree must every role have. For example, a senior developer should have the knowledge of defining UML classes in degree 3 (considering that the maximum degree is 3). Then, the manager has to define the roles of software engineers that is to assign every user with a role. Next, the manager can define the groups of roles, which are actually sets of roles characterized by an amount. For example, a group of roles could consist of one project manager, two junior developers and one senior developer. The definition of teams is the last step that the manager should make. A team is an entity of a group of role and consists of software engineers with certain roles that can work together and succeed in a common target. For example, a team could be consisted of Maria (who is a project manager), John, George (who are junior developers) and Anna (who is a senior developer). During the formation of teams and groups, the manager should take into consideration the software engineers' qualifications and how they can cooperate together to achieve a target. Teams can consist of other sub teams (hierarchical structure).

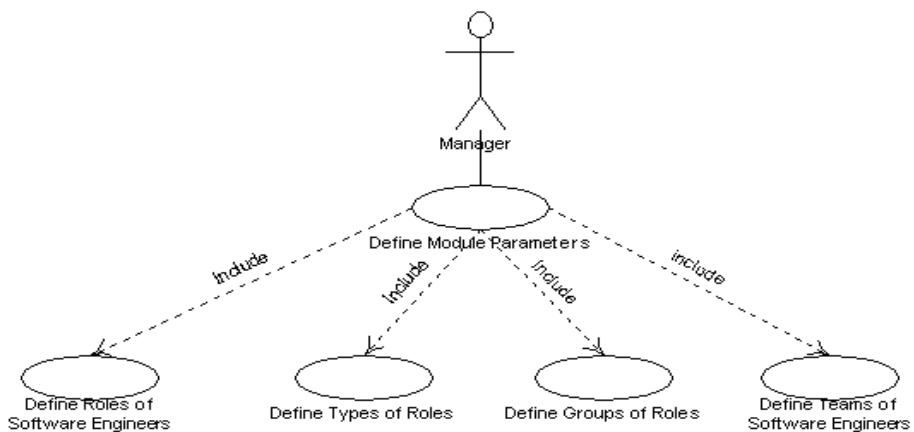


Figure 4. Use-case diagram of the Manager's Group-Role Tasks

This data is used by the Group-Role Module in order to advise the manager on the most effective ways that software engineers could collaborate and cooperate. Moreover, the Group-Role Module is constantly checking the appropriateness of the users for their assigned role. In other words, it supervises if the software engineers fulfill the

conditions of their assigned role. Then, the manager is informed, as s/he may need to reconsider the roles and teams.

5. Intelligent Help

The main objective of ISEA is to provide the users with intelligent help and for that it includes the Help and the Collaboration Modules. The intelligent help consists of help related to the educational progress and advice on the team construction and collaboration of the software engineers. The Help Module is responsible for help given to software engineers and the Collaboration Module for the advice given to the manager.

Help is user adaptive and is offered to the software engineers in the form of informative and interactive messages. Informative messages are adaptive in terms of their presentation. In ISEA, depending on the stereotype of the user, informative messages either appear in a separate popup window or they appear in the status bar. For example, if a user is found to be sceptical, the informative message will appear on the status bar, as it is unlikely not to be noticed by the user. On the contrary, if a user seems to be hurried, the informative message will appear as a modal popup window to ensure that s/he will read it.

Interactive messages are windows that contain a help context and, additionally, request the opinion of the user on it. This feedback is stored in ISEA database for the User Modeller to evaluate it. For example, ISEA has remarked that a software engineer, who evidently knows how to create a UML class and the average time needed for that is five minutes, the last thirty minutes s/he has made four mistakes during his/her effort to define two new UML classes. Furthermore, ISEA has noticed meaningless movements of the mouse and opening – closing of the same form for five times. Therefore, ISEA concludes that the particular software engineer is unconcentrated. This means that the software engineer has the necessary knowledge, but s/he cannot concentrate in order to implement it. This may happen due to a temporary bad psychological, intellectual or health state, a noisy environment or even a concentration problem. Hence, before the system decides which of these reasons causes the user to be unconcentrated, it asks for his/her opinion: “ISEA believes that you have a concentration problem. Do you consider it as an accidental/temporary phenomenon?”. If at the first time the user answers “Yes”, which means that s/he considers it as a temporary phenomenon or a mistaken presumption of the system, then ISEA will not haste to characterize him/her as unconcentrated. However, it will register this estimation and the answer of the user in the database and wait until it has more feedback. This registration will be very useful as in the course of time, in a similar situation, the answer of the user will have a different importance depending on his/her sincerity. Proceeding with this example, if ISEA notices for one more time the same phenomenon, asks the user the same question and his/her answer is positive again, then it will estimate that s/he is probably slip-prone and that s/he cannot evaluate successfully his/her efficiency.

The Collaboration Module produces advice for the manager. This advice is oriented to the ways the manager could achieve the most effective collaboration schemes between software engineers. The Collaboration Module evaluates the stereotypes of the individual software engineers and the formed structure of groups and teams in order to blend their strengths and weaknesses.

6. Conclusions

The coordination of a large number of software engineers is a very complex task for a human manager. This is even more the case when there are many novice software engineers among them. This task would be even harder if the software engineers should urgently be incorporated in the productive process of a software house. The manager would find helpful to have a software system that could provide the functionality and user interface of a case tool and simultaneously provide support to him/her and to the software engineers. To meet these requirements, we introduced ISEA, which is a software tool for constructing UML diagrams and at the same time support the manager and the software engineers adaptively. ISEA is based on a user-modelling component implemented with stereotypes. These stereotypes describe the UML knowledge, the performance type and the personality of the software engineers. ISEA evaluates every action of the user in the system, including the collected user feedback. ISEA also contains the Group-Role, the Help and the Collaboration Modules. All of these modules cooperate envisaging assisting the human factor to manage harmoniously and effectively teams of software engineers.

7. References

- [1] OMG Unified Modeling Language Specification, Object Management Group (1999).
- [2] Gregor Engels, Reiko Heckel and Stefan Sauer: "UML - A Universal Modeling Language?", Lecture Notes in Computer Science, Vol. 1825, p. 24 (2000).
- [3] Vesssey I and Sravanapudi A P: "CASE tools as collaborative support technologies", Comms ACM, 38, No 1, pp 83-95 (1995).
- [4] Gary Klein: "The strengths and limitations of teams for detecting problems", Cognition, Technology & Work, pp. 1 – 10 (2006).
- [5] Ron J.C.M. Salden, Fred Paas and Jeroen J.G. van Merriënboer: "A comparison of approaches to learning task selection in the training of complex cognitive skills", Computers in Human Behavior Elsevier Science, Vol. 22, pp 321-333 (2006).
- [6] Nilufar Baghaei and Antonija Mitrovic: "COLLECT-UML: Supporting Individual and Collaborative Learning of UML Class Diagrams in a Constraint-Based Intelligent Tutoring System", Lecture Notes in Computer Science, Vol. 3684, pp. 458 – 464 (2005).
- [7] Weiqin Chen: "Supporting teachers' intervention in collaborative knowledge building", Journal of Network and Computer Applications Elsevier Science, Vol. 29, pp. 200-215 (2006).
- [8] Peter Brusilovsky: "Adaptive Hypermedia", User Modeling and User-Adapted Interaction Kluwer Academic Publishers, Vol. 11, pp. 87-110 (2001).
- [9] Ian Beaumont and Peter Brusilovsky: "Adaptive Educational Hypermedia: From Ideas to Real Systems", Proceedings of ED-MEDIA 95 World Conference on Educational Multimedia and Hypermedia, pp.93-98 (1995).
- [10] Herman D. Surjono and John R. Maltby: "Adaptive Educational Hypermedia Based on Multiple Student Characteristics", Lecture Notes in Computer Science, Vol. 2783, pp. 442 – 449 (2003).
- [11] Rich, E: "Stereotypes and user modeling", International Journal of Cognitive Science, Vol. 3, pp. 329-354 (1979).

SEEC – A Software Error Estimation Method for Multi-component Software Projects

Timo KUUSELA, Tuomas MÄKILÄ
Department of Information Technology
University of Turku, Finland

Abstract. In this article, some existing error estimation methods are presented and their suitability for modern software development projects is analyzed. Based on this analysis, a new software error estimation method SEEC (Software Error Estimation by Component-wise comparison), specifically designed for multi-site, multi-component software projects, is introduced. The required activities and the mathematical formulation of the new method are presented. Finally, the integration of the SEEC method to a software development process is illustrated with the SPEM process modeling language.

Keywords. Software development, software project planning, software error estimation

Introduction

More and more often software development projects can last years, cost loads of money and require continuous effort of hundreds of people. The projects can be divided in several sites in terms of software process activities and responsibilities or in several parts in terms of software functionality itself. Messing around with a bunch of code by a fistful of talented programmers has turned to large and complex projects that hold enormous resources and bury various different risks.

These large software projects require increasingly planning and management related activities: resource and schedule planning, budgeting, risk management, and software reliability and quality management to mention a few. To be reliable, all the related decision-making has to be based on established practices and techniques. One part of these techniques includes different estimations that are produced to assist the important planning activities and that can be based on, for instance, history data, expert opinions or future indicators.

One crucial, but perhaps a bit underrated, category of estimation is software error estimation. That is, estimating the number and the cumulative rate of software errors in the developed software during the project. The later an error is found in the project, the

more cost it brings to be corrected. It is clear that estimating software defect amounts has everything to do with the required resources in testing, error management and implementation. A reliable total error amount estimate helps the project to be prepared to the upcoming errors and to avoid unexpected expenses. In addition, comparing the prevailing error amount of the software to the original estimate also gives direct indications of the current state of the software's reliability during the project.

Error estimation can be seen as an important activity in a mature software process. Producing a reliable estimate requires some time and resources (depending on the applied estimation method) but it can be an effective tool to assist the planning and analysis of the project. This article presents the basic categories of software error estimation and introduces some known techniques and models. Having stated that none of them live up to the demands of a modern software development project, a new estimation method is introduced. A view on how the new method can be applied in a real project is also presented and modeled with the *SPEM* (*Software Process Engineering Meta-model*) language [1].

1. Different Classifications of Error Estimation

Some error estimation methods exist in the literature, varying from simple mathematical techniques to more complex, closely project time-line related models. Some of them are meant to be used in the early stages of the software development project producing one *static* total amount estimate that is used throughout the whole project. Some have a more *dynamic* nature. They are updated and fine-tuned during the project to give a more up-to-date picture of the current state of the project [2].

Both categories of error estimation have their purposes. An estimate made in the beginning of the project can be used by the project management to guide the resource planning and scheduling. It can also be used in software maturity estimations later on in the project when the decisions about software releasing points are made. A dynamic estimate being updated during project's timeline, on the other hand, gives a different kind of view on the state of the project by predicting the project based on the current situation [3].

An extensive error estimate – either static or dynamic – can be further divided into two more or less independent parts: total error amount estimate and error rate estimate (see Table 1). The former holds the estimate of the total error amount of the final software. These errors originate in different phases of the development project and are usually found in the different testing phases. The latter tries to describe at which stage of the project the errors are found. This behavior is usually described with a cumulative error curve [3].

Table 1: Error estimation categories.

Different categories of error estimation	Static total amount estimation	Dynamic total amount estimation
	Static error rate estimation	Dynamic error rate estimation

Many modern software development processes use iterative way of working. Therefore also testing – that is, discovering errors – is something that is being done in the development project a) during a long period of time and b) in many levels of abstraction (as also the traditional V-model of testing suggests). Hence, both error rate estimation and dynamic error estimation have an important role in the field of error estimation. This article focuses, however, mainly to the *static total amount estimation*. The main goal is to find the right way to estimate the total amount of software errors in a big, multi-site project developing large, feature-rich software.

2. Existing Error Estimation Methods

The word *technique* is used here to describe the somewhat straightforward ways to estimate the total error amount of software. A technique is considered to be a statistical or mathematical calculation that merely utilizes numerical data describing the software. A *model*, on the other hand, is considered to a more complex estimation method including more expert judgment and software project related figures like estimated time-line, software size or even more precise project parameters.

2.1. Error Estimation Techniques

Past error density is one of the oldest and simplest ways to estimate the total error amount [4]. It relies on comparing the size of the software of past projects to the one of the upcoming project. Having the error data of past projects available and relying to the fact that the ratio of *errors to software size* is the same in past and future software, error estimate for the future software can be made based on an analogy.

Error seeding is also a widely documented technique that is based in traditional statistics [4]. As the name suggests, some errors are produced on purpose by an extra group of programmers while another group is trying to find them by testing the software. Knowing 1) the number of seeded errors, 2) the number of discovered seeded errors and 3) the total number of discovered errors, estimate of the total error amount can be made. It relies on the assumption that a small sample of errors – that is, the seeded errors – reflects the state (in terms of total error amount) of the whole software. It should be noticed that the error seeding technique can only be utilized after the software has already undergone some testing activities. The formulation of the technique is as follows:

$$N_{\text{errors},\text{total}} = (N_{\text{errors},\text{seeded},\text{total}} / N_{\text{errors},\text{seeded},\text{found}}) * N_{\text{errors},\text{found}} \quad (1)$$

In the *error pooling* technique, the testers are divided into two independent groups, both reporting errors to their own *pools* [4]. The testing should be profound and cover the whole functionality of the software. At any given point of time the amount of unique errors in both pools, and most importantly, the amount of common errors found from both of the pools, can be calculated. According to the approach of the error seeding technique, the total amount of software errors can now be estimated using the following formulation (not mathematically derived, but estimated itself):

$$N_{\text{errors},\text{total}} = (N_{\text{errors},A} * N_{\text{errors},B}) / N_{\text{errors},A \& B} \quad (2)$$

2.2. Error Estimation Models

The field of software error estimation models highlights somewhat different aspects of estimation. A wider range of data describing the current development project is utilized. In addition, the methods become more complex, making good use of things like expert judgment in addition to mere mathematical calculations.

One large category of error estimation models is formed by the different models based on the *Weibull distribution*, a time-dependent mathematical formulation [2]. It includes parameters of shape, scale and time and describes the dynamics of the cumulative error rate in different situations. In the environment of software development, the two most widely applicable special cases of the distribution are the *Rayleigh model* and the *exponential model* [2], [5]. It should be noticed, though, that the total error amount is only a parameter in these formulations, not the output. They cannot be applied to the total amount estimation as such and are therefore left without further studies in this context.

Probably the most well-known model for software quality and error amount estimation is *COQUALMO (Constructive Quality Model)*, published by Barry Boehm in 1997 and based on the widely acknowledged COCOMO (Constructive Cost Model) [6]. It has the somewhat same structure than COCOMO II but has its focus in software quality and error amounts instead of cost. At the time being published, it had already gone through some calibration with error data from certain real software projects.

COQUALMO is actually a combination of two sub-models: the Defect Introduction model and the Defect Removal model. The former estimates the total amount of software errors discovered from the software during the project, the latter estimates the amount of errors being removed from the software. In the context of this article, the former sub-model is the more interesting one.

The Defect Introduction model divides the incoming errors in three sources: requirements, design and coding. The input parameters of the mathematical formulation of the Defect Introduction model include a) software size, b) an error source specific calibration constant and c) an economy scale factor - all separately for the three error sources – as well as d) the so called defect introduction drivers (DID's) that are the main factors used to fine-tune the estimate and are partly inherited from the originating COCOMO II. This results in the following formulation:

$$N_{\text{errors},\text{total}} = \sum_{j=1}^3 A_j * (\text{Size}_j)^{B_j} * \prod_{i=1}^{21} \text{DID}_{ij} \quad (3)$$

where j relates to the three sources of errors, A_j is the calibration constant related to the j :th error source, Size_j is the size of the j :th error source, B_j is the economy scale factor of the j :th error source and DID_{ij} is the i :th DID of the j :th error source.

The starting point of the model is size that has to be estimated separately for the three different error sources. It can be expressed in lines of code, in function points or

in some other applicable way depending on the nature of all the error sources being estimated. The error source specific calibration constant and the economy scale factor are parameters that can be utilized to adjust the estimation based on available project data outside the scope of the DID's.

The DID's form the central control point of the model. They are divided in four main categories: platform, product, personnel and project. In these categories the DID's are further divided to altogether 21 separate drivers (like required reusability, platform volatility, programmer capability and process maturity). As the formulation shows, these drivers acting as multiplicative constants are the main medium used to fine-tune the estimate. They hold the organization and project specific information that is utilized to get the final estimate. The initial valuation of the DID's – the most essential thing having an effect on the estimation result – is based on expert judgment. As projects go by, the drivers must be calibrated with the available data.

2.3. Notable Shortcomings of the Existing Methods

Let us revise the modern software development environment to which an error estimation method should fit. First of all, the software can be remarkably large and consist of various independent components. It may include a lot of new features and technologies compared to preceding releases and/or to software of other companies. Secondly, the software can be developed by multiple suppliers working in various separate sites. The different suppliers may have their own processes and ways of working. Thirdly, organizations developing software this large usually manage several successive projects with the software somewhat related to each other. It would not be realistic to assume that an organization would only have been put up for one project. This factor of continuity is also something that has an effect on the ways to handle error estimation. So, what are the biggest deficiencies of the existing methods? Is one of them the solution for the whole problem?

Relying purely on the analogy between past and upcoming projects – as in the past error density technique – can be quite dangerous. As new features and technologies are introduced in the software, it is impossible to say whether it can directly be compared to the preceding released software of the organization in question or not. In addition, using only the size of the software to illustrate its complexity can lead to incorrect estimates.

The biggest problem with the error seeding technique is the demand of extra resources. While the normal implementation and testing activities are ongoing, other groups are needed to produce and to look after the seeded errors. It is highly debatable whether a software development organization would invest to error estimation by hiring several new professionals or not. The answer is most likely: not. In addition to the first problem, error seeding also contains the risk of seeded, even critical, errors to remain undiscovered in the software. These can cause unwanted functionality or side-effects later on in the software.

The error pooling technique has some of the same downsides than the error seeding. Firstly, two independent testing teams are required. Both of them are required to have the expertise to test the whole functionality of the software. Secondly, a lot of extra work (in terms of creating several overlapping error reports) is done for the sake of estimation. Thirdly, one more expert is needed to do the comparison between the

two error pools. Hence, the technique seems too resource-greedy to be actually utilized in real projects.

Different life-time models, like the models based on the Weibull distribution, are very useful when describing the cumulative error amounts during the development project, described by the error curve. They cannot, however, be used in the total amount estimation because they only utilize such information, do not produce it. Combined with a reliable way to estimate the total amount estimate they can, however, form a solid practice to end-to-end software error estimation.

The Defect Introduction model of the COQUALMO is the most realistic and extensive effort to match the demands of proper error estimation method. It perceives the software development project as a complex system from which several independent areas of making can be identified and separately valued. It has many clear weak points, though. First of all, the model focuses on calibration which means that it requires data from several upcoming consecutive projects to become reliable. It is not guaranteed that the model still fits the environment of the applying organization after the calibration time has passed and it is finally ready to be utilized. Secondly, the key input parameters like the DID's are highly supplier specific and are unwieldy adapted to describe the whole organization of a multi-supplier project. The multi-site factor also makes it hard to form a common practice to evaluate the size of the software which is the starting point of the whole estimation.

3. SEEC – A New Approach to Software Error Estimation

As the existing techniques and models do not seem to offer a complete solution, the principles of a whole new estimation method have to be settled. A new method may make good use of some of the upsides and inventions of the presented methods but it introduces a new way to divide and conquer the difficult field of error estimation. In the following, the most important aspects of software error estimation are discussed and the important decision-making related to the new estimation method is presented.

First of all, the *unit of estimation* should be established. The two alternatives are a) to deal with the whole software as a one large unit or b) to divide the software to well-defined components. The software development environment to which the new estimation method is targeted highly demands the latter approach to be used. The software is most likely developed in independent components holding specific features and technologies and it can be developed by multiple suppliers, internal or external to the managerial organization.

At the same time, one should keep in mind that the components and their suppliers are to be closely examined and valued when the final estimate is fine-tuned. After all, many different things may have an effect on the error amount of a software component. So, dividing the software to too many components can lead to a dead end when the final adjustment of the estimate is made. A proper level of granularity should therefore be the goal of the component division of the software.

Secondly, the *reference level* of the estimation should be decided. Some existing methods use the estimated software size as the starting point, some can be used only after the testing is started and some actual error amount data is available. But if one

wants the new method to be a reliable tool for the project planners, the reference level should be available in the beginning of the project and rely on existing data, not to be an estimate itself.

The answer to this problem is analogy: not to use it in the whole total amount estimation but to use it in the discovering of the reference error levels. In practice this means that the error amounts of the defined components in the past projects of the organization in question are chosen as the starting points for the estimation. In whatever manner a component is discovered and defined, using the same kind of tactics the error amount of that component in the preceding software is determined. If some past error data from several projects is available, it is only reasonable to make good use of as many of them as possible.

Third essential thing is to come up with a well-defined way to *fine-tune* the reference error amounts so that the final estimate would actually reflect the expected state of the upcoming software. The adjustment should not only be a collection of educated guesses but instead a solid practice that could be used and further developed through a flow of successive projects. Achieving this requires identification of the key factors having an effect on the error amount of a component. These may include concepts like changes in feature sets, changes in applied technologies, or changes in supplier functions. This article does not try to identify all the possible influencing factors but instead highlights the fact that the identification itself should be done and the identified drivers should somehow be anchored to enable the evolution of the estimation method. The intention is not to freeze the whole set of influencing factors but to have a certain basic set as a starting point instead. This collection can and should be extended in the context of a new evaluated project.

In the mathematical formulation of the new method, see Eq. (7), these so called *change factors* take the form of multiplicative constants used to operate the reference values component-wise. It is therefore clear that they have to be valuated. In the ground level it means giving some kind of value for every type and amount of change in the component or its supplier comparing to the preceding versions. After the set of identified change factors is valuated, correct values are selected based on analysis on preceding and upcoming products. Although the valuation and the selection of correct values are essential parts of the estimation activity itself and they have to be done before the method can be applied, they are also considered to be outside the scope of this article. Only an estimation framework is introduced leaving highly domain, project and organization specific details to further studies.

Dividing the software to components and comparing these components in the preceding products with the ones in the evaluated product captures the essence of the new estimation method. The initial goal was to discover a way to estimate software error amounts. Taking these facts into account, the new method is given the name SEEC - Software Error Estimation by Component-wise comparison.

Figure 1 illustrates how the SEEC method can be integrated as a part of a large software project. The two primary activities utilizing the method are project planning and project tracking. In the former activity, SEEC estimate can be applied for resource planning. In the latter activity, it can be used to predict the current level of software maturity and reliability, and, thus the progress of the project. The estimation activity

relies on past error and project history data. Therefore the process of the applying organization must include corresponding data gathering activities.

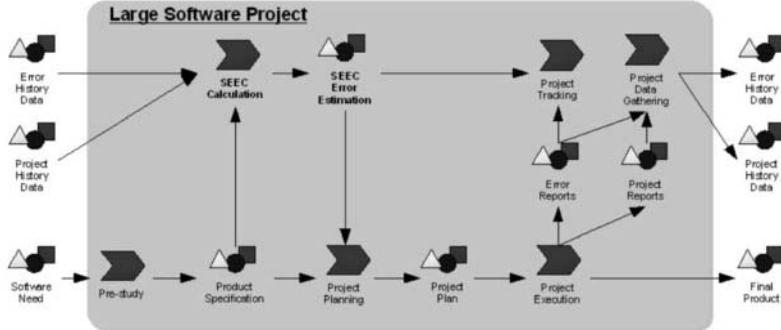


Figure 1. SEEC method as part of a large software project.

One more important aspect of estimation has to be pointed out. Software projects or processes are not machines that can be totally estimated and controlled in a watertight way. In addition, the pure numerical data does not always describe them in a comprehensive way. The recognition of these facts leads to the acknowledgement of the importance of the human factor, the expert judgment. It is needed in many points of the estimation, even with the new method. Both component division and the identification and valuation of the change factors require expert judgment. What makes the most essential difference between mere guessing and applying this method, though, is the logical and well-defined manner the method is constructed and instructed to be applied.

4. Required Activities and the Mathematical Formulation of SEEC

Now that the most essential aspects of software error estimation are discussed and related decisions concerning the new method made, the resulting approach has to be summarized as a re-usable formulation. The error estimation is done component-wise which indicates that the mathematical formula has to be a sigma expression. Error amounts of past products are used as the reference level so they form the core of the formulation. Change factors are used to fine-tune the estimate so they are present in the formula as corresponding multiplicative constants. The required working activities and the final formulation of the new method are presented in the following.

First, the component-division of the developed software is made. E.g. different documents like project plans (maybe draft versions at the time), feature lists or product specifications can be used to assist this activity. The main goal is to clearly define independent software components (probably delivered by different suppliers) in a proper level of generality.

Secondly, the software error amounts of the identified components in the preceding software products are determined. Assuming that the projects are well-

organized and documented, error data of past projects should be available. The outcomes of this activity are the reference error amounts:

$$REF_1 \dots REF_n \quad (4)$$

where n is the number of identified components and REF_i refers to the reference error amount of the i :th component.

Thirdly, based on careful analysis of the organization, project and the upcoming product itself, the complete list of change factors is established. In case of the project not being the first one of the organization in question deploying this method, this is done by going through the basic set of change factors and by completing it if necessary. In any case, the outcomes of this activity are the identified change factors:

Before the change factors can be applied they have to be valued. In practice this means that the values for different extents of change in the scope of that change factor have to be decided. Regarding to the change factors of the basic set, the valuations are only updated, if necessary. After this, correct values for every identified change factor related to every identified component are selected from the collection of valuations based on profound analysis on past and upcoming products. The outcomes of this activity are the change factor values for every identified component:

$$CFV_{11} \dots CFV_{nm} \quad (5)$$

where n is the number of identified components, m is the number of identified change factors and CFV_{nm} refers to the value of m :th change factor related to n :th component.

Putting the reference error amounts and the change factor values together results in the final formulation of the new estimation method:

$$N_{errors,total} = \sum_{i=1}^n \left(\prod_{j=1}^m CFV_{ij} \right) * REF_i \quad (6)$$

where n is the number of identified components, m is the number of identified change factors, CFV_{ij} refers to the value of m :th change factor related to the n :th component and REF_i is refers to the reference error amount of the i :th component.

Figure 2 illustrates the required activities and related work products, roles and tools of the SEEC method. Valuating change factors is the most demanding activity when applying SEEC in real software projects. Thus, a new role responsible for the valuation is introduced.

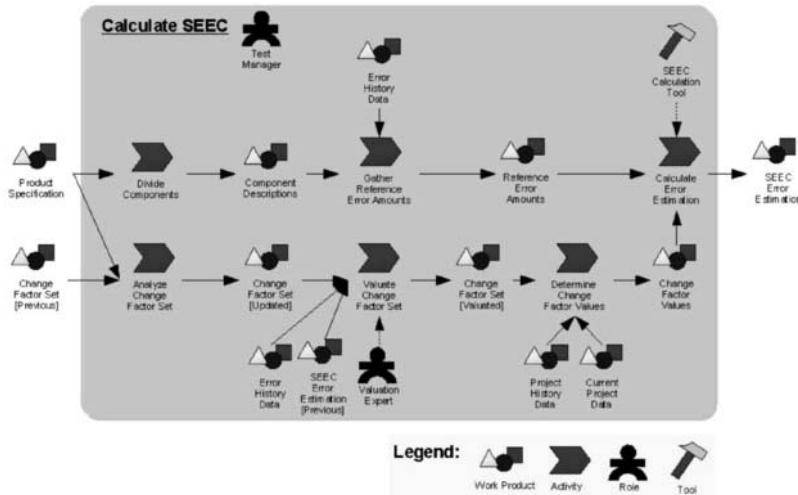


Figure 2. Internal activities, work products and roles of the SEEC method.

5. Conclusions

It was discovered that the existing error estimation methods are not fully sufficient for modern, multi-site software development projects including several different components. Therefore, a new way to estimate error in this kind of environment – the SEEC method – was presented. It serves as a framework offering the basic principles and activities to produce a reliable software error estimate. Before it can be applied in real software projects, it has to be customized to support the software process of the applying organization. Despite the fact that the overall structure of the method has been defined, individual activities have to be further examined in order to maximize the benefits of the method.

Also, illustration of software process metrics using the SPEM process modeling language was experimented in this article. It is evident that the graphical presentation helps to situate a solitary measurement method into the development process and to form the general view of the requirements and benefits of the method.

References

- [1] Object Management Group. 2005. *Software Process Engineering Metamodel Specification – version 1.1*. Object Management Group, January 2005. formal/05-01-06.
- [2] Kan, S. 1995. *Metrics and Models in Software Quality Engineering*. Reading, Massachusetts: Addison Wesley.
- [3] Kuusela, T. 2005. *Developing a Software Error Estimation Method for Series 60 Product Programs*. Master thesis, Turku University, March 2005.
- [4] McConnell, S. 1997. *Software Project Survival Guide*. Redmond, Washington: Microsoft Press.
- [5] Putnam, L. & Myers, W. 1992. *Measures of Excellence: Reliable Software on Time within Budget*. New Jersey: Prentice Hall.
- [6] Boehm, B. & Chulani, S. 1999. *Modeling Software Defect Introduction and Removal: COQUALMO*. Technical report, USC - Center for Software Engineering, 1999.

Support Method for Source Code Modification in Changing GUI Widgets

Junko Shirogane^a Kazuhiro Fukaya^b and Yoshiaki Fukazawa^b

^a *Tokyo Woman's Christian University*

^b *Waseda University*

Abstract. Recently, the expansion of the range of users has led to the increase in the importance of software usability. To realize usable software, it is efficient to repeat the process of evaluating and improving GUIs. In this process, it is often necessary to replace the GUI widgets. In this case, we have to change the connection between GUI codes and logic codes. However, it is difficult for developers to find and modify the codes to be changed in the source code. In this research, we propose a method of finding the modification codes in the source code automatically and of suggesting instructions on developers how to modify them.

Keywords. GUI, Widget, Usability, Source code modification

1. Introduction

In recent years, more people have been using computers and systems, because they can do various kinds of work. Therefore, it is important to develop software, focusing not only on what people can do but also on how usable it is to people, called usability. In particular, users directly operate Graphical User Interfaces (GUIs) of software, so the usability of GUIs has a great influence on the software's usability that end users consider. Therefore, making GUIs more usable is very important to develop usable software.

End users' experiences and preferences strongly influence the usability of GUIs that end users consider. Also, there are many cases where some problems of usability are found in actual use but not during the development of the software. Thus, it is difficult to develop enough usable GUIs without an iterative design and implementation phase. Therefore, to make GUIs more usable, it is effective to repeat iterative steps for developing GUIs[1]: use the GUIs as a trial, evaluate the usability of the GUIs, improve the GUIs on the basis of the evaluation, and modify the codes of connecting GUIs and other components.

Source codes of software can be classified into the following three types: GUI codes, logic codes and connection codes. The roles of GUI codes are GUI processing, such as widget generation and widget arrangement on windows. The roles of logic codes are to provide the body of the software, such as data processing and file handling. The roles of connection codes are data passing between GUI codes and logic codes. In many cases, it is necessary to change not only layouts but also the types of widgets and windows that widgets are put on. In this case, it is necessary to modify the connection codes between GUI codes and logic codes, and even logic codes themselves (called "modification codes"). In large-scale software, its source codes includes many modification codes,

which may spread widely in the source codes. In modifying source codes, developers may make mistakes in identifying modification codes and modifying them, and miss required modification codes. Recently, many developers have been using Integrated Development environments (IDEs), such as Eclipse[2] and NetBeans[3]. In changing GUIs, IDEs show not semantically valid candidates but near-spelling candidates for only compile error codes. Thus, it is difficult for developers to find all modification codes in the software and modify them correctly.

In our research, to make GUI changes easy and correct, we propose a method of automatically searching for modification codes and suggesting instructions on how to modify codes. Here, we focus on the Java programming language, and all widgets used as examples in this paper are included in the Java Swing package. However, our approach does not depend on the programming language and GUI packages. Also, we assume that the targets our method are the large-scale business application.

2. Preliminary

It is necessary to change the GUIs of software, such as when developers improve GUIs based on the usability evaluation or reflect changes of users' preferences and usage environment in actual use. In our method, to make GUI change easy, our system automatically searches for modification codes in source codes, and suggests instructions on how to modify those codes.



Figure 1. Processing after pushing button

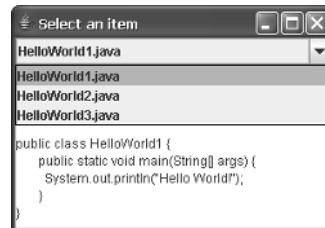


Figure 2. Processing after only selecting item

As an example of GUI change, we consider GUIs for which a user selects an item in a combo box and some texts are inserted in a text field on the basis of the selected item. These GUIs are shown in Figures 1 and 2. In Figure 1, first, a user selects an item from the combo box, and pushes the button for the right combo box. Then, some texts are inserted in the text field below. In Figure 2, first, a user selects an item from the combo box. Then, some texts can be inserted in the text field below. The operation steps of the GUI of Figure 2 are fewer than that of the GUI of Figure 1, so some users may consider the GUI of Figure 2 to be usable. In the GUI of Figure 1, users can confirm the selected item before processing, so some users may consider the GUI of Figure 1 to be usable. Thus, which GUI users consider to be usable is based on factors, such as users' preference, the actual use environment and processed data in the software. Therefore, it is considered that GUI change is required.

Modification codes from Figure 1 to Figure 2 are shown in Figure 3. To search for modification codes in the source codes and suggest instructions on how to modify them, we classify the functions of widgets, consider possible change patterns and classify the methods of each widget. In this paper, we call widgets before change (currently used widgets in the target software) "pre-widgets", and widgets after change "post-widgets".

```

public class SampleGUIClass2 extends JFrame implements ActionListener{
    private JComboBox combo;
    private JButton button;
    private JTextArea text;
    private Container pane;
    private JPanel panel;

    public SampleGUIClass2(){
        super("SAMPLE");
        String[] items = {"HelloJava1.java", "HelloJava2.java", "HelloJava3.java"};
        combo = new JComboBox(items);
        button = new JButton("DISPLAY");
        button.addActionListener(this);
        text = new JTextArea();
        text.setEditable(false);
        pane = new JPanel();
        pane.setLayout(new GridLayout(0,2));
        pane.add(combo);
        pane.add(button);
        panel.add(pane);
        panel.add(text);
    }

    public void actionPerformed(ActionEvent ae){
        String fileName = (String)combo.getSelectedItem();
        this.openFile(fileName);
    }
}

```

Figure 3. Modification codes from Figure 1 to Figure 2

2.1. Roles of widgets

Each type of widget is used for different purposes, such as text input and item selection. The types of post-widgets that developers can replace are restricted on the basis of the purpose of widgets. Therefore, we classify the types of widgets according to purposes, and our system shows the available types of post-widgets to developers when the pre-widget is given. We call these purposes "roles", and the classifications are written in a table (called "Widget Replacement Table (WRT)"). Now we define the following 7 roles:

Display role

- The result of processes and the confirmation of user input are displayed
- Example widgets: label and list (in Java Swing, JLabel and JList) widgets

Selection role

- Users select items from selection items
- Example widgets: combo box and radio button (in Java Swing, JComboBox and JRadioButton) widgets

Selection & Action role

- Users select items from selection items and then some processes are performed
- Example widgets: combo box and radio button (in Java Swing, JComboBox and JRadioButton) widgets

Input role

- Users input texts
- Example widgets: text field (in Java Swing, JTextField and JTextArea) widgets

Input & Action role

- Users input texts and then some processes are performed (in many cases, users input text, hit the Enter key on a keyboard, then some processes are performed)
- Example widgets: text field (in Java Swing, JTextField and JTextArea) widgets

Action role

- Some user events, such as pushing a button and hitting the Enter key, occur
- Example widgets: button and menu item (in Java Swing, JButton and JMenuItem) widgets

Container role

- This widget is used as a mat to put widgets on
- Example widgets: panel and frame (in Java Swing, JPanel and JFrame) widgets

2.2. Change patterns

In changing widgets, there are several cases, such as replacing the current widgets with another type of widgets (for example, a combo box widget is replaced with a list widget), changing the trigger of processes for input (for example, performing processes after pushing a button is changed to after only selecting an item), or moving widgets from one window to another. Also, there are cases in which developers replace not only one widget with another widget but also one widget with several widgets, several widgets with one widget and several widgets with several widgets. For example, both JList widgets and JRadioButton widgets have the selection role. Several selection items are registered to one JList widget. However, a selection item is registered to one JRadioButton widget. That is, in many cases, several JRadioButton widgets are used as the same number of selection items, and users select an item from several JRadioButton widgets. However, users select an item from one JList widget. Thus, when JRadioButton widgets need be replaced with JList widgets, several JRadioButton widgets need be replaced with one JList widget. Therefore, in replacing widgets, we define the available patterns that widgets can be changed into the basis of the number of widgets and the operation of GUIs as "Change patterns". Due to change patterns, our system can search for modification codes correctly. The current change patterns are the following 9 patterns:

Action inclusion pattern

- A pattern for changing one type of input or selection role widget and one type of action widget to one type of input & action or selection & action role widget
- For example: change GUIs in which a user selects an item and pushes a button, then some processes are performed, to GUIs in which users only select an item, then some processes are performed

Action separation pattern

- A pattern for changing one type of input & action or selection & action role widget to one type of input or selection role widget and one type of action role widget
- For example: change GUIs in which users select an item, then some processes are performed, to GUIs in which users select an item, push a button, then some processes are performed

One-to-one pattern

- A pattern for replacing a widget with another type of widget
- For example: replace a combo box widget with a list box widget

One-to-many pattern

- A pattern for replacing a widget with a set of another type of widget (all of post-widgets are the same type)
- For example: replace a combo box widget with a set of radio button widgets

Many-to-one pattern

- A pattern for replacing a set of widgets with another type of widget (all of pre-widgets are the same type)
- For example: replace a set of radio button widgets with a combo box widget

Many-to-many pattern

- A pattern for replacing a set of widgets with a set of another type of widgets (all of pre-widgets are the same type, and also all of post-widgets are the same type)
- For example: replace a set of check box widgets with a set of toggle button widgets

Moving to existing window pattern

- A pattern for moving a widget or a set of widgets from a window, on which the widget(s) are put on, to another window written in the source code

Moving to new window pattern

- A pattern for creating a window, and moving a widget or a set of widgets from a window to the created window

Deleting pattern

- A pattern for deleting a widget or a set of widgets

2.3. Widget replacement table

Each widget has several methods, such as those for getting values of the widget and setting the size of the widget. Some methods are used for the same purpose in all widgets, such as the getSize() method for getting the size of the widget and the setName(String) method for setting the name of the widget. Some methods are used for different purposes in each widget. Also, in widgets of the same role, some methods are used for the same purposes. These methods are not used for the same purpose in all widgets but are used for the same purpose only in widgets of the same role. For example, both JList widgets and JComboBox widgets have the selection role, and they have a method of getting a selected item. The method for this purpose is the getSelectedValue() method in JList widgets and the getSelectedItem() method in JComboBox widgets.

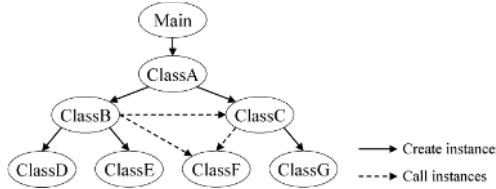
Table 1. Example of widget replacement table (Selection role)

Widget	Getting selected item	Getting widget size
JList	getSelectedValue()	getSize()
JComboBox	getSelectedItem()	getSize()
JRadioButton	if(<widget variable>.isSelected()) getText() (getName() or something);	getSize()

It is necessary to identify the purpose of each method in widgets of the same role and classify methods according to purpose. This classification is written in a WRT (Widget Replacement Table). In some cases, a method of a widget corresponds to some methods of another type of widget with the same role. For example, the getSelectedItem() method of JComboBox widgets is one method of realizing the purpose "getting a selected item". To realize this purpose in JRadioButton widgets, first, the target software identifies which JRadioButton widget is selected, then gets the text, name or something of the selected JRadioButton widget. That is, the getSelectedItem() method in JComboBox widgets corresponds to an if statement, the isSelected() method and the getText() method, the getName() method or something in JRadioButton widgets. In the WRT, these correspondences are also written. An example of the WRT is shown in Table 1.

2.4. Guidelines in searching for modification codes

In searching for modification codes, we define an "Instance Creation/Call Graph (ICCG)". This graph describes the relationship between created and creating instances of classes. An example of the ICCG is shown in Figure 4. In this graph, a node represents a class, and an arrow represents the relationship from creating instances to created instances. For example, instances of class B are created in class A, and instances of class F are called from classes B and C in this figure.

**Figure 4.** Example of instance creation/call graph

In our method, using an ICCG, our system identifies target classes to be searched on the basis of the class in which the variables of pre-widgets are declared. Then, our system generates a program dependence graph (PDG)[4] in each target class. A PDG describes the data dependence and control dependence between statements in the program. In our method, our system searches for modification codes by tracing the data dependence relationships in a PDG.

Our system analyzes source codes translated by JavaML[5] to search for modification codes. JavaML is a tool that translates Java source codes into XML format.

3. System architecture

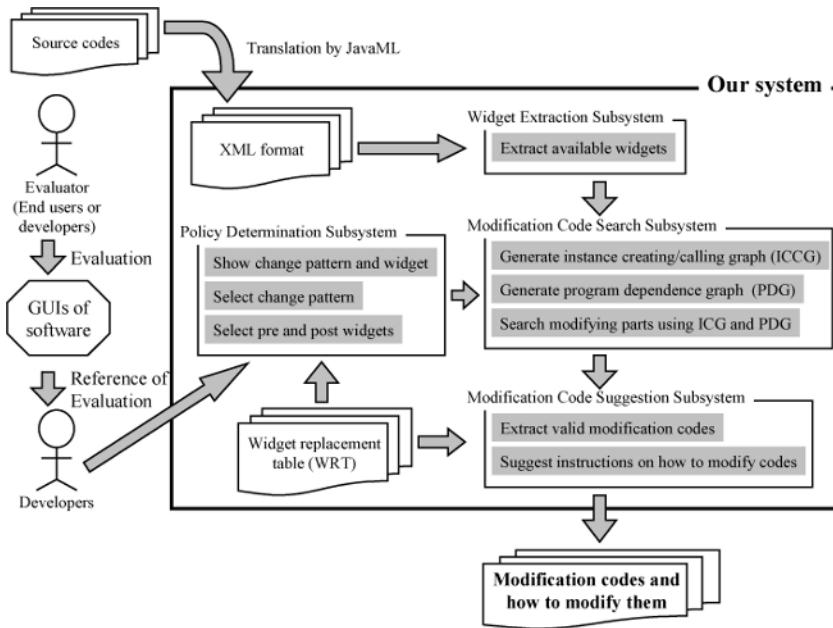
**Figure 5.** System architecture

Figure 5 shows the architecture of our system. Our system consists of the following four parts: widget extraction, policy determination, modification code search and modification code suggestion subsystems.

3.1. Widget Extraction

Our system translates source codes of target software in XML format using JavaML. Then, some information on each widget written in the source codes is extracted. The extracted information includes the followings: types (class names) of widgets, variable names of widgets and windows on which widgets are put.

3.2. Policy determination

To make GUIs more usable, the usability of GUIs are evaluated. On the basis of the evaluation, developers determine how GUIs are changed. First, our system shows candidates of pre-widgets extracted by the widget extraction subsystem. Next, our system shows change patterns on the basis of the selected pre-widget, and developers select one of them. Then, our system shows available candidates of the post-widget based on the selected pre-widget and change pattern, or shows available candidates of windows for replacing pre-widgets. Finally, developers select post-widgets or a window to replace.

3.3. Modification code search

Our system searches for candidates of modification codes on the basis of the change pattern and pre-widgets and post-widgets selected in the policy determination subsystem. Candidates are searched in the following two steps.

(1) Search target identification

Modification codes are not always contained only in a class in which pre-widgets are declared (called a "widget-declared class"), so it is necessary to identify target classes that need to be modified in other classes. In our method, target classes to be searched are identified using ICCG. Target classes to be searched include the followings. Here, as an example, class B in Figure 4 is regarded as the widget-declared class.

- A widget-declared class (class B in Figure 4)
- Parent level classes of widget-declared class on ICCG (i.e., Classes that create or call instances of the widget-declared class, class A in Figure 4)
- Child level classes of widget-declared class on ICCG (i.e., Classes of instances created or called from the widget-declared class, classes D and E in Figure 4)

(2) Target class search

Our system generates PDGs of the target classes and searches for modification codes using the PDGs. The procedure is as follows: 1. extract statements of defining and referring variables, 2. analyze data dependence relationships, 3. create PDGs and 4. search for modification codes using the PDGs.

A node represents a statement, and an edge represents data dependences in PDGs. The procedure for searching for modification codes involves tracing edges from a node of the definition of pre-widgets to other nodes. In the target classes except the widget-declared class, our system traces the PDGs from a node of creating or calling an instance or a node of a called method of the widget-declared class.

3.4. Modification code suggestion

It may not be necessary to modify all of codes searched by the modification code search subsystem. Therefore, our system extracts valid modification codes from searched codes, and suggests instructions on how to modify the codes.

(1) Extraction of valid modification codes

Valid modification codes include mainly the followings: variable declaration, variable reference method names of widgets, return type declaration in methods and type declaration of parameters in methods.

In some cases, method names need to be modified, and in other cases, method names do not need to be modified. These are identified on the basis of what pre-widgets and post-widgets are. For example, when a `JList` widget is replaced with a `JComboBox` widget, the `getSelectedValue()` method of the `JList` widget for getting a selected item needs to be replaced with the `getSelectedItem()` method of a `JComboBox` widget. However, the `getSize()` method for getting the widget size is inherited from the `JComponent` class, which is a super class of the `JList` and `JComboBox` classes, so it is not necessary to replace this method. These classifications of methods are written in WRT, so our system extracts valid modification codes from searched codes using WRT.

(2) Suggestion of instructions on how to modify codes to developers

Our system suggests instructions on how to modify for valid modification codes identified in (1) using WRT. In WRT, the classifications of methods and widgets and the available change pattern for each widget are written. Thus, our system can suggest corresponding instructions on how to modify each modification code using WRT. If there are no corresponding instructions in WRT, our system indicates only the necessity of code modification to developers. Figure 6 shows an example of replacing a `JList` widget with some `JRadioButton` widgets.

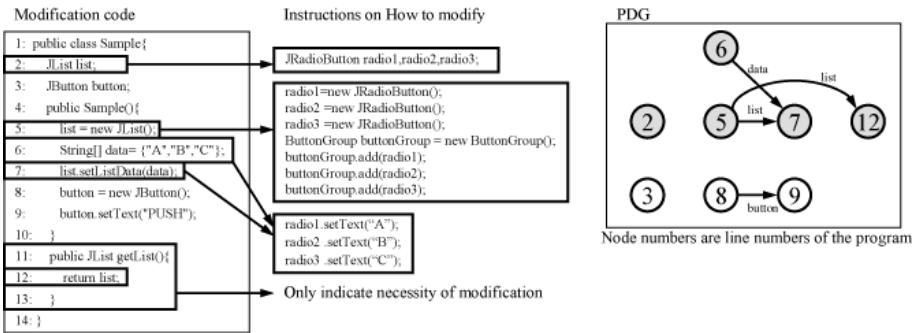


Figure 6. Modification of widget-declared class

4. Evaluation

We evaluate our system using three sample applications A, B, and C. Sample A is `Font2DTest` in the JDK 1.4 demo [6], sample B is `FileChooserDemo` in the JDK 1.4 demo [6] and sample C is our system. We replace `JComboBox` widgets with `JList` widgets in sample A, and `JRadioButton` widgets with `JComboBox` widgets in samples B and C.

4.1. Evaluation of searching for modification codes

We evaluate the correctness of searching for modification codes by modifying using Eclipse[2] and our system. In Eclipse, we open the source codes of the applications, replace widgets and count statements that Eclipse shows as having errors. In our system, we count statements that our system shows as modification codes. The results are shown in Table 2. (a) means the number of modification codes to be found, (b) means the number of modification codes to be shown correctly, (c) means the number of necessary modification codes not to be shown, and (d) means the number of codes to be shown incorrectly. In this evaluation, we also manually change the GUIs of the sample applica-

tions. The number of modification codes in manual modification can be considered as the correct number. The numbers of correct modification codes are 17, 19 and 31 in samples A, B and C, respectively.

Table 2. Evaluation of correct searching

	Sample A		Sample B		Sample C	
	Eclipse	Our System	Eclipse	Our System	Eclipse	Our System
(a)	10	17	19	19	21	32
(b)	9	13	19	19	21	28
(c)	8	4	0	0	10	3
(d)	1	4	0	0	0	4

In terms of (b), our system shows the number of modification codes more than those of Eclipse. The reason is that Eclipse shows only codes in which widget variables are used, whereas our system shows those codes and other related codes. Also, in our system, some necessary modification codes were not shown. When a method of a pre-widget is replaced with the corresponding method of a post-widget, there are some cases in which the number of parameters of a pre-widget's method is different from the number of parameters of post-widget's method. Our system cannot show these extra parameters. However, only 53% and 68% of correct modification codes in sample A and sample C respectively are shown by Eclipse. 76% and 90% of correct modification codes in sample A and sample C respectively are shown by our system. So, our system can find modification codes correctly much more than Eclipse.

if a lot of necessary modification codes are not to be shown, it is necessary for developers to search for these manually. Therefore, it is more important that necessary modification codes that are not to be shown are few.

4.2. Evaluation of suggesting instructions on how to modify codes

We evaluate the correctness of suggesting instructions on how to modify the modification codes. The results are shown in Table 3. (a) means the number of modification codes which our system should suggest the instructions, and (b) means the number of modification codes which our system suggests the modification instructions correctly.

Table 3. Evaluation of correct suggesting

	Sample A	Sample B	Sample C
(a)	15	19	27
(b)	8(53%)	12(63%)	19(70%)

According to this table, our system suggests correct instructions on how to modify codes at a rate of 50-70%, so our system is effective in modifying codes. Codes for which our system cannot suggest instructions on how to modify them are mainly in the cases of deleting codes and changing listeners. For example, it is necessary to change the ActionListener of the JComboBox widget to the ListSelectionListener of the JList widget in sample A. In this case, it is necessary to add an import statement, an implements statement and a method of ListSelectionListener. In order to improve this, it is necessary to increase the entries of a WRT.

According to Tables 2 and 3, our system can show modification codes and modification instructions more properly than the other method. Therefore, our method is more effective than the other method in changing GUI.

5. Related work

To support source code changes, some methods have been proposed. Ying et al. proposed a method of identifying relevant source codes in modifying program[7]. In this method, a set of files changed in past times are found using data mining techniques from the history of changing source codes. Zimmermann et al. also proposed a method of identifying relevant source codes in modifying program[8]. In this method, programmers' histories of source code changes are saved. When a programmer changes a function, the system shows another functions that other programmers changed with the former function.

Some methods for impact analysis have been proposed. Using impact analysis, modification codes based on program changes can be identified. Law et al. proposed a method for impact analysis using call graphs[9]. In this method, paths of call graphs are collected dynamically. Badri et al. also proposed a method for impact analysis[10]. In this method, call graphs are created statically, and the relevant codes in modifying are identified.

In these method, the concrete instructions on how to modify codes are not suggested, and valid modification codes may not be shown, that is, codes that do not need to be modified may be shown, and codes that need to be modified may not be shown.

6. Conclusion

In this paper, we propose a method of searching for modification codes in a source code and suggesting instructions on how to modify them in GUI change. Future work includes the followings:

- Increase types of widgets that our system can handle
- Detect other change pattern, if any
- Support for modifying indirect relevant codes of widgets related to changing GUIs

References

- [1] J. Nielsen, "Usability Engineering", Morgan Kaufmann Pub., 1994.
- [2] Eclipse: <http://www.eclipse.org/>
- [3] NetBeans: <http://www.netbeans.org/>
- [4] S. Horwitz, T. Reps, "The Use of Program Dependence Graphs in Software Engineering", Proceedings of the 14th International Conference on Software Engineering, 1992.
- [5] G. J. Badros, "JavaML: A Markup Language for Java Source Code", Proceedings of 9th International World Wide Web Conference, 2000.
- [6] JDK 1.4: <http://www.java.sun.com/>
- [7] A. T. T. Ying, G. C. Murphy, R. Ng and M. C. Chu-Carrol, "Predicting Source Code Changes by Mining Change History", IEEE Transactions on Software Engineering, Vol. 30, No. 9, 2004.
- [8] T. Zimmermann, P. Weibgerber, S. Diehl and A. Zeller, "Mining Version Histories to Guide Software Changes", IEEE Transactions on Software Engineering, Vol. 31, No. 6, 2005.
- [9] J. Law and G. Rothermel, "Whole Program Path-Based Dynamic Impact Analysis", Proceedings of 25th International Conference on Software Engineering, 2003.
- [10] L. Badri, M. Badri and D. St-Yves, "Supporting Predictive Change Impact Analysis: A Control Call Graph Based Technique", Proceedings of the 12th Asia-Pacific Software Engineering Conference, 2005.

A Method for Extracting Unexpected Scenarios of Embedded Systems

Toshiro MISE ^{a,b,1}, Yasufumi SHINYASHIKI ^{a,b,2}, Takako NAKATANI ^{c,3},
Naoyasu UBAYASHI ^{a,4}, Keiichi KATAMINE, ^{a,5} and Masaaki HASHIMOTO ^{a,6}

^a *Kyushu Institute of Technology*

^b *Matsushita Electronic Works, Ltd.*

^c *University of Tsukuba*

Abstract. Systems embedded in household electric appliances are used for long periods of time by various kinds of people. These environments sometimes cause unexpected failures. To save users from the appliances failures, every product is required to retain a high standard of quality and safety until their terminal period. We have developed a method named ESIM (Embedded System Improving Method) to extract the causes of failures and lead failure scenarios to make design specifications secure prior to development. The scope of the method covers hardware deterioration, partial failure, environmental exception, as well as illegal operations. In this article, we present a practical method to evaluate comprehensive failure scenarios derived from this method.

Keywords. Embedded system, Failure scenario, Unexpected phenomenon, Safety quality, Improving safety

1. Introduction

In our daily life, there is a lot of equipment containing software, especially in the case of household electric appliances, whose lifecycles extend over a period of ten years, as well as taking into account a multiplicity of users. We should be concerned about long use and the products' environment, which sometimes causes critical failure, thus putting users in danger. The Ministry of Economy, Trade and Industry investigated the reasons for correcting the embedded systems specifications in 2004. They reported that the imperfection of specifications reaches a high percentage of the whole: i.e. "imperfection of the requirements specifications" is 29.9%, and "imperfection of the design specifications" is 17.4%.

When we produce a new product line, engineers sometimes face a lot of specification changes after the design phase. Those changes affect on its productivity and costs as is commonly known. In the actual design phase, engineers sometimes realize the necessity

¹E-mail: mise@mail.mew.co.jp

²E-mail: yasufumi@mail.mew.co.jp

³E-mail nakatani@gssm.otsuka.tsukuba.ac.jp

⁴E-mail: ubayashi@ai.kyutech.ac.jp

⁵E-mail: katamine@ci.kyutech.ac.jp

⁶E-mail: hasimoto@ci.kyutech.ac.jp

to design the system to be able to watch and avoid failures. Those failure handlings account for over 70 percent of all program codes.

Those failure handling codes include exception handlings for expected behavior and handlings for unexpected behavior. *Expected scenario* is a storyline to show the basic functions of the system work. Expected scenarios are directed as behaviour of the product, or the responsibility of the user as stated in the user manual. *Unexpected phenomenon* is a consequence that should be avoided to protect the users and the product environment. Unexpected scenarios cause a chain reaction. Some chain reactions, called failure scenarios, lead to fatal errors, or a more critical situation. An unexpected phenomenon includes malfunctions caused by mis-operation, mischief, deterioration of circuits and/or machinery, partial failures, noise, changeable environment, and/or a combination of all the above. Therefore, we must analyze system behaviour based on the realistic usage of the product in order to predict unexpected phenomena[8].

In our product study, we researched the development process of the product focusing on the reasons to change the requirements after the design phase. The product had 500k steps in C and was developed in the waterfall model. The results are as follows:

- For additional definition of unexpected behaviors of the hardware: 24%
- For additional definition of unexpected usage: 19%
- For improving usability and operationability in order to preserve the system functionalities: 13%
- For adapting the software to the tuned up and/or updated hardware: 13%
- For improving maintainability and/or executionability: 10%
- Misc: 21%

According to the report, it said that the percentage of the requirements changes to cope with unexpected phenomena was 43%. We did not have any useful methods to solve these problems.

During the embedded software development stage, every company reviews and tests their products furiously. As a result, they rarely overlook any expected scenarios or exceptions in the expected usage. The reason why the percentage of the unexpected scenarios is high is that engineers tend to pay more attention to the expected scenarios during the analysis phase. If the embedded system becomes bigger and more complex, we should concentrate more on ways to extract failure possibilities and/or unexpected scenarios in order to maintain product safety. Software should be regarded as the means to detect a symptom and/or cause of failure in order to prevent failures. Thus, we have developed a method *ESIM* (Embedded Systems Improving Method) to support the pre-design phase activities to assist engineers in extracting failure possibilities caused by unexpected phenomena in embedded systems.

The rest of the paper is organized as follows: Section 2 presents related works prior to introducing the analysis process of ESIM in Section 3. Section 4 reports the results of an examination of a real embedded system in order to evaluate the method. Our discussion and conclusion follows in Section 4 and 5.

2. Related Works

FTA is used to analyze the causes of a failure specified at the root of a tree; from the root to the leaves in a stepwise manner[3]. FMEA is used to analyze the failures that occur as

a result of problems with the system components in a bottom-up manner[4]. We need a method to extract failure scenarios from atomic elements. Those scenarios are expressed by multiple phenomena rather than a simple couple of causes or a resultant phenomenon. Each individual method is not enough to analyze failure scenarios. To clarify the failure scenarios, the analysis should be proceeded by a top-down approach and a bottom-up approach repeatedly.

HAZOP (the Hazard and Operability) is a method for analyzing the safety of plants like oil refineries. HAZOP assumes deviational phenomena of the oil flow that has a negative impact on the plant, and evaluates plant safety. They apply their concerns to the phenomena as a process parameter, e.g. volume, pressure, temperature, and fluid levels. HAZOP also provides the guide words to focus the deviation, e.g. increased, decreased, lacked, reversed, etc. Then, the word derived from the guide words is used to analyze the impacts on the plant, and evaluates the safety[5]. One of the challenges in extracting unexpected scenarios is to imagine all possible situations. We use HAZOP's guide words approach to grasp an image of the unexpected phenomena.

In the field of requirement engineering, *Misuse Case* becomes a key word in negative scenarios[1]. Its actor is an agent with hostile intent toward the system under design. The method cannot collect comprehensive negative scenarios that are constructed from multiple failures. Lamsweerde and his colleagues applied KAOS to analyze obstacles in goal-oriented requirements elaboration[6]. To cope with a security problem, an anti-requirement is considered as a requirement of a malicious user that subverts an existing requirement. This problem is tackled by developing a framework to examine requirements and anti-requirements systematically[2].

An embedded system consists of hardware, software, and its users in certain environments. An unexpected scenario that we focus on may have a what/if chain of events, and various states. In the following sections, we introduce ESIM, and show the process of extracting those unexpected scenarios.

3. Unexpected phenomenon in a failure scenario

Unexpected scenarios should be extracted to ensure the embedded systems safety. We also consider productivity by avoiding the specification changes.

The analysis process of ESIM is shown in Figure 1. The analysis starts from defining the hardware components with software. Each component has links with other components. Those components and links are the places where failures may occur. When a failure occurs, it may cause a chain reaction. The chain will create an unexpected scenario, and some of the scenarios will be failure scenarios.

3.1. Defining hardware components and their relations

One of the novel points of ESIM is that the environment of the system is regarded as a component embedded in the system. For example, each kind of user can be a component that has a state, which in turn sends events that influence the system. Each component has expected phenomena defined in the product planning documents. ESIM also regards an interaction between components via their relation as a communication. Therefore, an interaction scenario consists of a communication chain in which one component sends

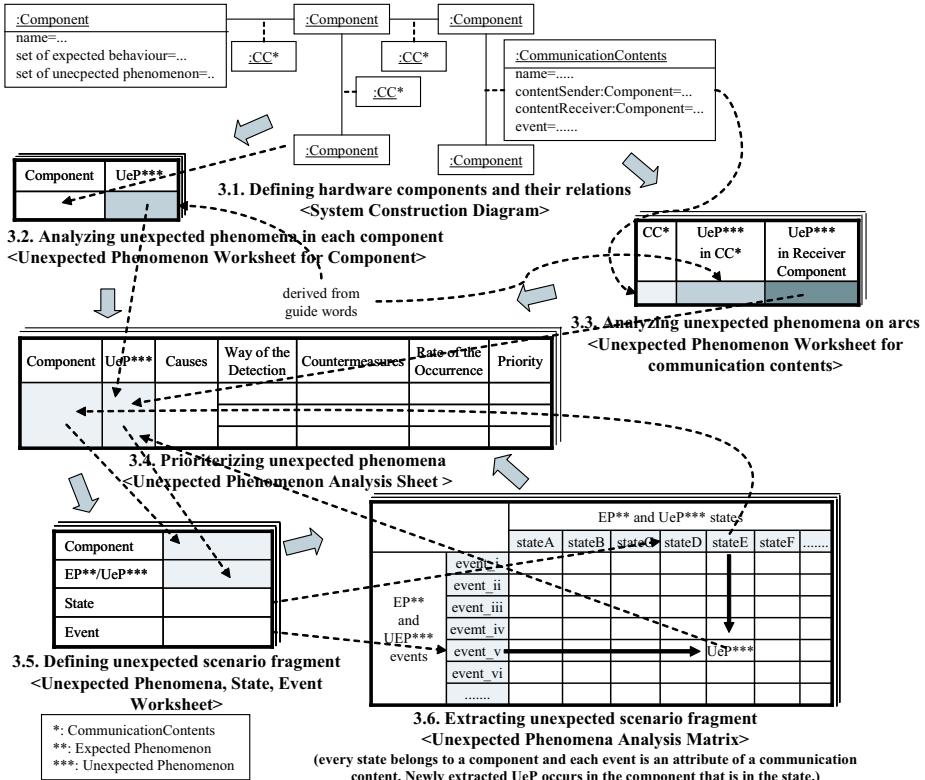


Figure 1. Analysis process and products of ESIM

communication content to another component. The ESIM provide a way to analyze communication contents on each arc after defining the system structure.

3.2. Analyzing unexpected phenomena in each component

An *unexpected phenomenon worksheet* for every component is a tool to record derived unexpected phenomena in each component. The worksheet has two columns; one for an identifier of a component, and one for an unexpected phenomenon. When we analyze an unexpected phenomenon, a guide word table shown in Table 1 supports us. The guide word table is developed by the past experiences of unexpected phenomena.

The first column of the guide word table provides the focal points of the analysis and the second column provides examples of unexpected phenomenon. Row A of the table is interpreted as a way to show that “its focal point” may cause “an unexpected phenomenon.” For example, if a component has a function, we can assume that *the function* may cause the component *not to work/to work insecurely/to be in frozen/to work unstably*. If the component consists of multiple sub-components, we can make an assumption based on the guided scenarios: e.g. *its part* may cause the component *to behave a malfunction/to behave abnormally/to terminate its function*.

Table 1. Guide words examples for components

Components	Focal Point	Guide Words for Components
Electrical Component	Behaviour	terminate, unstable, frozen,...
	Physical Failure	aging, partial failure,...
	Logical Failure	skipping signals, receiving noise, sending excess signals,...
Mechanical Component	Behaviour	terminate, unstable, locked,...
	Physical Failure	wared, deteriorated, stained,...
	Logical Failure	slipping, switching, shifting,...
Environment	Natural Environment	humidity, temperature, brightness, air pressure,...
	Mechanical Environment	vibration, pressure, acceleration,...
	Electrical Environment	electrical noise, radio noise, ...
	Load Environment	over load, obstacle, contrary load, ...
	Power Environment	declining power, power off, increasing power, surging,...
Operational Environment	construction environment	malconstruction,...
	setting up environment	malinstallation, mis-attachment,...
	misc.	lacking resources, collision, inconsistent operation,...
User	Intention	intentional maloperation, accidental maloperation,...
	Purpose	exceptional usage,...
	Characteristics	old, infant, handicapped,...

Table 2. Guide words examples for communication contents

Propagating data from/to components	Guide Words for Communication Contents
events, signals	out of range, undefined order, out of timing, ...
data, quality	out of range, meaningless,...
frequency	continuously, repeatedly, intermittently, occasionally, ...

3.3. Analyzing unexpected phenomena on arcs

Analyzing unexpected phenomena in components is not enough, since it takes a broad view to focus on phenomena neither on the arcs between components, nor chain reactions among the arcs of the system network. Arcs provide a sequence of propagation of the phenomenon.

An *unexpected phenomenon worksheet* for every communication content is a tool to record derived unexpected phenomena on the arc. The worksheet has three columns; one for an identifier of communication content, one for unexpected phenomena of communication contents, and one for a content receiver.

When analyzing unexpected phenomena, guide words can be applied. The second column of Table 2 provides guide words to analyze unexpected phenomena on an arc. Every arc has communication contents. An unexpected phenomenon is transmitted along the arcs from one component to another component by the communication content. Therefore, an unexpected phenomenon on an arc is defined as a property of the communication content.

3.4. Prioriterizing unexpected phenomena

Up to this point of the analysis, we have gotten unexpected phenomena in components and arcs. To prioriterize them, *the criticality, prioriterizing worksheet* is applied. The sheet has two parts. The first part is for analyzing causes of a phenomenon with three columns: a component identifier, an unexpected phenomenon, and its cause. If the cause is regarded as a never have occurred situation, the possibility of the phenomenon can be ignored. The second part is for analyzing the ways to catch the phenomenon, a safeguard against the phenomenon, a rate of the incidence, and priority whether the phenomenon should be considered or not. If the ways to catch the phenomenon and safeguard against the phenomenon are defined and are designed into the system, we can cut the chain of reaction at the component. An analyst can delay his/her final decision to install those functions into the system until they have all the facts of the chain of reaction. Even if the rate of the incidence is low, it may cause a critical incidence in the system.

3.5. Defining unexpected states and events

Before analyzing the combinations of the phenomena, we need an event and a state for each phenomenon. The state represents the situation of a component in which the unexpected phenomenon is occurring. The event takes place in order to send the phenomenon to other components. The event is an attribute of the communication contents in the system construction diagram. Thus, the owner of the state is a receiver component and the owner of the event is a sender component of the communication content. *The unexpected phenomena, state, and event worksheet* is a tool to define the state and the event for each phenomenon. The analyst can correctly perceive the reality of what has happened in the system by focusing on the relationship between an event and a state through the worksheet.

3.6. Extracting unexpected scenario fragment

An unexpected scenario is a construction of state-event-phenomenon, referred to as chained relationships. One sentence of the scenario is a scenario fragment. It consists of the state of a component, an event sent from another component, and a phenomenon that is caused by the event. *The unexpected phenomena analysis matrix* represents a state of a component in its column, an event in its row, and an unexpected phenomenon in its cell. A phenomenon in each cell is derived from a small what if scenario: “if the component in the state receives the event, a phenomenon will occur.” The phenomenon will be a part of the unexpected scenario or failure scenario as a scenario fragment. Then, the phenomenon is analyzed in *the criticality, prioriterizing worksheet*. If the phenomenon has a high priority, a new state and a new event are produced in *the unexpected phenomena, state, event worksheet*. Thus, the analysis process proceeds from prioriterizing unexpected phenomena to extracting unexpected scenario repeatedly. Finally, the unexpected scenario fragments compose the unexpected scenario. If it leads to fatal errors or a critical situation, we should set a check point in the system to monitor and terminate the scenario. Thus, in the final process, *the criticality, prioriterizing worksheet* shows the required specifications to save the product from the failure.

4. Evaluation of ESIM

4.1. Overview of the examination

We evaluate ESIM by comparing the results of cases with/without ESIM. The target embedded system is developed as a new product in C. It has few new technical issues in its hardware, however, we must add some complex functions that are not embedded in its previous software. The size of the software is middle class for controlling facilities. Team-A applied the method and wrote inquiry for the formal review of the specifications. The other team, Team-B, did not use the method but developed a prototype to eliminate the risk of the lack of the functionalities. After the prototype evaluation, the prototyping team updated the requirements specifications for the second version. Then, two members of the quality management section reviewed it and made inquiry for the formal review.

The purpose of both teams is to design the product with high safety quality. We evaluate our method, ESIM, by comparing inquiries submitted from two teams and analyzing how well they defined unexpected scenarios. The evaluation process is shown in Figure 2.

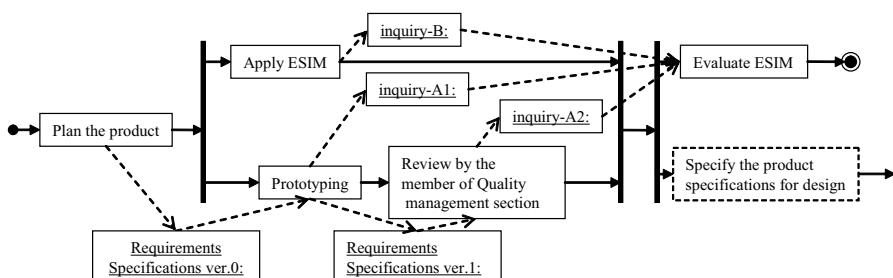


Figure 2. Examination flow

4.2. Results

4.2.1. Results of Team-A

One engineer took 8 hours to understand the first requirements specifications, and 24 hours to apply ESIM in analyzing unexpected scenarios. He extracted 25 expected states, 20 expected events, 22 unexpected states, and 24 unexpected events for *the unexpected phenomena analysis matrix*. Though the matrix had 2068 cells with 47 states and 44 events, 361 cells (17.5%) were occupied with unexpected phenomena. There were 44 complex unexpected phenomena of 361 phenomena. Other cells with no phenomena mean that nothing would happen, if the state in the column receives the event of the row. After the analysis, 48 items were listed as incomplete or undefined requirements to be considered.

4.2.2. Results of Team-B

According to the interview with a member of Team-B, they eliminated the same 22 items in Team-A's 48 items as, requirements not to be implemented in the system before prototyping. They also pointed out the same fourteen items in the other team's list as incomplete requirements during their prototyping. The rest of twelve items were not mentioned in the updated requirements specifications. Those items consisted of five items that were related to the inconsistent states among sensors caused by hardware failure; four items were rarely caused by complex phenomena, two items that were related to the unique operation, and the other items were related to the deterioration of components. Items that were identified in the results of Team-B, but were not mentioned in the list of Team-A, were related to the usability. In general, prototyping is effective to improve usability.

When two members of the quality management section reviewed the second version of the requirements specifications, they took eight hours to understand the specifications and four hours for analysis. We focused on whether they could identify the rest of twelve items. Actually, there were unique items. One of them originated from ambiguity within the first version of the requirements specifications. The other item related to the domain specific usage. The former item can be eliminated by verification of the specifications. The latter item designates that our method needs to mature to have domain specific knowledge. On the other hand, they listed five items as unexpected scenarios. Four of them were matched by the list of Team-A. The rest of them were listed according to the member's experience on the system failures. This item also tells us that the guide words are the key of our method.

Through the results of the examination, Team-A could identify more unexpected scenarios than Team-B.

4.2.3. Evaluation

The quality management members spent one-sixth of the time of Team-A for analysis. In interviews with the members, they said that even if they had spent more time, they would not have been able to point out other faults. This means they could not detect more specific situations.

After the examination, we taught ESIM to the quality management members. According to their trial application of ESIM to the same specifications, they identified the same unexpected scenarios as in the list of Team-A. Their list has almost the same concreteness as the list of Team-A. Therefore, we can say that ESIM was effective for engineers. ESIM may be able to provide the way, and give concrete images of system behaviors.

Though the engineers of Team-A had to spend 32 hours to make their inquiry, while another engineer of the product development team spent less time, for their inquiry: a user who has knowledge and/or experience in the domain will be able to complement the method, event though the present ESIM still needs more guide words.

In the results of Team-A, the *unexpected phenomena analysis matrix* was filled 17%. Before the examination, ESIM seemed to show a risk of exploding the *unexpected phenomena analysis matrix*, thus taking a long period for analysis. The results showed that the risk was reasonably small.

5. Discussion

The Unexpected phenomena occurred in a component are defined through the guide words. In *the unexpected phenomena analysis matrix*, we can lead possible side effects caused by those unexpected phenomena. According to our experiences, it is hard to aware those side effects caused by a phenomenon occurred in a component somewhere in the system. The columns of the matrix provide a perspective to check the possibility of side effects of an event for every state of the components, and the rows also provide a perspective to review unexpected phenomena of a state caused by every event. Those multiple perspectives are important to cover possible phenomena of the system, as a result, we will be able to guarantee a safety of the product environment.

The other our challenge is to concern the complex unexpected phenomena. Those phenomena have chain reactions. The matrix supports us to analyze those chains of reactions.

Figure 3 shows the example of an electric kettle. The kettle equips a filled water sensor that senses the water level. If the sensor senses the filled water state, it turns off the heater to prevent from running over of the boiled water. When a user supplies water to the kettle, a thermistor senses temperature falls. The event turns on the heater, then the kettle transits its state to the boiling water state. On the other hand, there is a possibility of an unexpected event that something tips the kettle. In case of the direction of its list, the filled water sensor cannot recognize that the real water level reaches to the danger. Now, the heater is still in the state of “boiling.” Finally, the boiled water runs over from the kettle and may burn the use. *The unexpected phenomena analysis matrix* supports to analyze this kind of scenario.

If we can aware the possibility of the unexpected scenario, we would be able to redefine the product specifications to cope with the scenario. Though this kind of scenario is more complex in the real product specifications, the result of our examination shows that the perspective of the combination of events and states is effective to realize the unexpected phenomena.

event \ state	The kettle is in an empty state.	The kettle is in the filled of water state.	The water level sensor is in the state of sensing the level.	The kettle is not in the filled of water state and the heater is on.	The water level sensor: sensing the level, but the real water level is out of recognition, The kettle is not in the filled of water state and the heater is on, The kettle is tipped.
Supply water	Turning on the heater	The water may run over.	Reaching the filled of water level	Reaching the filled of water level	Before receiving this event, the kettle may burn the user.
Water level sensor turns off the heater	SAFE	stop boiling water	SAFE	SAFE	SAFE
Something tips the kettle	SAFE	The water may run over.	Out of recognition of the real water level	The water level reaches to the filled water level	-----
A user touched the kettle.	SAFE	SAFE	SAFE	SAFE	The kettle may burn the user.

Figure 3. The analysis example of complex phenomena

6. Conclusions

In the embedded system, an unexpected phenomenon is caused in a component with a certain state receiving various events. One of the novel ideas of ESIM is that the system is divided into not only its components but also relationships between components. The phenomenon, that the system design specification does not mention, is decomposed into the state where the phenomenon is occurred and the event that propagate the phenomenon from a component to other components. *The unexpected phenomena analysis matrix* integrates those parts into one system focusing on the states and the events. Therefore, the matrix includes the whole possible phenomena inside the system. To prevent to explode *the unexpected phenomena analysis matrix*, we apply *the criticality, prioriterizing worksheet*. As the result of the analysis examination, the matrix space is saved from the explosion.

In our future work, improving the guide words and developing the ESIM tool (ESIM.T) are essential. The tool will support traceable analysis processes.

References

- [1] Alexander, I. : "Misuse cases, use cases with hostile intent," *IEEE Software*, vol.20, no.1, pp.55–66, 2003.
- [2] Crook, R., Ince, D., Lin, L., and Nuseibeh, B. : "Security Requirements Engineering: When Anti-Requirements Hit the Fan." *Proc. of the 10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02)*, pp.203–205, 2002.
- [3] Leveson, N. G. : "Fault Tree Analysis," *Safeware: System Safety and Computers*, Addison-Wesley, pp.317–326, 1995.
- [4] Leveson, N. G. : "Failure Modes and Effects Analysis," *Safeware: System Safety and Computers*, Addison-Wesley, pp.341–344, 1995.
- [5] Leveson, N. G. : "HaZards and Operability Analysis," *Safeware: System Safety and Computers*, Addison-Wesley, pp.335–341, 1995.
- [6] Lamsweerde, A. V. and Letier, E. : "Handling Obstacles in Goal-Oriented Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 26, no. 10, pp.978–1005, 2000.
- [7] Mise, T., Sinyashiki, Y., Hashimoto, M., Ubayashi, N., Katamine, K., and Nakatani, T. : "An Analysis Method with Failure Scenario Matrix for Specifying Unexpected Obstacles in Embedded Systems" *Proc. of the Asia-Pacific Software Engineering Conference (APSEC'05)*, pp.447–456, 2005.
- [8] Nakatani, T., Mise, T., Shinyashiki, Y., Katamine, K., Ubayashi, N., and Hashimoto, M. : "Toward defining a system boundary based on real world analysis," *Proc. of the FOSE2005*, Japan Society for Software Science and Technology, Kindai Kagaku Sha, pp.221–226, 2005 (in Japanese).
- [9] Sumi, T., Hirayama, M., and Ubayashi, N.: "Analysis of the external environment for embedded systems," *IPSJ SIG Technical Reports*, 2004-SE-146, pp.33–40, 2004 (in Japanese).

Automatic Creation of a Countermeasure Plan against Process Delay Based on Fast-Tracking

Rihito YAEGASHI ^{a,1}, Daisuke KINOSHITA ^b Yuichiro HAYASHI ^b
Keiichi NAKAMURA ^b Hiroaki HASHIURA ^b and Seiichi KOMIYA ^b

^a Toyota Technological Institute, Information Processing Center

^b Graduate School of Engineering, Shibaura Institute of Technology

Abstract. Recently, some techniques for a very short-period software development which is called “Agile” have been proposed. In order to avoid risks, it has been considered as a taboo that the Post-work is started before the Pre-work is completed. However, we might have to break a taboo to recover a process delay. Therefore, this paper proposes a method to make a successful schedule plan automatically. The process delay can be recovered by breaking the taboo. This paper also proposes a method which is able to make a countermeasure plan automatically against a process delay by means of ‘Fast-Tracking’, and finally proves that our method is effective in a software project management by showing an example of the system.

Keywords. Schedule Planning for Software Development, Software project management, Fast-Tracking.

Introduction

There are two means to make countermeasures against a process delay of a software development project. One is “Duration compression”[1] and the other is “SHUKUTAI”². The method of “Duration compression” includes two techniques of ‘Crashing’[1][2] and ‘Fast-Tracking’[1]. “SHUKUTAI” is a method to complete all the work before the fixed date by reducing some parts of modules and functions. ‘Crashing’ has been the most useful solution for a process delay so far. In case even the ‘Crashing’ was useless, “SHUKUTAI” was used. In order to avoid risks, it has been considered as a taboo that the Post-work is started before the Pre-work is completed. However, to complete a software development in a short period, it is necessary to recover a process delay even if it causes this taboo.

Therefore, this paper proposes a method to make a successful schedule plan automatically, so that a project, which is behind a schedule, is completed on schedule by means of ‘Fast-Tracking’, and finally proves that our method is effective in a software project management by showing an example of the system.

¹Toyota Technological Institute, Hisakata 2-12-1, Tenpaku-ku Nagoya-city, Aichi 468-8511, Japan; E-mail: rihito@toyota-ti.ac.jp.

²To change a contract with a client and to reduce functions and modules.

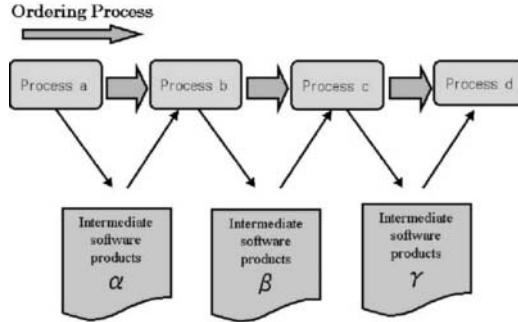


Figure 1. Ordering constraints based on Intermediate Software Products.

Table 1. An example of worker allocation patterns (in case of CT1).

Skill Level	Pattern-1	Pattern-2	Pattern-3
Rank a (Excellent)	1	1	
Rank b (Good)			2
Rank c (Poor)		1	
Rank d (Impossible)			
Necessary days	3	2	2

1. The Constraints which Problems of Schedule Planning Have and a Method to express the Constraints

This section discusses the constraints which problems of schedule planning have, and how to express them. Details are described in [3]. In this section, the authors refer to only “Constraints concerning an order of executed activities”.

There are some intermediate software products to determine the order of the works are conducted. Figure 1 shows one example of the constraints existing between software development works and intermediate software products. To conduct work ‘b’, an intermediate software product ‘α’, which is necessary to execute work ‘b’, must be completed before work ‘b’ is started. This constraint is called a ‘pre-condition’. Another constraint is called a ‘post-condition’. An intermediate software product ‘β’ which is produced by the execution of work ‘b’ must be obtained after an execution of work ‘b’.

2. How to Specify the Number of the Workers Allocated to Each Process

We evaluate the capability of each worker according to the four-level in grades of Excellent, Good, Poor, and Impossible in this research. Excellent-level means that the worker can conduct the work by himself quickly. Besides, he can teach other workers how to conduct the work during his own work. Good-level means that the worker can manage to conduct the work by himself. However, he cannot teach other workers. Poor-level means that the worker can manage to conduct the work with another worker’s instruction. Impossible-level means that the worker can not conduct the work even with another worker’s instruction. Usually, the number of workers allocated to each process is more than two. We suppose the allocation patterns of the workers in Table 1.

An example is shown in Table 1 and explained. Table 1 shows that there are three allocation patterns of the workers for this process. The Pattern-1 shows that this process

SA	SD1	SD2	DR	DD1	UT/ CT1	DD2	PE	UT/ CT2	DT	IT
Pattern-1 Pattern-2 Pattern-3										
Pattern-2	Pattern-1	Pattern-2	Pattern-3	Pattern2	Pattern-2	Pattern-3	Pattern-1	Pattern-3	Pattern-1	Pattern-2
A,B	A	A	A,B,C	B	B,C	D	C	B	D	E,F

Figure 2. A way to specify the number of the workers allocated to each process.

needs 3 days, if a worker whose skill is an Excellent-level conducts this process. The Pattern-2 shows that this process needs 2 days, if a worker whose skill is Excellent-level and a worker whose skill is a Poor-level conduct this process together. The Pattern-3 shows that this process needs 2 days, if two workers whose skills are Good-level conduct this process. The number of the workers is specified on the basis of the pattern of allocating workers as stated above. In this system, it is executed to allocate the workers by trying all the patterns. Let us suppose a plan (shown in Figure 2) of the activities order and the allocation patterns of the work. The plan created by using GA is only the first and the second layer. Figure 2 shows that the first layer expresses the execution order of activities and the second layer expresses the allocation patterns of the work. For example, Figure 2 shows that the allocation patterns of the work to CT1 have three patterns; Pattern-1, Pattern-2, Pattern-3. The third layer expresses the allocated workers according to the allocation patterns of the work (This data of the allocation patterns is stored in the database). In Figure 2, the same words, Pattern-1, Pattern-2, and Pattern-3 and so on, are used in all the process. However, the actual allocation patterns of the work differ from each process. The allocation patterns of the work to CT1 are shown in Table 1. In this case, these workers are allocated according to Pattern-3. The third layer shows that the allocated workers are only two people B and C who satisfy the constraints. The worker(s) allocated to each process is/are determined in this way.

3. How to Make a Countermeasure Plan against a Process Delay by Means of 'Fast-Tracking'

In our system, an order of executed activities has been decided based on "Constraints concerning an execution order of executed activities", and a schedule has been made according to the principle that Post-work does not begin before Pre-work is finished. It has been considered as a taboo that Post-work is started before Pre-work is completed in order to avoid risks. Therefore, this paper proposes a method to make a successful schedule plan automatically, so that a project, which is behind a schedule, is completed on schedule by means of 'Fast-Tracking'.

This section describes that it is possible to make the software development plan by means of 'Fast-Tracking', even if the process delay which cannot be recovered by 'Crashing'. Also, this section proposes the framework of a system which is able to make a countermeasure plan automatically against a process delay by means of 'Fast-Tracking'.

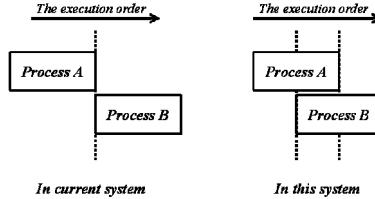


Figure 3. Newly Interpretation of the constraints to perform 'Fast-Tracking'.

3.1. Addition of constraints to execute 'Fast-Tracking'

Section1 and [3] described that the problems of schedule planning for a software development can be regarded as the problems which need to satisfy the conditions of the constraints and to allocate resources in the best way. In our system, in order to use 'Fast-Tracking', it is necessary to improve the conditions of 'Constraints concerning an order of executed activities' which described in Section 2.(1). Our system has a principle that a Post-work does not begin before a Pre-work is finished.

However, it is necessary to improve the conditions of 'Constraints concerning an order of executed activities' as the Post-work is able to begin before the Pre-work is finished in Figure 3. In this way, in some of the processes, it is possible to make a plan that the Post-work can be started before the Pre-work is finished. When you automatically make a countermeasure plan against a process delay by means of 'Fast-Tracking', it is necessary to add the new constraints in a Post-work which is started before a Pre-work is completed. It is necessary that the Post-work can begin before a Pre-work is finished without obtaining intermediate software products. The workers who manage the Post-work needs to be familiar to the work. It means that the workers of the Post-work needs to be familiar to the condition of intermediate software products of the Pre-work. It is reasonable because there is no guarantee that the intermediate products which are demanded in the Pre-work are obtained. In order to solve this problem, we need to add a new constraint that the worker of the Post-work must have the equal or better skill than the worker of Pre-work.

3.2. Estimates of a technical Risk in each process

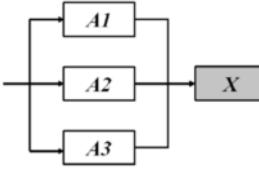
In this paper, a Technical Risk of each process is estimated by the value from 0 to 1. The value 1 has the highest possibility of the failure, and the value 0 has the highest possibility of the success.

3.3. A computational method of a Network Risk and a Total Risk

It is reasonable that the execution of 'Fast-Tracking' has the higher risk than the execution of a Post-work after a Pre-work is completed. It is because there is no guarantee to obtain intermediate products of the specification which is demanded in a Pre-work. However, in case that it is indispensable to execute 'Fast-Tracking', it is executed in the process which has the lowest Total Risk. The Total Risk means both a Network Risk which depends on positions in the network and a Technical Risk which each process has. A computational method of a Network Risk and a Total Risk is described as follows.

Table 2. Technical Risk.

Process	Technical Risk
S1	α
SD1	β
SD2	γ

**Figure 4.** The connected processes in series.**Figure 5.** The parallel processes.

A Technical Risks in the process A1, A2, and A3 is given beforehand as α , β , and γ in Table 2. It is necessary to complete all the works of A1, A2, and A3 so that work X is executed in series (Figure 4). Therefore, possibility A that all the works of A1, A2, and A3 are completed is as follows.

$$A = (1 - \alpha)(1 - \beta)(1 - \gamma)$$

In addition, risk R is as follows in series.

$$R = 1 - A = 1 - (1 - \alpha)(1 - \beta)(1 - \gamma)$$

It is also necessary to complete all the works of A1, A2, and A3 so that work X is executed in the parallel composition (Figure 5). Therefore, probability A that all the works of A1, A2, and A3 are completed is as follows.

$$A = (1 - \alpha)(1 - \beta)(1 - \gamma)$$

In addition, risk R X is as follows in the parallel composition.

$$R = 1 - A = 1 - (1 - \alpha)(1 - \beta)(1 - \gamma)$$

The computational method of the Network Risk is the same no matter how processes are connected.

$$R_i = 1 - \prod_{j=1}^j A_i \\ 1 - \prod_{j=1}^i (1 - r_j)$$

R_i : Network Risk A_i : Utilization rates of process i r_j : Technical Risk of process i

In this system, a Technical Risk at each process is estimated beforehand, and given to the system. The method of estimating a Technical Risk in each process is described in [5]. The Total Risk means both a network Risk which depends on positions in the

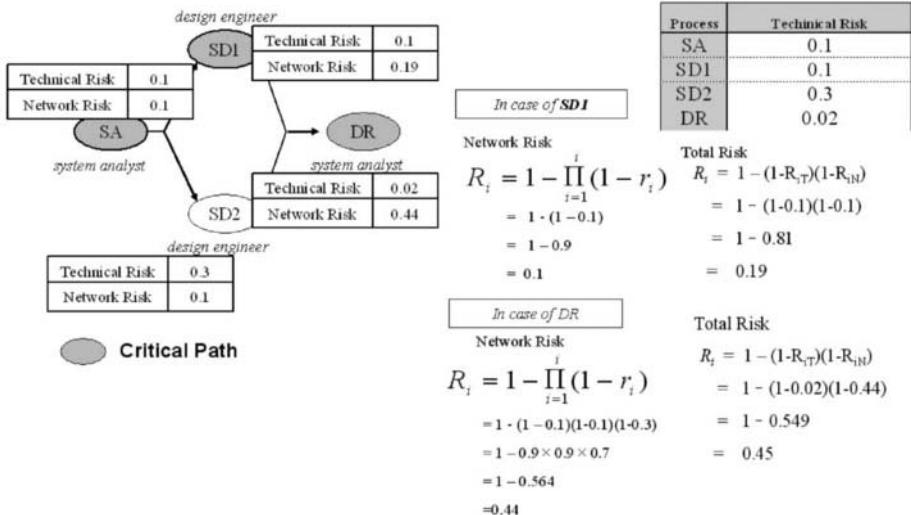


Figure 6. The measurement method of Network Risk and Total Risk.

network and a Technical Risk which each process has. In process i, a Technical Risk is R_{iT} and a Network Risk is R_{iN} , then a total Risk R_i is shown as follows.

$$R_i = (1 - R_{iT})(1 - R_{iN})$$

An appropriate reason for this expression is as follows. $(1 - R_{iT})$ corresponds to the utilization rates of process i which includes a Technical Risk R_{iT} . $(1 - R_{iN})$ corresponds to the utilization rates of process i which includes a Network Risk R_{iN} . On the other hand, $(1 - R_{iT})(1 - R_{iN})$ corresponds to the utilization rates of process i which includes a Technical Risk R_{iT} and Network Risk R_{iN} .

$$1 - R_i = (1 - R_{iT})(1 - R_{iN}) = 1 - R_{iT} - (1 - R_{iT})R_{iN} \leq 1 - R_{iT} \quad (i)$$

(i) is always approved, and $R_i \geq R_{iT}$ is always approved as well. Furthermore $R_i \geq R_{iN}$ is always approved. A risk of process i with both Technical Risk R_{iT} and Network Risk R_{iN} is never smaller than a risk of process i only with Technical Risk R_{iT} or Network Risk R_{iN} . This shows that it is not meaningless that both a Technical Risk and a Network Risk are considered at the same time.

Figure 6 shows that a Technical Risk of SD1 is 0.1. To start SD1, it is necessary that Pre-work SA is finished, and a Network Risk of SD1 0.1 is obtained. In case of DR, it is necessary that Pre-works of SA, SD1, SD2 are finished, and a Network Risk 0.44 is obtained.

It is assumed that a process delay occurs while the workers are executing the work of SA in Figure 6. Figure 6 shows that the method to decide which process might be executed the 'Fast-Tracking'. In this case, it is assumed that SD1 and DR are the processes on the critical path, and the 'Fast-Tracking' is executed in either SD1 or DR. Figure 6 also shows that the Total Risk of SD1 is 0.19 and the Total Risk of DR is 0.45. Therefore, the 'Fast-Tracking' is executed in SD1 in Figure 6.

4. Automatic creation of a schedule plan as countermeasures against a process delay and its evaluation

4.1. System evaluation by an example

In this section, it proves that it is effective to solve a schedule-planning problem by showing an execution example that a system automatically makes a schedule plan as countermeasures against a process delay by means of 'Fast-Tracking'.

"Hypothetical example"

Suppose there is a project to develop a demonstration system to present in an exhibition which is to be held from 1st of August. This project is organized on 2nd July and the system should be completed by 31st of July. The necessary activities for this system development are system Analysis, outline design, outline design review, detailed design, coding, individual testing, and comprehensive testing.

The constraints to be taken into consideration for planning the project are listed below. Since the demonstration system requires a high performance, the performance estimation should be done on part of the detailed design and the result should be reflected in the coding work. The performance estimation requires a performance analysis technology. The hardware that is utilized in the exhibition is a new machine under the development for this exhibition. It can be utilized from 26th July at the earliest. This machine should be transported to the exhibition hall by 31st July. Because of the constraint about the period that the machines will be used in the exhibition, it has been decided that a test support tool will be developed for an efficient testing. Under the conditions described above, a plan has developed and the project began.

The role of the human resources for each project are listed below

- The Project Manager is in charge of planning project, managing the progress, and giving approvals.
- The Design Engineer is generally responsible for designing and coding works.
- The quality Assurance Engineer is generally responsible for planning and executing tests.

A, B, C, D, G, and H is appointed as the Design Engineers, and E and F as the Quality Assurance Engineer. They are expected to perform the activities according to their roles. In the project, the following machines and tools are utilized: development machines, machines which are actually utilized in the exhibition, tools used for performance estimation, and tools which will be developed for an efficient testing. Each resource of this project has the attributes listed in Table 3 and Table 4, respectively.

Although the software development project was carried out according to the schedule shown in Figure 7, process delay is caused at a finishing point of 11th July. It is because worker A could not participate in the work because of the sickness, although worker A, B, and C take charge of DR (Design Review) in the original plan. Then, only the two workers B and C take charge of DR(Design Review), and DR needs four days to be finished. A critical path is the path of works without any slack time, and the critical path is measured by using PDM (Precedence Diagramming Method)[6]. A process DR (Design Review) is located on a critical path as you can understand by PDM shown in Figure 8. By the way, the process delay of a process DR means the process delay of the

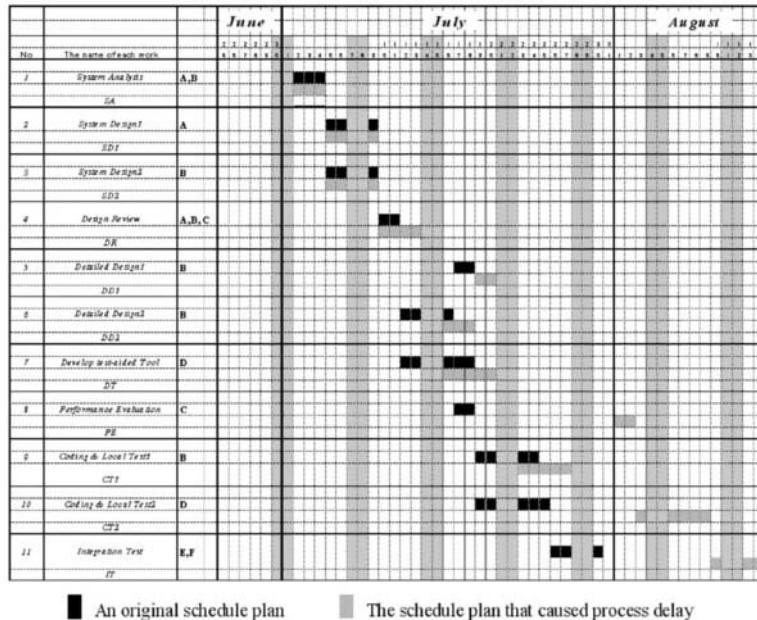


Figure 7. An original schedule plan and the schedule plan that caused process delay.

Table 3. The value of human resources attributes in the example project.

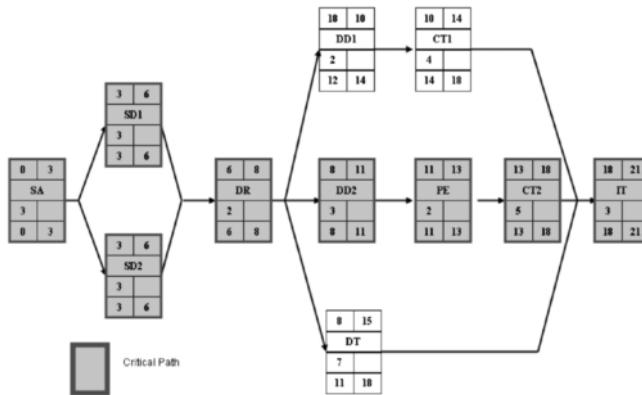
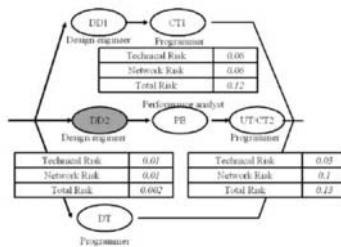
Name	Skill	Rank	Available Time	Cost
A	System Analyst	a	~7/13	2300
	Design Engineer	b		
	Programmer	c		
B	System Analyst	b	Any	3300
	Design Engineer	a		
	Programmer	a		
C	System Analyst	b	7/10-7/18 &1-	3300
	Design Engineer	a		
	Performance Analyst	a		
D	Design Engineer	b	7/2-7/25 7/28-	3300
	Programmer	a		
	QA Engineer	b		
E	QA Engineer	b	Any	3300
F	QA Engineer	b	Any	3300
G	Design Engineer	b	7/13-	3300
H	Design Engineer	b	7/12-17	3300

whole project, as the process delay of the process on the critical path means the delay of a whole project. Figure 7 shows that a delay of 1 day eventually causes a delay of 15 days, owing to the constraints concerning the available period of worker C, because worker C cannot conduct the work of PE(Performance Evaluation) until 1st August owing to the constraints concerning the available period of worker C who takes charge of the work of PE.

Therefore, a system automatically makes a schedule plan to eliminate the process delay by adding a new resource or several new resources (i.e. worker(s) in this case) to the process after 12th July. It is called 'Crashing' to eliminate a process delay by adding a new resource or several new resources to the process on a critical path. The system that

Table 4. The value of non-human resources attributes in the example project.

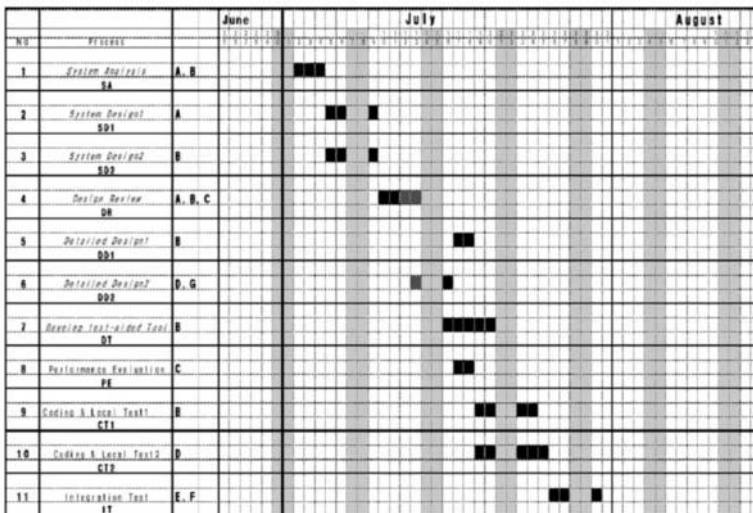
Name	Available time
Performance Analysis Tool	7/1(=7/18,8/1-8/5)
Test Support Tool	-7/23
Machine for Development	Any
Machine for Exhibition	7/24-7/30

**Figure 8.** An original schedule plan expressed based on PDM.**Figure 9.** The process which carries out 'Fast-Tracking'.

we have developed has not made a schedule plan as countermeasures against the process delay. It means the process delay cannot be recovered even by means of 'Crashing'.

Therefore, it is required to make a schedule plan as countermeasures against the process delay by means of 'Fast-Tracking'. The system that we have developed makes a schedule plan as a countermeasure (Figure 10). 'Fast-Tracking' is executed to DD2 in countermeasures. Figure 9 show that a Network Risk is calculated based on a Technical Risk, and a Total Risk is calculated based on both a Network Risk which depends on the position and a Technical Risk at each process. Figure 9 also shows that the Total Risk in DD2 is the lowest of DD2, PE, and CT2. Therefore, the 'Fast-Tracking' is executed to DD2.

The countermeasure (See Figure 10) show that the system development can be finished by 31st July. The countermeasure adds the extra worker G to DD2, and also 'Fast-Tracking' is executed to DD2. In the original plan (Figure 7), DD2 was supposed to be completed for three days by only worker B. In countermeasure, worker D and G take

**Figure 10.** Countermeasures

charge of DD2, and DD2 can be completed in two days. In addition, countermeasure 1 is executed so that DD2 is started before DR is completed. This is 'Fast-Tracking'.

5. Conclusion

It has been considered as a taboo that the Post-work is started before the Pre-work is completed in order to avoid risks. However, it is necessary to recover a process delay even if it causes this taboo to carry out software development in a very short-period. It is called agile nowadays. Therefore, this paper have proposed a method which is able to make a countermeasure plan automatically against a process delay by means of 'Fast-Tracking', and finally proved that our method is effective in a software project management by showing an example of the system.

References

- [1] Project Management Institute, "A Guide to the Project Management Body of Knowledge (PMBOK Guide) 2000 Edition," 2000.
- [2] R.Yaegashi, D.Kinoshita, H.Hashiura, K.Uenosono, S.Komiya, "Automatically Creating a Schedule Plan as Countermeasures by Means of 'Crashing' against Process Delay," Joint Conference on Knowledge-Base Software Engineering 2004 (JCKBSE'04), pp. 24–36, Aug. 2004.
- [3] S.Komiya, N.Sawabe, and A.Hazeyama, "Constraints-Based Schedule Planning for Software Development," IEICE Trans INF & Syst., vol. J79-D-I, No9, pp. 544–557, Sep. 1996.
- [4] A. Hazeyama, S. Komiya "Workload Management Facilities for Software Project Management," IEICE Tran INF & SYST, vol. E81-D.No12, pp. 1404–1414, Dec. 1998.
- [5] W.H.Roetzheim "Structured Computer Project Management," 1992.
- [6] M. Yoshizawa, S. Miyazawa "Modern Project Manegement Literacy I,II," IITEC, 2001.

Detecting Defects in Object Oriented Designs Using Design Metrics

Munkhnasan CHOINZON ^aand Yoshikazu UEDA ^a

^a *Ibaraki University, Japan*

Abstract. In order to decrease the cost, it is recommended to detect defects in the phase they occur. This paper presents a metrics-based approach to detect defects in OO designs. We first identify a list of OO design defects which have a significant impact on the design quality based on the violations of many design guidelines and best practices expressed by experts. Then, we define the metrics for automatically detecting each of these defects. Several number of well-known metrics are used. For some defects, however, there are no suitable design metrics. Therefore, we define new design metrics to detect these defects. Moreover, thresholds to judge whether a metrics value indicates a critical situation, or not are defined for each of the metrics. By defining metrics on each design defect, many design rules and heuristics, and flaws which are described qualitatively can be evaluated quantitatively. On the other hand, we will show that intended application of metrics becomes clear.

Keywords. Quality model, quality attributes, OO design defects, OO design metrics

1. Introduction

The last goal of the software development is to produce high-quality software. However, assessing the quality of a software at an early stage, not after a product is ready, is essential for saving the cost. If design defects are not fixed in the design phase, the cost for fixing them after delivery of the software is 100 times higher[3].

Today, object-oriented(OO) design plays an important role in the software development environment. Thus, models and methods which assess and evaluate the quality of a software design developed using OO technology are becoming a critical issue since OO paradigm introduces new design concepts such as encapsulation, inheritance, polymorphism, messaging, and composition, which are fundamentally different from those in structured methods.

With OO approach, design defects can be recognized at the end of the stage because OO methodologies require significant effort early in the development life-cycle to identify objects and classes, attributes and operations, and relationships.

OO design measurement can do help to identify design defects by measuring design characteristics.

In the literature, a large number of OO metrics have been proposed. For many of these metrics, however, interpretation and analysis of results are not clearly defined. In other words, there is no criteria or threshold on most metrics to judge whether a metrics

value indicates a critical situation or not. Therefore, intended application of metrics and its effectiveness are very difficult to be determined[7,8].

On the other hand, many OO design rules and heuristics have been proposed over the years. Violation of the design rule and heuristic causes the problems of the design phase which affects its quality. However, detecting violations of design heuristics by hand is very difficult and time-consuming in large systems, because definitions of these heuristics are given in the form of natural language. We believe that some of violations of the design heuristics can be detected automatically by using specific metrics.

The issues mentioned above are beyond the scope of this paper. In short, this paper introduces a metrics-based approach for detecting defects which have a significant impact on the OO design quality. Moreover, thresholds to help in identifying critical values are defined for each of the metrics.

The key contribution of this paper is the detection technique of a list of design rule violations-based defects of OO design.

The rest of the paper is organized as follows. The next section gives a brief summary of the research problems, followed by a section presenting the metrics-based approach to detect design defects. Finally, section 4 concludes the paper and suggests future work.

2. Research Problem

Several authors [1], [11], [17], [29], and [34] have presented a number of particular OO design problems that may occur when defining the classes and objects, relationships between classes, and class hierarchies. However, these works do not provide support to automatic identification of the design problems.

To assess the quality of software, many OO metrics have been proposed in the last years [5], [6], [12], [20], [21], [22], [27], and [32]. However, we can see that a few of them are very helpful for detecting faulty classes. We will focus here only on the design metrics that can be used early in the software life cycle. Chidamber and Kemerer[12] have proposed a suite of OO design metrics. This suite of metrics has been empirically validated by Basili et al.[2]. Their results show that most of the metrics of the suite are useful quality indicators to predict fault-prone classes.

Briand et al.[9] have made a empirical validation of all the OO design measures found in the literature to define the relationships between OO design measures and the probability of fault detection in system classes. The results show that the frequency of method invocations and the depth of inheritance hierarchies are very important quality factors of fault prone-classes.

Erni et al.[16] proposed a systematic approach to find problem spot using multi-metrics which is a set of simple metrics that are all related to the same component. To interpret metrics values, they used thresholds and trend analysis which help to identify outliers and critical values. However, in this approach, developer's inspection of their software is necessary for drawing conclusions from the metrics values.

An empirical study that describes application of measurement to design assessment was done by Colin Kirsopp and his colleagues[26]. In this study, quality assessment of a small system which consists of 15 classes was done by them, using measurement results and design heuristics. Analysis tool that installs a large number of metrics provides measurement of the system. From their study, for some metrics, it was difficult to inter-

pret their results into concrete problems whereas for some design problems, which are pre-defined by authors, suitable metrics to detect them were not collected. This shows that suitable metric selection and metrics collection for an application are very important for successful software measurement. Also around fifty percent of the problems detected were at the class-level. This may suggest that partitioning a system's functionality into a well designed set of classes should be considered carefully.

Most recently, UML-based, quality measurement tool for object-oriented design has been developed[33]. This tool contains a rich set of OO design measures covering the structural properties of design elements from all nine UML diagrams. To interpret the results derived from the software design measures obtained with SDMetrics Tool effectively for detection of design problems, data analysis techniques are needed. They suggest a number of data analysis technique such as benchmark and prediction model.

To summarize, current way of detecting a design problem with design measurement is done in a manner of doing inspection on the results derived from the metrics after doing measurement, collecting the metrics to be regarded as suitable for a given application. In other words, there is no pre-defined metrics for the particular design problem and what design problems exist in a given application is only determined by inspections after measurement. Therefore, it might happen that metrics could easily be misapplied or their results misinterpreted.

3. Development of Approach

This section introduces the approach for detecting defects in OO designs.

Within the framework of this work, following four sequential steps are carried out.

1. Survey of design defects into literature and define a list of design problems(defects).
2. Categorize the design defects found during the survey into corresponding design properties that are affected by that design defect.
3. Identify metrics to detect each design defect.
4. Identify threshold for each of the metrics.

These steps are described in detail in the following subsections.

3.1. Identifying OO Design Defects

We have made a wide survey into literature to collect the OO design defects that may occur when defining the classes and objects, attributes and methods of a class, relationships between classes, and class hierarchies.

During the survey, we found many design guidelines, rules, and flaws originating from various authors [1], [11], [13], [15], [17], [19], [25], [29], [30], and [34]. Violations of these design rules may contribute a potential design defect to the software. However, it is not always true that every violation of a design rule corresponds to a design defect, since heuristics are just guidelines, and not hard and fast rules that must be followed[30]. Depending on the application, violations of some design heuristics may be acceptable to some extent.

We collected forty seven OO design defects which best covered our interesting area for further examination. The rejected guidelines and flaws were either equivalent to an

already chosen one and thus redundant, or requires too detail information of implementation and thus impossible to detect from a design.

Riel[30] collected more than sixty guidelines for OO design. From his heuristics, sixteen are taken into account and violations of them are included in our list. We choose the design heuristic of which violation indicates a design problem and that is measurable and detectable in the design phase. Another design rules are defined by Eder[15], Gamma[19], Jonhson[25], and Meyer[29]. Violations of some design defects of their design rules are also included in the list. On the other hand, several authors [11], [13], [17], and [34] considered design solutions which lead to negative consequences. We are going to derive eighteen design defects from authors definitions of [1], [11], [13], [17], and [34], which are considered as detectable in design phase. Moreover, we derive four design defects from a number of metrics definitions proposed by Lorenz[27].

The complete list of design defects defined by our work together with references is shown in Table 1. All of the required information to detect these design defects can be taken from design diagrams.

3.2. Classifying Design Defects into OO Design Properties

In this section, design defects described in the previous subsection are categorized into OO design properties depending on what design property's quality aspect is violated. Design properties are characteristics to define the quality of an OO design from the perspective of the internal attributes.

We choose the following ten design properties as OO design internal quality characteristics: encapsulation, abstraction, cohesion, coupling, messaging, complexity, inheritance, composition, and polymorphism. Depending on the different projects or different designers, different sets of characteristics may be used[35]. However, we consider that these design properties are most general and most important characteristics for indicating internal quality of OO design.

Classification of design defects into design properties is listed in Table 2. Last column of Table 2 shows metrics which are used to detect the defects. These metrics are described in detail in Table 3 and 4. Each of the design defects on the design properties are ranked in an descending precedence from highest to lowest.

3.3. Defining Metrics to Detect Design Defects

Metrics is an important tool for detecting the defects. Each of the design defects identified in this section can be detected by metrics defined on each defect. Many design metrics for OO design have been proposed by the authors of [5], [12], [20], [21], [22], and [27]. Twenty two well-known metrics are used in the assessment of some design properties, such as size, encapsulation, and cohesion and defined in Table 3.

For the other design defects, there are no suitable metrics, therefore, we define eighteen new metrics. Table 4 summarizes the definitions of these new metrics.

Combination of metrics is used in detecting some design defects, for example, we could use a combination of LCOM, NOA, and NOM to identify D14 multifaceted abstraction.

3.4. Identifying Thresholds on Metrics

Once we have a metrics value, we need to judge whether the value indicates critical situation or not. Thresholds can help us do this judgement. If a design metrics exceeds a certain threshold, the design element can then be considered as “critical” and must be redesigned. Thresholds are defined as in [27] heuristic values used to set ranges of desirable and undesirable metric values for measured software. These thresholds are used to identify anomalies, which may or may not be an actual problem.

We define thresholds for all design metrics defined in the previous subsection. If a metrics value is equal to threshold or in a range of undesirable values defined by threshold, it means that there exists the defect and should be reconsidered. According to the Table 2, we can simply find what design defect it is. The definitions of the thresholds are shown in Table 3 and Table 4.

Below, we will explain about our criteria for selecting and defining threshold value on each metrics. We classify thresholds in our work into three groups and show in Table 5. Second column of Table 3 and Table 4 shows thresholds belong to which group, as shown in Table 5.

In our work, there is no general standard for thresholds of metrics. Different researchers proposed different threshold values for the same OO metrics. Besides this, Benlarbi et al.[10] pointed out that design elements with values below the threshold can still mean high fault-proneness, just not the highest. For these reasons, for thresholds groups 1 and 3, we define several thresholds instead of a single one. Then we divide thresholds on one metrics again into three subgroups(levels), so as to get more fine-grained threshold. To make clear these three subgroups meaning, we use linguistics values such as “little bad”, “bad”, and “very bad”.

We believe that detection of design elements which have still high probability to be “critical” even if their values below the threshold, can be increased by defining several thresholds on their metrics.

In group 1, fourteen thresholds are included. Most of them are defined based on the thresholds proposed by Lorenz and Kidd. They present a number of thresholds for OO measures based on their experiences with Smalltalk and C++ projects. As an example, for metrics NOM, threshold value is proposed by [1], [16], and [27] respectively 20, 50, and 60. These three of thresholds are quite different, so we define several thresholds for metrics NOM, according to our scheme as mentioned above. 3 subgroups of thresholds on metrics NOM are defined as follows: “little bad” ranges from 20 to 30, “bad” ranges from 31 to 50, and “very bad” is larger than 50. Lorenz et.al[27] define NMSM threshold of 9. And Rosenberg[31] define a CBO threshold of 5. CBO is the number of other classes to which a class is coupled. Based on these two thresholds, we define thresholds of NMSM: “little bad” ranges from 9 to 11, bad is between 12 and 14, and “very bad” is larger than 14. For metrics NAC, Lorenz et.al[27] suggest that ten to fifteen percent of total classes should be abstract. So three subgroups of thresholds of NAC are defined as follows: “little bad” is between 10 and 12, “bad” is between 8 and 10, and “very bad” is less than 8. The complete description of thresholds included in Group 1 are described in Table 3.

Group 2 is made up of fifteen thresholds which are clear from design defect descriptions, or design rules connected with the metrics. All thresholds in this group takes the value 0 or 1. Due to the lack of space, we discuss only about three thresholds of

this group. There is defect D20 “There exist methods which do not make sense in the class’s responsibilities”. For this defect, we define metrics NMMI “number of messages to a method itself”. If there is no message to the method itself, it means that method is useless. Thus threshold is 0. There is another defect D39 “A Subclass uses only part of base class’s interface”. This implies that the subclass does not inherit its data. So we define threshold of 0 for metrics NID “number of inherited attributes”, which detects this defect. There is also defect D45 “There are no message sends from the containing class to the contained objects”. If contained objects are not sent messages by the containing class, then they are useless information[30]. Therefore, we have to take 0 as the threshold of a NMCCC.

Group 3 consists of twelve thresholds that are proposed by authors. In other words, for these metrics, there are no thresholds by experts, and design defect and design rule which correspond to the metrics do not contain any quantitative statements. Out of twelve thresholds in this group, two are proposed based on the design rules by expert and shown in Table 4. Riel[30] suggests that all data in a base class should be private. Based on this heuristic, thresholds on metrics NPABC(D42) are defined as follows: “bad” is between 1 and 3, “very bad” is larger than 3. Webster[34] says that if base class is doing too little, it means no apparent generalization or code reuse. So we define threshold for metrics NPMBC(D35) that “little bad” is 2, “bad” is 1, and “very bad” is 0.

In the descriptions of defects D16, D24, D29, and D31, there is a same expression like “most/most of”- e.g. “ most of the methods in a class/most of the messages sent in a method”. Thresholds for metrics NMNP, NMSSO, NMHC, and NMHSP that detect these four defects are based on this expression and defined in Table 4. For example, thresholds for metrics NMNP are defined as follows: “little bad” ranges from 30 to 50 percent of total number of methods, “bad” is between 50 and 70 percent, “very bad” is between 70 and 100 percent.

The remaining thresholds of this group are defined based on the thresholds of metrics in Group 1 . For example, thresholds for metrics NPM(D9) are proposed based on the thresholds of metrics NOM, NPubM, and NPAM($NPM \cong NOM - NPubM - NPAM$). Thresholds for metrics NOAC(D11) are defined based on the thresholds of metrics DIT. Thresholds for metrics NPAM(D18) are defined based on the thresholds of metrics NPV($NPAM = NPV * 2$). Thresholds for metrics a MPC(D26) are defined based on the thresholds of metrics NPubM, NMSM, and NPAM($MPC \cong NPubM * NMSM + NPAM$). To define thresholds “little bad” on this metrcis, we used an upper threshold value of normal or desirable values of metrics NPubM equals to 4 and NMSM equals to 9. “Bad” is defined by using “little bad” thresholds of a NPubM and NMSM. “Very bad” is defined by using “bad” thresholds of a NPubM and NMSM. Thresholds for metrics a NPVUA(D21) are defined based on the thresholds of metrics CBO.

Without threshold, it is difficult to interpret and analyze the metric results.

4. Conclusions and Future Work

In this paper, we have proposed a metrics-based approach to identify design defects in an OO design.

We have made wide survey into literature to define the design defects that may lead to problems or errors later in the development life-cycle. Design defects derived from the

result of the survey are classified based on the design properties that they influence. This classification can provide a systematical search for quality design defects and is helpful for evaluating internal design properties as we discussed in the previous section.

Eighteen new metrics have been proposed since there are no suitable metrics to detect some design defects. Furthermore, we have proposed thresholds to help in judging whether a metrics value indicates problematic situation or not, on each metrics. There are no general standard threshold values for some metrics defined in this paper. Therefore, for these metrics, we have assigned thresholds each has three levels and each level has range. The levels of the thresholds allow the user to judge whether metrics value is acceptable or not by intuition.

All metrics information can be derived from design diagrams. By defining metrics on each design defect, these defects can be detected automatically. Consequently, many design heuristics, and flaws which are described qualitatively can be evaluated quantitatively. On the other hand, it can be seen that intended application of metrics has been clearly determined.

Most of the approaches detect design defects by making inspection on the results of the measurement. Whereas in our approach, what design defects exist in a system can be seen from the metrics results because metrics that detect each of the defects together with thresholds have been pre-defined as we explained above. Doing measurement in the design phase allows the design defects to be caught and fixed earlier, and thus the quality of a design of software system can be improved.

By this work, we have attempted to evaluate ten internal OO design attributes of Table 2 by means of verifying and detecting design defects that are pre-defined for each of the internal attributes, using metrics. Link to external attributes of OO design and empirical validation of our work are remained as future work. Before we link internal attributes to external one, we plan to get one comprehensive number for each design property by representing three levels of thresholds into the fuzzy logic set[36].

We have developed our own tool that helps collect metrics values more efficient and effective way.

References

- [1] M. Akroyd, "AntiPatterns Session Notes", Object World West, 1996.
- [2] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, 22(10), 751-761, 1996.
- [3] B. Boehm and V. Basili, "Software Defect Reduction Top 10 list", IEEE Computer, 34(1), 135- 137, 2001.
- [4] J. Bansya and G. Davis, "A Hierarchical Model for Object-Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, 28(1), 4-17, 2002.
- [5] J.M. Bieman and B.K. Kang, "Cohesion and Reuse in an Object-Oriented System", in Proc. ACM Symp. Software Reusability(SSR'94), 259-262, 1995.
- [6] L. Briand, P. Devanbu, and W. Melo, "An Investigation into Coupling Measures for C++", Proceedings of ICSE 97, Boston, USA, 1997.
- [7] L. Briand, J. Daly, and J. Wuest, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", Empirical Software Engineering - An International Journal, 1998 .
- [8] L. Briand, J. Daly, and J. Wuest, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", IEEE Transactions on Software Engineering, 25(1), 91-121, 1999.
- [9] L. Briand, J. Wuest, J. Daly, and V. Porter, "Exploring the Relationships Between Design Measures and Software Quality in Object Oriented Systems", Journal of Systems and Software, vol. 51, pp. 245-273, 2000.

- [10] S. Benlarbi, K.EI. Emam, N. Goel, and S.N. Rai, “Thresholds for Object-Oriented Measures”, 11th International Symposium on Software Reliability Engineering, 2000.
- [11] W.J. Brown, R.C. Malveau, H.W. McCormick, and T.J. Mowbray, “AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis”, John Wiley & Sons, 1998.
- [12] S.R. Chidamber and C.F. Kemerer, “A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, 21(3), 263-265, 1994.
- [13] “CodeSmell”[online], Available from: c2.com/cgi/wiki?CodeSmell[Accessed 01/03/2004].
- [14] R.G. Dromey “A Model for Software Product Quality”, IEEE Transactions on Software Engineering, 21(2), 146-162, Feb. 1995.
- [15] J. Eder, G. Kappel, and M. Schrefl, “Coupling and Cohesion in Object-Oriented systems”, Technical Report, University of Klagenfurt, Austria, 1994.
- [16] K. Erni and C. Lewerentz, “Applying Design-Metrics to Object-Oriented Frameworks”, Third IEEE International Symposium on Software Metrics, Germany, 1996.
- [17] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, “Refactoring: Improving the Design of Existing Code”, Addison-Wesley, 1999.
- [18] N. Fenton, “Software Measurement: A Necessary Scientific Basis”, IEEE Transactions on Software Engineering, 20(3), 199-206, 1994.
- [19] E. Gamma, R.Helm, R. Johnson, and J. Vlissides, “Design Patterns, Elements of reusable Object-Oriented Software”, Addison-Wesley, 1994.
- [20] Y.S. Lee, B.S. Liang, S.F. Wu, and F.J. Wang, “Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow”, in Proc. International Conference on Software Quality, Maribor, Slovenia, 1995.
- [21] W. Li and S. Henry, “Object-Oriented Metrics that Predict Maintainability , J. Systems and Software”, 23 (2), 111-122, 1993.
- [22] M. Hitz and B. Montazeri, “Chidamber and Kemerer’s Metrics Suite: A Measurement Theory Perspective”, IEEE Transactions on Software Engineering, 22(4), 267-270, 1996.
- [23] Software Product Evaluation - Quality Characteristics and Guidelines for their Use, ISO/IEC Standard ISO-9126, 1991.
- [24] B. Kitchenham and S.L. Pfleeger, “Software Quality: The Elusive Target”, IEEE Software, 13(1), 12-21, 1996.
- [25] R.E. Johnson and B. Foote, “Designing reusable classes”, Journal of Object-Oriented programming, 1(2), 22-35, June 1988.
- [26] C. Kirsopp, M. Shepperd, and S. Webster, “An Empirical Study into the Use of Measurement to Support OO Design Evaluation”, Sixth IEEE International Symposium on Software Metrics, 1999.
- [27] M. Lorenz and J. Kidd, “Object-Oriented Software Metrics: A Practical Guide”, Prentice Hall, 1994.
- [28] J.A. McCall, P.K. Richards, and G.F. Walters, “Factors in Software Quality”, vols. 1, 2, and 3, AD/A-049-014/015/055, Nat'l Tech. Information Service, Springfield, Va., 1977.
- [29] B. Meyer, “Object-Oriented Software Construction”, Prentice Hall, 2nd Edition, 1997.
- [30] A.J. Riel, “Object Oriented Design Heuristics”, Addison-Wesley, 1996.
- [31] L. Rosenberg, R. Stakpo, and A. Gallo, “Object-Oriented Metrics for Reliability”, Presentation at IEEE International Symposium on Software Metrics, 1999.
- [32] D.P. Tegarden, S.D. Sheetz, and D.E. Monarchi, “A Software Complexity Model of Object-Oriented Systems”, Decision Support Systems, 13(3-4), 241-262, 1995.
- [33] “The Software Design Metrics tool for the UML” [online], Available from: www.sdmetrics.com/ [Accessed 01/03/2004].
- [34] B. Webster, “Pitfalls Of Object Oriented Development”, M&T Books, 1995.
- [35] S.A. Whitmire, “Object-Oriented Design Measurement”, John Wiley and Sons, 1997.
- [36] L.A. Zadeh, “Fuzzy Sets”, Information and Control, 338-353, 1965.

Table 1. Description of Design Defects

#	Design Defect Name	Description
D1	Large class	With too many instance variables or with too many methods, or both [29].
D2	Large method	Number of(n.of) message sends is high[27], or n.of parameters is high[25].
D3	Small class	No variables and with too few methods[29], or too few variables[13] and with too few methods, or with too few variables and no method at all.
D4	Large interface	A Class offers too wide public interface[30].
D5	Poor encapsulated data	Public variables are declared in a class[30].
D6	Poor interface	No public methods in a class[13].
D7	Unencapsulated	A lot of global variables being used by class[27].
D8	Unhidden private method	Common-code private functions are put in the public interface[30].
D9	Exceeded method hiding	A Class has too many private(or protected) methods[13].
D10	Unsufficient abstract classes	N.of abstract base classes is very low[25].
D11	Complicated abstraction	Average n.of ancestors is high[4].
D12	Extra subclass	Abstract base class is defined when there is only one derived class for it[19].
D13	Nonrelated data and behavior	Data and behavior in a class are not related[30].
D14	Multifaceted abstraction	A Class captures several abstractions, not one[30].
D15	Data class	A Class contains only data[13].
D16	Unparameterized methods	Most of methods in class have no parameters and utilize class or global variables for processing[11].
D17	Method turned into class	A Method has been turned into class[30].
D18	Unrelated abstraction	Class has many accessor methods in its public interface[30].
D19	Not single task	Method does not do single task in the problem domain[15].
D20	Unutilized	There exist methods which don't make sense in class's responsibilities [15].
D21	Higher external variable accesses	Number of external variable accesses is high[25].
D22	Higher collaboration	N.of classes collaborating with given class is high[30].
D23	Bidirectional relation	Cyclic(two-way) dependencies between methods in different classes[13].
D24	Method in a wrong class	Most of the messages sent from same method is to the same object[13].
D25	Irrelevant class	There exist classes that are outside system[30].
D26	Too many message sending	N.of message sends between a class and its collaborators is high[30].
D27	Complex interface	N.of parameters per method is more than 6[25].
D28	Higher method complexity	N.of methods that must be called to carry out a given method is high[34].

#	Design Defect Name	Description
D29	Higher class complexity	Most of the methods in a class have higher complexity.
D30	Inconsistent	In a method, same parameter is passed twice[14].
D31	Redundant parameters	Same parameter is passed to most methods of class[13].
D32	Not enough commonality	Two or more classes share only common data[30].
D33	Poor subclassing	Percentage of inherited methods is low[27].
D34	Getting away from abstraction	N.of operations added by a subclass is high[27].
D35	Base classes do too little work	Base class has few or no public methods at all[34].
D36	Higher class hierarchy	The depth of the class hierarchy is very high[25].
D37	Redundant variable declaration	Subclasses inheriting from same base class have the same data[17].
D38	Misdeclared members of base class	Members which only relate to one of its subclasses are put in base class[17].
D39	Improper use of inheritance	A subclass uses only part of base class's interface or doesn't want to inherit data[17].
D40	Object turned into a subclass	Object of a class is turned into derived classes of the class[30].
D41	Variable turned into subclass	There exist subclasses that vary only in methods that return constant data[17].
D42	Weakening of data hiding	Most of the data in a base class are not private[30].
D43	Improper use of switch statement	There is a conditional that chooses different behavior depending on the type of an object [17].
D44	Higher relation	A Class contains many objects, more than six[30].
D45	Useless containment relation	If a class contains object of another class, then there are no message sends from the containing class to the contained objects[30].
D46	Redundant dependency	Objects that contained in the same containing class have uses relationships between them [30]
D47	Improper use of delegation	Using delegation and often writing many simple delegation for the entire interfaces[17].

Table 2. Design Defects and Metrics for Design Properties

D.Property	#: Metrics
Size	D1: NOA, NOM, D2: NMSM, NOP, D3: NOA, NOM
Encapsulation	D4: NPubM, D5: NPV, D6: NPubM, D7: NG, D8: NACUM, D9: NPM
Abstraction	D10: NAC, D11: NOAC, D12: NOS
Cohesion	D13: LCOM, D14: LCOM, NOA, NOM, D15: NOM, D16: MNP, NPV, NGV, D17: NOM, D18: NPAM, D19: NMSM, NOP, D20: NMMI
Coupling	D21: NPVUA, D22: CBO, D23: NTWD, D24: NMSSO, D25: NMR
Messaging	D26: MPC
Complexity	D27: NOP, D28: NMSM, D29: NMHC, D30: NSP, D31: NMHSP
Inheritance	D32: NIM*, D33:NIM, D34:NAM, D35:NPMBC, D42:NPABC, D37:NSAIS, D38:NITBM, D39:NID, D40:NOIDC, NOA [†] , NOM*, D41:NMIS, D36:DIT
Polymorphism	D43: NC
Composition	D44: DAC, D45: NMCCC, D46: NMOCC, D47: NMDC
Threshold for NIM* is 0, threshold for NOA [†] is 1, threshold for NOM* is 1.	

Table 3. Selected Well-known Design Metrics

Metrics	Definition and Threshold Group	Threshold of Undesirable Value
NOA	Number of(N.of) attributes(1)	7-9 little bad, 9< bad[27]
NOM	N.of methods in a class(1)	20-30 little bad, 31-50 bad, 50< very bad[27]
NPM	N.of private methods in class(3)	14-22 little bad, 22-34 bad, 34< very bad
NMSM	N.of message sends in method (1)	10-11 little bad, 12-14 bad, 14< very bad[27]
NOP	N.of parameters in a method(1)	5-6 little bad, 7-8< bad, 8< very bad[27]
NPubM	N.of public methods in a class(1)	6-8 little bad, 9-10 bad, 10< very bad[27]
NPV	N.of public variables in a class(1)	2-3 bad, 3< very bad[27]
NGV	N.of global variables(1)	2-3 little bad, 4-5 bad, 5< very bad[27]
NAC	Ratio of n.of abstract classes total n.of classes in a system(1)	<8% very bad, 8~10% bad, 10~12% little bad[27]
NOS	N.of subclasses in a superclass(2)	1 bad
NOAC	N.of ancestors of the class(3)	3-4 little bad, 5-6 bad, 6< very bad
LCOM	Lack of cohesion in methods(1)	<0.4 very bad, 0.4~0.6 bad, 0.6~0.8 little bad
NPAM	N.of public accessor methods(3)	4-6 bad, 6< very bad
DAC	Data abstraction coupling(1)	3-4 little bad, 5-6 bad, 6< very bad[27]
CBO	Coupling between classes(1)	6-7 little bad, 8-9 bad, 9< very bad[31]
NMR	N.of messages of other classes(2)	0 bad
MPC	Message passing coupling(3)	36-63 little bad, 64-96 bad, 96< very bad
NIM	Ratio of n.of inherited methods(1)	<0.4 very bad, 0.4~0.6 bad, 0.6~0.8 little bad
NAM	Ratio of n.of added methods(1)	0.2~0.4 bad, 0.4< very bad[27]
DIT	Depth of inheritance hierarchy(1)	4-6 bad, 6< very bad[27]
NID	N.of inherited attributes(2)	0 bad
NOIDC	N.of objects on derived class(2)	1 bad

Table 4. Definition of New Design Metrics

Metrics	Definition and Threshold Group	Threshold of Undesirable Value
NACUM	N.of another classes using a given method(2)	0 bad
NMMI	N.of messages to a method itself(2)	0 bad
NMNP	Ratio of the n.of methods do not have parameters in the class(3)	30~50% little bad, 50~70% bad, 70~100% very bad
NPVUA	N.of external variables(3)	3-5 bad, 5<very bad
NTWD	N.of two-way dependencies between methods in different classes(2)	1 bad, 2< very bad
NMSSO	Ratio of n.of messages sent to same object from same method(3)	60~70% little bad, 70~80% bad, 80~100% very bad
NSP	N.of same parameters in one method(2)	0< bad
NMHC	Ratio of n.of methods with high complexity to total n.of methods in a class(3)	50~60% little bad, 60~70% bad, 70~100% very bad
NMHSP	Ratio of n.of methods which have same parameters to total n.of methods(3)	50~60% little bad, 60~70% bad, 70~100% very bad
NPABC	N.of public attributes in base class(3)	1-3 bad, 3< very bad
NPMB	N.of public/protected methods in base class(3)	2 little bad, 1 bad, 0 very bad
NSAIS	N.of same attributes defined in all subclasses(2)	0< bad
NITBM	N.of inherited times of base class's members(2)	1 bad
NMIS	N.of methods defined in the subclasses(2)	1 bad
NMCCC	N.of messages from containing class to the contained objects(2)	0 bad
NMOCC	N.of messages between the objects contained in the same class(2)	0< bad
NMDC	Ratio of the n.of methods of delegate class to be used in delegating class(2)	90~100% bad
NC	N.of conditional chooses depending on the type of an object(2)	1-2 bad, 2< very bad

Table 5. Description of Thresholds

Group	Threshold Description
1	Thresholds which are defined based on the thresholds proposed by experts.
2	Thresholds which can be derived from statements in design defects, design rules.
3	Thresholds which are proposed by authors.

Exploring an Open Source Data Mining Environment for Software Product Quality Decision Making

Houria YAZID^a and Hakim LOUNIS^{a,1}

^a*Department of Computer Science, Université du Québec à Montréal, Canada*

Abstract. Software metrics play a major role in predicting software quality; they help project managers in decision-making. Indeed, software metrics provide a quantitative approach allowing the control and the improvement of the development process including the maintenance. The ISO/IEC international standard (14598) on software product quality states, “Internal metrics are of little value unless there is evidence that they are related to external quality”. Many empirical prediction models are presented in the literature; their goal is to investigate the relationship between internal metrics and external qualities, in order to assess software quality. In this paper, we explore different machine-learning (ML) algorithms provided by an open source data-mining environment. We analyse their capacities to produce accurate and usable predictive models.

Keywords: Software product quality, corrective maintenance, reusability, fault-proneness, metrics, machine-learning.

Introduction

The big issue in software engineering is to produce high quality software in time and in budget. To pass through this issue, measurement based on metrics should be done at different stages of the software development life cycle, with objectives as, software quality prediction or learning from past experiences; the ultimate goal is to improve the quality of the developed product. In fact, software metrics provide a quantitative approach to control the software development and then the software products quality [8]. They are necessary to identify design and implementation anomalies early in the software life cycle, and then to see where the resources are needed. They are also a crucial source of information for decision-making. Moreover, early availability of metrics is a key factor to a successful management of software development.

Our research is done in the context of the ISO/IEC international standard (14598) on software product quality; it states, “*Internal metrics are of little value unless there is evidence that they are related to external quality*”. We explore the relationship between three quality factors (or external attributes), *Maintainability*, *Reusability*, and *Fault Proneness*, and internal attributes as inheritance, coupling, cohesion, complexity, and size. The aim is to try to predict these factors via metrics associated to the considered internal attributes. Many different approaches have been proposed to build such

¹Department of Computer Science, Université du Québec à Montréal, CP8888, Succursale Centre-Ville, Montréal, H3C 3P8, Canada. HOURIA.YAZID.1@ens.etsmtl.ca and lounis.hakim@uqam.ca

empirical predictive models; these later can take different forms depending on the building technique, e.g., machine-learning techniques. In all cases, they allow affecting a value to a quality factor based on the values of a set of software measures.

Our work deals then with the construction of such predictive models for the three pre-cited quality factors. To build these models, different machine-learning algorithms are selected from the WEKA environment (Waikato Environment for Knowledge Analysis) [14]. In section 1, we present various works in the same topic. Then, in section 2, we describe the hypotheses to verify, followed in section 3, by an overview of the experimental environment and the selected machine-learning algorithms. In section 4, a short description of the experimental process is given, and results are presented and analyzed. Finally, we conclude with some perspectives and we discuss the applicability of such models in intelligence-artificial decision-making systems.

1. Related works

To control software quality, a great number of internal metrics have been defined but only few ones have been considered by the researchers. Most researchers have aimed to assess external attributes (i.e., quality factors) basing on internal metrics. Several approaches have been proposed to construct such models; for instance, Briand and al., in [5], tried to see if coupling measures, capturing all kinds of collaboration between classes, can help to analyze change impact. More recently, machine-learning techniques have been introduced with regard to their prediction capability and their wide use in other fields as, robotics, and vision for instance. Selby and Porter [12] were the first who use a machine-learning algorithm to predict errors during the maintenance phase. Other works explored different algorithms belonging to the divide and conquer family (e.g., C4.5) [2] [1]; in [1], Almeida and al. have investigated three machine-learning approaches, i.e., top down inductive decision tree, inductive logic programming, and covering. Four ML algorithms (NewID, C4.5, FOIL, and CN2) were selected, and the results showed that FOIL (a first-order inductive logic programming algorithm) provides the best accuracies for assessing corrective maintenance cost. In [10], more approaches and algorithms are explored; seven (7) ML algorithms adopting many approaches (divide and conquer, statistic, analogy, covering, soft computing) are examined to verify the relationship between three quality factors and internal measures. The authors established that some design measures could predict, with a high level of accuracy, fault-proneness components or reusable ones.

In our present research, we underline the relevance of machine-learning to assess software quality, and their usefulness for the construction of knowledge-based systems, which can provide a great help to managers when they deal with development decisions. In this paper, we explore 22 machine-learning algorithms, belonging to different approaches, in order to verify hypotheses stated in the next section.

2. The Hypotheses

We are interested in exploring the relationship between three external quality attributes (quality factors), *Maintainability*, *reusability*, and *fault proneness* from one side, and internal attributes as inheritance, coupling, cohesion, complexity, and size, on the other side. The following sub-sections present the hypotheses we want to verify.

2.1. Maintainability expressed as Corrective Maintenance Cost

Corrective maintenance concerns the error correction in code and in documentation. Pressman [13] states that maintenance consumes 60% of resources invested in the software development. This highlights the need to predict the cost of corrective maintenance. To do so, we have used data collected during the corrective maintenance phase of the General Support Software reusable components library located at the Flight Dynamics Division of NASA's Goddard Space Flight Center. The library is composed by 1 K of software components written in ADA and its size approaches 515 KSLOC where only 164 faulty components are used. We have also used the internal metrics extracted from faulty components, and which are mainly size and complexity metrics (e.g., cyclomatic complexity, Halstead's metrics, etc.). [1] gives the details of these metrics. The variable to assess is the cost for isolation and correction of a faulty component; this cost may be high or low. Thus, the components are classified according to their cost either high or low. In this case, the investigated hypothesis (called H_{CCM}) is: "*Components size and complexity have an impact on corrective maintenance costs*".

2.2. Components Reusability

Reusability has obviously an impact on software development cost and time. It will be interesting to determine reusable components starting from internal attributes like inheritance, coupling, and complexity. This could be done if we can settle that there is a cause-effect link between these internal product attributes and the degree of reusability. This link is expressed via three hypotheses mentioned in sub-sections 2.2.1, 2.2.2, and 2.2.3. To explore these hypotheses, we have used (1) data collected from a multi-agents system (LALO) written in C++ and developed at CRIM [11], and (2) extracted internal measures. Here, the variable to predict is defined as the reusability degree, which can be measured, by the necessary amount of work to reuse a component either from system to another one in the same domain, or from a version to another version of the same system. The developer's team (specialized in multi-agents systems development [11]) proposes the following classification of the reusability degree:

- Totally reusable: the component is generic to multi-agents domain;
- Reusable with minimum rework: a component is reusable with less than 25% of changes;
- Reusable with great rework: more than 25% code must be modified to reuse a component;
- Not reusable: components are specific to the considered system.

2.2.1. Coupling-Reusability Hypothesis

It is well admitted that low coupling and high cohesion can provide high quality software. In our case, the objective is to verify the hypothesis H_{re-cou} that is: "*Coupling between a class and its environment can affect its reusability*".

This hypothesis is verified for two levels of coupling: design and code. For the former, we exploit the metrics set defined by Briand et al [3]. These metrics consider all features of the object-oriented paradigm, such as, aggregation and friend relationship. 18 metrics are defined according to three facets: relationship, locus, and type. The relationship may be a friend relation, an inheritance one, or other. The locus

specifies if a class exports/imports services to/from other ones. Thus, the value of locus can be import or export. Finally, type defines the kind of interaction between classes: class-attribute, class-method, or method-method. For the code level metrics, Lounis and al. [9] have proposed 24 metrics to assess coupling at the code level for modular software systems, where a module is defined as a set of units, and each unit is a set of contiguous statements.

2.2.2. Inheritance-Reusability Hypothesis

Inheritance allows constructing a hierarchy of classes. The position of a class in hierarchy informs us whether it is generic, specific, etc. Then, the hypothesis H_{re-inh} is: “*The class position in an inheritance hierarchy can indicate its reusability degree*”.

To verify this hypothesis, the following metrics are considered: DIT (Depth of the inheritance tree), HIT (Height of the inheritance tree), NOA (Number of Ancestors), NOC (Number Of Children), NOD (Number of Descends), OIM (Number of Inherited Methods), and NOP (Number of Polymorphic methods). These metrics are defined in [11].

2.2.3. Complexity-Reusability Hypothesis

The complexity of a software component has an impact on its comprehension. We study with this hypothesis the impact it has on its reusability. Thus, we consider the following hypothesis (H_{re-com}): “*The complexity of a software component can affect its reusability*”.

The used metrics in this case are: NOM (Number Of local Methods in a class), RFC (Response For Class), CIS (Class Interface size), NOT (Number of Trivial Methods), NOP (Number of Polymorphic Methods), NOD (Number of Attributes), NAD (Number of Abstract Data Types), NRA (Number of Reference Attributes), NPA (Number of Public Attributes), NOO (Number of Overloaded Operators), NPM (Number of Parameters Per Method), and CSB (Class Size in Bytes). See [11] for more details on these metrics.

2.3. Components Fault-Proneness

The assessment of this factor allows focusing maintenance efforts on faulty components and then gradually improves software quality. Data are collected from the LALO system described in section 2.2; the dependent variable is defined as class fault proneness. We want to indirectly assess it using the internal measures of inheritance, coupling, and cohesion. In the next sub-sections, we state the 3 hypotheses and present the involved internal metrics.

2.3.1. Cohesion-Fault-Proneness Hypothesis

To verify this hypothesis, 10 cohesion metrics are considered: $LCOM_i$ (Lack of cohesion of methods, $i=1..5$), Co and Coh (LCOM variants), LCC (Loose Class Cohesion), TCC (Tight Class Cohesion), and ICH (Information-flow-based Cohesion). The hypothesis we state is H_{fp-coh} : “*Class cohesion can affect its fault-proneness*”.

More information about these metrics are given in [4].

2.3.2. Coupling-Fault-Proneness Hypothesis

To study this hypothesis, we considered 28 coupling metrics; 18 of them are defined in [3] (design coupling, 2.2.1). The 10 others are: coupling between object (CBO, and CBO'), response for class (RFC_{_1}, and RFC_{_α}), Message passing coupling (MPC), Information-flow-based coupling (ICP, IH-ICP, and NIH-ICP), and Data abstraction coupling (DAC, and, DAC') (See [4]). So, we have to verify the hypothesis $H_{fp\text{-cou}}$: “*Coupling between a class and its environment can affect its fault-proneness*”.

2.3.3. Inheritance-Fault-Proneness Hypothesis

We used as inheritance metrics the ones defined in section 2.2.2, plus 5 other metrics which are: AID (Average Inheritance Depth), NOP (Number Of Parents), NMO (Number of Methods overridden), NMA (Number of Methods Added), and SIX (Specialized IndeX) (See [4]).

We want to establish if these metrics are good indicators of classes’ fault-proneness or not. The hypothesis $H_{fp\text{-inh}}$ is then: “*The class position in an inheritance hierarchy can affect its fault-proneness*”.

To explore the 8 hypotheses, we have selected 22 machine-learning algorithms from an open source data-mining environment called WEKA. In the next section, a brief description of this environment and of the selected algorithms is presented; the experimental conditions (setup) are also given.

3. The Experimental Environment

3.1. Data-Mining Environment Description

Data mining is the collaboration of many fields, such as statistic, database, and artificial intelligence (mainly, machine-learning). It allows the extraction of pertinent patterns (or knowledge) from amount of data that can be heterogeneous. To achieve our experiments, we have used WEKA (Waikato Environment for Knowledge Analysis), an open source data-mining environment [14]. It provides tools for data pre-treatment, machine-learning algorithms, clusters, association rules, attribute selection, and 2D visualization. Mainly, it allows (i) achieving pre-treatment upon data set, (ii) applying machine-learning algorithms, and (iii) analyzing results and performances (accuracies). These algorithms can be used through three tools: Explorer, Experimenter, and KnowledgeFlow. As our objective is to execute one algorithm at a given time upon data-set and to analyse the results, Explorer seems to be the best choice for us.

3.2. The Machine-Learning Algorithms

Machine-learning is an important sub-field of Artificial Intelligence; it proposes many approaches like induction, analogy, probabilistic, and soft-computing. The inductive approach includes techniques as divide and conquer and covering. The divide and conquer technique is adopted by classifiers based on decision trees. ADTree, J48, and NBTree are selected variants of these classifiers and differ on how they construct the model. On the other hand, the covering approach generates decision rules; each rule

covers an examples subset. ConjunctiveRule, DecisionTable, and PART are chosen examples of algorithms belonging to this approach.

The analogy approach is mainly represented by instance-based techniques, where a new example is firstly compared to classified ones using a distance metric; the prediction (or class) of the closest classified example is assigned to the new one [14]. Ibk and KStar are algorithms of such an approach; they use respectively an Euclidian measure and an entropy-based similarity function.

The third approach (probabilistic) includes Bayesian networks, and logistic regression. Bayesian networks are based on the Bayes formula [7]. The chosen classifier BayesNet is an implementation of such a classical algorithm. On the other hand, NaiveBayes considers the fact that attributes can be independent.

The last approach regroups fuzzy logic and the neuronal network. FLR² is an example of an fuzzy logic based algorithm; it uses the notion of interval where each attribute is represented by an interval value. On the other hand, neuronal networks represent architectures of artificial neurones and the most widely used architecture is the multilayer perceptron. Finally, RBFNetwork and SMO (Sequential Minimal Optimization) present similarities with neuronal networks and are two variant techniques of SVM (Support Vector Machines); these techniques are based on dataset examples separation by a hyper plan.

Other algorithms can be found in Weka as, meta-classifiers, which allow combining different machine-learning algorithms in order to obtain better accuracies. We have used Bagging, Vote, AdaBoostM1, and Stacking. Bagging and AdaBoostM1 are similar and combine outputs of same type models (e.g., decision trees) in a single output model using a vote technique. But their running is different. As for Stacking, it combines different-kind models; it runs in two steps: firstly, it generates the base models (level 0), which outputs are entries for a meta-model that gives the final output.

3.3. Experiment Setting

To apply each algorithm on a data set, minimum settings are required. We have chosen a 10-fold cross-validation for its well-known advantages; it is helpful when the amount of data for training and testing is limited, which is our case: we try a fixed number of approximately equal partitions of the data, and each in turn is used for testing while the remainder is used for training. At the end, every instance has been used exactly once for testing. To evaluate the accuracies of each model, we have used three criteria: a success rate (percentage of correctly classified instances), the statistic variable Kappa, and the mean absolute error (MAE). These criteria are sufficient to analyze models accuracies. Other options can be tuned depending on researchers' requirements.

4. Hypotheses Verification

To verify the stated hypotheses, we have to transform the data-set in the ‘arff’ format, readable by WEKA, pre-treat the data if it’s necessary, apply a selected algorithm, analyze the results, and compare with other results in the literature.

² Note that we have used the version 3.4.0 of Weka.

4.1. Corrective Maintenance Cost Hypothesis Verification

Table 1 shows the best experiment results obtained when verifying hypothesis H_{ccm} .

Table 1. Results for the H_{ccm} hypothesis

Algorithms	Success rate %	Kappa	MAE
Logistic	56.7	0.12	0.45
Multilayer Perceptron	56.7	0.11	0.47
IB1	57.9	0.16	0.42
KStar	56.7	0.13	0.44

As we notice, obtained accuracies are very medium (success rate around 57%, kappa ranges from weak to medium, and MAE is medium in most results). Hence, we cannot establish that effectively there is a relationship between size and complexity of ADA components, from one side, and their corrective maintenance costs from the other side. However, some classifiers as PART and J48 generate knowledge, which can be used in decision-making processes. We have observed that the “inline_comments” metric is the most relevant to indicate whether the cost of faulty component is high or low. In fact, Grosby [6] states that the programming style and documentation can influence the comprehension of artefacts, and consequently, maintenance efforts. Less relevant are the “blank-lines” and “cyclomatic complexity” measures; they could be also used as good indicators of corrective maintenance costs. This result was also stated by Almeida and al. [1].

4.2. Fault-Proneness Hypotheses Verification

In this section, we examine the results obtained for the three fault-proneness hypotheses related respectively to cohesion, coupling, and inheritance, namely H_{fp-coh} , H_{fp-cou} , and H_{fp-inh} . We have considered only algorithms, which have a success rate greater than 70%.

4.2.1. H_{fp-coh} Hypothesis

Table 2 gives us the best model accuracies. For all the algorithms, the success rate varies from 60% to 76.3%, with 8 algorithms providing a success rate between 70% and 76.3%; kappa fluctuates between weak and good, and MAE isn't significant. Furthermore, algorithms based on rules and decision tree provide information about different relevant metrics. We have drawn that LCOM2, LCOM1, and LCOM5 are good indicators of fault-proneness. This is confirmed by [10] and [4]; it states that lack of cohesion can have an impact on fault proneness.

4.2.2. H_{fp-cou} Hypothesis

For this hypothesis, we have obtained the following results, presented in Table 3. The success rate is between 70% and 80%, kappa varies from weak to excellent, and MAE is considered between weak and medium. We have also found the same conclusions as in [4] and [10], i.e., the use of inheritance coupling and method invocation coupling (measured by RFC) can predict fault proneness of C++ classes.

4.2.3. $H_{fp\text{-inh}}$ Hypothesis

The results provided by this experiment are shown in Table 4. They are better than those of the latter experiments and the success rate varies between 70 and 75, kappa is considered above weak, and MAE is medium. On the other hand, NMI (number of methods inherited) and DIT (Depth of inheritance tree) are significant indicators of C++ classes' fault-proneness. This result confirms the conclusions in [10].

Table 2. Results for the $H_{pf\text{-coh}}$ hypothesis

Algorithms	Success rate %	Kappa	MAE
KStar	71.3	0.25	0.31
HyperPipes	76.3	0.33	0.50
VFI	76.3	0.33	0.48

Table 4. Results for the $H_{fp\text{-inh}}$ hypothesis

Algorithms	Success rate %	Kappa	MAE
Logistic	72.5	0.24	0.39
Multilayer Perceptron	73.8	0.24	0.36
KStar	71.3	0.25	0.31
AdaBoostM1	73.8	0.28	0.33
Bagging	73.8	0.17	0.39
Vote(J48)	72.5	0.17	0.37
HyperPipes	72.5	0.17	0.50
J48	71.3	0.17	0.37
NBTree	72.5	0.11	0.39
Decision Table	73.8	0.17	0.38
PART	75	0.29	0.32

Table 3. Results for the $H_{fp\text{-cou}}$ hypothesis

Algorithms	Success rate %	Kappa	MEA
Logistic	71.3	0.27	0.38
SMO	71.3	0.06	0.29
IBK (k=4)	73.8	0.26	0.38
Bagging	73.8	0.24	0.39
Vote	75	0.32	0.32
VFI	71.3	0.21	0.48
ADTree	72.5	0.29	0.39
J48	75	0.32	0.32
PART	80	0.47	0.28

4.3. Reusability Hypotheses Verification

In this section, the experiments have the objective of verifying three hypotheses related to internal quality attributes (inheritance, coupling at the two levels: code and design, and, size and complexity), and the degree of reusability of C++ classes.

4.3.1. $H_{re\text{-cou}}$ Hypothesis

As we have indicated in section 2.2.1, the experiments are done to verify the hypothesis $H_{re\text{-cou}}$ for the two levels of coupling: design and code. The first experiment results concern coupling at the design level and are given in Table 5. We obtain a success rate from 50% to 66.8%; kappa is between medium and excellent, and MAE is weak in general. We remark that instance-based algorithms have the best scores.

The algorithms that produce models have revealed that export coupling (measured by OMMEC, OCAEC, DCMEC, etc.) has an impact on reusability. This result is confirmed by some results mentioned in [11].

On the other hand, Table 6 gives results obtained at the code level. The accuracies are quite similar to those at the design level. We have also drawn that code level coupling measured by the user data type metric TYI seems to be a good indicator for the degree of reusability.

Table 5. Results for the Hre_cou hypothesis at the design level

Algorithms	Success rate %	Kappa	MAE
IB1	64.3	0.45	0.19
IBK (k=4)	60.7	0.42	0.24
KStar	66.8	0.49	0.17

Table 6. Results for the Hre_cou hypothesis at the code level

Algorithms	Success rate %	Kappa	MAE
IBK (k=4)	60.71	0.40	0.26
KStar	60.71	0.41	0.21
Bagging	64.29	0.45	0.27
Stacking	61.90	0.43	0.28
NBTree	64.29	0.43	0.26

4.3.2. *Hre_inh* Hypothesis

The experiment results are shown in the Table 7. The algorithms provide results with a success rate between 58.3% and 60.7%; kappa is above medium, and MAE is weak in general. Concerning the relevant metrics, DIT and NOP (Number of polymorphic methods) are produced by several models; they seem to be good indicators of the degree of reusability of a C++ class in the same domain (multi-agents systems). Moreover, we have noticed that a combination of the metrics NOP, DIT, and OIM (number of inherited methods) can clearly distinguish if a software component is totally reusable or requires 25% of code changes to be considered as reusable.

4.3.3. *Hre_sac* Hypothesis

Table 8 provides the experiment results. The results are a little lower than those shown in Table 5 and 6; kappa is excellent, and the MAE is weak. We can also distinguish that the metric NPM (number of parameters per method) is identified as a good indicator of reusability. In fact, combined with others like, NOP (number of polymorphic methods) and CSB (Class Size in Bytes), they allow us to determine if a software component is totally reusable or with 25% of code changes.

Table 7. Results for the Hre_inh hypothesis

Algorithms	Success rate %	Kappa	MAE
Logistic	58.3	0.33	0.28
IBK (k=4)	60.7	0.41	0.26
KStar	58.3	0.37	0.23

Table 8. Results for the Hre_sac hypothesis

Algorithms	Success rate %	Kappa	MAE
Multilayer Perceptron	59.52	0.37	0.23
IB1	60.71	0.41	0.21
KStar	59.52	0.39	0.21

5. Conclusion and Discussion

The main objective of this work is to improve the quality of software products, by learning from past projects and capturing some knowledge that will guide future developments. In this work, we have tried to show that machine-learning is a feasible

approach for software product quality prediction. The performances we have obtained vary depending on the targeted quality factor. Concerning the prediction of ADA components corrective maintenance costs, experiments present medium performances but underline that it's necessary to adopt (i) a good programming style and (ii) documentation. The second quality factor is about C++ classes' fault-proneness; it presents the best results in particular with coupling, with a success rate exceeding 70%. Finally, experiments on reusability present medium accuracies but MAE and kappa are very interesting; they can be used to select the best model. We have repeated a subset of experiments with the filter *attribute selection*, and have noticed that some results are better than without the attribute selection. So, it will be interesting to repeat all the experiments with a prior attribute selection.

Another important point of this work is that some models provide knowledge about internal attributes; this knowledge takes the form of rules or decision trees. It is explicit and can be integrated in a knowledge-based decision-making system architecture. This kind of architecture separates knowledge from procedures that manipulate it; they can be very useful for software product quality assessment.

However, models obtained have to be confirmed and generalized by more experiments on more software data, extracted from various and representative applications. This is the challenge to produce relevant and reusable models.

References

- [1] M.A. De Almeida, H. Lounis, W. Melo, "An Investigation on the Use of ML Models for Estimating Software Correctability," Journal of Software Engineering and Knowledge Engineering, Oct. 1999.
- [2] V. Basili, Condon, K. El Emam, R. B. Hendrick, W. L. Melo, "Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components," Proc. of the IEEE 19th Int'l. Conf. on S/W Eng., Boston, 1997.
- [3] L. C. Briand, P. Devanbu, W. L. Melo, "An Investigation into Coupling Measure for C++," Proc. of 19th Int. Conf. S/W Eng., 1997.
- [4] L.C. Briand, J. Wüst, H. Lounis, "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs," ISERN-98-29-Version 3.
- [5] L. C. Briand, J. Wüst, and H. Lounis, "Using Coupling Measurement for Impact Analysis in Object-Oriented Systems" in proceedings of the International Conference on Software Maintenance ICSM'99, Oxford, England, August 30 – September 3, 1999
- [6] M.E. Crosby, J. Scholtz, S. Wiedenbeck, "The Roles Beacons Play in Comprehension for Novice and Expert Programmers," Proc. of the 14th Annual Workshop of the Psychology of Programming Interest Group London. UK. pp 58-73.
- [7] G.H John., P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," In Proceedings of the Eleventh Conf. on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, 1995.
- [8] W. Li, S. Henry, "Object-Oriented Metrics that Predict Maintainability," J. Systems and Software, 23 (2), 111-122, 1993.
- [9] H. Lounis, W. Melo, "Identifying and Measuring Coupling on Modular Systems," Technique Report N. CRIM-97/05-79.
- [10] H. Lounis, L. Ait-Mehdine, "Machine-Learning Techniques for Software Product Quality Assessment," Proc. of the 4th IEEE Int. conf. on Quality Software, Sept. 2004, p. 102-109.
- [11] Y. Mao, H. A. Sahraoui, H. Lounis, *Reusability Hypothesis Verification using Machine Learning Techniques: A Case Study*. ASE 1998: 84-93, 1998.
- [12] A. Porter, R. Selby, "Empirically-guided software Development using metric-based classification trees," IEEE Software, 7(2): 46-54, March 1990.
- [13] R.S Pressman, *Software engineering: a practitioner's approach*, 4th ed. New York, N.Y. : McGraw-Hill, 1997.
- [14] I.H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*, Morgan Kaufmann Publishers, San Francisco, California, 2000.

Deep Semantics of Visual Languages

Pavel Grigorenko^a, Enn Tyugu^a

^a Institute of Cybernetics, Tallinn University of Technology
Akadeemia tee 21
12618 Tallinn
Estonia

Abstract. Visual specification of software is gaining popularity, but its usage is restricted by the lack of precise semantics of visual languages. In the present work, precise semantics of visual languages is defined. Three kinds of deep semantics of schemes are presented as different ways of usage of attribute models of schemes. Attribute models of a wide class of schemes are defined and higher-order attribute models are introduced. Dynamic evaluation of attributes is used in defining semantics of schemes.

Keywords. Software synthesis, higher-order attribute models, semantics of visual languages, shallow and deep semantics of schemes, dynamic evaluation of attributes.

1. Introduction

Our aim is to give a possibly general way to implement *deep semantics* of visual languages, i.e. to give tools that enable one to program in a systematic and sufficiently simple way transformations that automatically produce the meaning of a visually represented artifact. In the domain of programming languages this kind of semantics is supported by attribute grammars that enable one to program a stepwise transformation of a source text into the code. Attribute grammars have been already used for representation of semantics of visual specifications in some specific applications [1], [5].

We consider here visual languages that are not restricted to any specific domain, but still have a well-defined syntactic structure. Actually, we have to restrict us to the languages where a visual specification has a definite structure that can be represented as a graph. One could call such languages *scheme languages*. Then a sentence in a language of this kind is a *scheme*. Languages of schemes are often used in engineering domains, e.g. schemes of electrical, logical, mechanical etc. devices. Generally, considering a structure of a visual description of some artifact, one comes always to a scheme representing components and their relations (connections), although this scheme is not always explicit. Also considering the process of composing a picture on a screen of the computer, we always see that the picture is composed of elements of various types related to each other in some definite ways, often positionally. Even an unordered collection of icons on a desktop is a scheme in our meaning. The only relation that binds the icons in this case is the relation expressing that the icons belong to one and the same container – the desktop. Speaking about visual representation of schemes, we allow hidden (not visible) relations in a scheme.

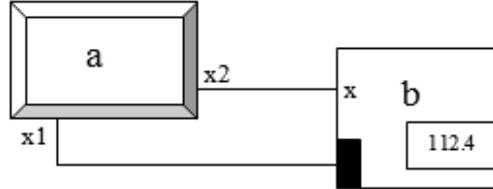


Figure 1. A scheme.

Fig. 1 shows basic features of a visual language: 1) visual identification of type of a component by shape of image, 2) connecting particular ports or variables ($x1$, $x2$, x) of components, introducing names of components (a , b), assigning value (112.4) to a field of a component. Some names are implicit, for example, the variable v behind the given value 112.4 and the name z of the port denoted by the dark element of the component b . The specification of this scheme in a textual language can be, for instance, as follows:

```
ClassP a;
ClassQ b;
a.x1 = b.z;
a.x2 = b.x;
b.v = 112.4;
```

This specification includes declarations of components, equalities that bind variables belonging to components and an assignment of value to a variable. (Values can appear also implicitly, for instance, as coordinates of components in a scheme.)

In the present work, we first introduce attribute models, including higher order attribute models, as well as attribute evaluation on these models. Thereafter we define shallow and deep semantics of visual specifications, and describe a method of representing the deep semantics of schemes. In the last sections, we give an example of deep semantics of a visual language and refer to the system CoCoViLa that is a tool for implementing the semantics of schemes.

2. Attribute models

In conventional attribute grammars [4], [6], attribute dependencies are related to productions of the syntactic part of a grammar. In our case we have no productions, and we are going to define a concept of attribute model that can represent both semantics of a single object or semantics of a scheme as a whole.

Definition 1. Attribute is a variable with a type.

Definition 2. Attribute dependency is a relation between attributes that is represented by one or several functional dependencies whose inputs and outputs are attributes that are bound by this relation.

Let us introduce two notations for functional dependency $(y_1, \dots, y_n) = f(x_1, \dots, x_m)$ where f is a function of m arguments computing a value of n -tuple. The variables x_1, \dots, x_m are *inputs* and y_1, \dots, y_n are *outputs* of the functional dependency. When we present only the type of the dependency, then we denote it by

$$x_1, \dots, x_m \rightarrow y_1, \dots, y_n,$$

which means for us that having x_1, \dots, x_m we can compute y_1, \dots, y_n .

When we present also the implementation f of the dependency, then we denote it by

$$x_1, \dots, x_m \rightarrow y_1, \dots, y_n f\}.$$

We assume here that attribute dependencies can be presented by equalities, structural relations, equations and preprogrammed procedures (methods of a class).

1. *Equality* $x = y$ can be rewritten as $x \rightarrow y; y \rightarrow x$.
2. *Structural relation* binding a tuple of variables x_1, \dots, x_m with a structured variable $x = (x_1, \dots, x_m)$ can be presented as $x_1, \dots, x_m \rightarrow x; x \rightarrow x_1, \dots, x_m$.
3. *Equation*, i.e. $x = y + z$ can be presented as a collection of functional dependencies, in the given example as $y, z \rightarrow x; x, y \rightarrow z; x, z \rightarrow y$.
4. *Preprogrammed procedure* with attributes x_1, \dots, x_m as parameters producing a value of attribute y can be presented as $x_1, \dots, x_m \rightarrow y$.

Definition 3. An attribute model M is a pair (A, R) , where A is a finite set of attributes and R is a finite set of attribute dependences binding these attributes.

Attribute models $M' = (A', R')$ and $M'' = (A'', R'')$ can be composed into a new attribute model binding some of their attributes by equalities. Let us have a set of equalities $s = \{M'.a = M'.b, \dots, M'.d = M''.e\}$ binding some attributes of models M' and M'' . By $\cup_s(M', M'')$ we denote an attribute model with the set of attributes $A' \cup A''$ and the set of attribute dependences $R' \cup R'' \cup s$. Let us generalize the composition of attribute models for more than two models as follows.

Definition 4. For a set of equalities s that bind some attributes of models M_1, \dots, M_n we denote by $\cup_s(M_1, \dots, M_n)$ an attribute model with the set of attributes $A_1 \cup \dots \cup A_n$ and the set of attribute dependences $R_1 \cup \dots \cup R_n \cup s$, and call it composition of M_1, \dots, M_n with bindings s .

Remark 2.1. When building a composition of attribute models, renaming of attributes may be required. A straightforward way to do it is to add the name of a model where an attribute came from to its name. This introduces composite names, e.g. $m.x, m.y$ for attributes x, y of an original model m .

Remark 2.2. Any attribute model can be presented in a *flattened form* where all attribute dependences are functional dependencies. In order to do this, one has to

consider relations between attributes as sets of functional dependencies and take their union for the set of attribute dependencies of the attribute model in the flattened form.

A simple undirected graph $G = (V, E)$ is called bipartite if there exists a partition of the vertex set $V = V_1 \cup V_2$ so that both V_1 and V_2 are independent sets. $G = (V_1 + V_2, E)$ denotes a bipartite graph with partitions V_1 and V_2 .

Remark 2.3. An attribute model (A, R) can be presented as a bipartite graph with partition of the set of nodes $A \cup R$. There is an edge (a, r) in the graph if and only if the attribute a is bound by the attribute dependency r .

Remark 2.4. An attribute model (A, R) in the flattened form can be presented as a directed bipartite graph with sets of nodes A and R . There is an arc from a to r , if and only if a is an input of r and an arc from r to a , if and only if a is an output of r .

3. Evaluation of attributes

Let U and V be two sets of attributes of an attribute model M . We call a pair (U, V) a computational problem on the attribute model M . The meaning of a computational problem (U, V) is that given values of attributes from U find values of attributes of V using attribute dependences of M . We show that there is a procedure that for any computational problem (U, V) on a attribute model (A, R) in the flattened form decides whether there is a way to compute values of attributes of V from given values of attributes of U , and in the case of the positive answer produces an algorithm for computing the values, i.e. produces an algorithm for solving the computational problem.

Definition 5. Value propagation is a procedure that for an attribute model M in flattened form and a set of attributes U that belong to this model decides which attributes are computable from U and produces a sequence of functional dependences that is an algorithm for computing values of these attributes.

A simple value propagation algorithm works step by step as follows. At each step it checks for each functional dependency whether its inputs are all computed and some of its outputs is not computed. In the positive case, the functional dependency will be added to the algorithm being built and all outputs of the functional dependency will be added to the set of computed attributes. Initially the set of computed attributes equals to the set of given attributes U and algorithm (i.e. the sequence of functional dependencies) is empty.

Remark 3.1. There are very efficient algorithms for value propagation that work in linear time with respect to the size of attribute model, see [8].

Remark 3.2. Value propagation is a procedure that decides for a computational problem whether it is solvable, and in the case of positive answer gives an algorithm of solving the problem.

Remark 3.3. There is a procedure that gives a minimal algorithm for solving a computational problem, see [3].

Remark 3.4. Set of all solvable computational problems on an attribute model is well defined, hence we can decide that an attribute model is an attribute dependency with algorithms for solving computational problems as its set of functional dependencies.

4. Higher-order attribute models

Unfortunately, attribute models as defined above are not very expressive. One can compose only linear sequences of functional dependencies for solving computational problems on them. We are going to define now more expressive attribute models that enable one to synthesize more complex algorithms. First let us notice that we can formally define computational problems on some set of attributes, without making precise which functional dependencies one can use for solving the problems.

Definition 6. Higher-order functional dependency on a given set of attributes is a functional dependency that has also computational problems on this set of attributes as inputs.

Definition 7. Higher-order attribute model is a pair (A, R) where A is a set of attributes and R is a set of attribute dependencies that include some higher-order functional dependencies on the set of attributes A .

Considering types of functional dependencies, one can see that besides functional types like $x, y \rightarrow z$ we have now also types like $(u \rightarrow v), x \rightarrow y$, where $u \rightarrow v$ represents a computational problem. This extension makes a big difference in the following: Higher-order attribute models are so expressive that enable one to synthesize recursive, branching and cyclic programs. In order to do this one has to present some control structures, e.g. loops and conditional branching as higher-order functional dependencies. Detecting solvability of a problem and synthesizing an algorithm on a higher-order attribute model has exponential time complexity. The problems on higher-order attribute models can be analyzed by the same technique that has been applied for higher-order constraint networks, see [8], therefore we are not going to discuss them here in any detail, but we will use the higher-order attribute models in semantics of schemes.

5. Syntax and textual presentation of schemes

Let us have a finite number of types. A *node type* is an expression $t(a, \dots, b)$, where t is a type and a, \dots, b are typed variables, i.e. pairs of the form (x, t') , where x is a variable and t' is its type. The variables a, \dots, b are called *ports* of the node type $t(a, \dots, b)$. A scheme is a set of pairs (u, t) called *nodes*, where u is a name (identifier) and t is a node type, and a set of equalities $u.a=v.b$ called *bindings*, where a, b are ports of one and the

same type of nodes u, v respectively, and a set of valuations $u.a = v$, where v is a value of suitable type.

A scheme as defined above can be presented by a text in a very simple language that has three kinds of statements:

1. declaration of a component:

$<\text{type}> <\text{idennitifier}>$

This declaration specifies a component of a scheme with given type, and its name given by identifier.

2. binding:

$<\text{name of component}>. <\text{name of port}> = <\text{name of component}>. <\text{name of port}>$

This statement specifies an equality between variables of components. These variables are also attributes of attribute models of components.

3. valuation:

$<\text{name of component}>. <\text{name of port}> = <\text{value}>$

This statement defines a functional dependency with no inputs and one output that gets a constant value.

6. Semantics of schemes

6.1. Shallow semantics

When we consider a particular problem-domain, a scheme representing something in this domain typically has what one could call the *real* semantics, the meaning of the scheme that a specialist of that domain recognizes and understands. The obvious aim is to represent this knowledge (at least to some extent) in a computer. For this, we have to reason about the semantics of schemes on several different levels.

Firstly, one can consider the shallow semantics of a scheme – the textual representation of the graph underlying the scheme. On this level, all deeper information about the scheme is disregarded. To be more precise – it is hidden in the types (classes) of objects.

Definition 8. The shallow semantics of schemes is a function SS , which, for each scheme G , returns a string including exactly the following:

- Type obj ; whenever G includes an object named obj of type $Type$.
- $obj.x = v$; whenever an attribute x of object obj has the value v in scheme G .
- $obj1.x = obj2.y$; whenever there is an edge between ports x and y , and these ports are associated with objects $obj1$ and $obj2$, respectively.

6.2. Deep semantics

Definition 9. Attribute model of a scheme is the composition of attribute models of its nodes bound by the equalities of its shallow semantics and including additional functional dependencies for values given in the scheme. Its attributes, functional dependencies and attribute dependencies are called also attributes, functional dependencies and attribute dependencies of the scheme.

More formally, if a scheme has nodes obj_1, \dots, obj_n with attribute models M_1, \dots, M_n respectively, and the equalities of shallow semantics constitute a set $s = \{ obj_i.a = obj_j.b, \dots, obj_s.d = obj_k.e \}$, and valuations are $obj_q.x=v_1, \dots, obj_p.y=v_l$, then the attribute model is $\cup s(M_1, \dots, M_n) \cup \{ obj_q.x=v_1, \dots, obj_p.y=v_l \}$.

Let U and V be two sets of attributes of scheme. We call a pair (U, V) a computational problem on the scheme. We know that there is a procedure that for any computational problem (U, V) on a scheme decides whether there is a way to compute values of attributes of V from given values of attributes of U , and in the case of the positive answer produces an algorithm for computing the values, i.e. produces an algorithm for solving the computational problem, see attribute evaluation in Section 3. Let us denote by $S1$ the function producing an algorithm for any solvable computational problem on a scheme.

Definition 10. The deep semantics $DS1$ of schemes is a composition of the shallow semantic function SS and of the semantic function $S1$ that defines computability on a scheme.

If $U1 \subseteq U2$ and $V2 \subseteq V1$ then we say that the computational problem $(U1, V1)$ is larger than the computational problem $(U2, V2)$. Let us denote by $S2$ the function that produces an algorithm for solving the largest solvable computational problem with empty set of input attributes on the scheme.

Definition 11. The deep semantics $DS2$ of schemes is a composition of the shallow semantic function SS and of the semantic function $S2$ which defines largest computability on a scheme.

The third semantic function is defined as an attribute evaluator of an attribute grammar. Up to now we have silently considered attributes as meaningful variables of a problem domain. However, in the most general case attributes can be just variables bearing some semantic information. The real meaning of a scheme can be computed as a value of a distinguished attribute, let us call it a *scheme attribute* on the attribute model of the scheme. We denote by $S3$ the function that, given a scheme, computes its scheme attribute value. The scheme attribute corresponds to the synthesized attribute of a nonterminal symbol representing the whole program in the conventional case of attribute grammars of programming languages, and in our case it represents the meaning of the whole scheme. The only essential difference between our approach and the conventional dynamic attribute evaluation is that we have to use a more sophisticated attribute evaluator, because instead of an abstract syntax tree we have a graph representing a scheme that in a general case is not a tree, and we accept higher-order attributes as well.

Definition 12. The deep semantics DS3 of schemes is a composition of the shallow semantic function SS and of the semantic function S3 that computes the value of a scheme attribute.

We have to notice that implementing DS3 is a much harder task than implementing DS1 and DS2. It is really a compiler-writing task for a visual language, whereas the latter two semantic functions are implemented easily by specifying domain-oriented functional dependencies.

7. Example

The goal of this example will be to demonstrate the use of attribute models and the attribute evaluation process. Firstly, we define attribute models corresponding to the objects of problem domain, then give a computational problem and finally provide a solution as a result of attribute evaluation. The language for the specification of attribute models that includes equalities, structural relations, equations and preprogrammed functional dependences (see Section 2) will be extended with declarations of components as used in textual specifications of schemes (Section 5) i.e. declaration of component(s) has the form $\langle type \rangle \langle identifier \rangle, \dots$. We are going to use only textual specifications of attribute models from now on.

The problem domain for this example is direct current circuits. Let us define an attribute model of resistor, using the Ohm's law, by the following text:

```
Res:
  numeric u, u1, u2, r, i;
  u = u2-u1;                                (1)
  u = i*r;                                  (2)
  p1 = (u1, i);                            (3)
  p2 = (u2, i);                            (4)
```

Let us have a scheme with two resistors R1 and R2 connected that have resistances 6 and 12, as well as the current R1.i=0.5 and potential R1.u1=0. This is expressed by the specification:

```
Res R1, R2;
R1.p2=R2.p1;                                (5)
R1.r=6;                                     (6)
R2.r=12;                                     (7)
R1.i=0.5;                                    (8)
R1.u1=0;                                     (9)
```

The attribute model of this scheme is a composition of two attribute models of resistors. It is shown in Figure 2 as a bipartite graph, including 13 attribute dependencies. If we decide to show the attribute model in the flattened form, we have to draw 28 functional dependencies. According to the deep semantics DS2 one can calculate values of all attributes. This can be demonstrated by value propagation on the attribute model. One of the possible sequences of application of functional dependencies is 9[R1.u1=0], 6[R1.r=6], 7[R2.r=12], 8[R1.i=0.5], 5[R2.i=R1.i], 2[R1.u=(R1.i*R1.r)], 1[R1.u2=R1.u+R1.u1], 2[R2.u=R2.i*R2.r], 5[R2.u1=R1.u2], 1[R2.u2=R2.u+R2.u1],

where $5[R2.i=R1.i]$ denotes application of the functional dependency $R2.i=R1.i$ after flattening of the attribute dependency number 5, etc.

8. Implementation

The semantics of visual languages defined above has been implemented in a tool CoCoViLa [10], [11]. From a user's point of view this framework consists of two components: *Class Editor* and *Scheme Editor*. The *Class Editor* is used for defining attribute models of components of schemes as well as their visual and interactive aspects.

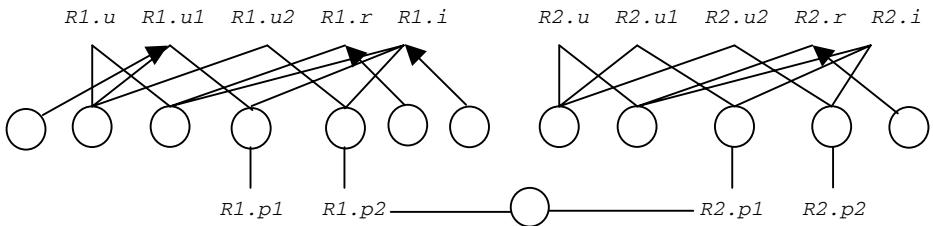


Figure 2. The composition of two attribute models of resistors

The *Scheme Editor* is a tool for usage of visual languages, i.e. developing schemes, compiling and running programs. It includes an attribute evaluator and implements all three deep semantics defined in the present paper. The *Scheme Editor* provides an interface for visual programming – building a scheme from visual images of components. The environment generated for a particular visual language allows the user to draw, edit and use schemes in computations through language-specific menus and toolbars.

9. Example continued

In order to demonstrate CoCoViLa, we extend the example from the section 7 by adding two more concepts into our language of circuits – parallel and series connections. The specifications of attribute models presented in CoCoViLa are as follows:

```
class Resistor {
    /*@ specification Resistor {
        double u, u1, u2, r, i;
        u = u2-u1;
        u = i*r;
        alias p1 = (u1, i, r);
        alias p2 = (u2, i, r);
    }@*/
}
```

```

class Parallel {
    /*@ specification Parallel {
        double u, u1, u2, i, i1, i2, r, r1, r2;
        i = i1 + i2;
        1/r = 1/r1 + 1/r2;
        u = i*r;
        u = u2 - u1;
        alias p1 = (u1, i, r);
        alias p2 = (u2, i, r);
        alias p3 = (u1, i1, r1);
        alias p4 = (u1, i2, r2);
        alias p5 = (u2, i1, r1);
        alias p6 = (u2, i2, r2);
    }@*/
}
class Series {
    /*@ specification Series {
        double u, u1, u2, u3, r, i, r1, r2;
        r = r1 + r2;
        u = i*r;
        u = u2 - u1;
        alias p1 = (u1, i, r);
        alias p2 = (u2, i, r);
        alias p3 = (u1, i, r1);
        alias p4 = (u3, i, r1);
        alias p5 = (u3, i, r2);
        alias p6 = (u2, i, r2);
    }@*/
}

```

Having developed the visual language we are able to load it in the Scheme Editor and build schemes by putting visual objects on the drawing canvas and connecting them through ports. Figure 3 represents a scheme containing parallel and series connection of three resistors. We see here also a pop-up window of attribute values for resistor_1.

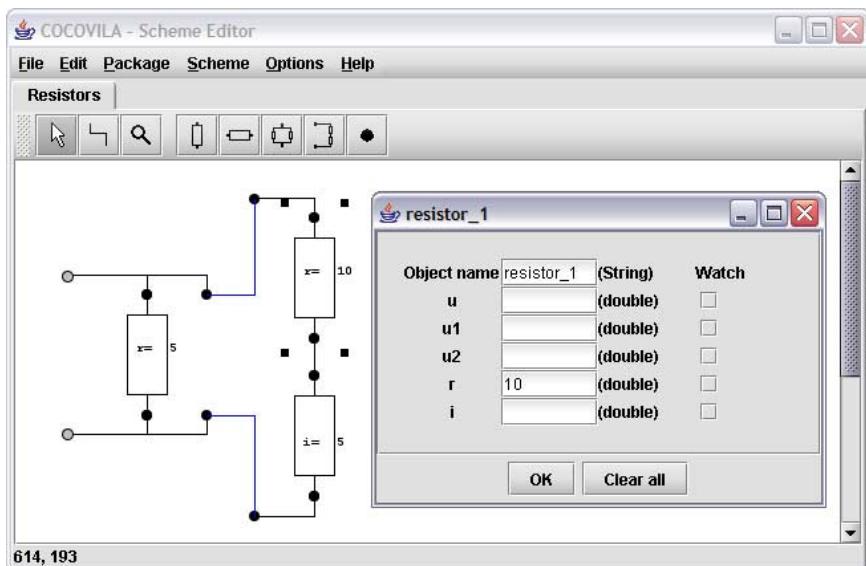


Figure 3. A scheme

The corresponding shallow meaning of a given scheme is as follows:

```
Parallel parallel_1;
parallel_1.u1 = 0;
parallel_1.u2 = 100;
Resistor resistor_3;
resistor_3.r = 5;
Resistor resistor_1;
resistor_1.r = 10;
Series series_1;
Resistor resistor_2;
resistor_2.i = 5;
parallel_1.p3 = resistor_3.p1;
parallel_1.p5 = resistor_3.p2;
resistor_1.p1 = series_1.p3;
resistor_1.p2 = series_1.p4;
resistor_2.p1 = series_1.p5;
resistor_2.p2 = series_1.p6;
series_1.p1 = parallel_1.p4;
series_1.p2 = parallel_1.p6;
```

After invoking the attribute evaluator, an algorithm containing 32 computational steps is produced. This algorithm corresponds to the deep semantics DS2 of the scheme, namely it solves the largest solvable problem on the scheme. Figure 4 shows the results of applying this algorithm as a visual feedback in the Scheme Editor window, as well as values of attributes of resistor_1 in the pop-up window.

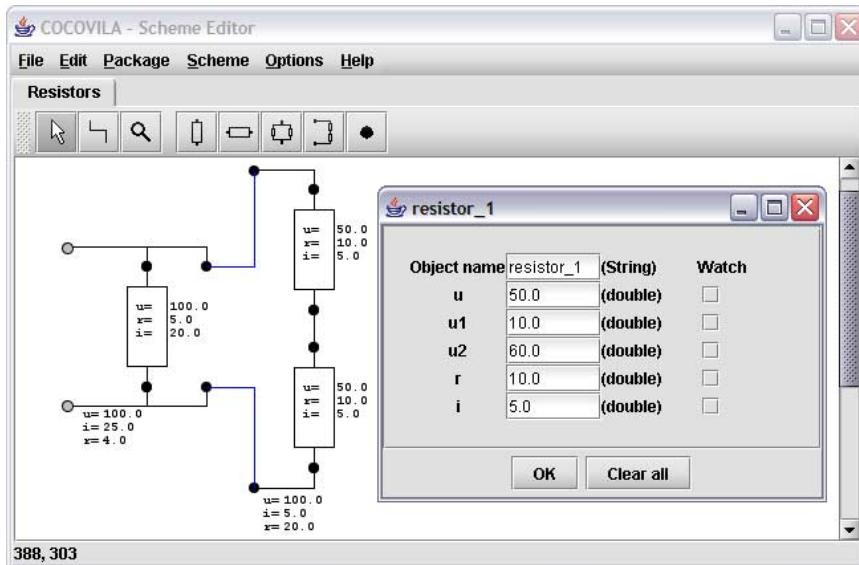


Figure 4. A scheme after evaluation

10. Related work

There are a number of works where attributes are associated with nodes of a graph. Götler [2] presents a formalization of the notion *graphic*. A graphic is considered to consist of a graph describing the overall structure and a set of attributes describing the shape, placement, etc. of the nodes and edges of the underlying graph. The formal handling of graphics is done by attributing the rules of graph grammars and by passing the attributes up and down the derivation tree of the graphic.

The paper from Alpern, et al. [1] introduces the concept of *attributed graph specification (AGS)* and develops a theory of strongly typed graphs. Graphs are modeled as collections of boxes connected by cables. Cables and boxes contain ports. A cable and a box are connected through ports. A graph is specified as a composition of boxes and cables. Attributes are associated with boxes, and attribute evaluation rules are attached to the composition rules. An attribute evaluation rule associated with a box composition can additionally specify a direction of flow of the attribute values from one port to another port. We use the similar technique in the construction of schemes, i.e. boxes are visual components with ports, and cables are the bindings.

Mandayam [5] introduces PDL as performance modeling language based on Attributed Nodes-Only Grammars. ANGs are a subset of graph grammars that are extended with attributes and their evaluation rules. The PDL is used for the specification of performance attributes supplied with *computable* functions for the evaluation of these attributes and indicating *temporal* relationships between design instances and the evaluation of attribute values. The proposed language allows to attach attributes to objects in the design and to propagate attribute values up, down and laterally across various levels in the design hierarchy. Like the specifications introduced in [1], design objects in PDL are *modules*, *carriers* and *ports*. Carriers represent wires connecting modules using ports. Performance attributes are classified into two categories: *static* and *dynamic*. Static attributes do not depend on dynamic attributes and once evaluated, do not change with a change in the data. The concept of controlled cyclic dependencies among attribute occurrences is introduced and a mechanism to conceptually break cycles in order to determine an evaluation sequence is presented.

The NUT system [9] is a programming tool supporting declarative programming in a high-level language, automatic program synthesis and visual specification of problems by means of schemes. The central part of the system is Structural Synthesis of Programs (SSP), which uses intuitionistic propositional logic. NUT restricts its attention to constructing programs from pre-programmed modules, rather than from primitive instructions of a programming language. The NUT specification language is an object-oriented language extended with features for program synthesis, the pre-programmed modules are methods of classes supplied with specifications.

11. Conclusions

In this work we have proposed precise definitions of three kinds of semantics of schemes that constitute a wide class of visual languages. We have used attribute models in order to attach meaning to schemes. We have extended the attribute models by introducing a higher-order feature – subtasks as inputs of attribute dependencies. The subtasks require automatic synthesis of algorithms for their solution. The semantics of

schemes is implemented by using a dynamic attribute evaluator that works as an automatic program synthesizer. This work extends the results in attributed graphs in two ways: first, by introducing precise semantics of schemes, and second, by extending the attribute models. Our semantics of schemes has been implemented in a software tool CoCoViLa that is publicly available from <http://www.cs.ioc.ee/~cocovila/>. Finally, it has to be said some words about scalability of our method. Thanks to the linear time complexity of value propagation, the scalability of the attribute semantics without higher-order attribute relations is excellent. We have practically solved computational problems on attribute models with thousands of attribute dependences. In the general case, the search time depends very much on the number of higher-order attribute dependences in the synthesized attribute evaluation algorithm.

Acknowledgements

This work has been supported by the Estonian Science Foundation grant No. 6886.

References

1. Alpern, B., Carle, A., Rosen, B., Sweeney, P., Zadeck, K. *Graph Attribution as a Specification Paradigm*. ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, pp. 121–129, 1988.
2. Götlér, H. *Attributed Graph Grammars for Graphics*. Graph Attribution and their Application to Computer Science, vol. 153 of LNCS, pp. 130-142, Springer-Verlag, 1983.
3. Harf M., Tyugu E. Algorithms for Structural Synthesis of Programs. Programming and Computer Software, No.4, pp. 3 – 11, 1980.
4. Knuth, D. *Semantics of Context-free Grammars*. Mathematical Systems Theory, vol 2, pp. 127-145, 1968.
5. Mandayam R. *Performance Modeling of VLSI Systems*. PhD thesis, University of Cincinnati, Cincinnati, OH, 1994.
6. Penjam, J. *Computational and Attribute Models of Formal Languages*. Theoretical Computer Science, vol. 71, pp. 241 – 264, 1990.
7. Tyugu E. *Attribute Models Of Design Objects*. Proceedings IFIP TC 5 / WG 5.2 Workshop on Formal Design Methods for CAD, Tallinn, Estonia, pp. 16-19, 1993.
8. Tyugu E., Uustalu T. *Higher-Order Functional Constraint Networks*. Constraint Programming. NATO ASI Series F : Computer and Systems Sciences, Vol. 131, Springer-Verlag, pp. 116-139, 1994
9. Tyugu E., Valt R. *Visual programming in NUT*. Journal of visual languages and programming, Vol. 8, pp. 523 – 544, 1997.
10. Grigorenko P., Saabas A., Tyugu E. *Visual Tool for Generative Programming*. Proc. of the Joint 10th European Software Engineering Conference (ESEC) and the 13th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-13). ACM Publ., p. 249 – 252, 2005.
11. Grigorenko P., Saabas A., Tyugu E. *COCOVILA – Compiler-Compiler for Visual Languages*. J. Boyland, G. Hedin. Fifth Workshop on Language Descriptions Tools and Applications LDTA2005. ETAPS, pp. 101 – 105, 2005.

Integrated System Analysis Environment for the Continuous Consistency and Completeness Checking

Erki EESSAAR

*Department of Informatics, Tallinn University of Technology, Raja 15, 12618 Tallinn,
ESTONIA
eessaar@staff.ttu.ee*

Abstract. Modelers create models of a system during information system development. Models should be correct, complete and consistent in order to be most useful. Modeler must have possibility to use software that helps to automate these kinds of checks in order to raise the speed of work and quality of its results. This article describes the software system that allows to perform strategic- and detailed analysis of a database-centric information system without using complicated visual notations. System supports methodological framework for the Enterprise Information System (EIS) strategic analysis (like the one that is described by Roost et al. [1]). User of the proposed system records specification in a database as a single integrated model. We propose database queries that help to find consistency and completeness (CC) problems of the recorded models.

Keywords. Model multiplicity, metamodeling, consistency, completeness, CASE

Introduction

Successful development of an information system takes a lot of effort. Important part of this work is modeling of the system. Common approach is to create different types of models that describe different aspects (dimensions) of the system with the different level of abstraction. An example is the Zachman framework for the information systems architecture [2]. Each model type is used in order to describe one aspect of a system. For example, system is described in terms of different views in case of visual modeling language UML. Each view has one or more corresponding diagram types. UML version 1.5 specifies *nine* types of diagrams [3] and UML 2.0 specifies *thirteen* types of diagrams [4]. In the different projects only subsets of these diagram types are used, depending on the goals. But more than one type of diagrams is needed in order to describe the static structure as well as the behaviour of the system. For example, Larman [5] presents a possible structure of a system analysis specification. The specification must contain UML diagrams (visual models) as well as textual models:

- Use case model (diagrams and textual specification of use-cases).
- Domain model (diagrams and textual specification of conceptual classes and attributes).
- System sequence diagram.
- Contracts of the system operations.

These models together also constitute a model. Final versions of the models that describe a system have to be syntactically and semantically correct, complete and consistent within itself and with each other in order to be most useful. If model is the combination of diagrams and textual description, then there can be inconsistencies between these components.

Examples of the inconsistencies *within one* model:

- Use cases/actors have different names in a diagram and in a text.
- There are different amount of use cases/actors in a diagram and in a text.
- A use case is associated with the different actors in a diagram and in a text.

Examples of the inconsistencies across different models:

- Names of the actors are different in a use case model and in the sequence diagrams that describe system operations.
- Names of the elements of a domain model differ from the names that are used in the pre- or post conditions of the contracts of the system operations.

Examples of the completeness problems:

- A domain model is missing.
- A use case diagram is not accompanied with a textual specification.
- An operation contract doesn't have the post-conditions.

Checking of the models in order to find such problems can be at least partially automated by the software system. Models could be inconsistent and incomplete during the development process. But it should be possible at any time to get information about these problems. It helps to gradually improve the quality of the models. Pedagogical pattern *Built in Failure* [6] suggests that teacher should remove the fear of failure as a barrier to learning by making failure a part of the learning process. A hypothesis that must be controlled in the future is that the checking functionality would also change a modeling tool to a valuable learning tool. It is because continuous feedback from the system instead of a teacher reduces fear to make mistakes that hampers the learning process. It also eases the work of a teacher who doesn't have to check consistency and completeness (CC) problems manually.

But CASE systems today do not provide enough support for checking consistency between different types of models [7], [8] and between evolving versions of models [9]. Instead it is mostly the manual work and takes quite a long time and a lot of effort. And even if CASE tools provide some support it is still a "daunting tasks beyond anyone's cognitive ability" [10] because of big amount of different types of models.

An author of this article teaches database design in a university. A part of the course work is a term project. Students have to create a strategic- and detailed analysis of an information system and a prototype of its software. Current structure of the project documentation has been used during the last four years. For example, 75 projects where presented in the spring of 2005. Teacher reviewed projects together with their authors and pointed to the mistakes. Average length of a review of one project was about 20 minutes. Average interval between first checking of a project by the teacher and acceptance of the project was 4.2 days. Students improved their projects during this period and sometimes they did it repeatedly. Students used word processor in order to write textual models and CASE-tools or diagram editors in order to draw visual models. These systems don't support automatic consistency and completeness checks. Presented projects reflected this situation and contained many such deficiencies. Another problem is that students have difficulties to understand how all

these different models are connected with each other. Therefore the system is needed that gives fast and precise feedback to the students.

Delen et al. [8] write: "Second, there is a need for a completely Web-based integrated modeling environment for distributed collaborative users." Our *main contribution* is description of the system that allows to perform strategic- and detailed analysis of a database-centric information system without using complicated visual notations. This system should ease creation of an independent work by the students and correction of it by the teachers. It can also be used in the real-world information system development projects. System should support *methodological framework* for the *Enterprise Information System (EIS) strategic analysis* (for example, framework that is described by Roost et al. [1]). Person with the modeler role will record system specification in a *database* as an *integrated model* using one tool. Modeler doesn't have to create a *collection* of weakly connected models by using different tools any more. Modeler will use the modeling language that is simplified synthesis of the different system specification languages (UML [3], SMX [11] and OPM [10] among others). The structure of the database will be derived from the fixed metamodel of this language. We plan to use DBMS that allows to use *SQL* language. Because of the fixed metamodel this system will be more similar to a CASE than to a Meta-CASE tool. Data about the various aspects of a system will be recorded in a database using a *form-based web interface*. We propose *queries* that find consistency or completeness (CC) problems that are present in a specification. Our approach eliminates problems with the inconsistencies between diagrammatic and textual representations. They are kind of views to the information in the database that could be generated by the system at any time. We don't have yet completely implemented the system but have started to create its prototype. We see that such a system will help students/teachers in the learning/teaching process and therefore continue its development. It will also help to improve quality of the result of the real information system development projects.

The rest of the paper is organized as follows. Section 1 gives an overview of the existing work about checking consistency and completeness of the models. Section 2 presents the metamodel of the modeling language that our system will use. Section 3 lists queries that help to identify consistency and completeness problems of the model. Section 4 contains discussion and comparison of our work with the existing work in the field. Section 5 summarizes this article.

1. Related Works

UML is nowadays *de facto* standard for describing the information systems. Problems of consistency between UML diagrams have for example acknowledged by Engels and Groewegen [12] who describe open issues in the object-oriented development.

Modeler has to learn a lot of different notations, rules and guidelines that are used in the different types of models. McLeod [13] notes that UML 1.1 contains 233 discrete concepts. Still he proposes rich visual notation for process models which could replace UML dynamic diagrams [13]. If we study one UML or other similar model, then we have to constantly look the other models in order to fully understand it. Switching between different pages/ files/ packages is inconvenient as well as mentally challenging and wearying. "Multiplicity of representational styles impedes communication between modeling professionals and their clients." [14] Visual models are usually created with the different CASE or model drawing tools (Rational Rose, ERWin, ArgoUML, Visio,

Dia etc.) and are accompanied with the textual specifications that are created using a text editor. We have to have this software in our computer if we want to thoroughly study these models or modify them. Modification of one model requires modifications of dependent models as well. Multiplicity of modeling software and files that contain models causes often creation of the model from scratch instead of reusing the existing ones. Some approaches that are used in order to achieve correct and consistent models:

1. Formulation of the guiding rules that a modeler who creates different types of models should follow. For example, Glinz [15] describes rules that help to minimize inconsistencies between a class model and a use case model.
2. Usage of the cross references between the different types of models. For example, Glinz [15] proposes to use references to a class model in the scenarios of use cases.
3. Usage of the specific models which contain cross references between other models. Example of such a model is a CRUD matrix. It shows associations between object types and processes [16] and helps to check their consistency.
4. Usage of the systems that evaluate models created by the CASE tools or assist user of a CASE tool.
5. Usage of the modeling notations and systems that use one type of model in order to specify multiple aspects of a system.

Drawback of the approaches 2 and 3 is that without a tool support, references in a model or new kinds of models may themselves have CC problems. CASE systems can have supporting tools that check the models or provide active assistance to its users (approach 4). They may transform the diagrams into some other form of representation in order to analyze them. For example, generated description logics statements [9] are analysed by using description logic query tool. Richters and Gogolla [7] describe the system which translates UML models to the statements of UML-based Specification Environment language. Models are then analyzed by simulating the case when model elements have instances. Agents based system *WayPointer* [17] monitors the use case model, which is created by some CASE tool, for completeness, consistency and correctness. It can point to the problems and offer recommendations.

Agarwal and Sinha [18] conclude that developers don't rate any of the UML diagrams as very high in terms of usability. It might be caused by the usage of the several model types which leads to the inconsistencies between various parts of system specification [10], [19]. Acknowledgement of the "model multiplicity problem" is not new. A single model-based approach is superior to the multiple model approaches for late requirements engineering through implementation according to Paige and Ostroff [20]. Already Jäderlund [11] describes a methodology for the holistic system development that uses so-called system matrices in order to describe the system. System matrix (SMX) incorporates multiple views of the system. It provides methods for checking correctness, completeness, and consistency (CCC check) of the system.

More recently model multiplicity problem has been addressed by introducing Object-Process Methodology (OPM) [10], [19], which is a holistic system modeling, development and evolution approach. OPM uses Object-Process Diagrams (OP diagrams) for the graphic specification and Object-Process Language (OPL) for the textual specification of the system. OPM uses one integrated type of model in order to describe structural, functional and behavioural aspects of the system [10]. CASE tool Object-Process Case Tool (OPCAT) that supports OPM has been developed [19].

Yet another example of the modeling languages that corresponds to the single-model principle is *Eiffel* [20]. Delen et al. [8] propose system *Modelmosaic* that allows

to create different types of models. It records models and relationships between the elements of different types of models in a single integrated information base. These relationships, that are recorded as business rules, allow to generate new models from the existing ones.

2. Description of the Modeling Language

A model is created by using some language. Specification of the semi-formal language should contain descriptions of the abstract syntax, well-formedness rules and semantics [21]. For example, metamodel that describes the abstract syntax of UML is presented as a set of class diagrams [3], [4]. Well-formedness rules of UML are expressed using OCL constraints and its semantics are described using free-form text. In this section we present the metamodel of the language that will be used for specifying information systems in our proposed system. Diagrams that present fragments of the metamodel are accompanied with the free-form textual descriptions that explain some of the underlying concepts. Structure of the database for recording specifications of the information system will be derived from this metamodel. In the section 3 we present queries that help to check the well-formedness of the recorded models.

Interested parties can participate in the information system development projects in the different roles (see Figure 1).

An information system (IS) is described using three types of subsystems according to the methodological framework for the Enterprise Information System (EIS) strategic analysis [1]. These types are: *areas of competence*, *functional subsystems* and data centric subsystems that are also called *registers* (see Figure 2). A functional subsystem corresponds to one or more business processes [1]. "A register is a logical data-centric view of a business object that holds the state and transactions data of the object and provides related recording and query services." [1] *Administrative subsystems* help to perform administrative tasks of the organizations. Examples are subsystems for the management of data about the workers and documents. These kinds of subsystems are part of many different information systems. *Business subsystems* help to perform specific business tasks of the organizations. These tasks are reason why this organization is founded in the first place. For example, university IS has subsystems for the management of data about the students, curriculums and study results.

Functional subsystems use the services of one or more registers by reading and modifying data in them. Subjects who have some role in an IS use the services of one or more functional subsystems that belong to the area of competence of their role [1].

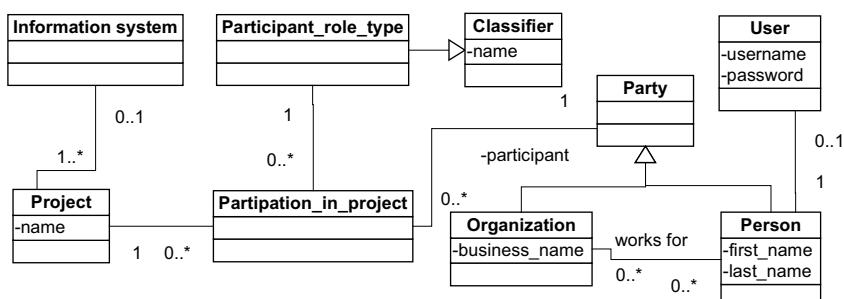


Figure 1. Metamodel of the projects and participants

We provide possibility to specify non-functional requirements of the system (see Figure 2) by using structure described in *Volere Requirements Model* [22].

Functional requirements of the information system can be specified as use cases. Corresponding fragment of the metamodel (see Figure 4) is created based on the guidelines of Larman [5] and Cockburn [23]. Each use case belongs to some functional subsystem (see Figure 4).

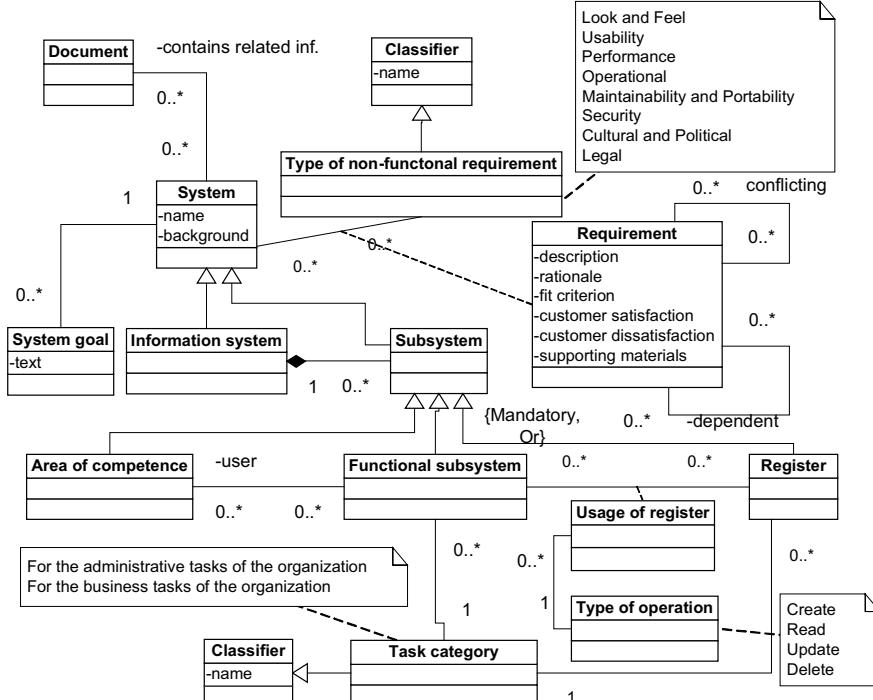


Figure 2. Metamodel of the subsystems

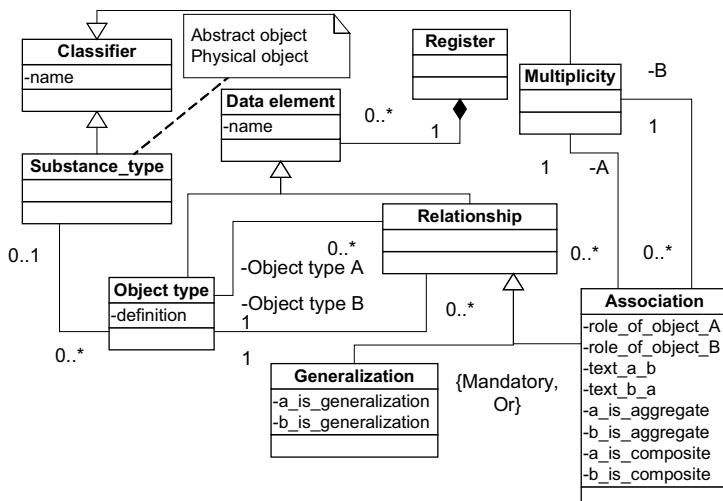


Figure 3. Metamodel of the data elements

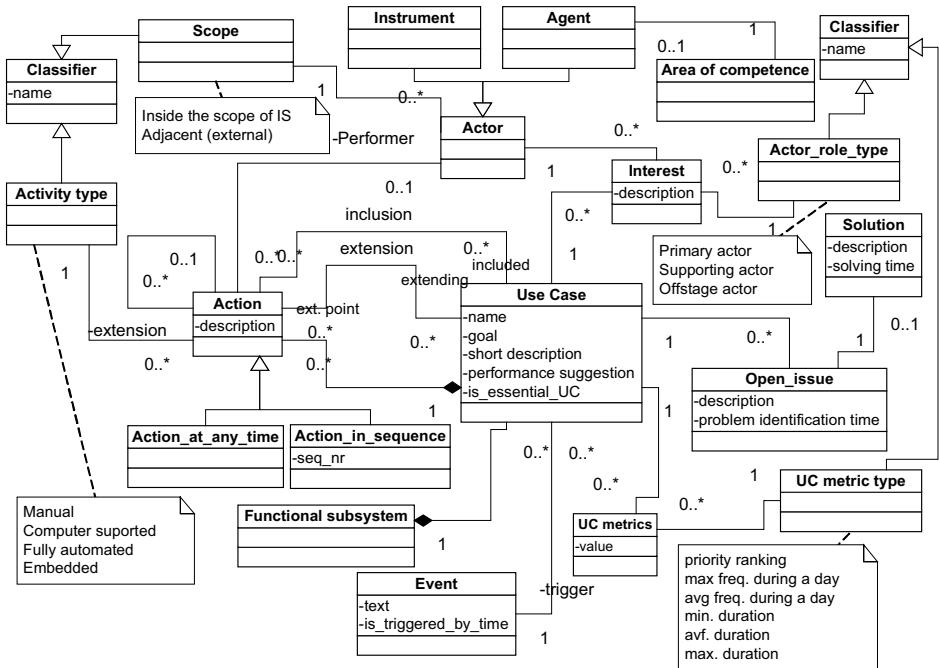


Figure 4. Metamodel of the use cases

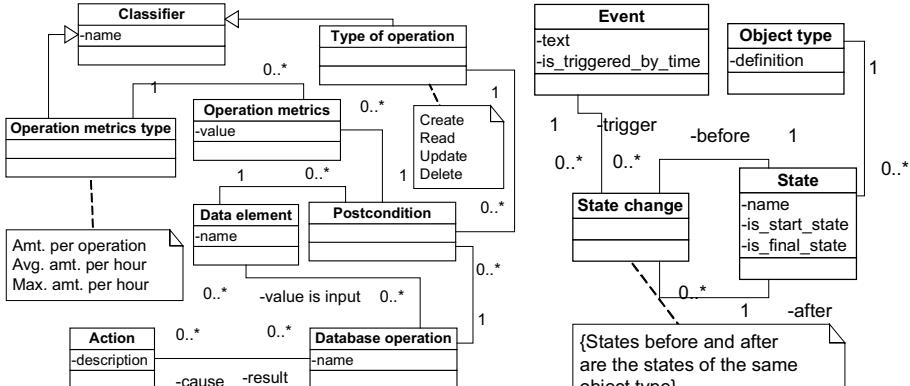


Figure 5. Metamodel of the database operations

Figure 6. Metamodel of the state changes

Each use case describes scenarios that consist of actions. Most of actions are performed sequentially. Some actions can be performed at any point of the scenario. Use cases can be related by using either extension or inclusion relationships.

Actions may be performed by the actors who are either agents or instruments. Agent corresponds to the area of competence. Registers that are used by a functional subsystem are specified in terms of the data elements (see Figure 3) and contracts of the database operations (see Figure 5). Action that is part of the use case can cause execution of the database operation. Result of the operation is described in terms of the post-conditions. Use case is triggered by an event (see Figure 4). Events cause change of the state of the objects (see Figure 6).

3. Consistency and Completeness (CC) Checks

Structure of the database determines the elements that can be associated. It also enforces the constraints of the metamodel according to which participation and/or cardinality is 1 at some relationship end. General principle of our system is that consistency and completeness of the models will not be ensured by the database constraints. It gives more freedom to the modeler. Occurrences of each problem will be found by using a query or a set of queries. Next we present *some* CC checks that the system must be able to perform. A query must find the cases where the CC rule is *not fulfilled*. Another possibility is to create a query that simply returns the value *false* if a model contains a problem. It must be possible to use these queries at any moment.

1. An IS consists of *at least one* area of competence (AC), functional subsystem (FS) and registry subsystems (RS).
2. An AC has exactly one corresponding actor.
3. An actor (and therefore an AC) uses services of *at least one* FS.
4. Services of a FS are used by *at least one* actor.
5. An IS (through some of its FS) is used by *at least one* non-adjacent actor.
6. A FS uses services of *at least one* RS.
7. A RS has *at least one* FS that reads its data.
8. A RS has at least one FS that adds new data to it.
9. An IS has *at least one* administrative FS and RS and business FS and RS.
10. An IS (as a whole or through some of its FS) has *at least one* non-functional requirement from each of the different requirements types.
11. Functional requirements to a FS are described by using *at least one* use case.
12. A use case describes process what has between *n* and *m* actions (steps). Numbers *n* and *m* should be dynamically changeable.
13. A use case is associated with the description about the interest of a primary actor in the context of the use case.
14. A use case has ideally *zero* unsolved open issues.
15. A use case that is not an essential use case is associated with *at least one* database operation through some action.
16. A data element is created and read by *at least one* database operation (association through a post-condition).
17. A database operation has at least one post-condition.
18. A database operation is associated with *at least one* action that is part of the non-essential use case. Each database operation is used by *at least one* use case.
19. Associations between FSs and RSs are consistent with the association of the use cases (that belong to some FS) and data objects (that belong to some RS).
20. It is possible to get from the start state of an object to any other state of an object by using state transitions.
21. An object type has *exactly one* start state and *one or more* end-states.
22. Relationships have mark about aggregation/composition/generalization *at most at one* end. In this case other end has no such marks.

Queries can also be used in order to find suspicious parts of the specification that may or may not be the mistakes. For example, query can search registers and data elements that are not subject of the update or delete operations. Queries help to collect metrics about the IS (like proposed by [24]). For example, we can count amount of use cases in different FS-s. If some FS has a big amount of use cases compared to others then it shows that the FSs are unbalanced and some big FS needs to be further divided.

4. Discussion and Comparisons

Commercial tools Modelmosaic [8] and EA WebModeler record models in a *database*. EA WebModeler [25] provides *form-based web interface* for creating models. Our system is not unique in this regard. UML is widely used notation and therefore it is reasonable to teach it even after we start to use this system. For the teaching and presentation purposes it is sometimes useful to see system specifications in the form of UML diagrams. Possible solution is the program feature which generates XMI files that contain models, based on the data in the database. These files can be opened using a CASE tool. Similar feature exists in EA WebModeler [25].

A modelling language that follows the principle of the single model must satisfy following three criteria: conceptual integrity, consistency of views, wide spectrum applicability [20]. Our proposed solution satisfies "*conceptual integrity*" criteria because models are recorded in one logical database. Each model element is recorded only once. Our proposed solution satisfies "*consistency of views*" criteria because checking of the consistency of different views of a model is automated. Our proposed solution partly satisfies "*wide-spectrum applicability*" criteria. This system is used in order to perform strategic and detailed analysis of the system but not design or implementation. In the next iterations system could be extended in order to allow generation of the stored procedures based on the database operations and table specifications based on the data-elements.

Why can't we use existing *free* software in order to model systems by using single model type? Examples of such systems are SystemSpecifier [26] that allows to create system matrices and OPCAT [19] that allows to create OPM models. First reason is that they don't fully support the methodological framework for the Enterprise Information System (EIS) strategic analysis. They don't allow to specify different types of subsystems and their interconnections. In addition, proposed system is different from OPCAT and SystemSpecifier because models are recorded in a database but not *directly* in the files. It helps to avoid well known problems of the file based systems like separation-, isolation- and duplication of data. More than one modeler can work with the same model at a time. Problem of OPCAT is that it doesn't provide explicitly CCC check functionality. Our proposed system allows to use queries from the database in order to find consistency and completeness problems. Usage of the integrated model prevents repeating recording of the same information and thus helps to avoid inconsistencies. Queries can be used in order to classify elements of a model. For example, it is possible to determine whether a use case is concrete, abstract, base or addition [5, p. 388] by making query about its relationships with the other use cases. Another example is that we don't have to separately record the events that influence the registers. Instead query can find the events that trigger the use cases which steps use the services of the register. Our proposed system can also be used in order to collect information about the work amount and performance of the modelers. It would also be a useful *e-learning tool*, because it is planned to be a web-based system.

5. Conclusions

In this article we have described principles of the system that helps to perform strategic and detailed analysis of the information systems. This system allows to record one integrated model of the system into a database by using form-based web interface. It

provides queries for finding consistency and completeness (CC) problems of the model. These queries can be used at any moment during the modeling process. We have created a partial prototype of the system for managing subsystems (see Figure 2). This prototype also allows to manage and execute queries (see Section 3).

If the system will be ready, then we can perform the usability study in order to evaluate which way users prefer to describe the system – using visual diagrams with little support to CC check or using textual descriptions with the extensive CC checks.

References

- [1] M. Roost, R. Kuusik, K. Rava, T. Veskiöja, Enterprise Information System Strategic Analysis and Development: Forming Information System Development Space in an Enterprise. Proceedings of the International Conference on Computational Intelligence, (2004) 215-219.
- [2] J. A. Zachman, A framework for information systems architecture, IBM Systems Journal, Vol. 26, issue 3, (1987) 276 – 292.
- [3] OMG Unified Modeling Language Specification. March 2003. Version 1.5. formal/03-03-01.
- [4] OMG Unified Modeling Language Specification. August 2005. Version 2.0. formal/05-07-04.
- [5] C. Larman, Applying UML and patterns: an introduction to object-oriented analysis and design and the Unified Process. Prentice Hall, Upper Saddle River, USA, 2002.
- [6] J. Eckstein, J. Bergin, K. Marquardt, M. L. Manns, H. Sharp, E. Wallingford, Patterns for Experimental Learning, Proceedings of EuroPLoP 2001, (2001) Retrieved 16.03.2006 from <http://www.pedagogicalpatterns.org/current/experientiallearning.pdf>
- [7] M. Richters, M. Gogolla, Validating UML Models and OCL Constraints. UML 2000 (2000), 265-277.
- [8] D. Delen, N. P. Dalal, P. C. Benjamin, Integrated modeling: the key to holistic understanding of the enterprise. Commun. ACM 48, Vol. 4, (2005), 107-112.
- [9] R. Van Der Straeten, T. Mens, J. Simmonds, V. Jonckers, Using Description Logic to Maintain Consistency between UML Models. UML 2003 (2003) 326-340.
- [10] D. Dori, Why Significant Change in UML is Unlikely, Communications of the ACM, Nov.2002, 82-85.
- [11] C. Jäderlund, Systematrix. Complete SMX handbook. Stockholm, 1981.
- [12] G. Engels and L. Groenewegen, Object-oriented modeling: A roadmap, Future Of Software Engineering 2000, 105–116.
- [13] G. McLeod, Beyond Use Cases, Proceedings of EMMSAD'00: 5th IFIP WG8.1 Int. Workshop on Evaluation of Modeling Methods in System Analysis and Design, Kista, Sweden.
- [14] A. M. Geoffrion, Computer-based modeling environments, European Journal on Operational Research 41 (1989), 33–43.
- [15] M. Glinz, A Lightweight Approach to Consistency of Scenarios and Class Models. Proceedings of the 4th International Conference on Requirements Engineering (ICRE'00) (2000), 49-58.
- [16] D. Brandon, Crud matrices for detailed object oriented design. J. Comput. Small Coll. 18, 2 (Dec. 2002), 306-322.
- [17] R. Racko, A Cool Tool Tool, Software Development Magazine, May 2004, Vol. 12, Part 5, 21-26.
- [18] R. Agarwal and A. P. Sinha, Object-oriented modeling with UML: a study of developers' perceptions, Commun. ACM, Vol. 46, No. 9, (2003), 248-256.
- [19] D. Dori, I. Reinhartz-Berger and A. Sturm, OPCAT - A Bimodal CASE Tool for Object- Process Based System Development, Proceedings of International Conference on Enterprise Information Systems, (2003), 286-291.
- [20] R. Paige, J. Ostroff: The Single Model Principle. Fifth IEEE International Symposium on Requirements Engineering (RE'01), (2001), 292-293.
- [21] J. Greenfield, K. Short, Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. John Wiley & Sons, USA, 2000.
- [22] S. Robertson, J. Robertson, Mastering the requirements process. Addison-Wesley, USA, 1999.
- [23] A. Cockburn, Basic Use Case Template, Versaion 2, October 26, 1998, Retrieved 11.03.2006 from <http://alistair.cockburn.us/usecases/uctempla.doc>
- [24] H. Kim, C. Boldyreff, Developing software metrics applicable to UML Models, Proceedings of the 6th International Workshop on Quantitative Approaches in Object-Oriented Software Engineering, (2002).
- [25] Agilense Enterprise Architecture Frameworks. Retrieved 12.03.2006 from http://www.agilense.com/documents/agilense_frameworks.doc
- [26] Systematik holistik metodik. Retrieved 05.03.2006 from <http://www.systematik.se/>

Defect rate profile in large software-systems

Elena-Ramona MODROIU and Ina SCHIEFERDECKER

FOKUS Fraunhofer Institute, Berlin, Germany

Abstract. Software reliability, an issue raised long time ago by software engineering researchers, is still a problem that must be faced nowadays. Our approach is to handle reliability of software products making use of a statistical distribution based model and historical data of similar projects. We analyze the software error metrics of large software projects developed in a multinational company in the last five years, searching for a Rayleigh pattern in the defect rate profile. If we can prove empirically such a pattern exists in certain environment, we have a rapid and powerful theoretical model to support project estimation decisions early in the development life cycle when historical data on similar projects is available.

Keywords. Project estimation, software reliability, error rate profile, knowledge acquisition, historical data

Introduction

Quality of a software product encompasses different aspects like stability, portability, maintainability, usability among which reliability plays an important role. Software reliability can be thought of as the probability that a software to function without failure for a specified period of time, in a specified environment [6]. To evaluate reliability, there are employed basic metrics over defect occurrences in time such as tracking defect counts in a cumulative manner. We expect to find a Rayleigh pattern in the defect occurrences and thus to ensure a theoretical framework for modeling of defects during the software testing cycle. In the following sections we show that the results of our analysis confirm our expectation.

Earlier in 1982, by studying 10 error histories, Trachtenberg was among the first ones to point out there might be the Rayleigh curve underlying the defect rate behavior in software products [1]. Schick-Wolverton predicted Rayleigh-shaped failure rates proposing a software reliability model derived from the assumption that the error detectability is linearly increasing with time [3]. This last model is part of a set of classical reliability models classified as linear (Jelinski-Moranda, Shooman, Musa), geometrical (Moranda, Ramamoorthy-Bastani) and Rayleigh (Schick-Wolverton). Later in 1990, Trachtenberg proposed a general framework to unify all these classical reliability models and showed that also the general theory predicts Rayleigh-shaped failure rates assuming that workload is increasing linearly with time [5]. Also QSM company published a few white papers and articles on the in-house defect estimation approach as being based on the Rayleigh distribution function to forecast the discovery rate of defects as a function of time throughout the software development process [2].

One of our goals is to compare new software development to previous developments. To achieve this we propose to determine the characteristics governing our environment, analyzing the knowledge provided by the measurement programs integrated in the development process. We have the chance to confirm the existence of similar environments among organizations generalizing the results we got by making use also of the findings reported in the afore mentioned research works. In the same time we extend the observation to different types of testing corresponding to the main phases in the product building process. This study was inspired by our team research work in the domain of quality of tests.

The paper is organized as follows. Section 1 describes the projects under study and software metrics that were used to measure them. Section 2 gives a short mathematical overview of Rayleigh model and the algorithm used to non-linearly fit the raw data to the statistical distribution. The results of non linear fitting are presented in section 3. We discuss further the importance of the results we got in the context of software projects estimation and the restrictions in applying a Rayleigh model. Section 4 summarizes the conclusions and future work.

1. Experiment settings

In this paper the focus is on studying the defect rate throughout the development life cycle of large, heterogeneous, multi-release software projects in the world of modern communication networks. A similar case-study is reported in [4]. We have now the chance to confirm the results among different companies and different software development processes.

The projects under study are large. They consist of many modules, are developed around the world within a multinational company and the effort employed represents a few hundreds man-years. Each project is a subsystem of a more complex software product, a network platform for mobile communications. As the mobile communication market has continuously grown in the last 10 years, also the technology was improved and new features had to be implemented in the software product. This resulted in new releases of the product almost each 2 years. Each release was in the category of high complexity project class due to new innovative hardware or software technologies, new software tools to be used, new algorithms to be created or component interfaces to be defined. A basic measurement program - common to CMMI Level 2 and 3 organizations - was in place, so we had plenty of data on defects discovered along each testing phase in the validation and verification process.

Our software product is made up of four main subsystems and data from two consecutive releases in the last 5 years was available. This means we have under study a set of 8 projects.

We attempt to discover proper knowledge to support project estimation decisions from following categories of historical data: size metrics (effective lines of code ELOC not including empty lines or comments), defect metrics (error finding status EFS, defects per development phase DDP) and effort metrics.

2. Data Processing Method

The software developing process is based on the V-Model graphically represented in figure 1. V-Model defines what, how and when has to be done during software development. It became a standard for German federal administration and defense projects, but also many companies have chosen to use it. There are certain phases that make up the life cycle of a software product. First part of the V corresponds to the building of the system, while the second part consists of the main testing types, each test activity being mapped to a certain building phase. Entry and exit points of the execution phases are marked formally by definition of generic milestones as listed in table 1.

Milestone	Short Description
M1	Product Definition
M2	Completion of Analysis
M3	Completion of Design
M4	Completion of Implementation
M5	Completion of Unit Test
M6	Completion of Component Integration Test
M7	Completion of System Integration Test
M8	Completion of System Test
M9	Completion of Customer Acceptance Test

Table 1. Milestones definition.

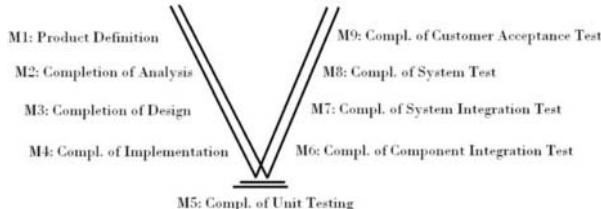


Figure 1. V-Model. Types of testing.

The error data, recorded per week in a cumulative manner during main testing activities - component integration test, system integration test, system test - will be analyzed against a particular case of Weibull distribution

$$W(m; k, \lambda) = 1 - e^{-(\frac{m}{\lambda})^k} \quad (1)$$

where $\lambda = \sqrt{2}\beta$ and $k = 2$ known as Rayleigh distribution function

$$R(x; \beta) = 1 - e^{-\frac{x^2}{2\beta^2}} \quad (2)$$

The fitting of the Rayleigh distribution to raw error data was done using Levenberg-Marquardt algorithm (LMA), present in almost any software that provides generic curve fitting tool. The LMA algorithm interpolates between Gauss-Newton algorithm (GNA) and the method of gradient descendant. It is an iterative procedure more robust than GNA, i.e. in many cases it finds a solution even if it starts very far off the final minimum.

3. Analysis of Results

In this section there are presented and analyzed the progress data collected independently by each test team. As the implementation is ready, first checks for code sanity are done by the code developers themselves. It is generally agreed that as late as a defect is found, more expensive will be to have it fixed. This phase is not going to be analyzed in this study, the focus being on defects discovered by specialized testing teams. Three main testing activities are well defined by verification and validation process and these are: component integration, system integration and system test. It follows then customer acceptance test, executed by clients, but this remains outside of the scope of our case-study.

The Rayleigh curve fitted to the defect data is

$$R(x) = a_0 \left(1 - e^{-\frac{x^2}{2a_1^2}}\right) \quad (3)$$

where a_0 stands for the total number of errors to be discovered and a_1 is the scale parameter of the curve and represents the time when Rayleigh probability density function reaches a maximum.

3.1. Component Integration Test

Component Integration Test is performed to integrate one subsystem of the larger project. As we are talking of very large subsystems, these were also made up of modules that have to be assembled and tested together to deliver a reliable component for the end product. The modules are already tested for stand alone functionality and correctness in the unit testing phase before to become available for component integration. This phase begins once milestone M5 is declared, and goes beyond M6 due to later change requests implementation and bug fixes of errors that did not represented a requirement criteria to M6 declaration.

Figure 2 presents the defect rate along component integration test for two projects: subsystem-1 and subsystem-2 of release A. The correlation coefficients between the fitted Rayleigh curve and the raw error data were 0.997 and 0.96. The results got for all projects in our study are listed in table 2.

Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.997
	Component 2	0.96
	Component 3	0.962
	Component 4	0.984
Release B	Component 1	0.994
	Component 2	0.998
	Component 3	0.983
	Component 4	0.984

Table 2. Component Integration Test

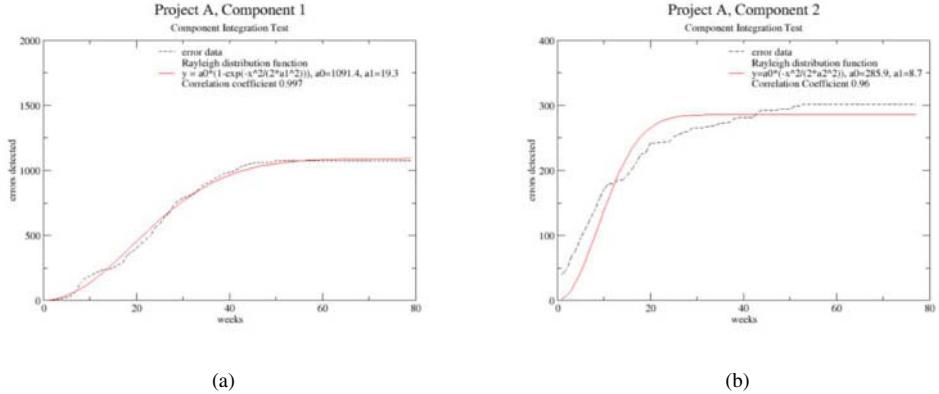


Figure 2. Progress Data: Software defects

3.2. System Integration Test

System Integration Test is performed to integrate all subsystems and get the final software product. Test cases are executed to check the interoperability of the subsystems, focusing on testing the interfaces in between. The phase begins once milestone M6 is declared and continues until product is delivered to customer acceptance.

Figure 3 presents the defect rate along system integration test for two projects: subsystem-1 and subsystem-2 of release A. The correlation coefficients between the fitted Rayleigh curve and the raw error data were 0.98 and 0.994. Table 3 summarizes the results obtained for all 8 projects.

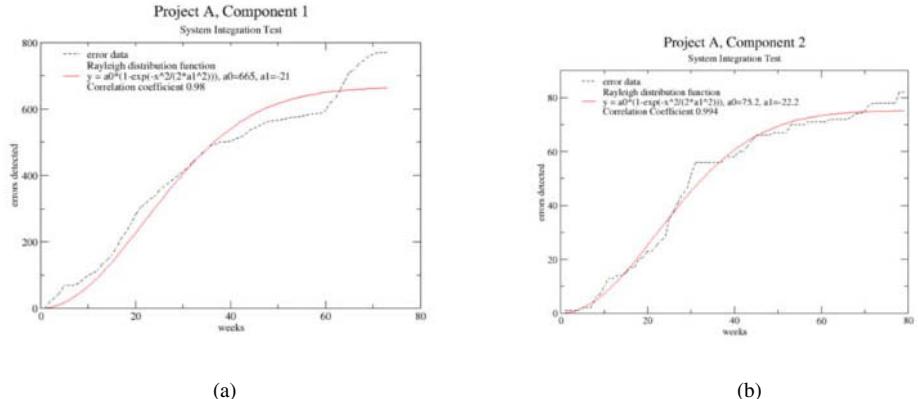


Figure 3. Progress Data: Software defects

With an exception of a correlation coefficient of 0.94 in the case of component 4 in the release A, all the others coefficients were very high.

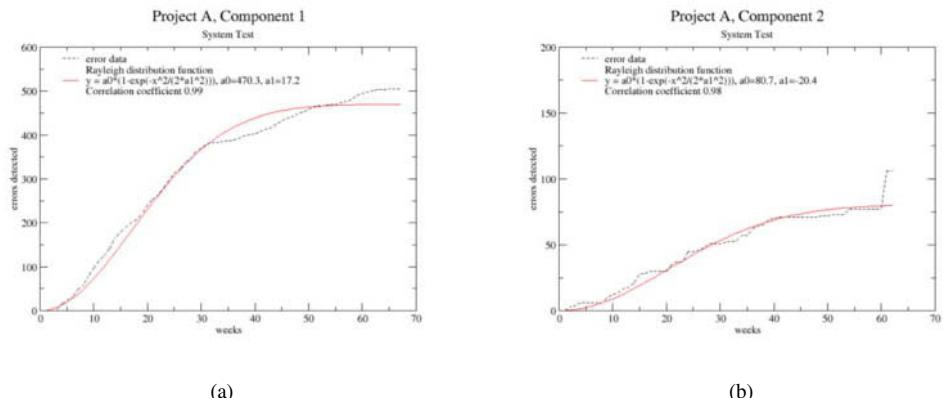
Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.98
	Component 2	0.994
	Component 3	0.994
	Component 4	0.945
Release B	Component 1	0.995
	Component 2	0.973
	Component 3	0.977
	Component 4	0.981

Table 3. System Integration Test

3.3. System Test

In System Test, test cases are planned and executed to check for conformance with the requirements of the entire software product. Also non-functional checks such as performance, volume, stress, robustness are among major activities now. System Test begins once milestone M7 is declared and continues until product is delivered for customer acceptance tests.

Figure 4 presents the defect rate along system test activities of release A, for subsystem-1 and subsystem-2. The correlation coefficients between the fitted Rayleigh curve and the raw error data were 0.98 and 0.994. Table 4 summarizes all the results. It can be noticed, the highest correlation coefficients to Rayleigh curve were obtained for System Test phase.

**Figure 4.** Progress Data: Software defects

3.4. Verification and Validation Phase

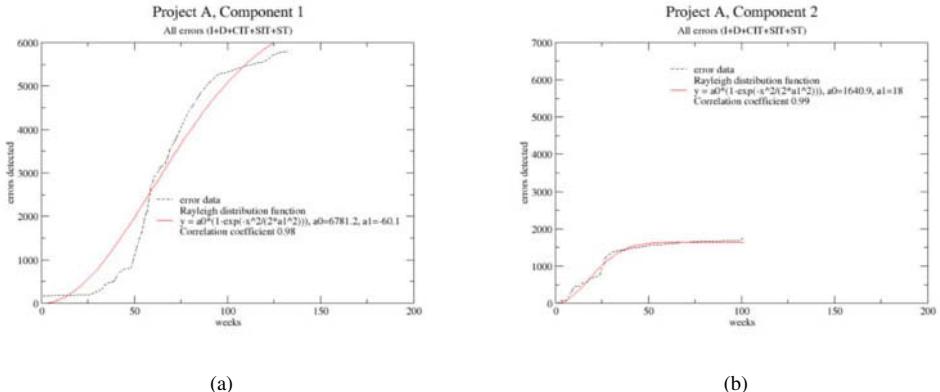
Verification and Validation corresponds to the second part of the software development process according to the V-Model and includes unit test, component integration test, system integration test, system test and customer acceptance test.

Figure 5 presents the defect rate along all testing phases for the same two projects, subsystem-1 and subsystem-2 of release A. The correlation coefficients between the fitted

Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.99
	Component 2	0.98
	Component 3	0.98
	Component 4	0.98
Release B	Component 1	0.995
	Component 2	0.991
	Component 3	0.985
	Component 4	0.995

Table 4. System Test

Rayleigh curve and the raw error data were 0.98 and 0.994. Table 5 lists the results got for all 8 projects selected for this study. One could notice very high correlation coefficients to Rayleigh curve, comparable to system test phase.

**Figure 5.** Progress Data: Software defects

Product	Project	Correlation coefficient to Rayleigh distribution
Release A	Component 1	0.98
	Component 2	0.99
	Component 3	0.98
	Component 4	0.99
Release B	Component 1	0.997
	Component 2	0.995
	Component 3	0.997
	Component 4	0.996

Table 5. Verification and Validation Phase

3.5. Decision Support in Project Estimation

This case study confirms there is a pattern in the defect rate profile and there is one governed by Rayleigh distribution function. The result was found for very large, heteroge-

neous software projects, as reported also in [4], and within an organization which has adopted V-Model as a standard development process.

Once proved the defect rate shows a certain distribution for projects of a company, this becomes a simple but powerful instrument for similar project estimations. It supports manager's decisions as when ready for product release ensuring a minimum required mean time to defect, what effort to pair up to testing to guarantee required quality in a given time interval. One way to achieve this target is by collecting for each project data sets as size, effort and β , where β is the Rayleigh distribution parameter computed for the project given the error progress reports. When a new project starts, assuming size and effort are estimated we can get from table the corresponding value for β and thus being able to plot an estimated Rayleigh-shaped error rate. Later on when data from the current project becomes available it can be compared to the estimated Rayleigh curve, having the possibility to identify deviations and analyze their cause, getting the project back on track. Also having such a table allows a manager what-if analysis. If the project size is about 2000 KLOC and an effort of 300 man-month is available, then using Rayleigh curve with the corresponding β from table it might be found that the project could be delivered in 2 years time. But what if the available effort is not only 300 man-month, but 400 man-month? What-if scenarios could be used by project managers if they had available data collected from previous similar projects. But this requires a big history of projects. It would be very useful if we could find a relationship between size, effort and then correlate it with β being able to interpolate the whole range of values size- effort- β . This is a future task do be done, when detailed effort data becomes available.

Similarity of two projects is decided by a series of attributes that could be classified into a few groups: domain of functionality, development environment and project complexity. We consider projects to belong to different domains of functionality such as banking applications, game applications, network communication platforms etc. Development environments are defined by the development model adopted within the organization, team experience and available software resources, while project complexity is given by the degree of innovation, size, multi-site need of development and duration of a project. The list of above mentioned attributes to characterize the similarity of two projects is not an exhaustive one, depending on the degree of similarity one would like to achieve, more detailed attributes could be added.

The Rayleigh model can be applied with two restrictions. The first one refers to the fact that defect rate during development phase must be positively correlated with defect rate in the field. The second one is about the errors' injection as being constant, and thus more defects discovered earlier means fewer defects are discovered later. Both this assumptions are related to the concept of "doing right the first time" [7].

4. Conclusions

The projects in this study presented a defect rate governed by a Rayleigh curve. This has been found valid for all testing activities seen as one single test phase and also for single testing types as component integration, system integration and system test. Higher correlation coefficients were obtained for system test and verification-validation scenario. This is in agreement with previous studies [2,3,4] which reported the same underlying pattern for all errors or for errors found by system test. Beside this, it has been shown

that also the other test types are characterized by a Rayleigh defect rate distribution. One could look at the verification and validation process as a sequence in time of 3 Rayleigh curves - having as starting points milestones M5, M6, respectively M7 - and in the same time the verification and validation process overall is represented by a Rayleigh curve.

It has been found that the Rayleigh curve is appropriate in modeling our environment of heterogeneous, complex projects developed in a multinational organization with cost of a few hundreds man-years. Other studies reported in the '80 the same model being appropriate for software development environments. Others reported the same findings more recently. This means a Rayleigh estimator can be applied among different organizations.

As a next step we would like to build a Rayleigh forecaster, a composite model built with data from same class of projects, and measure its performance when applied to new projects in the same class.

Certainly there are limitations in implementing such a forecaster, due to availability of data. For CMMI 2 or higher level organizations where measurements programs are implemented, this problem disappears. We see now that establishing a measurement program integrated in the development process helps any organization achieve an in-depth understanding of its software development environment, ensuring a solid background for further improvements in project planning and estimation.

References

- [1] M. Trachtenberg, Discovering how to ensure software reliability, *RCA Engineer*, Jan.-Feb. 1982, 53-57.
- [2] Lawrence H. Putnam and Ware Myres, Familiar Metric Management - Reliability, www.qsm.com.
- [3] George J. Schick and Ray W. Wolverton, An analysis of Competing Software Reliability Models, *IEEE Trans. Software Eng. (1978)*, vol 4, no 2, 104-120.
- [4] J.W.E. Greene, Ensuring Delivery of Highly Reliable Complex Software Releases, *Quantitative Software Management LTD*, www.qsm.com 2003.
- [5] M. Trachtenberg, A General Theory of Software-Reliability Modeling, *IEEE Transactions on Reliability*, volume 39, number 1, April 1990.
- [6] Dick B. Simmons and Newton C. Ellis and Hiroko Fujihara and Way Kuo, Software Measurement: A visualization Toolkit For Project Control and Process Improvement, *Prentice Hall PTR*, 1997.
- [7] M. Neil and P. Krause and N. Fenton, Software Quality Prediction Using Bayesian Networks.

Information Flow Diagram and Analysis Method for Unexpected Obstacle Specification of Embedded Software

Hidehiro KAMETANI ^a, Yasufumi SHINYASHIKI ^{b,a}, Toshiro MISE ^{b,a},
Masa-aki HASHIMOTO ^{a,1}, Naoyasu UBAYASHI ^a, Keiichi KATAMINE ^a
and Takako NAKATANI ^c

^a *Kyushu Institute of Technology, Japan*

^b *Matsushita Electronic Works, Ltd., Japan*

^c *University of Tsukuba, Japan*

Abstract. This paper proposes an IFD (Information Flow Diagram) and an analysis method using the IFD for finding unexpected obstacles during the specification process of embedded software. Recently, embedded software has become increasingly larger in scale and more complicated, while time available for software development has become shorter. Both these industry trends affect the quality of the software. To improve the quality, it is necessary to find unexpected obstacles early in the specification process. This paper also describes a case study and discusses IFD and the analysis method.

Keywords. embedded software, specification, unexpected obstacles, information flow diagram, analysis method

1. Introduction

Embedded software has become increasingly larger in scale and more complicated, while the software development cycle has shortened. Both these industry trends affect the quality of the software. Various users include embedded systems in their specific environments[1,2], and the operations of the systems must be secure and usable over long periods. For this reason, more than 70% of the software is assigned to functions for handling exceptions such as communication errors, signal noise, operator error and device failures. However, software specifications sometimes omit handling such unexpected problems because the knowledge of hardware and operational environment of systems necessary for the specification is not available[3,4,5]. Omitting such exception handling specifications can cause serious problems, such as human accidents caused by poor quality software and delays due to redesign in the software development process.

We can expect an improvement in the quality and productivity of embedded software development if we precisely and effectively specify the software for handling unexpected exceptions. The exceptions sometimes missed in the software specification pro-

¹680-4 Kawazu, Iizuka, 820-8502 Japan; E-mail: hasimoto@ai.kyutech.ac.jp

cess are called “unexpected obstacles”. In this paper, we define “expected specifications” and “unexpected obstacle specifications” as follows: The former specifies the system behavior usually described in the software operation manual, and explicitly defined at the start of the architectural design. The latter concerns any deviations from the behavior determined by the former. The deviations are as follows: fading and failure of system hardware, wrong operation and overload caused by system users, temperature and radio noise in the natural environment.

We studied a specification method[6] for finding unexpected obstacles by formalizing one of the methods that is tacitly used by experts of embedded software development. In this method, we assume that the exceptions are caused by the abnormal flow of information in embedded systems and their operational environments. Therefore, we propose an *IFD (Information Flow Diagram)* composed of device and process diagrams, and an analysis method using the IFD that applies three failure analysis techniques: *FMEA (Failure Modes and Effect Analysis)*[7], *FTA (Fault Tree Analysis)*[8] and *Guide Words*[9] in an integrated manner. In this research area, misuse cases, abuse frames and goal-oriented obstacle handling have been already studied. However, IFD combined with the FMEA, FTA and Guide Words has not yet been studied. Although Failure Scenario Matrix uses FMEA, FTA and Guide Words, it uses a state transition table instead of information flow diagram. In this paper, Section 2 describes embedded systems models. Section 3 and 4 describe IFD and the analysis method respectively. Section 5 explains a case study. Section 6 discusses them.

2. Modeling of the embedded system

This section describes general ideas of modeling of an embedded system.

2.1. Caching of an embedded system

In a business processing system, system behavior is determined by software built into the limited kinds of hardware such as CPU, memory and hard disk. On the other hand, in an embedded system there is a variety of hardware used in different products. Therefore it becomes important to pay attention to each device, and to model the operation environment for an embedded system.

An embedded system works by receiving a signal from a user in the operation environment. In other words users and operation environment elements become actors of the system. The signals sent by actors go to each device, and are recognized and handled by the embedded software in the microcomputer. Once activated by the signal, the system can perform its intended function.

In a normal situation an actor gives a command to a system. The system makes the output by interpreting the command precisely and processes to satisfy it. On the other hand, where there is malfunction, the system misinterprets the required behavior of the command. In addition, it is malfunction of a system to perform actions for a signal from a wrong actor. From the system point of view of this process, the system receives information, interprets it, handles it and outputs information again. Unfortunately, correct information changes into wrong information in some situations. In other words, communicating wrong information is a cause of malfunction in the system or operational environment.

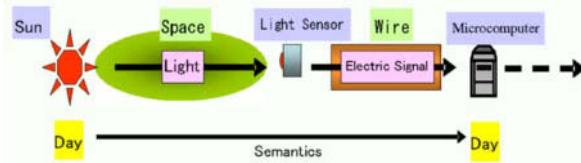


Figure 1. General ideas of information communication.

In other situations, it is possible to cause a malfunction of the system by creating obstacles resulting from communication of false information. Furthermore, other devices can receive the false information, and then interpret, handle and output it. This scenario is called “failure scenario”.

We consider it is important to pay attention to information flows communicated in a system and operational environment during unexpected obstacles analysis. Therefore, information communicated to each device in a system in an operational environment can be visualized by the flows shown in a diagram. In addition, analysis of information flows on the diagram enables analysis of unexpected obstacles by extracting “failure scenarios”.

2.2. General ideas of information communication

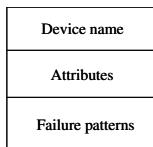
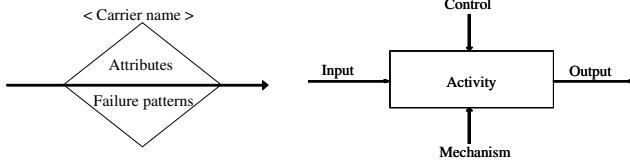
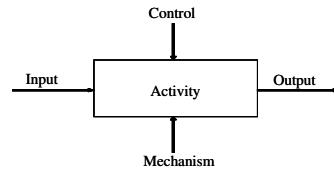
This section describes general ideas of communication in an embedded system. We understand that an embedded system works by communicating information. When you look closely at connections between certain electric devices, they work by communicating electrical signals. Actually the electrical signals carry not only the physical parameters such as voltage or frequency but also information for a system to do make decisions. For example, the signals include information such as the values of sensing parameters and the signal of motor ON. In other words, electrical signals carry information between devices and mediums of physics such as light or temperature and between the operational environment and the embedded systems. The signals act as “carrier” and convey the information necessary for the system to know what to do. Figure 1 shows the general concepts of information communication.

3. Information Flow Diagram

We introduce the concept of IFD. An IFD is comprised of a device diagram and process diagram. The former is a diagram for describing devices and their connections. The latter is a diagram for modeling processes executed on the devices that appear in the device diagram.

3.1. Device diagram

A device diagram represents a modeling domain that illustrates devices in embedded systems and objects such as users in the operational environment. Each device is an entity represented by a name, attributes, and failure patterns. A failure pattern shows categories of problems including noise, disconnection, and deterioration. An attribute describes the property of a target device. Figure 2 shows the notation of a device in the diagram.

**Figure 2.** Device.**Figure 3.** Carrier.**Figure 4.** IDEF0.

Devices physically connected to each other are shown by using arrows to indicate the direction of information flows. The carrier of an information flow is described above the corresponding arrow. A carrier, denoted by a diamond, comprises the name, attributes, and failure patterns. Figure 3 shows the notation of a carrier.

3.2. Process diagram

In IFD, the functions of an embedded system are modeled using a process diagram whose notation is based on *IDEF0* (*Integrated Definition methods*), an activity modeling technique. IDEF0 is a method for describing static models that represent functional activities such as business processes. In IDEF0, a process can be described by a series of activities connected to each other by input, output, control and mechanism arrows. An activity receives input information, does some work by using facilities of specified mechanisms, and generates output information. The control applied to an activity shows conditions for executing its process. Figure 4 shows the IDEF0 process notation.

A process diagram in IFD is described as processes executed on devices. Each of the processes corresponds to an activity in IDEF0. Information flow in IFD is basically represented using input and output arrows. Some output arrows are linked to control arrows if the output information is applied as the control information. A process has a mechanism arrow from the device on which the process is executed.

A process diagram can be refined stepwise. First, a top-level process diagram is described, then, each process in the diagram is refined as necessary. Figure 5 illustrates an example of an IFD. The upper part is a process diagram, and the lower part a device diagram. This diagram shows requirements that describe the normal behavior of the electric pot.

4. An analysis method for finding unexpected obstacles

In this section, we introduce an IFD-based analysis method for finding unexpected obstacles. This method integrates IFD with three failure analysis techniques including FMEA, FTA, and Guide Words in HAZOP (the Hazard and Operability studies).

4.1. An application of FMEA

FMEA is an easy to use and yet powerful proactive engineering quality method that helps you to identify and counter weak points in the early conception phase of products and processes. The structured approach makes it easy to use and even for the non-specialist is a valuable tool. First, we select a device in a device diagram, then pick up the failure pattern of the device, and describe information flow caused by the failure pattern. After

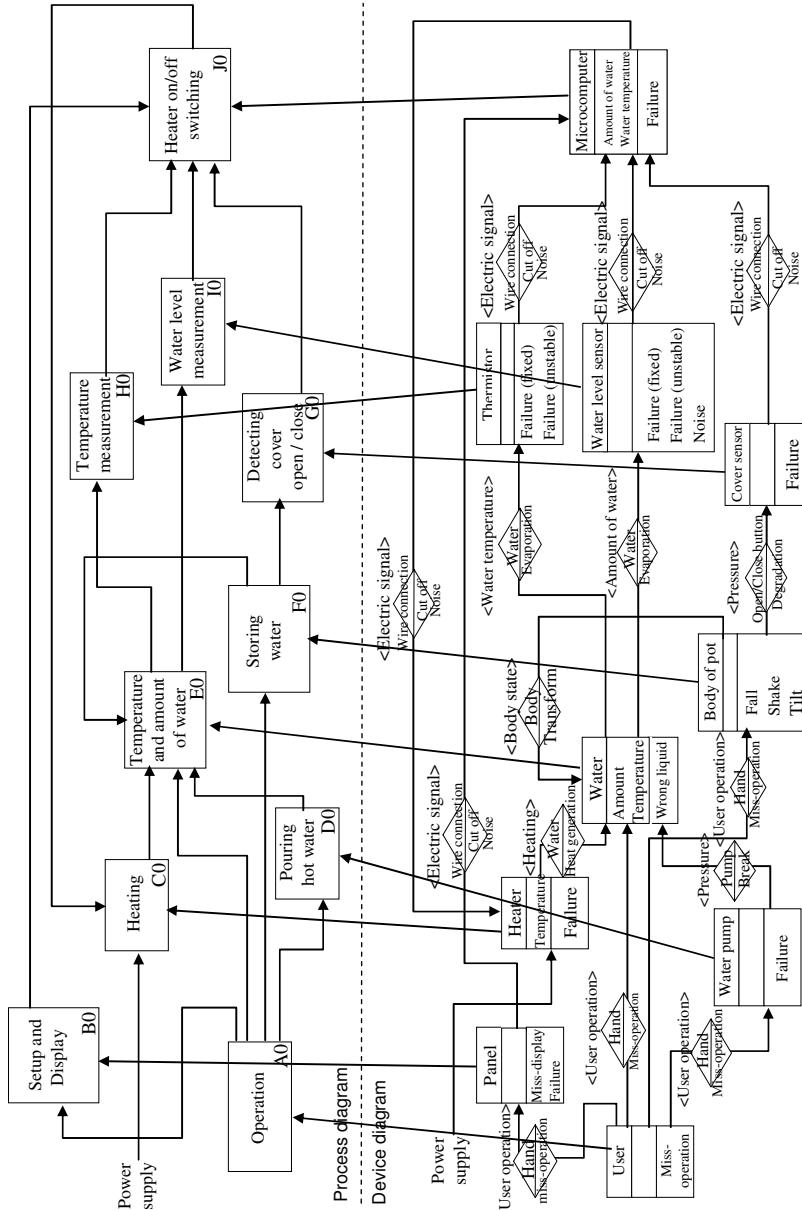


Figure 5. Example of IFD.

that, we analyze what effect an obstacle in the pattern has on the system. There is no problem if the obstacle is correctly dealt with by software or hardware, but the IFD must include an information flow path from the obstacle to the final failure. This flow path indicates a failure scenario caused by the device failure. As shown below, by integrating FMEA and IFD, we can identify failure scenarios.

Table 1. Failure list.

Quality Element	Failure	Carrier
Safety	Fire	Heat
	Burn injury	Heat

...

Table 2. Example of guide words.

Meaning of Information	Value shifted greatly / Value shifted small Value contradicted from the situation
Amount of Information	Too much / Too little Too long / Too short
Timing of Information	Too early / Too slow Synchronous / Asynchronous Too long / Too short
Composition of Information	The order is different The composition is different
Subject of Information	Not a true subject
State of Information	Lost, Irregular, Fixed, Unstable, Oscillated

4.2. An application of FTA

FTA is a method that consists of an analysis of possible causes starting at a system level and working down through the system, sub-system, equipment and component, identifying all possible causes. To integrate FTA with IFD, we introduce a new kind of table called a *failure list*. The list is composed of three constructs: quality items, failures disturbing the quality items, and carriers that cause the failures. The quality items are usually specified as non-functional requirements of the system. Table 1 shows an example of a failure list.

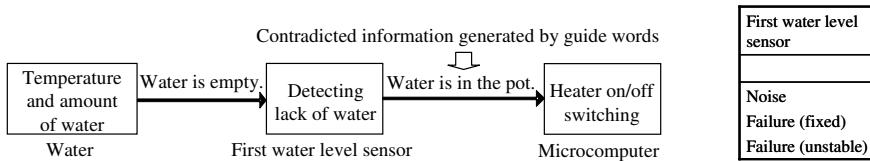
Applying FTA, we can find an unexpected scenario that causes a failure. First, we select a carrier that causes a failure in the failure list. A path of the carrier is found from the IFD. The situation that causes the failure on the pass is analyzed then described in IFD. We repeat for all causes gathered from the failure patterns of devices, and information flow from a cause to a failure become a failure scenario.

4.3. An application of Guide Words in HAZOP

Originally, HAZOP was designed as a technique for analyzing dangerous factors of chemical plants. Here we apply guide words to the analysis of unexpected obstacles in embedded systems. Although guide words are applied to the flow of material in chemical plants, we use guide words for information flow of expected obstacles in IFD. Table 2 shows an example of guide words. A guide word shows only a possibility of an obstacle, but they are effective for extracting obstacle scenarios because they indicate reasons for deviations and obstacles caused by the deviations.

4.4. A systematic analysis method

A systematic analysis method is described with IFD. Initially we make an IFD of expected obstacles and apply a guide word for a normal information flow in IFD. We assume that the normal information is deviated according to a guide word and analyze what kind of effect the deviation information has on the system. There are two cases where the deviation information is handled normally and finally reaches failures. In the case of deviation for reaching failures, we analyze causes of the deviating. If a failure pattern occurs during a breakdown, there is the possibility that the failures will occur. On the other hand, if there is no failure pattern, a process does not have to be added for the deviation because the deviation from the guide word does not occur. The path from the obstacles

**Figure 6.** Application of guide word.**Figure 7.** First water level sensor.

in the failure pattern to the failures, expresses a scenario to follow an information flow along the path to reach an obstacle. This failure scenario becomes the result of analysis.

In addition, because it is often hard to identify unexpected obstacles, they are easy to overlook at the time of the FMEA, and causes of failures easily overlooked at the time of FTA. FTA is different from FMEA in a direction of analysis, but an information flow to reach a failure from an obstacle in a failure scenario of an analysis result does not change. In other words FTA is connected to FMEA at a point within an analysis process. FMEA and FTA are not applied alone but applied with guide words. FMEA and FTA with guide words catch the situation easily overlooked by the individual application. The unexpected obstacles that were hard to be discovered can be extracted.

5. A case study

This section describes a case study with the analysis method of unexpected obstacles.

5.1. An application example

SESSAME (the Society of Embedded Software Skill Acquisition for Managers and Engineers) provides example requirement specifications for an electric pot as a kind of embedded system. Electric pots are a type of consumer goods, and consumer goods should be designed with quality characteristic such as high reliability, safety and fault tolerance. Thus, we selected the requirement specifications as a target to reveal the undefined requirements of the pot for satisfying the quality characteristics above. This section describes the case study conducted on an electric pot by using the analysis method.

5.2. Case analysis

When the pot is boiling water and water disappears by evaporation, the pot works to stop a heater. In an implementation of this function, the pot stops the heater when the pot is empty and the first water level sensor does not detect water. Using the analysis method of unexpected obstacles, the implementation method is analyzed to evaluate its suitability. At first, for output information of the first water level sensor, the guide word “contradicted” is applied. When a pot is empty, each water level sensor outputs the information “does not have water”, and assumes the output information “water is in the pot” is “contradicted” using the guide word application. Figure 6 illustrates an application of the guide word “contradicted” with IFD.

Next we analyze the kind of obstacles that occur because of this situation. The microcomputer judges that there is water in the pot based on the input information. The

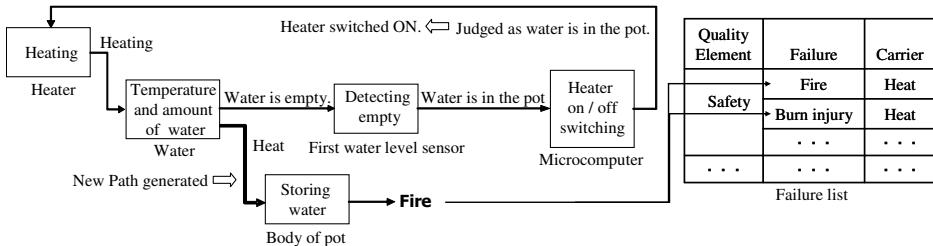


Figure 8. Information flow leading to fire.

system continues boiling. However, the heater gets abnormally hot because there is no water in the pot, heat reaches the body of pot and the pot catches fire. Figure 8 illustrates an information flow of the situation with IFD.

We analyze whether there is the case where the first water level sensor sends the information that it is “contradicted”, like the guide word. Figure 7 illustrates the first water level sensor in IFD. An obstacle meaning that the output value of the sensor is fixed by failure regardless of the input value is specified as a failure pattern “failure (fixed)” in device diagram. Therefore, the sensor can send the information “water is in the pot” to the microcomputer even if the sensors detected no water. This means that there is a failure pattern corresponding to the guide word “contradicted” in the first water level sensor.

From the above, the following failure scenario is provided: When water of a pot is emptied during boiling, the system must turn off the power supply to the heater. However, when the obstacle occurs in the sensor, the system judges the existence of water in the pot and continues heating until the pot catches fire. From this failure scenario, we reach a conclusion that judging the presence of water in a pot from only a water level sensor is insufficient. For example, a method that judges the presence of water using the temperature raising ratio besides the water level sensor is better.

5.3. A combination of failure patterns

A combination of failure patterns of devices is not considered in FMEA. However, an analysis of the influence of combinations of failure patterns in the system is necessary to identify unexpected obstacles for high quality system. Therefore, we analyze the influence of combinations of device failure patterns on IFD. However, a number of combinations explodes when we consider all combinations of failure patterns on the IFD, so we need an effective method to combine failure patterns.

Each device processes its input information for outputs. In other words, the device is affected only by its input information. Therefore, we analyze combinations with failure patterns of devices outputting the above-mentioned input information. In this way, we can reduce the analysis combinations of failure patterns. Therefore, you must enumerate input information for each device.

We describe a method to enumerate input information without exception in the following: Each device process is classified as follows: (1) a device process that handles both input and output, (2) a device process that has only outputs, (3) a device process that has only inputs. Firstly, we begin the enumeration by considering original source of information in the system. The process above (2) is an original source of informa-

tion, and there are many cases where it occurs as a user or operational environmental element. Secondly, we specify all the output information sent from the process. At this time, we enumerate all the output information determined by the expected specifications and unexpected obstacle specifications of the system. Thirdly, we select a process which inputs one of the above-mentioned output information. Then, we enumerate all the output information sent from the process by combining all the input information and failure patterns of the process, and considering the function of the process. Fourthly, we repeat this procedure along the lower flow of information. Finally, an information flow diagram including combinations of obstacles of all devices is obtained.

There are several serious levels of failure patterns of devices. For example, a device does not work at all when a critical failure happens in the device. Of course, there are unserious failures such as fading of light sensor. By combining the serious levels, we can also reduce the analysis combinations of failure patterns. When a critical failure happens in a certain device, the device does not work even if there are other failure patterns associated with the device. Knowing this situation means that it is not necessary to think about combinations of other failure patterns.

6. Discussion

We discuss the IFD and the analysis method proposed in this paper:

(1) Combining existing analysis methods

An analysis method of unexpected obstacles has been described that uses FMEA, FTA, and Guide Words. Analysis of unexpected obstacles requires experience and skill, which is easily applicable by FMEA, FTA and Guide Words. This analysis method can reduce the element of lack in the specification of unexpected obstacles.

This analysis method supports both bottom up and top down analysis. The bottom up analysis using FMEA and Guide Words is not goal-oriented, and the top down analysis with FTA is goal-oriented. Unexpected obstacles can occur anywhere without any specific goal. Therefore the non goal-oriented analysis is useful for unexpected obstacles specification. However, results of the goal-oriented and non goal-oriented analysis are sometimes equal to each other. The redundancy of analysis is permitted in companies because they can achieve high product quality.

(2) Usefulness of IFD

Experts of embedded software development can tacitly construct failure scenarios by inferring and connecting phenomena caused by failures. However, non experts cannot do such inference without effective tools. Therefore, they sometimes miss unexpected obstacle specifications. Obstacle information can be included in the information output from devices and processes. The flow direction of the information on the diagram can lead the inference and construction of failure scenarios since the flow direction is the same as the progress direction of the failure scenarios. Moreover, as described in Sub-section 3.5, combination of failure patterns is effectively analyzed. Therefore, IFD and the analysis method are useful for finding unexpected obstacles.

In the case of large software development, it is very difficult to understand all specifications of the software. IFD is helpful for understanding them as it illustrates all the specifications statically. Moreover, IFD can be made and analyzed partially according to the hierarchy of IDEF0. It is also helpful for handling large software.

(3) Future study

Experts' knowledge of failure patterns of devices and processes, their effects and handling them is necessary for the analysis method using IFD. The knowledge should be formalized according to IFD and the analysis method, and stored in a database to help specification by non experts. An information flow path can make a loop of positive or negative feedback such as automatic control, or vanish by malfunction of carrier, or be born by electromagnetic induction. These phenomena should be analyzed on IFD using graph theory. We also proposed Failure Scenario Matrix[6] as alternative analysis method to IFD for the same purpose. The relationship between them should be studied.

7. Conclusion

This paper has proposed IFD and an analysis method using it for finding unexpected obstacles in embedded software. The flow direction of information is the same as the progress direction of the failure scenarios. Therefore, IFD is useful for inferring and constructing failure scenarios to find unexpected obstacles. IFD is also helpful for handling large software since it illustrates the software statically. The analysis method has combined FMEA, FTA and Guide Words. Therefore, the method supports both bottom up and top down analysis to identify more unexpected obstacle specifications.

We will study the knowledge database of obstacles, analysis method using graph theory and the relationship between IFD and Failure Scenario Matrix in the future.

References

- [1] Nakatani, T., Mise, T., Shinyashiki, Y., Katamine, K., Ubayashi, N., and Hashimoto, M. : "Toward defining a system boundary based on real world analysis." *Proc. of the FOSE2005*, Japan Society for Software Science and Technology, Kindai Kagaku Sha, pp.221–226, 2005 (in Japanese).
- [2] Sumi, T., Hirayama, M., and Ubayashi, N.: "Analysis of the external environment for embedded systems," *IPSJ SIG Technical Reports*, 2004-SE-146, pp.33–40, 2004 (in Japanese).
- [3] Alexander, I. : "Misuse cases, use cases with hostile intent," *IEEE Software*, vol.20, no.1, pp.55–66, 2003.
- [4] Crook, R., Ince, D., Lin, L., and Nuseibeh, B. : "Security Requirements Engineering: When Anti-Requirements Hit the Fan," *Proc. of the 10th Anniversary Joint IEEE International Requirements Engineering Conference (RE'02)*, pp.203–205, 2002.
- [5] Lamsweerde, A. V. and Letier, E. : "Handling Obstacles in Goal-Oriented Requirements Engineering," *IEEE Transactions on Software Engineering*, vol. 26, no. 10, pp.978–1005, 2000.
- [6] Mise, T., Sinyashiki, Y., Hashimoto, M., Ubayashi, N., Katamine, K., and Nakatani, T. : "An Analysis Method with Failure Scenario Matrix for Specifying Unexpected Obstacles in Embedded Systems" *Proc. of the Asia-Pacific Software Engineering Conference (APSEC'05)*, pp.447–456, 2005.
- [7] Leveson, N. G. : "Failure Modes and Effects Analysis," *Safeware: System Safety and Computers*, Addison-Wesley, pp.341–344, 1995.
- [8] Leveson, N. G. : "Fault Tree Analysis," *Safeware: System Safety and Computers*, Addison-Wesley, pp.317–326, 1995.
- [9] Leveson, N. G. : "HaZards and Operability Analysis," *Safeware: System Safety and Computers*, Addison-Wesley, pp.335–341, 1995.

A Generation Method of Exceptional Scenarios from a Normal Scenario

Atsushi Ohnishi

Department of Computer Science, Ritsumeikan University, Shiga 525-8577, JAPAN

Abstract. This paper proposes a method to generate exceptional scenarios from a normal scenario written with a scenario language. This method includes (1) generation of exceptional plans and (2) generation of exceptional scenario by a user's selection of these plans. The proposed method enables users to decrease the omission of the possible exceptional scenarios in the early stages of development. The method will be illustrated with some examples.

Keywords. Requirements elicitation, scenario analysis, scenario generation

1 Introduction

Scenarios are important in software development, particularly in requirements engineering, by providing concrete system description [18]. In particular, scenarios are useful for system developers in defining system behaviors and validating the customers' requirements. In many cases, scenarios are foundation for system development. Incorrect scenarios will have a negative impact on the overall system development process. However scenarios are informal and it is difficult to verify the correctness of scenarios. The errors in incorrect scenarios may include (1) vague representations, (2) lack of necessary events, (3) extra events, and (4) wrong sequence among events.

The author has developed a scenario language for describing scenarios. In this scenario language, simple action traces are embellished including typed frames based on a simple case grammar of actions and for describing the sequence among events [12][13]. Since this language is a controlled language, the vagueness of the scenario written in this language can be reduced. Furthermore, the scenario with this language can be transformed into internal representation. In the transformation, both the lack of cases and the illegal usage of noun types can be detected, and concrete words will be assigned to pronouns and omitted indispensable cases [11][13]. As a result, the scenario with this language can avoid the errors typed 1 previously mentioned.

Scenarios can be classified into (1) normal scenario, (2) alternative scenario, and (3) exceptional scenario. A normal scenario represents the normal and typical behavior of the target system, while an alternative scenario represents normal but untypical behavior of the system and an exceptional scenario represents abnormal behavior of the system. In order to grasp whole behaviors of the system, not only normal scenarios, but also alternative/exceptional scenarios should be specified in the requirements definition phase. However it is difficult to detect alternative scenarios and exceptional scenarios, whereas it is easy to think of normal ones.

This paper focuses on how to generate exceptional scenarios from a normal

scenario. It adopts scenario language prepared beforehand to write scenarios, because previously proposed scenario language is a control language and it is easy to analysis scenarios with that scenario language.

The rest of this paper is organized into 5 sections. Section 2 introduces the outline of the scenario language, and gives a scenario example. Section 3 illustrates a generation method of exceptional scenarios from a normal scenario with examples. Section 4 describes the evaluation of the method. In section 5, a discussion of related works is presented. Finally, in section 6, the author provides some concluding remarks and points out the future work.

2 Scenario Language

2.1 Outline

The scenario language has already been introduced [12][13]. In this paper, a brief description of this language will be given for convenience.

A scenario can be regarded as a sequence of events. Events are behaviors employed by users or the system for accomplishing their goals. It is assumed that each event has just one verb, and that each verb has its own case structure [6]. The scenario language has been developed based on this concept. Verbs and their own case structures depend on problem domains, but the roles of cases are independent of problem domains. The roles include agent, object, recipient, instrument, source, etc. [6][11].

Requirements frames [11] were provided previously, in which their verbs and own case structures are specified. The requirements frame depends on problem domains. Each action has its case structure, and each event can be automatically transformed into internal representation based on the frame. In the transformation, concrete words will be assigned to pronouns and omitted indispensable cases. With Requirements Frame, users can detect both the lack of cases and the illegal usage of noun types [11].

It is assumed that four kinds of time sequences among events: 1) sequential, 2) selective, 3) iterative, and 4) parallel. Actually most events are sequential events.

This scenario language defines the semantic of verbs with their case structure. For example, data flow verb has source, goal, agent, and instrument cases. Since such case structure can define the abstraction level, scenario with our scenario language becomes the almost same level of the abstraction.

2.2 Scenario Example

In this subsection, a scenario of train ticket reservation of a railway company is employed. Figure 1 shows a scenario of customer's purchasing a ticket of express train at a service center of a railway company. This scenario is written in our scenario language based on a video that records behaviors of both a user and a staff at a service center of a railway company [14].

A title of the scenario is given in the first line of the scenario in Fig.1. View-

points of the scenario are specified in the third line. In this paper, viewpoints mean active objects such as human, system appearing in the scenario. There exist two viewpoints, namely staff, and customer. The order of the specified viewpoints means the priority of specified viewpoints. In this example, the object of the first priority is staff, and the second one is customer. In such a case, the first priority object becomes the subject of an event.

In this scenario, almost all events are sequential, except for just one selective event (the 9th event). Selection can be expressed with if-then syntax like program languages. Actually, event number is only for reader's convenience and not necessary.

[Title: A customer purchases a train ticket of reservation seat]
[Viewpoints: Staff, customer]

1. *A staff asks a customer about leaving station and destination as customer's request.*
2. *He sends the customer's request to reservation center via private line.*
3. *He retrieves available trains with the request.*
4. *He informs the customer of a list of available trains.*
5. *The customer selects a train that he/she will get.*
6. *The staff retrieves available seats of the train.*
7. *He shows a list of available seats of the train.*
8. *The customer selects a seat of the train.*
9. **If** (there exists a seat selected by the customer) **then** the staff reserves the seat with the terminal.
10. *He gets a permission to issue a ticket of the seat from the center.*
11. *The customer paid for the ticket by cash.*
12. *He gives the ticket to the customer*

Fig. 1: Scenario example.

2.3 Analysis of Events

Each event is transformed into internal representation. For example, the 2nd event “He sends the customer's request to reservation center via private line” can be transformed into internal representation shown in Table 1.

In this event, the verb “send” corresponds to the concept “data flow.” The data flow concept has its own case structure with four cases, namely source case, goal case, object case and instrument case. Sender corresponds to the source case and receiver corresponds to the goal case. Data transferred from source case to goal case corresponds to the object case. Device for sending data corresponds to the instrument case. In this event, “customer's request” corresponds to the object case. Since the pronoun “he” in the event should be “staff,” concrete noun “staff” is assigned in the source case.

Let us consider other surface representations of event, “Customer's request is sent from staff to reservation center via private line” and “reservation center receives customer's request from staff via private line.” These events are syntactically different but semantically the same. Each of these two events can be transformed into the same internal representation shown in Table 1. In this sense, the internal representation is independent of surface representation of an event.

Table 1: Internal representation of the 2nd event.**Concept: Data Flow**

source	goal	object	instrument
Staff	Reservation center	Customer's request	Private line

3. Generation of Exceptional Scenarios

There exist several causes of abnormal behavior of systems, such as hardware errors, software errors, power supply failures, and human errors. Among of them, let us focus on human errors, because human errors are frequent causes of abnormal behaviors. Human actors are classified into two types, that is, novice type and experienced one. It is assumed that novice type actors tend to make mistakes. Actions are grouped into two types, simple actions and complicated actions. It is said that complicated actions lead to mistakes. It is possible to generate an exceptional scenario using the combination of these types.

3.1 Generation method of exceptional scenarios

Actors of events can be human or non-human, such as systems. Human actors are classified into novice users and experienced ones. The probability of making mistakes for these two human actors (novice user and experienced user) and system actors are shown in Table 2. Actions are grouped into two types, namely simple actions and complicated actions. Complicated actions are data flow, check and select actions. The probability of making mistakes of these two typed actions is shown in Table 3. With the combination of categorized actors and actions, users can select events that tend to be erroneous.

Possible exceptions for each of actions and corresponding exceptional scenario templates are also provided. The outline of the generation method is described as the following steps.

Table 2: Probability of making mistakes by actors.

Actor type	Probability of making mistakes
Novice	High
Experienced	Medium
System	Low

Table 3: Probability of making mistakes by actions.

Action type	Probability of making mistakes
Data flow, check, select	Medium
Others	Low

It is assumed that a normal scenario is written in our scenario language in advance as shown in step 0.

Step 0: Scenario writer describes a normal scenario with our scenario language.

Step 1: The normal scenario is transformed into internal representation. In this step each events is transformed into internal representation based on requirements frame.

Step 2: Events with both a complicated action and a novice actor are selected. Possible exceptions for each of actions are provided in advance. Scenario writers check a list of possible exceptions.

Step 3: In accordance with each of the checked exceptions, a template of exceptional scenario is derived from exceptions DB. Some cases are missing in the scenario template, but missing cases are automatically compensated using the internal representation of the corresponding event.

Step 4: Exceptional scenarios are provided to the scenario writer. He/she can revise or customize them.

If there exist multiple events causing mistakes, the step 2, 3 and 4 will be applied to each of the events. These steps are repeated until there are no more events that have not been applied.

3.2 Example of generating exceptional scenario

The above 4 steps are illustrated with the example shown in Fig. 1. Here it is assumed that customer is a novice type actor and staff is an experienced type actor. The complicated level of actions is also defined as shown in Table 4.

Table 4: Complicated level of actions.

Action type	Concept	Verbs
Complicated	Data flow	Ask, send, get, give, inform
	Selection	Select, choose, pick
Simple	Retrieval	Retrieve
	Payment	Pay
	Show	Show, display
	Reservation	Reserve, book

In the step 2, data flow events whose source case or goal case is “customer” and selection events whose agent case is “customer” in Figure 1 are selected as candidate of exceptional events. The selected events are shown in Figure 2.

A list of possible exceptions for data flow concept and selection concept are also provided. Figure 3(a) shows a list of possible exceptions for data flow concept. Fig. 3(b) shows a list of possible exceptions for action of selection.

- 1. A staff asks a customer about leaving station and destination as customer’s request.
- 4. He informs the customer of a list of available trains.
- 5. The customer selects a train that he/she will get.
- 8. The customer selects a seat of the train.
- 12. The staff gives the ticket to the customer

Fig. 2: Selected events for exceptions.

1. Check whether (object) has some errors.
2. If (source) is novice, then check whether (source) fails to send (object).
3. If (goal) is novice, then check whether (goal) fails to receive (object).
4. If there exists (instrument), check whether (instrument) is damaged.

Fig. 3(a): Possible exceptions for data flow.

1. Check whether candidates of selection (objects) are incorrect
2. Check whether selected (goal) is incorrect
3. Check whether the number of selected (goal) is incorrect
4. Check whether (agent) selects wrong (goal)

Fig. 3(b): Possible exceptions for selection.

By applying the possible exceptions of Fig. 3(a) to the 1st event of Fig. 2, only two exceptions are derived. The 3rd exception is not appropriate, because the goal case “staff” is not a novice actor. The 4th exception is not appropriate, too, because there is no instrument in the event. The derived exceptions are shown in Fig. 4. It is assumed that scenario writer selects the first exception.



Check whether “leaving station and destination as customer’s request” has some errors.



Check whether customer fails to send “leaving station and destination as customer’s request.”

Fig. 4: Derived exceptions for the 1st event.

In the step3, a template of exceptional scenario is retrieved from exceptions DB in accordance with the selected exceptions. Fig. 5 shows a template of exceptional scenario when data flow object has some errors.

In this case, “staff” corresponds to the goal case, “customer” corresponds to the source case, and “leaving station and destination as customer’s request” corresponds to the object case. By compensating these nouns to the template, users can get an exceptional scenario shown in Fig. 6. In the step4, the scenario writer may revise or customize the derived scenario.

[Title: (Object) has some errors in data flow]

[Viewpoints: (source), (goal)]

1. (goal) informs (source) about errors in (object)
2. (source) corrects (object)
3. (source) send (object) to (goal) again.

Fig. 5: Template of exceptional scenario in case of errors in data flow.

[Title: “leaving station and destination as customer’s request” has some errors in data flow]

[Viewpoints: customer, staff]

1. Staff informs customer about errors in leaving station and destination as customer’s request.
2. Customer corrects leaving station and destination as customer’s request.
3. Customer sends leaving station and destination as customer’s request to staff again.

Fig. 6: Derived exceptional scenario.

3.3 Supporting tool for making exceptional scenario

A supporting tool based on the method on a Windows 2000 PC with VisualBasic.NET has been developed. This tool transforms a normal scenario into exceptional scenarios. The left-hand side of Fig. 7 shows possible exceptions for a specified data flow event. The right-hand side of this figure shows a normal scenario with XML format. In the left-hand side of Fig. 8, user selects the first exception. The right-hand side of Fig. 8 shows derived an exceptional scenario in accordance with the selected exception.

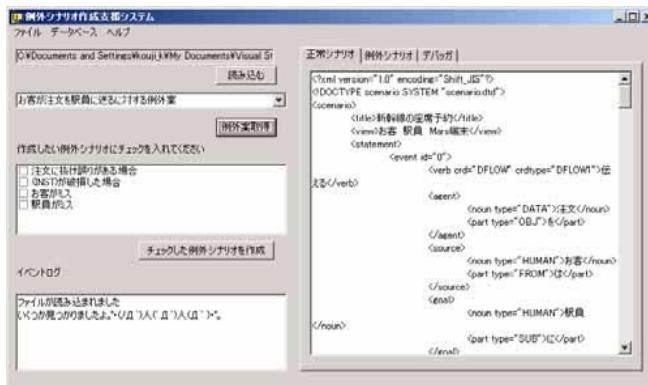


Fig. 7: Original scenario and possible exceptions.



Fig. 8: Derived exceptional scenario.

4. Evaluation

In order to evaluate the method, the following experiment was performed. Two scenario-based software projects are adopted. In these two projects, analysts wrote not only a normal scenario for each project, but also other scenarios, i.e., alternative scenarios and exceptional scenarios. The proposed method is applied to normal scenarios and got exceptional scenarios. Then, it is compared to other scenarios that developed in the projects with automatically generated scenarios. Since original normal scenar-

ios are written with natural language, the normal scenarios are rewritten with the scenario language prior to the experiments.

4.1 Project A: Development of accepting orders for cosmetician

In this project, one normal scenario, no alternative scenarios, and 45 exceptional scenarios are specified. Table 5 shows concepts of actions and the number of occurrences specified in the normal scenario.

Table 5: Concepts and verbs in the normal scenario.

Concepts	The number of occurrence
Data flow	21
Select	1
Check	4
Add	5
Delete	0
Update	4
Modify	1
Retrieve	2
Calculate	3

By applying the method of generating exceptional scenarios, 48 exceptional scenarios were obtained. By comparing original 45 exceptional scenarios with automatically generated scenarios, the author found that corresponding 36 scenarios are same respectively, 9 scenarios are newly generated and effective, 9 scenarios are not generated, and 3 scenarios are new but meaningless as shown in Table 6.

Table 6: Result of exceptional scenarios of project A.

	Total	Same	New	Not generated
Original	45	36	-	9
Method	48	36	9+3	-

As for nine not generated scenarios, because the concepts “calculate” and “confirm” are not regarded as complicated ones, so mistakes related to these two concepts were not detected. By registering these concepts in the table of complicated verbs and providing events of coping with errors of calculation and confirmation, exceptional scenarios can be obtained. As for three meaningless scenarios, exceptional scenarios including wrong operation were generated. This problem can be solved by revising a template of exceptional scenarios.

4.2 Project B: Development of an insurance system

In this project, one normal scenario, 4 alternative scenarios, and 5 exceptional scenarios are specified. By applying proposed method of generating exceptional scenarios, 7 exceptional scenarios can be obtained. By comparing original exceptional scenarios with generated scenarios, it is found that 3 scenarios are same respectively, 4 scenarios are newly generated and effective, and 2 scenarios are not generated. Table 7

shows the above result. As for the not generated two scenarios, because the concept “calculate” is not regarded as complicated one, so mistakes related to this concept was not detected. By registering this concept in the table of complicated verbs and providing events of overcoming calculation errors, exceptional scenario can be successfully obtained.

Table 7: Result of exceptional scenarios of project B.

	Total	Same	New	Not generated
Original	5	3	-	2
Method	7	3	4	-

5. Related works

Ben Achour proposed guidance for correcting scenarios, based on a set of rules [2]. These rules aim at the clarification, completion and conceptualization of scenarios, and help the scenario author to improve the scenarios until an acceptable level in terms of the scenario models. Ben Achour's rules can only check whether the scenarios are well written according to the scenario models. The author proposes a generation method of exceptional scenarios from a normal scenario.

Ian Alexander proposed a scenario-driven search method to find more exceptions [3]. In his approach, a model answer was prepared with knowledge of all exception cases identified by stakeholders. For each event, related exceptions are listed as a model answer. His model answer, however, strongly depends on a specific domain.

Neil Maiden et al. proposed classes of exceptions for use cases [9]. These classes are generic exceptions, permutations exceptions, permutation options, and problem exceptions. With these classes, alternative courses are generated. For communication actions, 5 problem exceptions are prepared, that is, human agents, machine agents, human-machine interactions, human-human communication, and machine-machine communication. They proposed a generation method of alternative paths for each normal sequence from exception types for events and generic requirements with abnormal patterns [17]. The author's proposed approach for generating exceptional scenario is similar, but more precise model based on both case structure of actions and actor types to generate exceptional scenarios are provided

In the author's previous work [11], building software requirements from textual requirements in Japanese is proposed, based on a typology of concepts very similar to the semantic roles of the case grammar [5]. Another related work is Ben Achour's use of case grammar in scenario analysis [1] [2]. Ben Achour focuses on how textual scenarios could be integrated into different existing methods, and proposes guidance for writing scenarios. He provides style and content guidelines referring to conceptual and linguistic model of scenarios, based on the case grammar. These works demonstrate that the case grammar is suitable to the semantic characterization of any design models as well as the semantic characterization of any natural language sentence.

6. Conclusion

The author has proposed a generating method of exceptional scenarios. A guideline of possible mistakes with the properties of actors and actions is provided. With this guideline and its templates, users can get exceptional scenarios. The proposed method contributes to lessen developers' work of making several scenarios and to improve the quality of scenarios. The method was demonstrated by the example and was evaluated. The evaluation results show that the method is valid and effective in software development.

As a future work, the author will research a generation method of templates of exceptional scenarios. Since templates may depend on not only problem domain but also individual jobs, automatic generation of optimized templates should be provided to improve the quality of generated exceptional scenarios.

Acknowledgement: The author thanks to Mr. Koji Kitamoto, Mr. Taishi Yamamoto, Mr. Erwin Widodo, Dr. Hiroya Itoga and all the members of scenario project, Software Engineering Lab., Ritsumeikan U. for their contributions to this research.

References

- [1] Achour, C. B.: Linguistic Instruments for the Integration of Scenarios in Requirements Engineering, Proc. of the Third REFSQ, pp. 93-106 (1997).
- [2] Achour, C. B.: Guiding Scenario Authoring, Proc. of the Eight European-Japanese Conference on Information Modeling and Knowledge Bases, pp.181-200 (1998).
- [3] Alexander, I.: Scenario-Driven Search Finds More Exceptions, Proc. 11th International Workshop on Database and Expert Systems Applications, pp.991-994 (2000).
- [4] Cockburn, A.: Writing Effective Use Cases, Addison-Wesley, USA, (2001).
- [5] Cramp, D.G., Carson E.R.: Assessing Health Policy Strategies: A Model-Based Approach to Decision Support, Proc. International Conference on System, Man and Cybernetics, Vol.3, pp.69-73 (1995).
- [6] Fillmore, C.J.: The Case for Case, in Universals in Linguistic Theory, Bach and Harms eds., Holt, Rinehart and Winston, (1968).
- [7] Jackson, M.: Problems and requirements, Proc. 2nd ISRE, IEEE Computer Soc., pp.2-8 (1995).
- [8] Leite, J.C.S.P. et al.: Enhancing a requirements Baseline with Scenarios, Proc. of the 3rd IEEE International Symposium on Requirements Engineering, pp.44-53 (1997).
- [9] Maiden, N.A.M., Manning' M.K., Ryan M.: CREWS-SAVRE: Systematic Scenarios Generation and Use, Proc. 3rd International Conference on Requirements Engineering (ICRE), pp.148-155, (1998).
- [10] Maiden, N.A.M., Hare, M.: Problem Domain Categories in Requirements Engineering, International Journal of Human-Computer Studies, 49, pp.281-304 (1998).
- [11] Ohnishi, A.: Software Requirements Specification Database Based on Requirements Frame Model, Proc. of the IEEE second ICRE'96, pp.221-228 (1996).
- [12] Ohnishi, A., Potts, C.: Grounding Scenarios in Frame-Based Action Semantics, Proc. of 7th International Workshop on Requirements Engineering: Foundation of Software Quality, pp.177-182 (2001).
- [13] Ohnishi, A., Zhang, H., Fujimoto, and H.: Transformation and Integration Method of Scenarios, Proc. of 26th International Computer Software and Applications Conference (COMPSAC), pp.224-229 (2002).
- [14] Railway Information System: JR System, http://www.jrs.co.jp/keiki/en/index_main.html, (2001).
- [15] Ridao, M., Doorn, J., Leite, J.C.S.P.: Domain Independent Regularities in Scenarios, Proc. of the Fifth IEEE International Symposium on Requirements Engineering (RE '01), pp.120-127 (2001).
- [16] Sutcliffe, A.G, Ryan, M.: Experience with SCRAM, a Scenario Requirements Analysis Method, Proc. of the 3rd ICRE International Conference on Requirements Engineering, pp.164-171 (1998).
- [17] Sutcliffe, A. G., Maiden, N. A. M., Minocha S., Manuel D.: Supporting Scenario-Based Requirements Engineering, IEEE Trans. Software Engineering, Vol.24, No.12, pp.1072-1088, 1998.
- [18] Weidenhaupt, K., Pohl, K., Jarke, M., Haumer, P.: Scenarios in System Development: Current Practice, IEEE Software, March, pp.34-45 (1998).

An Effect of Comment Statements on Source Code Corrective Maintenance

Hirohisa AMAN¹, Hirokazu OKAZAKI and Hiroyuki YAMADA

Graduate School of Science and Engineering, Ehime University

Abstract. This paper focuses on the density of comment statements in a source code, and statistically analyzes an impact of the comment statement density (CSD) on the source code change rate (SCR) through a corrective maintenance. More thorough writing of comments encourages the developers to review their source code more carefully, then that leads to quality improvements of those code. An empirical study is performed with 1,820 version-upgrade cases in an open-source software, Eclipse. The empirical results show that SCR tends to increase when CSD is less than 59.5%.

Keywords. Corrective maintenance, comment statement, statistical test, metric

1. Introduction

Successful development of high-quality software requires comprehensive and careful review (inspection) of the source code[1]. However it is difficult to perform detailed reviews on all source code because of some actual restrictions such as insufficient manpower and delayed development. A lack of detailed reviews has the potential for releasing low-quality software that includes some faults. To prevent deterioration in software quality, there are many activities that oblige the developers to practice a particular coding rule[2]. One of the most basic activities is a thorough writing of comment statements[3]. The writing of comments encourages the developers to review their source code more carefully, and that leads to quality improvements of those code. An effect of such activities would appear as a reduction in efforts of the source code corrective maintenance. More writing of comments tends to aid more detailed code-review, but we have not a criterion for deciding how many comments should be written in a source code. This paper focuses on the density of comment statements in a source code, and statistically find a criterion of acceptable comment statement density through an empirical study.

2. Corrective Maintenance and Comment Statement

2.1. Corrective Maintenance

Software version-upgrades mainly include the following two types of activity:

¹Corresponding Author: Hirohisa AMAN, Graduate School of Science and Engineering, Ehime University, 3 Bunkyo-cho, Matsuyama, Ehime, 790-8577 Japan; E-mail: aman@computer.org.

1. Fault fixing, and source code refactoring.
2. Functionality enhancement.

This paper will call the first type “corrective maintenance”, and the second one “perfective maintenance”, respectively. We focus on the corrective maintenance that is an activity to improve software reliability, and that is a critical and costly activity. It is important to develop a software that will require no or small efforts in the corrective maintenance. In this paper, we will discuss the corrective maintenance for object-oriented software (object-classes) written in Java. Since a corrective maintenance and a perfective maintenance are often mixed into a version-upgrade of a class, we pick up the version-upgrade cases that perform only corrective maintenance, based on some class design features:

“In a version-upgrade of a class, if the class design features (design metric values; see below) remain unchanged through the upgrade, then we regard the upgrade case as a corrective maintenance case”.

We use three metrics shown in Table 1 for picking-up corrective maintenance cases. A change in DIT[4] value means a redesign of the *is-a* relationship. Variations in NCM and PIM[5] values indicate changes in the functionalities to be provided to other classes/objects. Needless to say, the three metrics are not enough to completely specify the all corrective maintenance cases. Since actual version-upgrade cases include various cases, the complete picking-up requires the semantic analysis of the changes performed through the version-upgrades. Note that the above metrics suite is a rough guideline for automatically picking-up corrective maintenance cases from a lot of version-upgrade cases; the accurate picking-up will require the detailed semantic analysis and rigorous management of source code version-upgrade.

This paper consider the following “source code change rate (SCR)” to be an index of corrective maintenance effort:

$$\text{SCR} = \frac{\text{the number of source code lines changed through version-upgrade}}{\text{LOC of the class before upgrade}} , \quad (1)$$

where LOC is the number of lines of code excepting the comment nor blank lines.

2.2. Comment Statement

The comment statements are non-executable (ignored) program statements and they are notations for aiding an understanding of the source code. The writing of comments encourages the developers to review their source code, and that leads to quality improvements of those code. In Java language, we can write three types of comment:

1. End Of Line Comment (EOLC) : the contents from “//” to the end of line character are considered to be comments.
2. Documentation Comment (DC) : the all text from “/**” to “*/” are comments.

Table 1. Class design metrics to be used for picking up corrective maintenance cases.

metric	description
DIT	the depth level in the inheritance tree
NCM	the number of class methods in the class
PIM	the number of public instance methods in the class

3. Traditional Comment (TC) : the contents between “`/*`” and “`*/`” are comments.

EOLC is used for describing what the code fragments are intending. While the definition of DC is included in that of TC, they have different purposes in their uses: DC is a description of method/class functionality, but TC is often used to ignore a part of the source code. We assume that EOLC and DC have effects of quality improvement on the source code. Our assumption will be empirically studied in the following section.

We consider the following “comment statement density (CSD)” to be an index of thoroughness of writing comments:

$$\text{CSD} = \frac{\text{the number of source code lines corresponding to EOLC or DC}}{\text{LOC of the class}} . \quad (2)$$

3. Empirical Study

To find a criterion of acceptable CSD (see Eq.(2)), we performed an empirical study on a relationship between SCR (see Eq.(1)) and CSD, with a well-known open source software, “Eclipse”[6]. We collected a lot of classes (source code) corresponding to the version-upgrades “2.0 → 2.1”, “2.1 → 3.0” and “3.0 → 3.1” of Eclipse. The followings are the steps of our empirical work and their results:

1. Collection of analysis data (ver.2.0 → 2.1 and 2.1 → 3.0)

We measured 14,106 classes included in Eclipse ver.2.0, 2.1 and 3.0, using the metrics DIT, NCM and PIM. Then we selected the classes such that

- all metric values were remained unchanged, and
- one or more source code lines were changed

in their version-upgrades. We also calculated SCRs and CSDs of the classes.

2. Data filtering

Small classes sometimes include little comment statements because of their small sizes of source code. Since small classes would be noise data and/or outliers in our statistical analysis, we excluded small classes from our analysis data. In this empirical work, we regarded the classes whose LOC are less than 83 (the median of LOC) as the small ones. After the filtering, we got 1,115 version-upgrade cases of non-small classes as our analysis data.

3. Statistical test

The analysis data could be classified into two groups by a threshold of CSD, τ (see Figure 1): “Group 1: CSD in the older version class is less than τ ; Group 2: the others.”

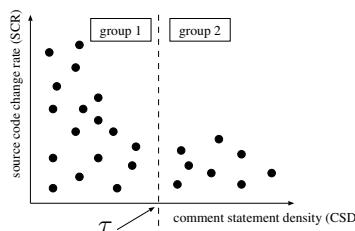


Figure 1. Classification of analysis data.

For $\tau = 0.10, 0.11, \dots, 1.00$ (*i.e.*, $\tau = 0.1 + 0.01n$ for $n = 0, 1, \dots, 90$), we performed the following statistical test:

- Null hypothesis: SCR of the group 1 is equal to that of the group 2;
- Alternative hypothesis: SCR of the group 1 is greater than that of the group 2.

Since the data distribution had not the normality, our test method was Wilcoxon test[7], a nonparametric test, instead of *t*-test.

In the above tests, some τ 's could reject the null hypothesis at 5% significant level. The median, a measure of central tendency, was 0.595. We consider 0.595 to be a threshold of acceptable CSD.

4. Validation (ver.3.0 → 3.1)

We collected 705 version-upgrade cases in the same way as the steps 1 and 2. Then we classified those data into two groups by the threshold value 0.595, and performed the above statistical test. The *p*-value was 0.00893, *i.e.*, the null hypothesis was rejected at 5% significant level. We could see that the classification of classes by the CSD threshold 0.595 was statistically significant.

Our empirical results show that the classes whose $CSD < 0.595$ have a tendency to be more changed in their corrective maintenance. It is a development criterion that we should write about 60% comments in our source code for controlling the corrective maintenance effort.

4. Conclusion and Future Work

This paper has focused on that a thorough writing of comments encourages the developers to review their source code, and such comment writing leads to quality improvements of those code. The empirical study with Eclipse has showed that the source code change in corrective maintenance tends to increase when the comment statement density (CSD) is less than 59.5%. It is a development criterion that we should write 59.5% or more comments in our source code. Note that a writing of more comments does not directly mean a quality improvement. That is a rough guideline for comment writing in a coding rule to be practiced at a scene of software development. Our future work includes an impact analysis of code clone[8] and automatically-generated code on the CSD threshold.

References

- [1] G. Sabaliauskaitė, S. Kusumoto and K. Inoue, Extended metrics to evaluate cost effectiveness of software inspection, *IEICE Trans. Inf. & Syst.* **E87-D** (2) (2004), 475–480.
- [2] N. Ueda and M. Ito, Activities for improving source code quality in embedded software development, *Proc. Embedded Software Symposium 2004* (2004), 54–57. (in Japanese)
- [3] S.L. Pfleeger, *Software Engineering: Theory and Practice*, Prentice-Hall, London, 2001.
- [4] S.R. Chidamber and C.F. Kemerer, A metrics suite for object oriented design, *IEEE Trans. Softw. Eng.* **20** (6) (1994), 476–493.
- [5] M. Lorenz and J. Kidd, *Object Oriented Software Metrics*, Prentice-Hall, N.J., 1994.
- [6] Eclipse, <http://www.eclipse.org/>.
- [7] S. Peter, *Quick Statistics: An Introduction to Non-Parametric Methods*, New York, 1981.
- [8] T. Kamiya, S. Kusumoto and K. Inoue, CCFinder: A multilingual token-based code clone detection system for large scale source code, *IEEE Trans. Softw. Eng.* **28** (7) (2002), 657–670.

Catalogue of Design Patterns

Lubomír Majtás¹

Institute of Informatics and Software Engineering, FIIT STU, Slovakia

Abstract. The paper deals with development of new catalogue specialized on design patterns. It specifies its content as a summary of extensions to well-known GoF catalogue. In the next part it focuses on representation of collected knowledge. Finally author decided to use technologies provided by Semantic Web and specifies the structure of ontology representing the content of the catalogue.

Keywords. Design patterns, catalogue, ontology

1. Introduction

Design patterns [1] are based on empirical knowledge of their authors and they are proven by long-time usage as a well arranged problem solutions. They became frequently used, what leads to their continuous development. Today we can say, that they offer solutions for significant amount of design problems which the software developer has to deal with. Common practice showed that pattern knowledge increase abilities of software developer, it leads to the clearer, better maintainable source code; it shortens time needed for implementation and testing and supports document creation [2]. These qualities lead to large pattern popularity.

In order to improve pattern accessibility to software developers, there were created some catalogues that store and present knowledge about patterns. They usually contain the list of patterns with descriptions and usage examples. It is expected that the user of these catalogues has a serious knowledge about patterns and after close catalogue study, he would be able to decide by self which pattern is the best for his problem. This kind of work with catalogue is not ideal. The user has to analyze common solutions in common conditions and then by process of reverse application attach pattern to his project. It would be much better, if the catalogue would be able to inform the user about the alternatives, suggest which one is the most appropriate and whether it would be better to use different solution.

This paper shows the development of a new catalogue that tries to minimize imperfections of other ones. This new catalogue should actively cooperate with his user in process of knowledge presentation, what should lead to decrease of time needed to find the searched solution. While developing we had to solve two different tasks: to specify the content of the catalogue and to determine the way of knowledge storing. The paper will provide the reader through the both steps of the development.

¹ Corresponding author: Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia; E-mail: majtas@fit.stuba.sk.

2. Content Specification

The first part of content definition is the specifying of the target group of its users. This catalogue was aimed for two users groups: educating people (they wish to improve their general knowledge in the domain of design patterns) and software developers (they would like to apply the concrete pattern in their project). Each group asks for different kind of information, so the catalogue was developed to accomplish demands of both.

There are two different ways of development of the catalogue content. The first one means development of absolutely new catalogue, the second one represents extension of existing catalogue. There exists high-quality design pattern catalogue with extensive pattern description created by GoF [1]. It contains well arranged structure of pattern description that became very frequently used in community of design pattern users. That is why we decided to improve old GoF catalogue rather than to try to develop a new one.

The process of the catalogue improvement consists of identification of disadvantages that should be eliminated and extending it by more knowledge and features that will make it suitable for larger community of users. Table 1 specifies the content extensions that have been applied to the new catalogue comparing the GoF one.

Table 1. Extensions of catalogue's content

Extension	Extension description
Navigation to the pattern	Extension helps beginners to find quickly the pattern which should provide the most suitable solution according their needs and conditions. It is being done by decision tree [3], which asks the user about his needs, and offers him possible choices. The user gets through the tree by choosing from the given possibilities the one that is the closest to his needs or problem – by choosing the possibilities he gets from the most general question to the concrete design pattern providing the most suitable functionality for him.
Extended pattern categorization	GoF defined criterions Scope and Purpose according which they categorized all patterns. The other authors defined another criterions and categories that can help with pattern cataloguing. Addition of new categorizations has two positive effects for the catalogue: it enhances search capabilities of the catalogue and it allows presenting all categories, to which the concrete pattern belongs in one place, what can support with faster understanding of the pattern's nature. The catalogue's categorization has been extended by these criterions [4]: <i>Applies to, Purpose, Scope and Time to apply</i> .
Categorization of pattern relationship	One of the disadvantages of the GoF catalogue is the absence of categorization of relationships between patterns. The new catalogue solves this problem by adding own pattern relationship categorization, it specifies these relationship categories: <i>Uses, Combines, Alternative and Similarity</i> .
Usage indications for different development phase	Software developers should consider usage of design patterns in every development phase: from the first analysis, up to the refactoring of the exiting source code. Indications of patterns usage are not often the same for all development phases, so the catalogue was extended by indications for these phases [3]: <i>Analysis, Design and Refactoring</i> .
Support of Pattern hybridization	Catalogue presents results of the new technology called Patterns hybridization [5], which is an approach to build systems out of patterns and their interactions. The authors defined Pattern hybridization as a mechanism to compose different patterns for generating a hybrid pattern whose intent is to solve a high level design problem in a generic context. Catalogue supports hybridization process by suggesting the patterns that can together create a hybrid pattern according the relationship <i>Uses</i> or <i>Combines</i> .

3. Representation of the Catalogue's Knowledge Base

We decided to represent the knowledge base of the catalogue by the ontology. There was a possibility to represent it by classic relational database, but the ontology provides more benefits. Knowledge of the ontology can be shared across the internet, so the other users can gain of it or it can be extended very easily by the others. We decided to use the W3C's OWL ontology [6].

The structure of the ontology had to fulfill two conditions: to represent knowledge in the structured way as much as possible and to allow its further expansion by patterns on the other level of abstraction. Figure 1 presents the structure of the ontology.

The main class of the ontology is the Pattern class. It is the ancestor for all classes representing patterns of specified abstraction level. At the moment it has only one child: DesignPattern class. In the future, when the ontology will carry knowledge about other patterns, the Pattern class will have more descendants.

The DesignPattern class represents the design patterns – its individuals are concrete GoF patterns. The datatype properties carry the base information about the pattern: its descriptions, usage indications for different development phase and links to its scheme and old GoF description. The object properties “in_relationship” and “is_related_by” associates the class with the class Relationship representing the inter-pattern relationships. This class holds the textual description of the relationship by datatype properties and the category of the relationship by the object property “is_type_of_relationship”. Individuals of RelationshipTypes class represent the relationship categories.

Class NavigationQuestion contains all navigation questions that help the user to find the most suitable pattern. Individuals have relationships with previous question (property “from_question”) and following questions (property “following_questions”), what allows to traverse through the decision tree to the final pattern (property “navigate_to”).

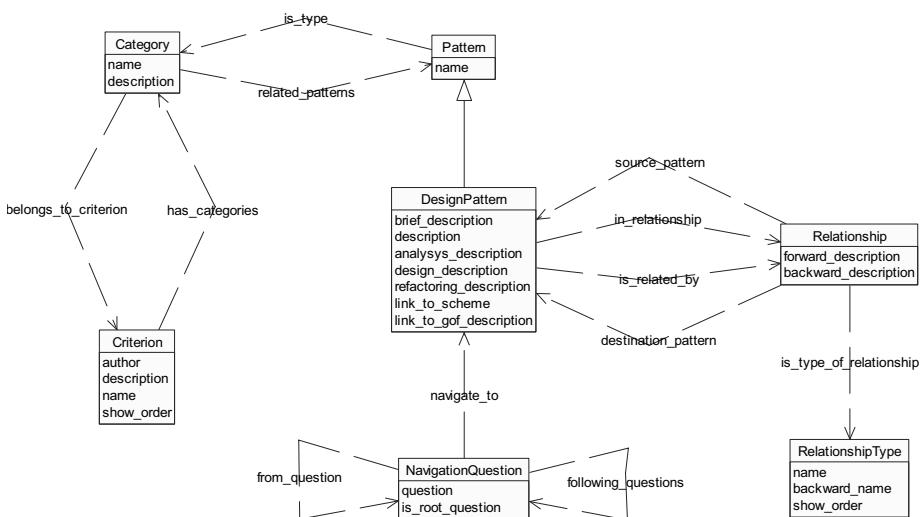


Figure 1. Structure of the ontology representing the knowledge of the catalogue.

Classes Criterion and Category represents enhanced pattern categorization, where class Criterion holds criterions and Category holds its categories.

The defined ontology is able to carry all knowledge defined in the section 2 in very effective way.

4. Conclusion and Future Work

The catalogue provides integrated and extended collection of knowledge concerning design pattern, so it can be useful tool for users allowing them to search for relevant information. It offers not only knowledge about software reuse, but also supports possibility of the knowledge reuse while it employs technologies offered by Semantic Web initiative. Thanks to them it is able to share the knowledge with other users and computer agents via Internet. For “regular” users it presents information in user-friendly way as a web application while for computer agents it allows direct access to the ontology. This should sustain wide accessibility of its content for everyone who is interested in it, what is the basis of its further evolution or possible success.

There are many possible ways of the development: to improve the ontology, to extend the catalogue’s content or functionality. Ontology improvement should make it structurally “compatible” [7] with other ontology dealing the problematic. The content of the catalogue can include the patterns on different level of abstraction such as ADT or Architectonic patterns (the ontology supposes it). Different way of content development can mean extension of knowledge about patterns already included in the catalogue. Great progress would be if the ontology could carry information not only about the patterns functionality or features but also about its structural nature such as in [8]. With this knowledge, the catalogue could be able to provide source code generation of the patterns, what would accelerate work efficiency of its users significantly.

Acknowledgement: This work was partially supported by the Slovak State Programme of Research and Development "Establishing of Information Society" under the contract No. 1025/04 and the Scientific Grant Agency of Republic of Slovakia, grant No. VG1/3102/06.

References

- [1] Gamma, E., Helm, R., Johnson, R., Vlissides J.: *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley professional, 1995. ISBN 0-201-63361-2
- [2] Tichy, W. F.: A catalogue of General-Purpose Software Design Patterns. In: *Technology of Object-Oriented Languages and Systems*, 1997, pages: 330 – 339
- [3] Shalloway, A., Trott, J.: *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Addison-Wesley professional, 2001. ISBN 0-201-71594-5
- [4] Kardell, M.: *A Classification of Object-Oriented Design Patterns*, Umeå, Sweden Master thesis on Department of Computing Science, Umeå University.
- [5] Ram, J., Reddy, J. K., Rajasree M. S.: Pattern Hybridization: Breeding New Designs Out of Pattern Interactions. *ACM SIGSOFT Software Engineering Notes*, Vol. 29, No. 3 (2004)
- [6] Smith, Michael K.; Welty, Chris; McGuinness, Deborah L.: OWL Web Ontology Language Guide. <http://www.w3.org/TR/owl-guide/> (27th March 2006)
- [7] Noy, N.: Order from Chaos. *ACM Queue*, Vol. 3, No. 8, (2005)
- [8] Dietrich, J., Elgar Ch.: A Formal Description of Design Patterns Using OWL. In: *Australian Software Engineering Conference (ASWEC'05)*, 2005, pages: 243 – 250

Reuse of Patterns' Source Code

Jaroslav JAKUBÍK¹, Pavol NÁVRAT
*Faculty of Informatics and Information Technology,
Slovak University of Technology, Bratislava*

Abstract. Patterns are used in different phases of software development, from analysis through architecture, from detailed design to implementation. By using patterns, some problems are identified. One of them is how to reuse parts of patterns at the source code level. This paper is concerned with design patterns especially with reusing general parts of patterns by using different techniques and technologies. We analyze different approaches based on frameworks, libraries, CASE tools and language extensions.

Keywords. Design patterns, reuse, CASE tools, pattern library

1. Motivation

There exist a lot of different catalogues and collections of patterns like [1], [2], [3], whether design patterns or others, concerned with a concrete phase of the software development cycle (analysis, design [3][4]) or concerned with a concrete domain. Using patterns is one of the best practices in software development in general.

In these catalogues, the description of patterns is informal but structured filled up with different models expressed by the UML diagrams. Based on this description, it is possible to recognize basic principles of a pattern, cooperated participants and some of relations between participants. Sometimes, a simple example of implementation is a part of a catalogue.

If the example mentioned above is not satisfactory to understand all features of patterns, it is necessary to find another source for selected patterns. Thereby, different variations of pattern implementation are created. These variations are sometimes modified incorrectly and they can invade basic principles of an applied pattern. We are looking towards to reuse of patterns' source codes. This reuse is not as easy as it seems because patterns connects general parts with domain parts. These two parts of patterns are tightly coupled.

There exist different approaches to reuse parts of patterns based on CASE tools, libraries or other technologies. Each of them has pros and cons and it is only a designer's or developer's decision which approach to use.

¹ Corresponding author: Faculty of Informatics and Information Technologies, Slovak University of Technology, Ilkovičova 3, 812 19 Bratislava, Slovakia; E-mail: jakubik@fiit.stuba.sk.

2. Patterns reuse in frameworks

Framework is software product, package which creates group of products. Framework is assigned to simplifier software development process in concrete domain. Development support is mainly realized by reuse [5].

By using standard object-oriented frameworks, it is necessary to know domain of developed application and framework mechanisms on similar level as framework developer. Application developer is more over restricted to framework language.

2.1. FACE framework

FACE (Framework Adaptive Composition Environment) is an environment for creating an application. FACE is based on the framework.

Applications are created on the basis of modeling approach. Building of an application is done by building a model which uses modeling primitives defined by a framework developer.

FACE connects abstractions at the pattern level with their implementation and supports incremental development process. FACE partially solves a design – implementation gap identified in [5].

Each pattern in FACE is decomposed to a kernel schema and a primal schema. The primal schema consists of abstract classes and their relationships. The primal schema defines the basic structure of the kernel schema. The kernel schema extends primal schema with concrete classes and operations.

A general and reused part of a pattern is located in the primal scheme. The primal scheme is supplemented with the concrete domain's specific parts in an application model.

The source code of an application is generated until a consistent model is ready.

3. Patterns reuse in libraries

One of another possible ways how to reuse pattern implementation is to save general parts of a pattern to the library and reuse it with language constructs. In [6], authors introduce library with name DPL (Design Pattern Library) in Beta language.

Beta is specific object-oriented programming language. Beta disposes of specific nonstandard features and resources, for example virtual lists or renaming.

Authors use resources for creation of complete design pattern library according to [3].

In [6] we can find the description of whole DPL library and description of the creation process as well. In the library, every pattern is divided into two parts. One of them is a part of the library. It is the general part, which is domain independent. The second is the domain specific part and belongs to a concrete application. The domain specific part fills up and adapts the general part for a concrete application, in which the pattern is used.

Authors use special programming language Beta, which allows simple and transparent pattern decomposition to the general and domain specific parts. There exist no special constructs as renaming and virtual list in standardized object oriented languages like C++ or Java. We need to override these constructions with standard resources like inheritances and associations.

The decomposition process for both variations of Composite pattern is presented in [7] using templates and multiple inheritances in C++.

4. Patterns reuse by using CASE tools

In the following option how to reuse parts of patterns we use CASE tools - especially module for patterns' support and module for source code generation. We concern with CASE tools comparison based on patterns' support. In [8] we define criteria of comparison. Based on these criteria, we compare five selected CASE tools patterns' support, we focus on patterns source code reuse in defined design patterns.

Compared CASE tools support almost all of 23 design patterns according to [3] or modification of these patterns for the concrete implementation language, for example for JAVA according to [1]. None of them supports mechanisms for using micro-architectures and pattern visibility in the whole design.

Source code generation in case of design patterns is in Rational XDE and Together at the satisfactory level. We can say that this kind of patterns reuse is one of the most accessible ways how patterns' parts can be reused.

5. Pattern reuse by using language extensions

One of the possibilities how to decompose domain specific and general parts of a pattern can be language extension with specific constructions, which are used for simple manipulation and for completing domain specific parts. In this case, a general part of a pattern is again located in the library. And in the application, special constructions are used for pattern's adaptation to a concrete application context. These specific constructions are not primary assigned to the use of patterns.

This approach is illustrated in paper [9]. By using concerns, authors define adapters, which are used for general pattern modification and used in the concrete domain. In paper [9] authors defines six types of adaptation (interface implementation, fusion of classes and methods, adding new methods, adding class).

This extension is designed without restrictions in programming language layer. So it is possible to use it with other languages and not only with JAVA.

As an illustration, authors use example of Observer pattern as one concern. Observer pattern is adapted to simple graphic user interface (GUI) represented with some buttons and labels.

By using these principles, it is possible to implement all design patterns in the same way as Observer (using concerns). General parts of patterns can be saved in the library and can be reused by using special constructs (adaptation). By using adapters, patterns can be specialized for needs of a concrete application or domain.

6. Conclusions

Design patterns, as it is reflected in the title, define reuse on the design layer. By using previous methods, it is possible to connect design patterns from the design layer with their implementation to the implementation layer. Furthermore, mentioned methods

provide pattern decomposition to domain specific and general parts. The general part is then reused in different application domains.

CASE tools use their own structures for saving information about design patterns. Design patterns are mostly applied with using a class diagram and are modified for a concrete domain. In spite of different possibilities, pattern visibility is not sufficiently supported. Design is blind and uselessly complicated by using more patterns.

Libraries decompose a design pattern into general and domain specific parts in an explicit way. The general part of a pattern is included in the library and the domain specific part is connected to the general part by using language resources. Design patterns narrowly connect both reflected parts and so it is very difficult to have general, generic and transparent solution in the context of standard object-oriented language.

By language extensions with special constructs we obtain access to nonstandard possibilities, which can be used for pattern reuse. General parts of patterns are again externalized to libraries and used later in the application with standard language constructions or new language extensions. This approach supports pattern visibility and does not make model nontransparent. Moreover, libraries support using micro-architectures in the design without explicit knowledge of a designer.

We would like to use language extension approach to build up whole pattern library. This creation depends on the study of patterns structure, their categorization and detailed properties.

Acknowledgement: This work was partially supported by the Slovak State Programme of Research and Development "Establishing of Information Society" under contract No. 1025/04 and the Scientific Grant Agency of Republic of Slovakia, grant No. VG1/3102/06.

References

- [1] Alur, D., Crupi, J., Malks, D.: Core J2EE Patterns: Best Practices and Design Strategies. Prentice Hall / Sun Microsystems Press, 1st edition, June 2001, ISBN 0-130-64884-1.
- [2] Cooper, J.: Java Design Patterns: A Tutorial. Addison-Wesley Personal Education, February 2000, Massachusetts, ISBN 0-201-485394.
- [3] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns, Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, 1995, Massachusetts. ISBN 0-201-63361-2.
- [4] Bushman F., Meunier R., Rohnert H., Sommerland P., Stal M., A system of patterns. John Wiley & Sons Ltd., 1996, Anglicko, ISBN 0-471-95869-7.
- [5] Meijler, T., Demeyer, S., Engel, R.: Making design patterns explicit in FACE. Proceedings of the 6th European conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering, Zurich 1997, pp. 94 – 110. ISBN 0163-5948.
- [6] Agerbo, E., Cornils, A.: Theory of language support for design patterns. Master's thesis, 1997, Computer Science Department, Aarhus University, Denmark.
- [7] Jakubík J., Izolácia všeobecných častí vzoru Composite, Objekty 2005, Sborník príspevkov desátého ročníku konference. Václav Snášel (ed.), Vydala Fakulta elektrotechniky a informatiky, VŠB – Technická univerzita Ostrava, 2005, pp. 74 – 84. ISBN 80-248-0595-2.
- [8] Jakubík J., Comparison CASE Tools Based on Design Patterns Source Code Support, IIT SRC 2006, Proceedings in Informatics and Information Technologies Student Research Conference 2006, Slovak University of Technology in Bratislava 2006, pp. 197 – 203. ISBN 80-227-2395-9.
- [9] Lahire, P., Quintian, L.: New perspective to improve reusability in object-oriented languages. In: Journal of Object Technology, Vol. 5, No. 1, January – February 2006

Goal Algebra for IT Alignment

Shuichiro Yamamoto

NTT DATA Research and Development Head Quarters, Tokyo, JAPAN

Abstract. The goal graph is used to decompose goals in Goal Oriented Requirements Engineering. It is not clear how to validate global optimality of goal graphs. In this paper, the goal algebra is proposed to formalize goal decomposition and provide graph interpretation for the goal algebra. The proposed method can reduce redundant goals and then ensure the global optimality of goal graphs using the goal dependency.

1. Introduction

Software requirements specification is used to validate the correctness of the design and code for the specifications. However, it is not sufficient to validate the correctness of the requirements specifications. This is the reason why business goals are used to verify the correctness of the requirements specifications [1][2]. In the course of developing information systems, it is necessary that functional requirements satisfy the business strategic goals. It is important to clarify dependencies among goals and reduce redundant functional requirements.

This paper is organized as follows. First, the goal graph structure is described. Then the basic concept of algebraic expressions is explained. Next, the goal interpretation is shown to develop goal graphs based on the goal algebra. The goal reduction conditions are considered to optimize goal graphs. At last the effectiveness and issues of the goal algebra are mentioned for conclusion.

2. 3 tired Goal Graph

The 3 tired Goal Graph is structured by three tires, the business strategic goals, soft goals and hard goals as shown in Figure 1. The business strategic goals should be achieved from the business view points. The soft goals are non functional goals that are decomposed into sub goals. The soft goal with no sub soft goal is called as the terminal soft goal. The hard goals represent functional requirements. The business goals are corresponds to soft goals that achieve them. In the same way, the terminal soft goals are corresponds to hard goals to achieve the terminal soft goals.

There are two types of relationships among soft goals. The decomposition relationship includes AND decomposition and OR decomposition. If a soft goal g is decomposed to x and y with AND relationship, both x and y are necessary for achieving g. If a soft goal g is decomposed to x and y with OR relationship, either x or y is necessary for achieving g. The dependency relationship gives inter dependency between two different soft goals.

3. Goal algebra

3.1. Goal term

[Definition] Goal term

- (1) The goal g is a goal term.
- (2) $(g \leftarrow a+b)$ is a goal term when a, b are terms and g is a goal.
- (3) $(g \leftarrow a*b)$ is a goal term when a, b are terms and g is a goal.

[Definition] Semantics of goal term

- (1) If the form of goal term is $(g \leftarrow a)$, the meaning of the term is that the goal term a is necessary for the goal g .
- (2) If the form of goal term is $(g \leftarrow a+b)$, the meaning of the term is that the goal term a or b is necessary for the goal g .
- (3) If the form of goal term is $(g \leftarrow a*b)$, the meaning of the term is that the both goal term a and b are necessary for the goal g .

[Rule] distribution

$$(g \leftarrow (h \leftarrow a+b)*c) \Rightarrow (g \leftarrow a*c + b*c)$$

$$(g \leftarrow (h \leftarrow a*b)+c) \Rightarrow (g \leftarrow (a+c)*(b+c))$$

[Rule] absorption

$$(g \leftarrow (h \leftarrow a+b)*a) \Rightarrow (g \leftarrow a)$$

$$(g \leftarrow (h \leftarrow a*b)+a) \Rightarrow (g \leftarrow a)$$

In the same way, commutative and associative rules are also hold. These rules reduce the left part of expressions as the redundant goals. The following rule also removes redundant goals.

[Rule] parenthesis deletion

$$(g \leftarrow (h \leftarrow a+b)+c) \Rightarrow (g \leftarrow a+b+c)$$

$$(g \leftarrow (h \leftarrow a*b)*c) \Rightarrow (g \leftarrow a*b*c)$$

3.2. Graph Interpretation of goal term

For goal g , goal terms a and b , the goal graph for the goal term $(g \leftarrow a+b)$ is defined as follows.

The parent goal is g , sub goals of g are the goals of left part of goal terms a and b . The decomposition type of g and its sub goals is OR relationship. If the left part of a is omitted, the right part is used as the sub goal. And if the right part is goal itself, the goal is used as the sub goal of g . In the same way, the goal graph for the term $(g \leftarrow a*b)$ can be defined.

4. Optimization using the Goal Algebra

4.1. Goal Dependency

[Definition] Goal Dependency Relationship

If goal a holds then goal b also holds, b depends on a . The expression $a < b$ shows this dependency relationship between a and b .

[Condition] Goal reduction

If $a < b$, goal a or b can be deleted under the following conditions.

(C1) if $(g \leftarrow a^*X + b^*Y)$ then $X = Y$

(C2) if $(g \leftarrow b^*Y)$ then $Y = a^*X$

If $a < b$ then $a+b$ is equals to b . And a^*b is also equals to a . On the other hand, the goals a and b should not be deleted without these conditions.

4.2. Example

The goal term $(g \leftarrow (x \leftarrow a^*(z \leftarrow b^*c)) + (y \leftarrow a + b + d))$ can be interpreted to the goal graph shown in Figure 2. It can be reduced as follows.

$$\begin{aligned}
 & (g \leftarrow (x \leftarrow a^*(z \leftarrow b^*c)) + (y \leftarrow a + b + d)) \\
 \Rightarrow & (g \leftarrow (x \leftarrow a^*b^*c) + (y \leftarrow a + b + d)) \quad (\text{goal } z \text{ is reduced by the conjunction rule}) \\
 \Rightarrow & (g \leftarrow a^*b^*c + a + b + d) \quad (\text{goal } x \text{ and } y \text{ are reduced by the distribution rule}) \\
 \Rightarrow & (g \leftarrow a + b + d) \quad (\text{goal } a^*b^*c \text{ are reduced by the absorption rule}) \\
 \Rightarrow & (g \leftarrow b + d) \quad (\text{goal } a \text{ and } c \text{ are reduced by the dependency rule})
 \end{aligned}$$

5. Effectiveness

5.1. Global Optimality

Suppose two locally optimized 3 tired goal graphs are given. If we want to integrate these two goal graph and minimize IT operations, it is necessary to reserve to achieve business goals of the graphs. The first step to integration is to analyze dependency relationship between soft goals of these two three tired goal graphs. If we find an inter dependency among the soft goals we can easily reduce soft goal part of graphs and then eliminate the descendant of the terminal soft goals. The hard goals correspond to eliminated terminal soft goals are also able to delete. This shows the proposed method could be effectively used to globally optimize the goal graph.

5.2. Goal graph equivalence

For large software developments the sizes of goal graphs are also tended to be large. Therefore management of many different portions of goal graphs needs to evolve these graphs. It is useful to manage large number of goal graphs if a convenient evaluation method to decide the equivalence between portions of goal graphs is provided. The proposed method could be extended to verify the equivalence of different goal graphs. For example, if we want to know the equivalence of the goal graph G and H, the correspondent goal terms for G and H can be calculated. Suppose these goal terms as T for G and S for H. If there is a transformation to a same goal term from T and S based on the rules defined in section 3, the correspondent G and H are equal, because they can be reduced to the goal graph interpreted from the same goal term.

6. Conclusion

In this paper, the goal reduction method based on goal algebra is proposed. It also

proposed the hard goal reduction method using three tired goal graph architecture. Moreover, it is clarified the conditions for reducing goals based on dependency relationship among soft goals. Although the proposed method is for the goal graphs of NFR framework, it could be extended to other GORE approaches such as i* and KAOS [2] because the goal algebra is based on general formalism.

It is also necessary to examine the effectiveness of the method for the practical information system development.

References

- [1] Chun, L., Nixon, B., Yu, E. and Mylopoulos, J., NON-FUNCTIONAL REQUIREMENTS IN SOFTWARE ENGINEERING, Kluwer Academic Publishers, 2000.
- [2] Lamsweerde, A., Goal-Oriented Requirements Engineering: A Guided Tour, RE'01, pp.249-263, 2001.

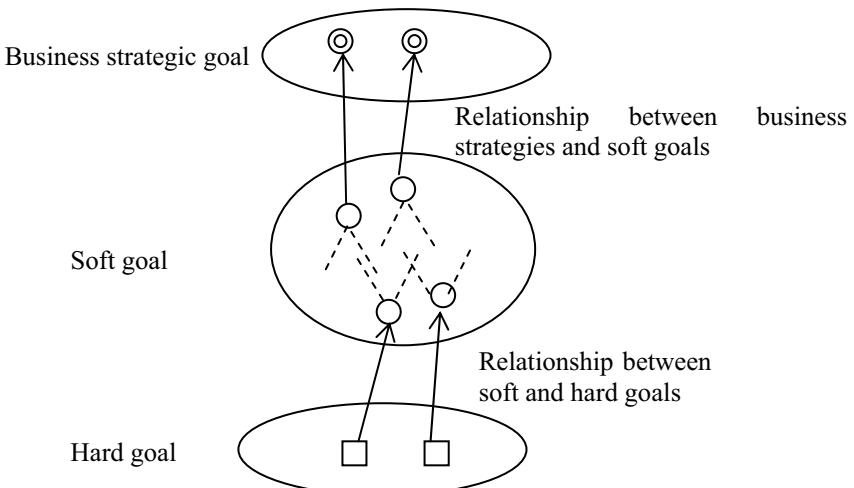


Figure 1 3 tiered Goal Graph

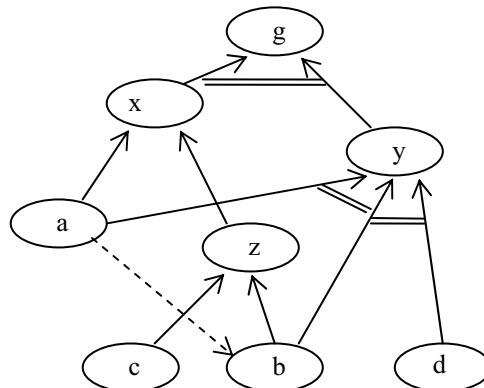


Figure 2 Reducible Goal Graph

Business Process Analysis using Reference Model

Shinobu SAITO, Shuichiro YAMAMOTO
NTT DATA CORPORATION Research and Development Headquarters

Abstract. Reference models are used to analyze and improve business processes. But it is not clear to use reference models effectively. We will propose a business process analysis method using reference model. So the decomposition checkpoints are proposed to clarify the problems in business processes. Then case studies are also presented.

Keywords. Business Process, Reference Model

1. Introduction

Reference models are used to analyze and improve business processes. But these reference models suggested only business process templates for some specific domains [1][2]. It is not clear to use reference model effectively for identifying the problems and getting the improvement ideas [4][5]. Recognizing these backgrounds, we proposed a business process analysis method using reference model. In this paper we focus on manufacturing domain. First, the reference model which we used in this paper is described. Then the basic concept for analysis using reference model is proposed. And the decomposition checkpoints for identifying the problems and getting the improvement ideas are explained. Next, two examples are studied to show the effectiveness of the proposed analysis method. At last, the future works are mentioned for conclusion.

2. Reference Model

In this paper SCOR model [3][7] is used as reference model. The SCOR model is reference model for Supply Chain Management. That model is structured by 5 process types “Plan”, “Sources”, “Make”, “Return”, and “Deliver”. That model focuses on three environments 1: Make-to-Stock, 2: Make-to-Order, and 3: Engineer-to-Order. As a result M1 becomes Make Make-to-Stock Product. M2 becomes Make Make-to-Order Product and M3 becomes Make Engineer-to-Order Product. In this paper M2 model is used for following case study (Figure 1). But The SCOR model is high abstraction level. So the customized analysis method is need when SCOR model is used for business process analysis.



Figure 1. Reference Model (M2: Make to Order)

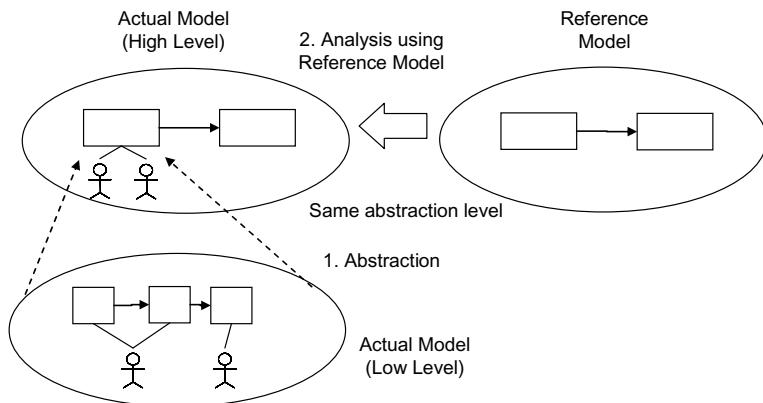


Figure 2. Two layers analysis method

3. Business Process Analysis

A Proposed analysis method is following two steps (Figure 2).

3.1. Abstraction

SCOR model is too abstract (high-level). But actual business processes are so detailed (low-level). Therefore, it is necessary that actual business processes are abstracted to compare actual business processes with reference model. And in actual business processes there must be the relationships among business processes, Actors and Resources. When business process abstraction is done, it is important that these relationships are kept for the following analysis with reference model.

3.2. Analysis using Reference Model

We propose the decomposition checkpoints for identifying the problems and getting the improvement ideas. For example checkpoints are Actor decomposition, Process decomposition and Resource decomposition. In this paper Actor decomposition and Process decomposition is studied. These decomposition points are candidates casing problems in actual business process. So, detailed analysis on these decomposition points is supposed to get the improvement ideas. 2 decomposition checkpoints are followings.

3.2.1. [Checkpoint1] actor decomposition

The point is where many actors take participate of a process which is equivalent to one primitive process in reference model (Figure 3). So, the more the number of actors which take participate of that process is increasing, the more it may be difficult to control that process. Therefore the relationship and cooperation between these actors must be done properly in that point. If not, it may be likely that problems happen.

3.2.2. [Checkpoint2] process decomposition

The point is where two and more process, all of which are equivalent to one primitive process in reference model, exists in actual business processes. Although the task of that process is defined to be done one time by reference model, the task of that process is decomposed and done several times in actual business processes. Therefore the procedure and the method may be different between these decomposed processes. And more, each process may be done based on different information. If those things happen in that point, the problem risk may rise.

4. Case Study

Two case studies in manufacturing factory are following.

Figure5 shows a result of analysis based on actor decomposition. 2 actors, Machine Control Worker and Manufacturing Worker, take participate of a “Machine set-up and Processing material”, which is equivalent to one primitive process “M2.3 Produce and Test” in reference model. First, Machine Control Worker dose set-up to machine. Next, Manufacturing Worker does processing materials with set-up machine. In that point it is necessary to investigate the matter whether there is a problem in the cooperation between two actors. If operation direction between two actors isn’t done timely, there may be a waiting time among actors.

Figure6 shows a result of analysis based on process decomposition. 2 processes, “Schedule for all shops” and “Schedule for each shop” exist, those of which are equivalent to one primitive process “M2.1 Schedule Production Activities” in reference model. First, Production Control Manager schedules production planning for all shops. Next, Shop Leaders individually reschedules their shop’s production planning to shop’s environment. In that point, it is necessary to investigate the matter whether there are differences in the planning procedure between decomposed processes. And it is also important to investigate the matter whether they individually make a decision based on different information. If Shop Leaders has less information than Production Control Manager, There rescheduling plans might be only contribute their own shops. So, that may have a bad influence on enterprise wide optimization.

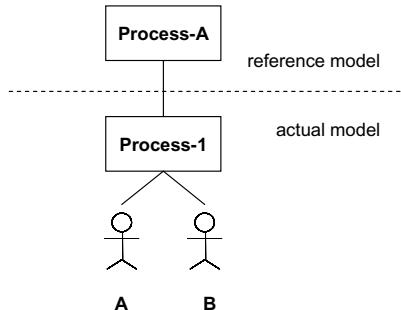
5. Conclusion

In this paper a business process analysis method using reference model is proposed. It also proposed decomposition checkpoints which clarified the problems and the improvement ideas in business processes. And case studies are explained. But much

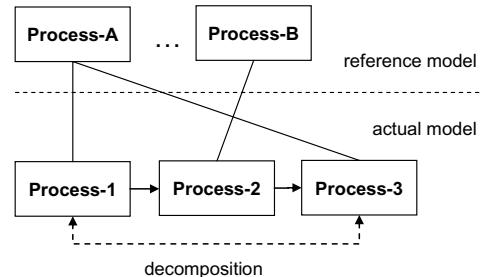
remains to be done in this analysis. It should also consider analyzing business process data [6]. Therefore future works are necessary to integrate both reference model analysis and data analysis.

References

- [1] Heinrich Seidlmeier : Aris: Business Process Frameworks, Springer Verlag, 2000
- [2] Thomas A. Curran et al: Sap R/3 Business Blueprint: Understanding the Business Process Reference Model, Prentice Hall, 1999
- [3] <http://www.supply-chain.org/>
- [4] <http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html>
- [5] <http://ccs.mit.edu/ph/>
- [6] Ryogo Ito,Tomoyuki Kawaguchi, Takahira Yamaguchi: A Support Environment for Business Process Reengineering Using a Fine Grain-Size of Enterprise Ontology, SIG-KBS, vol46, pp29-34, 2000
- [7] Peter Bolstorff, Robert Rosenbaum, Supply Chain Excellence: A Handbook for Dramatic Improvement Using the Scor Model, Amacom Books, 2003



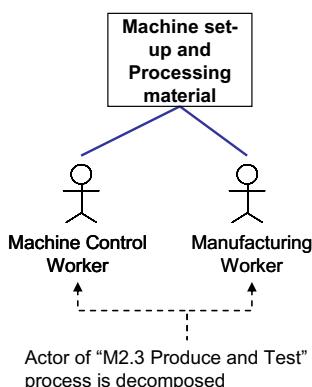
Process-1 is equal to Process-A in reference model



Process-1 and 3 are equal to Process-A in reference model
Process-2 is equal to Process-B in reference model

Figure 3. [checkpoint 1] Actor decompositon

Figure 4. [checkpoint 2] Process decompositon



Actor of "M2.3 Produce and Test" process is decomposed

Figure 5. example of checkpoint 1

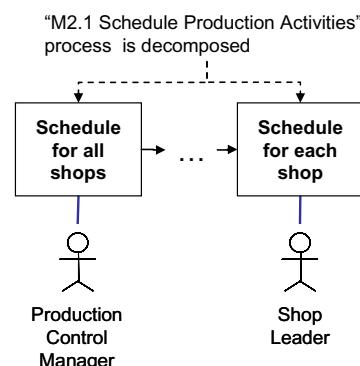


Figure 6. example of checkpoint 2

2. Semantic Web

This page intentionally left blank

Re-engineering OntoSem Ontology Towards OWL DL Compliance

Guntis BARZDINS, Normunds GRUZITIS and Renars KUDINS
Institute of Mathematics and Computer Science, University of Latvia
Raina bulv. 29, Riga, LV-1459, Latvia
guntis@latnet.lv, normundsg@ailab.lv, renars.kudins@gmail.com

Abstract. Re-engineering of successful pre-OWL ontologies or other formal ER or UML system models towards OWL DL compliance opens new possibilities in ontology debugging, enabled by the formal semantics and automated reasoners developed for OWL DL. Meanwhile the transformation of legacy ontologies to OWL DL is a challenging and interesting task, which we illustrate in this paper on the example of OntoSem — a lexical common-sense ontology and an application framework for deep semantic analysis of natural language texts.

Keywords. Common sense ontology, OWL DL, reasoning, natural language processing, text-meaning representation.

Introduction

Reported research is based on the SemTi-Kamols project ¹ at the Institute of Mathematics and Computer Science, University of Latvia, which aims at developing resources and methodologies for efficient integration of Latvian language and the latest semantic web technologies. If successful, this approach promises to open new possibilities, enabled by automatic meaning extraction from texts or generating semantically correct translations. This project is part of a larger “Semantic Latvia” initiative [1], aiming at coordinated development of semantically rich ontology-driven applications in Latvia.

For ontology-driven application development, the key advantage of conforming to W3C Semantic Web standards and particularly to OWL (Web Ontology Language) [2] is the eventual opportunity to integrate multiple ontologies into the “global” semantic web, as well as the possibility to apply an ever-growing arsenal of powerful tools being developed for OWL. The focus of this paper is on OWL DL, a sublanguage of OWL with strictly defined semantics rooted in description logic. OWL DL formal semantics enables ontology debugging [12] and application development based on formal logic and automated reasoners, such as RacerPro [4], FaCT++ [6], or Pellet [11]. These advantages is a good reason for attempting to re-engineer successful legacy (pre-OWL) ontologies (or other formal models, such as ER-database models or UML system models) to OWL DL compliance. In our view, this shall be a true “gold rush” waiting to happen, because a successful conceptualization of some application domain for the

¹ Project is funded by the National Research Program and partially supported by European Social Fund.
Anno 2005. Project website: <http://www.semti-kamols.lv>

purpose of building a formal ontology is a notoriously difficult task and any reuse possibility there is highly rewarding.

Re-engineering of a legacy ontology towards OWL DL compliance consists of two stages: transformation of the original ontology into OWL DL notation (Section 2) and semantic debugging of the transformed ontology by means of formal OWL DL reasoning techniques (Section 3). It is important to keep in mind that in most cases the ultimate goal of the whole ontology transformation exercise is not the OWL DL compliance itself, but rather the possibility to create debugged and highly consistent ontologies — a goal hardly achievable in the pre-OWL era.

1. The OntoSem ontology

In this paper we are discussing transformation towards OWL DL compliance² of a legacy common sense lexical ontology OntoSem [10], developed at New Mexico State University, Carnegie Mellon University and University of Maryland Baltimore County over a period of more than ten years. OntoSem ontology, descendant of the well-known Mikrokosmos³ ontology [9], is represented by ~8000 concepts, which inherit either from the “object” and “event” branches, and ~350 types of “properties” (relations and attributes), which are used to describe (distinguish) the concepts. The upper levels of the ontology are depicted in Figure 1.

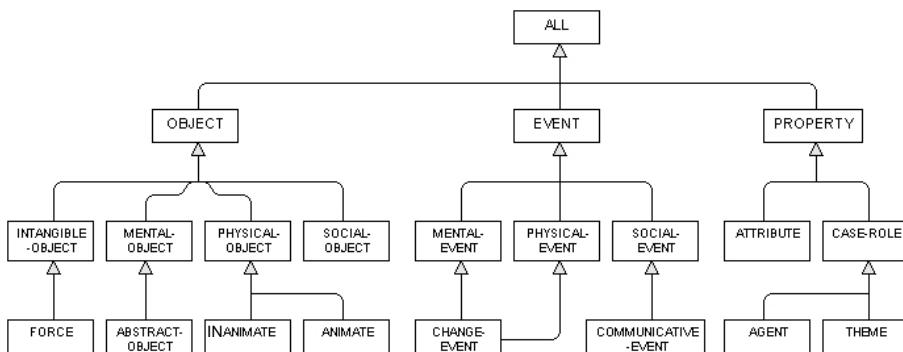


Figure 1. Simplified fragment of the upper levels of the OntoSem ontology, which tries to capture the most general object, event and property concepts referred to by the natural language texts.

Besides ontology itself, the OntoSem group at University of Maryland Baltimore County has developed also the theory of ontological semantics [9] about meaning in natural language — a systematic, non ad-hoc, comprehensive approach to natural language processing, which uses language-independent ontology as the central resource for extracting and representing meaning of texts in the so called Text Meaning Representation (TMR) as well as generating texts based on representations of their meaning. The OntoSem application framework, which has grown on the basis of consequent projects in a period of two decades, is an implementation of the theory.

² It shall be noted that OntoSem authors have converted OntoSem ontology into OWL format [7], but this conversion has not resulted in the OWL DL compliant ontology.

³ <http://crl.nmsu.edu/Research/Projects/mikro/index.html>

Multilingualism of the text-processing environment is supported via mappings of entries of language-specific lexicons to ontological constructions. In Section 4 of this paper we briefly discuss this application framework and its re-implementation based on the converted OWL DL ontology and how it can contribute to further ontology testing.

The reason for choosing specifically OntoSem ontology and application framework for the further discussion, is that to our knowledge currently it is the most successful attempt to integrate lexical knowledge of natural language with the full-featured ontological “world model”. Unlike many other lexico-ontological approaches, which primarily concentrate on building large lexical taxonomies (like WordNet [3]) without a clear application in mind, the OntoSem framework incorporates not only a detailed high-level ontology, but also an ontology driven application for automatic creation of coherent TMR from natural language sentences. The fact that OntoSem framework has been already successfully used for the number of non-English languages [8] gave assurance that it is sufficiently language independent to be applicable also for Latvian.

2. Transformation of OntoSem ontology into OWL DL syntax

The re-engineering step described in this section is specific to the legacy ontology considered. The original OntoSem ontology is described in the proprietary frame-based LISP notation and is available online at [10]. Its semantics is based on the common-sense notation utilized, as well as is partially encoded into the text analysis applications driven by this ontology. As we will see below, by comparison to OWL DL, this legacy level of formalism omits many crucial aspects of the domain being described and thus seriously restricts usability of the ontology.

The general conversion of OntoSem ontology into OWL notation has been accomplished already by the OntoSem development group and described in [7], but their efforts fall short of converting the ontology into the fully formal OWL DL sublanguage. In this paper we are filling this gap by converting the OntoSem ontology into pure OWL DL. Since the general syntax conversion rules are the same as described in [7], we will only discuss the problematic issues specific to OWL DL conversion.

Figure 2 shows the basic syntactic differences between the frame-based LISP notation used by OntoSem, and the corresponding OWL DL encoding. At the same time Figure 2 illustrates also more essential changes imposed by formal semantics of OWL DL. Namely, we are forced to explicitly state that “location” of “soccer” event is a “unionOf” concepts “playing-field” and “sports-area” (and not, for example, “intersectionOf” or “complementOf”) — note, that this information is not explicitly present in the original ontology, although this appears to be the intended meaning.

Next issue is that defining an OWL DL “allValuesFrom” restriction on property “location” of “soccer” event being “unionOf” “playing-field” and “sports-area” in most cases is not sufficient. This restriction means “in case soccer event has a location, then it is either playing-field or sports-area”. In reality we might want to state also “any soccer event has a location, which is either playing-field or sports-area”. For the second assertion we would need to add also a “someValuesFrom” restriction on the “location” of “soccer” being the same “unionOf” “playing-field” and “sports-area”. In Figure 2 the second restriction is omitted (for the sake of simplicity and due to some other properties it indeed is optional); this issue will be discussed also in Section 3.

“soccer” in the original OntoSem ontology	“soccer” in the OWL DL ontology
(make-frame soccer (agent (inv (striker))) (is-a (value (sports-discipline))) (instrument (inv (soccer-ball))) (location (sem (playing-field sports-area))))	<owl:Class rdf:about="#soccer"> <rdfs:subClassOf rdf:resource="#sports-discipline"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty rdf:resource="#location"/> <owl:allValuesFrom> <owl:Class> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#playing-field"/> <owl:Class rdf:about="#sports-area"/> </owl:unionOf> </owl:Class> </owl:allValuesFrom> </owl:Restriction> </rdfs:subClassOf> </owl:Class>

Figure 2. Definition of the concept “soccer” in the original LISP-like syntax of OntoSem ontology and the corresponding transformation into OWL DL.

Inverse relations highlight one more problem. In Figure 2, relation “(agent (inv (striker)))” has been created automatically based on the explicit definition of the inverse relation “agent-of” under the “striker” concept in the legacy OntoSem ontology:

```
(make-frame striker
  (is-a (value (athlete)))
  (agent-of (sem (soccer))))
```

Semantics of such automatically added inverse relations is ambiguous. Does it imply a restriction on the “agent” of the “soccer” to be only a “striker”? Or can “soccer” still have other “agent” types, e.g. “goalkeeper” and “defenseman”, which are subclasses of “athlete” — a more general “agent” type inherited from the “soccer’s” super-class “sports-discipline”? Second interpretation is assumed in the OWL DL translation shown in Figure 2 (restriction for “agent” is not explicitly present in the “soccer” definition, because in OWL DL it is automatically inherited from the super-class “sports-discipline”). Meanwhile, the first interpretation might have been appropriate for some single-player sports disciplines (e.g. “large-hill-ski-jumping”), where restriction to the specific skills (e.g. “ski-jumper”) is appropriate instead of permitting the generic “athlete” inherited from the “sports-discipline”.

However, the most difficult step in the transformation towards OWL DL is the mandatory requirement to specify which ontology concepts are disjoint (mutually exclusive). For example, OntoSem concepts representing social roles “bride” and “groom” are disjoint, while social roles “singer” and “dancer” are not. Unfortunately, the disjointness information is not present in the original OntoSem ontology at all, which means that it has to be added manually during the transformation to OWL DL. As we will show in Section 3, adding correct “disjointWith” information to the ontology is essential both for lexical disambiguation and for reasoning (by default,

classes in OWL DL are allowed to overlap, unless they are specifically defined to be “*disjointWith*” each other). Without adding “*disjointWith*” statements where appropriate, reasoning is limited to simple taxonomy checking. Of course, it is virtually impossible to automate adding of the “*disjointWith*” statements, as this is one of the key steps on the way from informal to formal ontology semantics. Practical experience shows, that it is rational to follow top-down approach here, as stating disjointness between ontology upper level concepts will make immediate impact on subclasses as well. Obviously, the process should be performed with support of a prearranged methodology to gain consistent and objective usage of “*disjointWith*” statements.

These examples are by no means a complete list of problems encountered during the transformation, but they highlight the issues specific to OWL DL conversion. We are skipping here description of more conventional OntoSem to OWL syntax conversion issues, which are already well described in [7].

3. Debugging of the transformed OWL DL ontology

Once the ontology is finally converted into the OWL DL syntax, we can start reaping the fruits of this exercise — by applying an automated OWL DL reasoner, like RacerPro, we can automatically find logical errors and redundancies present in the ontology. Common types of ontology errors that can be found are: inheritance anomalies (classification), classes not permitted to have any individuals due to erroneous restrictions (satisfiability/consistency), and errors related to class individuals. Needless to say, in a large ontology like OntoSem (with thousands of classes and properties), finding all such bugs manually would be a virtually impossible task.

The number of errors we discovered in OntoSem ontology by automated reasoning ranges in hundreds, depending on the error types we are interested in. We will show here few examples to illustrate common error types, which can be discovered thanks to OWL DL formal semantics. Note that all of these errors have slipped the attention of the OntoSem ontology developers over the course of a decade!

The most basic reasoning test for any OWL DL ontology is consistency check — are there any classes, which are not allowed to have any individuals? This test relies on the “*disjointWith*” information added to the ontology in the previous section. For example, by assuming that “physical-object” and “mental-object” classes are “*disjointWith*” each other, the RacerPro reasoner finds the OntoSem ontology class “document” being inconsistent (see Figure 3). Class “document” (“*anything printed or hand-written that is relied upon or used as proof of something*”) is described as a physical object (“artifact”) as well as mental object (“representational-object”). If physical and mental objects are made disjoint (by means of our common sense), we end up in an inconsistency — disjoint classes cannot have common instances. To correct this issue, our proposal is to enroll different associations instead of generalization.

This allows us to get rid of wrong multiple inheritances without loosing knowledge of concept. When talking about documents, we can assume that they are “stored-on” physical object — paper by default. This, of course, does not mean that the range of “stored-on” must be restricted to the physical objects only; range could be objects with appropriate restrictions defined for concrete class contexts. This makes sense, because documents can be stored on paper, as well as electronically at logical addresses (URLs).

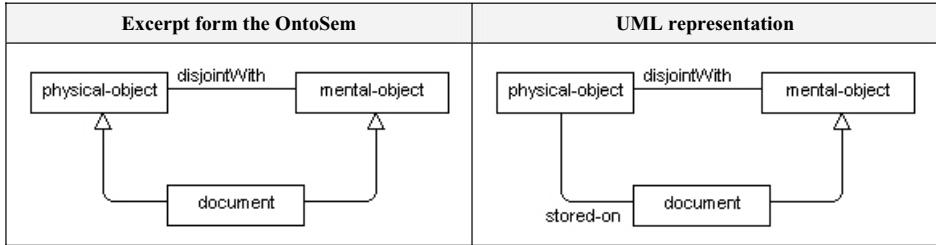


Figure 3. Unsatisfiability error found by automated reasoning. This is a typical classification error due to multiple inheritance. It can be resolved by substituting one of the generalization associations with a semantically more precise (e.g. “stored-on”) association.

The next basic reasoning test is subclass hierarchy check — are there any subclass classification bugs? Figure 4 illustrates one of the subclass classification redundancies found by RacerPro reasoner.

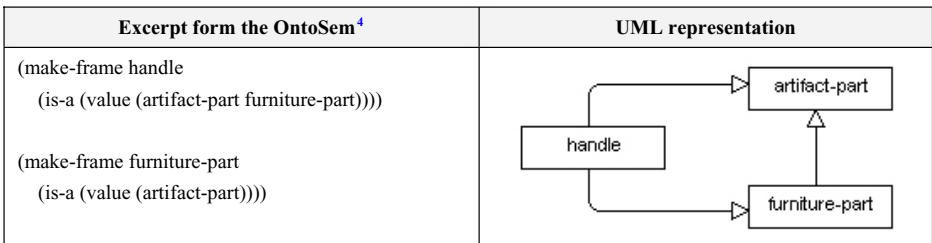


Figure 4. Classification error found by automated reasoning over the ontology.

A more difficult to find error type is illustrated in the Figure 5. In this example the unsatisfiability error is hidden behind the inheritance — here the relation “theme” of class “eat-meal” has direct “someValuesFrom” restriction towards “ingestible” and inherited “allValuesFrom” restriction towards “abstract-object”. Since “ingestible” and “abstract-object” are subclasses of “physical-object” and “mental-object” (which we have defined to be “disjointWith” each other), this leads to the unsatisfiability of the class “eat-meal” (it can’t be instantiated). Also this error is found by the RacerPro reasoner, provided that “someValuesFrom” and “allValuesFrom” restrictions are correctly added to the ontology (see Section 2). Note that Figure 5 shows only those restrictions involved in the inconsistency.

The illustrated error is typical for lexical ontologies, and originates from the polysemy of the concept “eat-meal” and its property “theme”, which in different contexts can describe either a social event or a nutrition event. The solution to such kind of problems is to either split the “eat-meal” concept into two different concepts (e.g. “eat-ingest” and “eat-socialize”), or to split “theme” relation into two different relations (e.g. “ingestible-theme” and “social-theme”).

⁴ Although debugging is being performed exploiting the OWL DL form of the ontology, we provide excerpts in the LISP syntax for readability as far it concerns only syntactical differences.

Excerpt from the OntoSem	UML representation of the transformed version
(make-frame eat-meal (is-a (value (social-event))) (theme (sem (ingestible))))	<pre> classDiagram class abstract-object class mental-object class physical-object class ingestible class object class social-event abstract-object --> social-event : (allValuesFrom) ∀ theme abstract-object --> mental-object mental-object --> physical-object mental-object --> object : disjointWith ingestible --> physical-object ingestible --> eat-meal : (someValuesFrom) ∃ theme eat-meal --> social-event </pre>
(make-frame social-event (is-a (value (event))) (theme (sem (abstract-object))))	
(make-frame ingestible (is-a (value (inanimate)))))	
(make-frame inanimate (is-a (value (physical-object)))))	
(make-frame abstract-object (is-a (value (mental-object)))))	

Figure 5. Unsatisfiability error found by automated reasoning over the ontology.

Finally, ontology might contain strange constructs, which are not necessarily formal errors, but nevertheless deserve to be identified as warnings, because they point to a likely error, or at least an unnecessary redundancy in the ontology. Such constructs cannot be found by a formal reasoner like RacerPro, but are easily identified by OWL DL ontology tests built into OWL-plugin of the Protégé ontology editor [5]. For example, OntoSem property “length” is found to contain redundant domain “literary-style”, not permitted by the super-property “size”:

```

(make-frame length
  (is-a (value (size)))
  (domain (sem (physical-object literary-style)))))

(make-frame size
  (is-a (value (scalar-physical-object-attribute)))
  (domain (sem (physical-object))))
  
```

4. Adaptation of the OntoSem lexical application framework

The original OntoSem ontology has been designed for use by the related ontology-driven lexical application for deep semantic analysis of natural language texts. By embarking on re-engineering the OntoSem ontology to OWL DL semantics, we also have re-implemented the related application, which we will call here txt2owl. Figure 6 shows the overall structure of the txt2owl application.

The main product of txt2owl application is conversion of the natural language sentences into their text meaning representations (TMR), which effectively is a list of OntoSem ontology individuals related by the corresponding ontology properties. Our txt2owl implementation accepts sentences in restricted Latvian, maps them through the Latvian morphology, syntax and lexicon databases into OntoSem ontology individuals

and properties, and finally applies a specialized reasoner to validate the result according to the restrictions defined in OntoSem ontology.

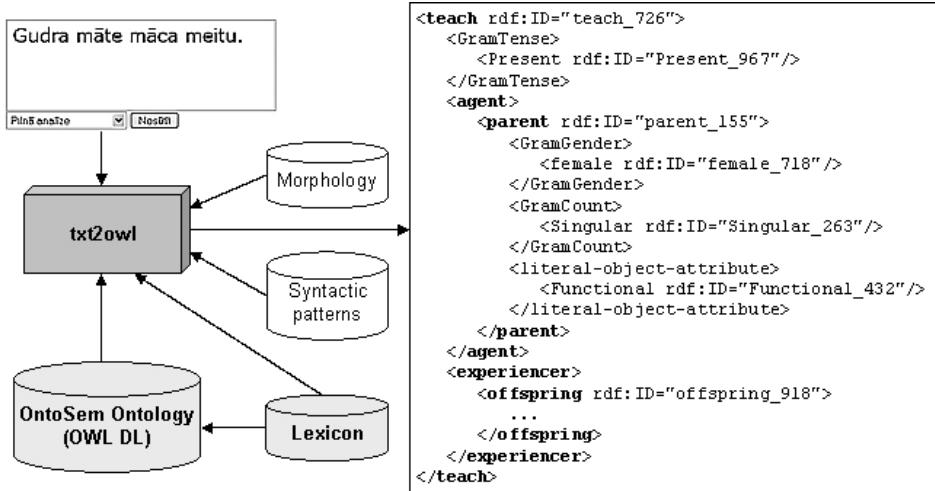


Figure 6. The structure of the txt2owl application generating ontological text meaning representation (TMR) for Latvian language sentences.

Txt2owl application is implemented in SWI-Prolog (which includes convenient libraries for OWL/RDF manipulation) and involves a lot of backtracking through the lexicon mappings in the OWL DL validation process. Although parsing of a single sentence into TMR takes just few seconds, for larger corpus processing the application is being ported also to the grid environment. The example in Figure 7 shows TMR created automatically for the sentence “*gudra māte māca meitu*” (“clever mother teaches daughter”).

```

<teach rdf:ID="teach_341">
  <GramTense><present rdf:ID="present_373"/></GramTense>
  <agent>
    <parent rdf:ID="parent_155">
      <GramGender><female rdf:ID="female_222"/></GramGender>
      <GramCount><singular rdf:ID="singular_594"/></GramCount>
      <literal-object-attribute>
        <clever rdf:ID="clever_558"/>
      </literal-object-attribute>
    </parent>
  </agent>
  <experiencer>
    <offspring rdf:ID="offspring_964">
      <GramGender><female rdf:ID="female_691"/></GramGender>
      <GramCount><singular rdf:ID="singular_562"/></GramCount>
    </offspring>
  </experiencer>
</teach>

```

Figure 7. Text meaning representation (TMR) produced by txt2owl application according to OWL DL/OntoSem ontology from the Latvian sentence “*gudra māte māca meitu*”. Since the generated TMR conveys full meaning of the original sentence, it can be translated into English sentence “clever mother teaches daughter” with no knowledge of Latvian.

The described txt2owl application can be exploited also for validating ontology from common sense point of view — apart from preventing logical bugs in the ontology, it is important to detect real-world inconsistencies as well. This can be achieved by checking ontology mappings of test suites containing real-world examples. In case of a lexical ontology, the most appropriate source for test-case acquisition is the language itself — common assertions expressed in the form of natural language sentences.

Manual validation of large ontology is time-consuming task and might not ensure wide coverage. Instead, an application like txt2owl fed with language-specific lexicon where word senses are aligned to appropriate ontology concepts⁵, makes it possible to automate the test-case generation process illustrated below. Moreover, such approach could add more objectivity to the test data in contrast to merely manual approach that involves human subjectivity factor.

Candidates for comprehensive and reliable knowledge sources include, but are not limited to: balanced text corpora (provides also frequencies of utterance), encyclopedias, or dictionaries, which contain definitional phrases and typical examples assigned to word senses, usually given in a rather canonical form. Test-cases can be seen as graph patterns, which we are trying to map to the ontology. For instance, typical patterns that can be found in example phrases given for verb senses are:

EVENT—AGENT—OBJECT (e.g. “*a boy runs*”)
 EVENT—THEME—OBJECT (e.g. “*to prepare food*”)
 OBJECT—AGENT—EVENT—THEME—OBJECT (e.g. “*a student reads a book*”)

Below are illustrated the steps required in test-case acquisition by a simplified definition of word *honey*: “*a sweet sticky fluid made by bees*”. For this example txt2owl application by means of morphological analysis, syntactic parsing and mapping of syntactic roles into semantic roles (properties), could produce a test-case (extended TMR including combinations of concept and property mappings in case of a syntactically/semantically ambiguous sentence) shown in Figure 8.

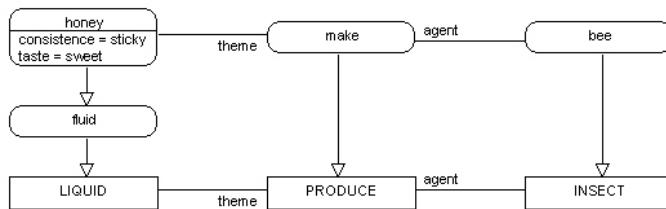


Figure 8. Test-case pattern for “produce” event, derived from the definition of *honey* in a dictionary. Ontological concepts (in upper case) are linked with the running words through the txt2owl lexicon.

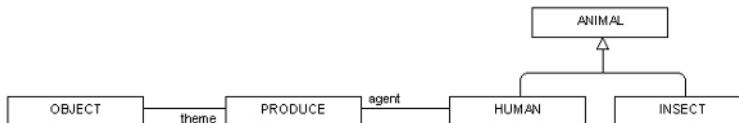


Figure 9. OntoSem ontology fragment relevant to test-case in Figure 8. Test fails — too restrictive range for property “agent” discovered.

⁵ We have skipped description of the model of a language-specific lexicon as it is out of the scope of this paper. See [7, 8, 9] for more details.

It is easy to see that the original OntoSem ontology (relevant fragment shown in Figure 9) does not permit the test-case depicted in Figure 8: the range of “agent” property is restricted too narrowly to include only “human”. To prevent this deficiency and get the test-case accepted, ontology needs to be modified by relaxing restriction on “agent” towards “animal”.

Conclusions

This paper describes the work in progress. The legacy OntoSem ontology has been converted into OWL DL syntax and debugged through the techniques described in this paper. Meanwhile several aspects of the original OntoSem framework are not yet ported to txt2owl application, such as fact repository, co-reference resolution and procedural semantic routines. Nevertheless, the core functionality of TMR creation is completed and demonstrates the viability of the OWL DL based approach, which greatly simplifies re-implementation of the OntoSem application framework through the use of standard OWL DL toolset libraries and automated reasoners.

References

- [1] Barzdins J., Barzdins G., Balodis R., Cerans K., Kalnins A., Opmanis M., Podnieks K. *Towards Semantic Latvia* // In: Seventh International Baltic Conference on Data Bases and Information Systems, DB&IS'2006 (submitted).
- [2] Dean D., Schreiber G., Bechhofer S., van Harmelen F., Hendler J., Horrocks I., McGuiness D., Patel-Schneider P., Stein L. A. *OWL Web Ontology Language Reference*. W3C Recommendation, 2004. Available at URL: <http://www.w3.org/TR/owl-ref> [date of citation: 2006-03-20].
- [3] Fellbaum C. (Ed.) *WordNet: An Electronic Lexical Database*. Cambridge: The MIT Press, 1998.
- [4] Haarslev V., Möller R. *RACER System Description*. // In: Goré R., Leitsch A., Nipkow T. (Eds.) International Joint Conference on Automated Reasoning (IJCAR 2001), Siena. Springer-Verlag, 2001, pp. 701–705.
- [5] Horridge M., Knublauch H., Rector A., Stevens R., Wroe C. *A Practical Guide to Building OWL Ontologies Using the Protégé -OWL Plugin and CO-ODE Tools*. The University of Manchester, 2004. Available at URL: <http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf> [date of citation: 2006-03-20].
- [6] Horrocks I. FaCT++ web site. Available at URL: <http://owl.man.ac.uk/factplusplus> [date of citation: 2006-03-20].
- [7] Java A., Finin T., Nirenburg S. *Integrating Language Understanding Agents Into the Semantic Web* // AAAI Press, 2005. Available at URL: <http://ebiquity.umbc.edu/get/a/publication/221.pdf> [date of citation: 2006-03-20].
- [8] Nirenburg S., McShane M., Beale S. *Increasing Understanding: Interpreting Events of Change* // In: Proceedings of the OntoLex 2005 Workshop, 2005, pp 43-52.
- [9] Nirenburg S., Raskin V. *Ontological Semantics*. Cambridge: The MIT Press, 2004.
- [10] Nirenburg S., Raskin V. Version of OntoSem ontology. Available at URL: <http://thoth.ilit.umbc.edu/CMSC-771/mikro-ontology.lisp> [date of citation: 2006-03-20].
- [11] Parsia B., Sirin E.. *Pellet: An OWL DL reasoner* // In: Möller R., Haarslev V. (Eds.) Proceedings of the International Workshop on Description Logics, 2004.
- [12] Wang H., Horridge M., Rector A., Drummond N., Seidenberg J. *Debugging OWL-DL Ontologies: A Heuristic Approach*. // In: Gil Y. et al. (Eds.) International Semantic Web Conference (ISWC 2005), LNCS 3729, 2005, pp. 745–757.

Searching over Public Administration Legal Documents Using Ontologies

Diego BERRUETA ^a, Jose Emilio LABRA ^b and Luis POLO ^a

^a CTIC Foundation, Gijón, Spain ¹

^b Department of Computer Science, University of Oviedo

Abstract. In this paper, we apply Semantic Web technologies to the creation of an improved search engine over legal and public administration documents. Conventional search strategies based on syntactic matching of tokens offer little help when the users' vocabulary and the documents' vocabulary differ. This is often the case in public administration documents. We present a semantic search tool that fills this gap using Semantic Web technologies, in particular, ontologies and controlled vocabularies, and a hybrid search approach, avoiding the expensive tagging of documents.

Keywords. Semantic Web, Spread Activation Algorithms, Synsets, Concepts, Syntactic Search, Semantic Search

1. Introduction

The vast quantity of information available in the world wide web made indispensable the development of search engines and information localization systems. Some of these systems are daily used tools of the majority of people and have become one of the most profitable companies in the Internet sector. These systems have a general domain and try to offer good results for general searches. They have to tackle with documents written in different languages, about quite different subjects and published by very different people and organizations.

In the case of a Public Administration, the type of published documents is more uniform, with one or a few official languages, about some given subjects and with a more controlled generation process. Nevertheless, Public Administrations publish everyday lots of documents that citizens are supposed to read, such as new laws, announcements, notifications or subventions. This information was usually published in printed bulletins, but is increasingly being published on the web. In such a controlled environment, it could be possible to apply specialised approaches which could improve the search results and user satisfaction.

Most of the Public Administration documents are written in legal and administrative jargon, far from ordinary language, and this represents a hindrance for the communication between citizens and Public Administration. This is a barrier that renders the tradi-

¹This work is partially funded by the CTIC Foundation through a contract with the Principality of Asturias and by the CTIC project TIN2004-03453

tional syntactic search almost useless, as terms that are considered synonyms by citizens have a clearly different meaning for the expert lawmaker.

The arrival of the Information Society has eased the citizen's access to Administration informative publications. But the published information localization rests chiefly upon identification of certain keywords introduced by the user. This way of accessing documents can become profitless, hence the dearth of citizen's habit regarding legal and administrative terminology.

The Semantic Web initiative powered by the W3C tries to provide an enlargement of the present-day Web. This technology allows the users to access the information easily, efficiently and quickly by means of suitable software services.

Among these technologies, we have decided to use an ontology-based model to ease the access to Public Administration documents. In this way, it is possible to develop new services that remove the language handicap, improving the chances of retrieving the desired information.

The paper is organised as follows. In Section 2 we present some motivation. Section 3 describes our ontology based search approach. In section 4, we describe the architecture and the development process that we have followed. Section 5 describes some related work and finally, Section 6 presents the conclusions and future work.

2. Motivation

CTIC Foundation (Centre for the Development of Information and Communication Technologies in Asturias) is a private non-profit institution founded by the Regional Government of the Principality of Asturias and a Consortium of regional private companies in the field of Information and Communication Technologies. The CTIC Research Department, in collaboration with University of Oviedo, has focused on one of the unidirectional communication channels between Public Administration and citizens, namely the *Boletín Oficial*² of the Principality of Asturias (BOPA).

This channel was seen as great opportunity to apply semantics, as it combines a number of technical challenges and a linguistic challenge. Anyone is potentially interested in the information pieces in the BOPA, either driven by personal or professional interests, but only a few know the difference between an order, a law and a decree, and these are only some examples. Additionally, a deep knowledge of Public Administration internal structure is often required to determine which department is responsible for the particular area of interest.

Our primary goal is to provide a practical search tool that enables the user to find the desired information, even if he does not have the knowledge of the administrative domain and vocabulary.

3. Ontology-based Search Approach

The standard initiative in semantic web rests upon the use of metadata to describe any resource, in this case, legal documents. However, it is too expensive to tag a vast amount

²The *Boletín Oficial*, or Official Gazette, is a daily newspaper published by Spanish national and regional Governments. It contains the verbatim listings of new legal texts and compilations of all the Administration announcements.

of documents (increasing daily), and it is also difficult to maintain the metadata over changes or additions in the domain. From the very start of the project, we gave up this approach, and we decided to introduce semantics in a different way.

3.1. Ontologies

After extensive analysis of the Administrative domain, and using Protégé [1] editor, we have built two kinds of OWL-DL [2,3] ontologies with different purposes:

1. A legal and administrative ontology. This ontology formalises the basic structure of BOPA and Regional Public Administration. It captures the relationships between the different departments of the administration and the type of legal texts they can issue.
2. A set of particular domain ontologies. Each one captures a small and well defined area of general knowledge, actually bringing the human expert knowledge to the system.

Every concept has associated a synonym set, similar to those presented in the WordNet architecture [4] but without such complexity. These synsets are the bridge between the ontology and users on one hand, and the ontology and the document base on the other.

The semantic structure was chosen following the psychological bases of Communication Theory, specially the definition of “context” in Relevance Theory [5], treating each subset of the domain ontologies as a common context to both citizen and search engine. That is to say, the concept set constituting the ontology is relevant both for the user search intentions and for the query process.

3.2. Search Process

The search process uses a hybrid syntactic and semantic search [6]. More precisely, it automatically transforms a semantic query into an equivalent syntactic query, exploiting the relationships in the ontologies and some linguistic knowledge. A user query is composed of a set of concepts from a unique context (see step 1 in Figure 1). Both concepts and context are selected by the user through an interface that hides the complexity of the underlying ontologies. No special abilities are required, since the user interface is not harder to use than conventional search engine.

These user-selected concepts are the input of an spreading activation algorithm [7] applied to the concepts in the ontologies (step 2). This algorithm, developed in the Artificial Intelligence area, works essentially as a graph explorer. Given an initial set of nodes, the algorithm traverses the arcs, activating nodes which are closely related.

In our particular case, the concepts are activated in the ontologies by the query forming the initial set. Each of these concepts receives a score, the top score. Hereafter, the spread activation algorithm scores only the concepts closely related by semantic links. The more we move away from the initial concepts the more this scoring decreases. The algorithm ends when the concept score is too low or when there are no more concepts to explore.

The output is a larger concept list, sorted by relevance. In this sense, the chain of concepts that forms the most relevant path between user-activated concepts receives an

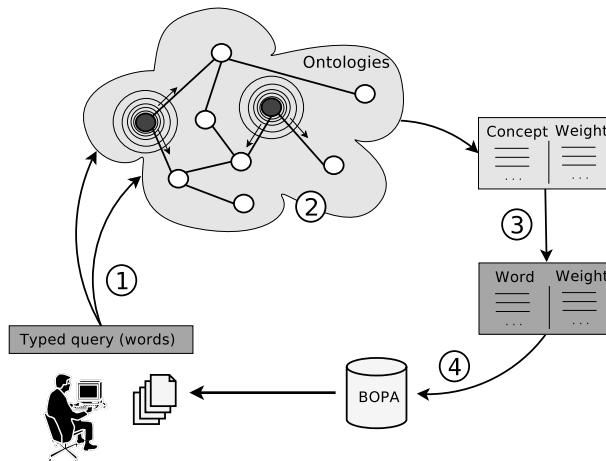


Figure 1. Representation of the semantic search process.

enhanced score. This technique is used in order to improve the score of some concepts trying to better reflect the original user search intentions.

In the third step, the concept list is then transformed to a word list. Each concept has an associated synset (see Section 3.1), specially designed to match the legal and administrative vocabulary used in BOPA.

As the fourth and final step, a syntactic search query is built and executed over the XML documents with a conventional search engine. Hopefully, it should bring the desired results.

In spite of using a traditional syntactic search engine, the semantic value of our approach lies in the enrichment of the query by means of the relations between the concepts in our knowledge base. This semantics increments the number of returned documents because of the synsets, merging the results of several queries that users should perform one by one in traditional search engines. For instance, searching for “accessibility” also brings results for “disabilities”, “blindness”, “handicapped” and announces of subventions for the adaptation of public vehicles and buildings, even if there is no syntactical match with the original search term typed by the user.

More interesting results are obtained from multi-concept queries. For instance, combining “chlorine” and “health” returns as result the swimming-pool regulations.

We have implemented this algorithm in Java using OWL-API [8]. Although the number of search words can be large, current syntactic search engines (such as Lucene [9]) are highly optimized and perform well.

3.3. Services Provided to Citizens

On the top of these ontologies and this algorithm, we have built three services: semantic search interface, alert subscription and a concept browser.

1. The semantic search service is the bridge service filling the linguistic gap. It provides a simple user interface to perform semantic queries using the algorithm described above.

2. The alert subscription service allows the user to subscribe to any arbitrary search results, and to be notified as new similar information becomes available.
3. Ontologies can be exploited in more ways than just searching pieces of information in the bulletin. As they reflect a model of Public Administration operation, citizens can gain more knowledge about the Administration browsing the contents of the ontologies. We have built two concept browsers, using SVG and plain HTML. Exploring the ontology relations also helps to explain the semantic search results.

4. Architecture and Development

In addition to the research topics described in the previous section, we provide some engineering notes about the application itself and its development process.

4.1. Application Architecture

Our application is a multi-tier application built with J2EE technology, making heavy usage of the industry patterns [10].

The data access tier interacts with multiple data sources, such as a relational database, a XML-native database, some syntactical indexes managed by Lucene and the original web server where the BOPA is still published.

The business tier integrates several vertical subsystems, with low coupling, such as syntactical analysis and search engine, semantical search engine, ontology processing and data fetching.

The main user interface is built using the Struts framework, and communicates with the business tier through Enterprise JavaBeans and SOAP web services.

4.2. Development Process

A team of 5-6 people has been working in this project for nine months. This is a multidisciplinary team of Computer Engineers and Linguistic experts. The development process has been driven by an agile methodology (Extreme Programming [11]), which fits particularly well into projects with a high innovation component. In this way, a working product was already available as soon as three months after the beginning of the project. Since that initial prototype, the team has focused on adding more features, or user stories, in several short (two weeks) iterations.

By following an agile approach, the team has been able to react to frequent changes in the scope of the project (which should be no surprise in a research project) and even changes to the team size.

4.3. Recovering Information from HTML

In the semantic web vision, documents are published in the web with their metadata expressed in an appropriate language, such as RDF. This is not the case nowadays, as a vast amount of the contents of the web are expressed in HTML, focusing in presentation issues and taking small, if any, care of preserving semantics.

The web version of the BOPA is a clear example of the previous problem. By historical reasons, the main interest of its publisher is the page composition of the printed version of the bulletin. The HTML version is generated as a subproduct of the traditional bulletin publishing chain, using the “export to HTML” feature of the word processing software.

As a result, the published HTML has poor quality. At the beginning of our work, some of the main problems we found were non-validating HTML, mixed presentation and data and inconsistent markup of titles and article separators.

We set up a fetching and cleaning process as follows:

1. We fetch the articles from the existing web servers. The file-article correspondence is not one to one, so our robot follows the links and detects the article separators. We gather pages from two different web servers because they contain complementary data. Unfortunately, matching the data from each web server is not straightforward, so we apply some heuristic techniques to determine the mappings.
2. Using the JTidy library (a Java port of the W3C HTML Tidy), we transform the HTML into valid XHTML documents.
3. By applying XSL stylesheets, we extract the data from the XHTML. The data is consolidated in XML documents, and inserted into a native XML database.

4.4. Application Integration

Our application is designed with interoperability in mind. We provide several facilities allowing the integration of new applications:

- The most important features of the application (in particular, the search interface and data access) are exported as SOAP web services. At the moment of this writing, there are two applications consuming these web services: an alternative user interface for blind people and a search interface for the desktop.
- The RSS format is a popular mechanism for syndication. We export a RSS channel which is updated daily with the contents of the new bulletins. We are also considering the generation of RSS channels for the results of the most frequent queries.
- In our system, the user can access the bulletin and article data in HTML and PDF, but also in RDF format. In this way, we feed the semantic web.

4.5. Evaluation of Search Results

It is obvious that the only valid source of feedback about the quality of the search results is the citizen (the end user). Unfortunately, explicitly asking the user for feedback provides only a minimum amount of responses. On the other hand, a lot of indirect feedback can be mined from the server logs and the traces of user activity. Our system implements this in order to evaluate the quality of the search results. We expect to obtain lots of valuable indirect feedback in the next months, as the application is progressively deployed into the production environment.

We also have some training sets, containing the expected results of some popular queries, selected and ranked by human experts. We wrote a simple tool to compare the search results provided by our application and the training sets. This tool provides metrics about the quality of the search results.

5. Related Work

The development of information retrieval methods for web documents has been a topic of research from the beginning of the Web [12,13]. The application of semantic web technologies to improve search engines and knowledge management systems has also been a subject of great interest in the last years [14,15,16,17,18,19]. Other approach is based on performing statistical term analysis over the document basis that is latter used for query expansion [20].

In the legal domain there is a lot of ongoing work [21] and specially some approaches that apply semantic web technologies. Gilardoni *et al.* [22] have developed a system which adds a semantic layer to a knowledge management system to assist lawyers in their work. That approach is complementary to our approach in the sense that our focus is to help citizens in general to find legal documents.

Breuker *et al.* [23] proposed a set of ontologies in the e-COURT project. They develop an initial core ontology and several specialised legal ontologies for different local courts. They provide a simple mechanism for query expansion with word sense disambiguation [24]. In their case, the goal of the project is to help in the transcription of hearings of criminal trials. It may be interesting to link the legal concepts in their ontologies with some of the legal concepts in the Public Administration domain or in the European Community Legislative texts [25].

Saias and Quaresma [26] propose a methodology to transform a traditional information system to a semantic aware one and they apply it to legal documents. Although the methodology is similar to our approach, they use a number of different techniques to enrich the queries, to define the legal ontologies and to translate those ontologies to Prolog for inference.

The hybrid syntactic and semantic search approach has been proposed in [6] where they also use the spreading activation algorithm. However, we do not do spreading activation between documents because in our case they are not linked. Our spreading is restricted to concepts in the ontology, knowing nothing about the data in the documents.

In [27], the authors propose another hybrid approach which combines syntactic search using Lucene with some semantic matching using WordNet. In our approach, we extend the matching to ontology concepts providing more semantic capabilities.

6. Conclusions and Future Work

Combining various techniques (syntactic search engine, spread activation, ontologies and controlled vocabulary thesauri), we have built a semantic search tool over a big document base (over 30,000 documents and almost 150,000 different terms).

We believe that our approach from an ontology based model is fundamentally more efficient than a traditional syntactic search engine. Using the ontology conceptual network, our prototype has the necessary knowledge to find documents which are semantically related to user's intentions. The results we have obtained so far confirm this fact.

This search improving rests upon two factors. On one hand, the combination of semantic exploring techniques (Spread Activation Algorithm) and traditional syntactic processes. On the other hand, the connection between the citizens vocabulary and administrative and legal vocabulary through the ontological bridge.

At the moment of this writing, the application is working with more than 500 different concepts and over 2,000 terms in the controlled vocabulary. We plan to improve our work in several directions:

- Using the tools for search results quality evaluation, we aim to fine-tuning the spreading activation algorithm and the other components of the search process.
- We are also widening the document base to cover generic public administration web pages, and building new ontologies and vocabularies for these new domains.
- Another aim is to improve the user experience bringing the interaction closer to natural language. In order to achieve this goal, WordNet seems to be the most powerful tool [4,28], so we are researching how to link our concepts with WordNet senses.
- There are some query expansion approaches which take into account the system and user history [29,30]. In our case, we are planning to capture a lot of information from the user interaction given that the application domain is localised and that there are a lot of subscribed users.
- Ontology interoperability is one of the keys of the semantic web vision. The DOLCE semantics [31,32] provides a framework to integrate knowledge of different kinds of ontologies. We are planning to align our ontologies under this framework. Another topic for future research is the connection between ontologies [33]

Acknowledgements

We want to thank the members of the development: Iván Frade, Jose María Álvarez, Emilio Rubiera, Miguel Cuevas, Natalia Cueto and Manuel Cañón, and the other researchers from the University of Oviedo: Enrique del Teso, Guillermo Lorenzo, Roger Bosch and Agustín Cernuda del Río.

References

- [1] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protégé OWL plugin: An open development environment for semantic web applications. In *Lecture Notes in Computer Science*, volume 3298, pages 229–243, January 2004.
- [2] Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language overview. Technical report, W3C, 2004.
- [3] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
- [4] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
- [5] Dan Sperber and Deirdre Wilson. *Relevance: Communication and cognition*. Harvard University Press and Oxford: Blackwell, 1986.
- [6] Cristiano Rocha, Daniel Schwabe, and Marcus Poggi de Aragão. A hybrid approach for searching in the semantic web. In *WWW*, pages 374–383, 2004.
- [7] F. Crestani. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, (11):453–482, 1997.
- [8] Sean Bechhofer, Raphael Volz, and Phillip W. Lord. Cooking the Semantic Web with the OWL API. In *International Semantic Web Conference*, pages 659–675, 2003.
- [9] Otis Gospodnetić and Erik Hatcher. *Lucene in Action*. Manning, 2005.
- [10] Deepak Alur, John Crupi, and Dan Malks. *Core J2EE Patterns: Best Practices and Design Strategies*. Sun Microsystems, 2003.

- [11] Kent Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 1999.
- [12] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
- [13] Mei Kobayashi and Koichi Takeda. Information retrieval on the web. *ACM Computing Surveys*, 32(2):144–173, 2000.
- [14] Sara Comai, Ernesto Damiani, and Letizia Tanca. Semantics-aware querying in the WWW: The WG-Log web query system. In *ICMC'S*, Vol. 2, pages 317–322, 1999.
- [15] B. Berendt, A. Hotho, and G. Stumme. Towards semantic web mining. In I. Horrocks and J. Hendler, editors, *International Semantic Web Conference (ISWC 2002)*, 2002.
- [16] Sean Bechhofer, Les Carr, Carole A. Goble, Simon Kampa, and Timothy Miles-Board. The semantics of semantic annotation. In *On the Move to Meaningful Internet Systems*, volume 2519 of *Lecture Notes in Computer Science*, pages 1152–1167. Springer, 2002.
- [17] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal C Doshi, and Joel Sachs. Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management*. ACM Press, 2004.
- [18] Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov. KIM: a semantic platform for information extraction and retrieval. *Nat. Lang. Eng.*, 10(3-4):375–392, 2004.
- [19] David Huynh, Stefano Mazzocchi, and David Karger. Piggy bank: Experience the semantic web inside your web browser. In *International Semantic Web Conference (ISWC)*, 2005.
- [20] Yonggang Qiu and Hans-Peter Frei. Concept-based query expansion. In *Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval*, pages 160–169, Pittsburgh, US, 1993.
- [21] R. Benjamins, P. Casanovas, A. Gangemi, and B. Selic, editors. *Law and the Semantic Web*, volume 3369. Lecture Notes in Artificial Intelligence, 2005.
- [22] Luca Gilardoni, Christian Biasuzzi, Massimo Ferraro, Roberto Fonti, and Piercarlo Slavazza. Lkms - a legal knowledge management system exploiting semantic web technologies. In *International Semantic Web Conference*, pages 872–886, 2005.
- [23] Joost Breuker, Abdullatif Elhag, Emil Petkov, and Radboud Winkels. Ontologies for legal information serving and knowledge management. In Trevor Bench-Capon, Aspassia Dascalopulu, and Radboud Winkels, editors, *Legal Knowledge and Information Systems*. IOS Press, 2002.
- [24] N. Ide and J. Veronis. Word sense disambiguation. *Special Issue on Computational Linguistics*, 24(1), 1998.
- [25] Sylvie Despres and Sylvie Szulman. Construction of a legal ontology from a european community legislative text. In T. Gordon, editor, *Legal Knowledge and Information Systems*. IOS Press, 2004.
- [26] José Sáias and Paulo Quaresma. Semantic enrichment of a web legal information retrieval system. In Trevor Bench-Capon, Aspassia Dascalopulu, and Radboud Winkels, editors, *Legal Knowledge and Information Systems*. IOS Press, 2002.
- [27] D. Ravishankar, K. Thirunarayan, and T. Immaneni. A modular approach to document indexing and semantic search. In ACTA Press, editor, *Web Technologies, Applications, and Services*, pages 494–500, 2005.
- [28] Piek Vossen. Eurowordnet, general document. Technical report, University of Amsterdam, 1999.
- [29] H. Cui, J. Wen, J. Nie, and W. Ma. Query expansion by mining user logs. *IEEE Transaction on Knowledge and Data Engineering*, 15(4):829–839, July 2003.
- [30] Erika F. Lima and Jan O. Pedersen. Phrase recognition and expansion for short, precision-biased queries based on a query log. In *ACM SIG Information Retrieval*, pages 145–152, 1999.
- [31] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. *The WonderWeb Library of Foundational Ontologies (D18)*. Laboratory for Applied Ontology - ISTC-CNR, 2003.
- [32] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with dolce. In *EKAW*, 2002.
- [33] Bernardo Cuenca Grau, Bijan Parsia, and Evren Sirin. Working with multiple ontologies on the semantic web. In *Proceedings of the 3th International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture Notes in Computer Science*. Springer, 2004.

Semantics Driven Development of Software Systems Based on Business Ontologies

Keiichi Kondo^{a,1}, Shogo Hoshii^{a,1}, Takeshi Morita^{a,1}, Takahira Yamaguchi^{a,1},
 Noriaki Izumi^{b,2} and Kōiti Hasida^{b,2}

^a*Department of Administration Engineering, Keio University, Japan*
^b*National Institute of Advanced Industrial Science and Technology (AIST), Japan*

Abstract. For users and developers to form agreements about software system development, we propose a software development method that combines agile software methodologies and knowledge sharing technologies based on business ontologies. These business ontologies enable users and developers to form agreements about system specification by using the ontologies to address three aspects of business application for local governments: legal aspects, business processes, and software design. Our method specifically examines construction of domain models based on business ontologies of the early stage of development to reduce the cost of describing specifications. To recover the completeness of specifications, we propose a Web process architecture, which enables us to transfer a domain model to a design system specification. We have implemented a full environment using JAVA language, which enables us to generate software systems from a set of work processes.

Keywords. Business ontologies, Web application, enterprise software, software development

1. Introduction

Numerous software development methodologies have been proposed in the fields of information system analysis [2], design [9], and implementation [7]. Nevertheless, methodologies for improved sharing of information have not been forthcoming, such as requirements and specifications for consensus among users and developers. Ontologies have been discussed as an important facilitating technology for sharing and reusing knowledge among humans and computers [5][13]. To reuse business-related knowledge such as terminologies and requirement definitions, various repositories have been provided based on ontologies.

¹ Department of Administration Engineering, Keio University, 3-14-1 Hiyoshi Kohoku-ku, Yokohama-shi Kanagawa-ken 223-8522, Japan; E-mail:{keiichi, shogo, t_morita, yamaguti}@ae.keio.ac.jp

² National Institute of Advanced Industrial Science and Technology (AIST), 1-18-13 Sotokanda Chiyoda-ku Tokyo 101-0021, Japan; E-mail:{n.izumi, hasida.k@aist.go.jp}

From the viewpoint of the formality of requirement definitions, Enterprise Ontologies have been proposed [4][14] in the field of artificial intelligence. Because of their strong formality, they contribute to the reuse of business models. Simultaneously, they cover only general and abstract concepts for which it is extremely difficult to produce concrete business models with operability.

In the field of Management Science, a famous result is MIT's e-Business Process Handbook, called the Process Handbook for short [10]. The Process Handbook is a substantial contribution as a business repository. It contains approximately 4,600 definitions of business tasks from abstract ones to the specialized ones for business over the Internet. It is, however, not useful for automated machine processing because its aim is supporting human decision-makers so that task definitions are not required to be detailed or sufficiently formalized for machine comprehension.

The field of software engineering has fostered a blossoming of a new software methodology: agile methods. During the early stage of popularity of agile methods in the late 1990s, Extreme Programming (XP) gained attention. Consequent to XP, numerous methods were proposed and accepted such as SCRUM, CRYSTAL, and FDD.

The most immediate difference is that they are less document-oriented, but code-oriented: the key part of documentation is source code. There is, however, no clear guideline about documentation.

To support business application developments, several software libraries, frameworks, and CASE tools have been developed. Software libraries and frameworks provide us with off-the-shelf development that reduces the cost of application development.

However, no clear relationship exists between business models and software libraries. Although some CASE tools help us to develop business applications, the integrated process of business model management and application development is only slightly achieved so that the characteristics of business models seem to be lost in the developed business application.

To summarize the above, a gap exists between knowledge-oriented and software-oriented technologies: how to capture a business model and how to build a business application. In other words, system analysis methods do not foster agreement formation among users and developers.

This factor is expected to reflect a difference of formality levels of descriptions. To reduce system construction costs, a developer requires better accuracy regarding specifications. This renders development documents strict about formalism, by increasing the types of diagrams, and makes it difficult for non-developers to understand.

From the standpoint of the importance for users and developers to form common agreements about system specifications [1], we propose a software development method that combines agile software methodologies and knowledge-sharing technologies based on business ontologies. Business ontologies enable users and developers to form agreements about system specifications by producing ontologies that address three aspects of business applications for local government: legal aspects, business processes, and software design.

To reduce the costs of describing specifications, our method specifically emphasizes construction of domain models based on business ontologies in the early stage of development, instead of building terminologies, domain models, workflow models, and requirement specifications of software development [11].

This cost reduction measure results in some loss of completeness of specification descriptions. To recover completeness about specifications, we propose a Web process architecture that strictly links domain models and database schemas. Web process architecture frees us from detailed analyses about database table structures. It also constrains the software framework and module structures that are regarded as software architecture and feature structures of agile methods. As a result, business ontologies can guide us to transfer a domain model to a design specification. In addition, Web process architecture simplifies workflow models because Web page transitions can represent detailed business flows for users and developers. For validating systems, legal aspect ontology enables us to verify the soundness of the resultant system for actual business in actual law.

Because of the loss of some formalism of requirement specifications, especially about activity diagrams and sequence diagrams, some matters remain that are related to verification of systems. Business ontologies enable developers to make tests on small pieces of software because primitives of domain models, based on business domain ontology, have roles of features in Feature Driven Development (FDD). Consequently, the issue on verification is resolved by iterative testing of software in agile methods.

We have implemented a full environment using JAVA language, which enables us to generate software systems from a set of work processes. Our environment also supports the reuse of existing web-applications. Through application of the environment to case studies, we confirm that our proposed framework can integrate personal work processes into an enterprise application that reflects requirements of business processes using Web process execution.

2. Construction of Business Ontologies

2.1. Detailed Business on Medical Affairs of Local Governments

The target domain of this paper is the government business on managing a medical institute such as hospitals or clinics. An overview of the business is presented in the figure below.

The target business is to issue permission when a new hospital is planned for construction because a governor's permission (or approval) is required. The business is also to carry out acceptance procedures, such as a notification for change, abolishing and stoppage of the approved matter. Because a governor does not actually examine the contents of permission, the person

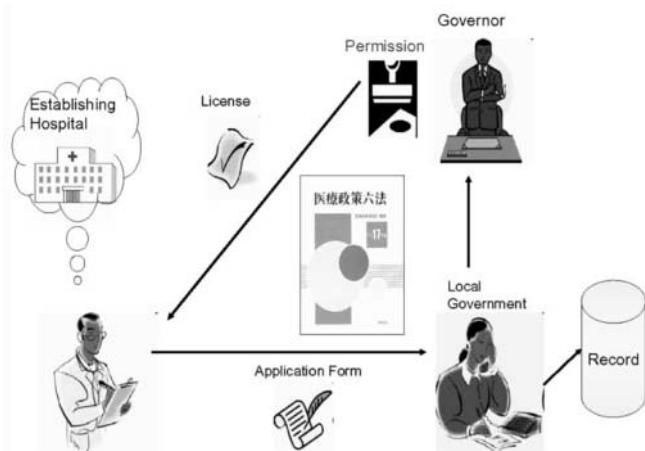


Figure 1. Application Domain

in charge of a local government carries it out.

In addition, the details of business and the standard of examination are prescribed by the Medical Service Law and its enforced regulations.

2.2. Architecture of Business Ontologies

Several phases exist in business application development, from business modeling to software development. In fact, we must define business tasks and their sequence (business process /workflow), to clarify why a local government must provide citizens with task execution as services (legal aspects), and to implement the detailed workflow as software application (application design) [8].

From the standpoint of the variety of the development phases, the business ontologies capture three different aspects of workflow system: legal aspects, business processes, and software design. The business ontologies are as follows:

- business domain ontology providing a structure of domain entities and their attributes,
- business task ontology, defining the *has-a* structure of work primitives, and
- legal aspect ontology, addressing the reason why a task should be provided.

In this study, users and developers are specifically dedicated to constructing domain model based on business ontologies: the task model consists of a sequence of business tasks. Our domain model is easier for users (non-developers) to understand. It is because domain model consists of primitives of domain ontology whose labelled node is assigned by glossary of business.

2.3 Business Domain Ontology Based on Document Form Analysis

This section describes how we construct business domain ontology from documents which local government uses.

2.3.1 Extraction of an application form to a constituent factor

First, we extract the entry matter of forms from the documents, collect them to produce a tabular form, and assign the data type of SQL.

2.3.2 Class division of constituent factors:

Second, we specify what holds each entry elements as attributes.

2.3.3 Investigation of a has-a relation

A conceptual inclusive relation is regarded as a *has-a* relation, which clarifies a dependency among conceptual classes.

2.3.4 Separation of roles from entities

Regarding a hospital or a clinic, numerous cases exist in which one person holds diverse posts of a civil servant, an administrator, and an attending physician. In such cases, the address of doctor A of the hospital H, the address of the establishment person of the hospital H, and the address of the administrator of the hospital H point to the same person. Presuming that doctor A moves and the address changes, an

establishment person's address must also change, along with an administrator's address. If this is described as an attribute of a different class, even though the doctor's address will be changed, an establishment person's address is not changed. As a result, attributes, such as the address and a name, are regarded as individual ones, and the doctor, the establishment person, and the administrator are regarded as role concepts which is dissociate from the individual entities. This view of the world is shown in Figure 2.

2.3.5. Investigation of a is-a relation

According to business manuals and application forms, the identified class concept in similar use introduces a higher rank class. Furthermore, the obtained is-a class is refined using DODDLE-J [15], which is a Japanese version of a semi-automatic ontology construction tool extract ontological concept from WordNet [6]. DODDLE-J uses the Japanese concept dictionary EDR [16].

2.3.6. Resultant business domain ontology

The ontology, which is obtained using the above processing, is shown below. The figure specifically depicts only

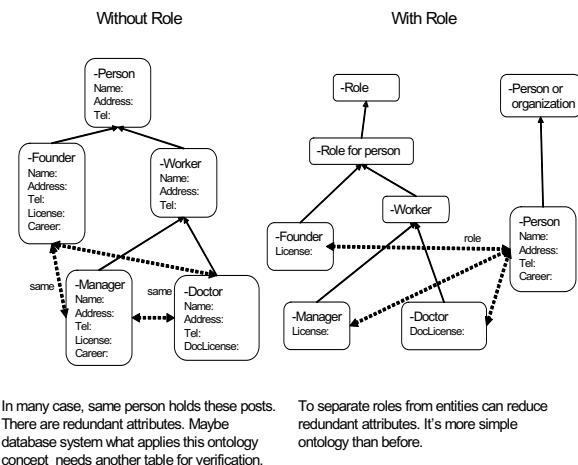


Figure 2. Separation of roles from entities

Example: Expanded and Simplified a Part

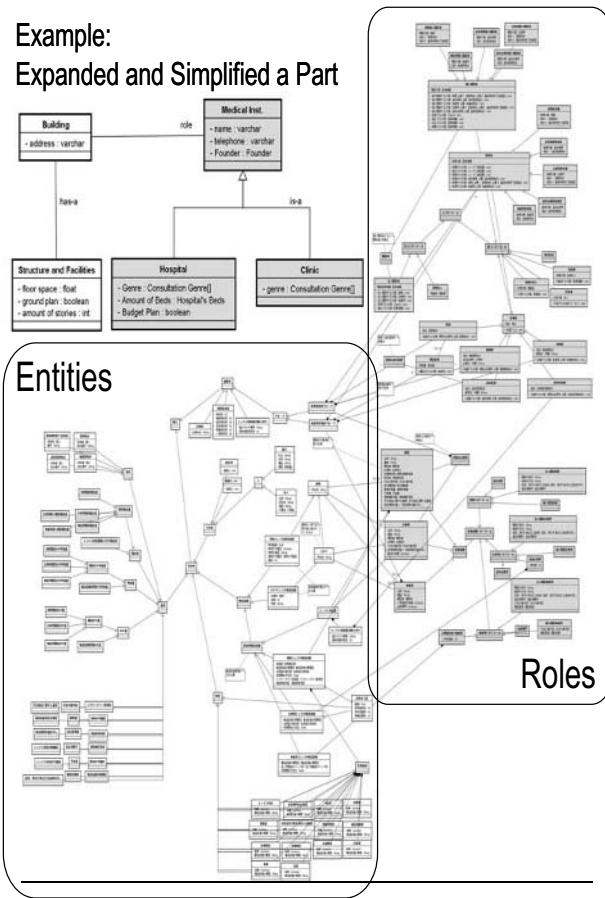


Figure 3. Obtained domain ontology

the task of establishment because it is too large to show in a single figure. In Figure 3, the colored class is a role concept and non-colored class is an entity. The figure shows 152 concepts.

2.4 Business Task Ontology Based on Web Process Architecture

In other words, business task serve two different roles in business models: unit of business process which is able to be composed, and start point of decomposition into the software features.

In this paper, by concentrating thin-client applications, we model the above vision about double aspects of a business task as a Web process architecture, as shown in Figure 4.

The double side view, as introduced above, is regarded as the difference between the human side and database side. The former side is taken as a single Web page transition with one-button action. This view is deduced so that a task may be captured, as portrayed in Figure 5.

The latter side is a unit of process that is disassembled into the detailed database functions shown in Figure 6.

According to the above vision, an example about local government business and its back-end system is illustrated in Figure 7.

In this example, a citizen might submit a registration document to a local government office. The office staff processes the submission using a Web-page like interface system that stores the record in the database.

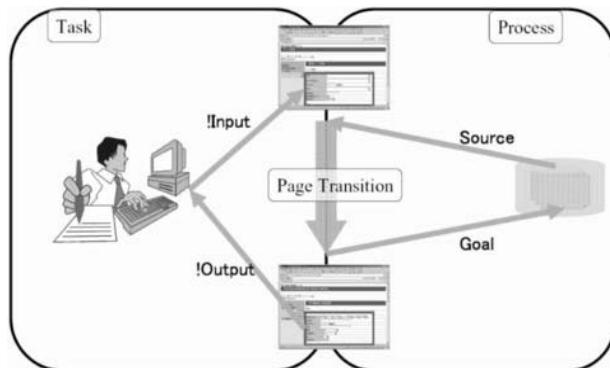


Figure 4. Web process architecture

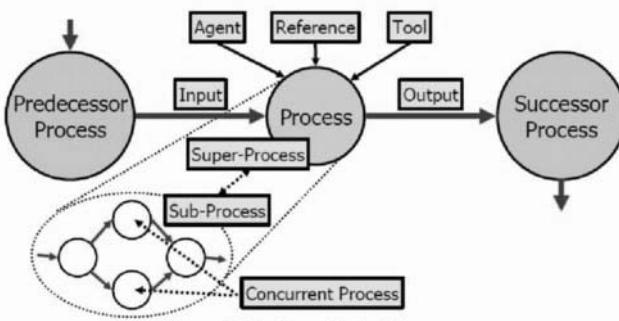


Figure 5. Business Task Schema

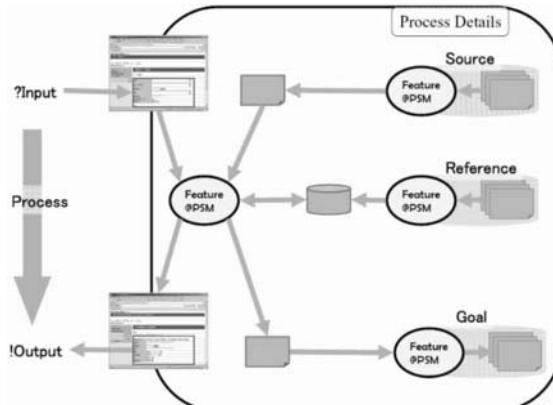


Figure 6. Detailed Web Process

2.5 Legal Aspect Ontology of Business Tasks

Identically to extracting domain ontology from application forms and documents, legal aspect ontology is obtained, from the Medical Service Law and its construction rules, as shown in Figure 8.

Through detailed analysis of the Medical Service Law and construction rules, the structure of legal aspect ontology differs from the scheme of business task ontology in the following respect: the legal primitive consists of a main part, super-primitive and sub-primitive, functional operator, term or period, conditions, and pre-primitive and post-primitive; the main part of all legal primitives comprises two sort of agent such as *doer* and *doer-ed*, object and operation.

Example: Registration
Outline of our study's object

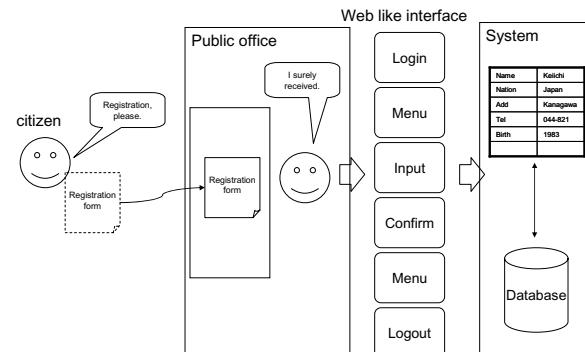


Figure 7. Example of Web Process



Figure 8. Resultant Structure of Legal Aspect Ontology

3. Semantic Feature Driven Development Based on Ontologies

3.1 Semantic Complement of Feature Driven Development

In this paper, we specifically describe Feature Driven Development (FDD) as a target development method whose semantics must be enhanced. In fact, FDD was developed by the long time Peter Coad. Similarly to other adaptive methodologies, it examines short iterations that deliver tangible functionality.

FDD has five processes. The first three are undertaken at the beginning of the project.

- Develop an Overall Model
 - Build a Features List
 - Plan by Feature
- The last two are performed during each iteration cycle.
- Design by Feature

- Build by Feature

Each process is broken down into tasks and is given verification criteria. From FDD, the only principle of feature selection is that the iterations are two weeks long.

In order to enhance the FDD semantics and formalism, we regard features as primitives of ontologies. This perspective enables us to reduce knowledge intensive work by shifting generating and testing features to selecting ontology primitives. Furthermore, selecting ontology primitives means that developers, who design feature lists, do not have to work making documents about the features. Documentation is the highest cost work in the development process because primitives and their combination must be familiar with both users and developers.

3.2 Model Construction and Specification Decomposition Based on Business Ontologies

In this section, we describe how business ontologies work and interoperate with Figure 9.

As explained above, by analyzing application forms and related documents, domain ontology is constructed as shown in the panel at the upper right of the above figure. Based on the domain ontology, business manuals of public office work are modeled as business task ontology, with a *has-a* structure, such as “Establish a Clinic” and “Establish a Hospital”. The Medical Service Law is formalized as a legal aspect ontology, with an *is-a* structure, shown in the bottom right of the figure. Each primitive task must be provided for reasons given by a node of the legal aspect ontology.

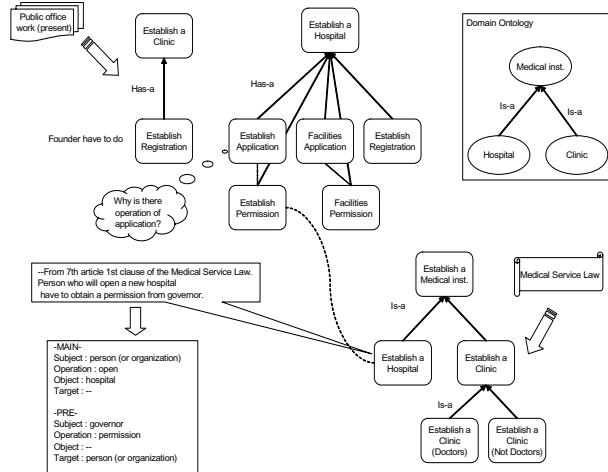


Figure 9. Overview of Business Ontologies

4. Proofing Experiment on Development of Local Government Application

We applied the proto-typing environment to case studies from the construction of business ontologies to building domain and task models based on application forms and business manuals that were provided to us by a local government on the condition of anonymity. The obtained sheet of Web process semantics is shown in Figure 10.

We compared implementations of case studies provided by our provided version and a former one, which is implemented as client-server system. Figure 11 is screenshots of this system.

Group	Feature	Source	test	Term	Condition	Action	Feature	Program	Note & Log	Name	Type
1	Group 1: Features										
2											
3											
4											
5											
6											
7											
8											
9											
10											
11											
12											
13											
14											
15											
16											
17											
18	1.Hand info database										
19	1.Hand info database										
20	1.Hand info database										
21	1.Hand info database										
22	1.Hand info database										
23	1.Hand info database										
24	1.Hand info database										
25	1.Hand info database										
26	1.Hand info database										
27	1.Hand info database										
28	1.Hand info database										
29	1.Hand info database										
30	1.Hand info database										
31	1.Hand info database										
32	1.Hand info database										
33	1.Hand info database										
34	1.Hand info database										
35	1.Hand info database										
36	1.Hand info database										
37	1.Hand info database										
38	1.Hand info database										
39	1.Hand info database										
40	1.Hand info database										
41	1.Hand info database										
42	1.Hand info database										
43	1.Hand info database										
44	1.Hand info database										
45	1.Hand info database										
46	1.Hand info database										
47	1.Hand info database										
48	1.Hand info database										
49	1.Hand info database										
50	1.Hand info database										
51	1.Hand info database										
52	1.Hand info database										
53	1.Hand info database										
54	1.Hand info database										
55	1.Hand info database										
56	1.Hand info database										
57	1.Hand info database										
58	1.Hand info database										
59	1.Hand info database										
60	1.Hand info database										
61	1.Hand info database										
62	1.Hand info database										
63	1.Hand info database										
64	1.Hand info database										
65	1.Hand info database										
66	1.Hand info database										
67	1.Hand info database										
68	1.Hand info database										
69	1.Hand info database										
70	1.Hand info database										
71	1.Hand info database										
72	1.Hand info database										
73	1.Hand info database										
74	1.Hand info database										
75	1.Hand info database										
76	1.Hand info database										
77	1.Hand info database										
78	1.Hand info database										
79	1.Hand info database										
80	1.Hand info database										
81	1.Hand info database										
82	1.Hand info database										
83	1.Hand info database										
84	1.Hand info database										
85	1.Hand info database										
86	1.Hand info database										
87	1.Hand info database										
88	1.Hand info database										
89	1.Hand info database										
90	1.Hand info database										
91	1.Hand info database										
92	1.Hand info database										
93	1.Hand info database										
94	1.Hand info database										
95	1.Hand info database										
96	1.Hand info database										
97	1.Hand info database										
98	1.Hand info database										
99	1.Hand info database										
100	1.Hand info database										
101	1.Hand info database										
102	1.Hand info database										
103	1.Hand info database										
104	1.Hand info database										
105	1.Hand info database										
106	1.Hand info database										
107	1.Hand info database										
108	1.Hand info database										
109	1.Hand info database										
110	1.Hand info database										
111	1.Hand info database										
112	1.Hand info database										
113	1.Hand info database										
114	1.Hand info database										
115	1.Hand info database										
116	1.Hand info database										
117	1.Hand info database										
118	1.Hand info database										
119	1.Hand info database										
120	1.Hand info database										
121	1.Hand info database										
122	1.Hand info database										
123	1.Hand info database										
124	1.Hand info database										
125	1.Hand info database										
126	1.Hand info database										
127	1.Hand info database										
128	1.Hand info database										
129	1.Hand info database										
130	1.Hand info database										
131	1.Hand info database										
132	1.Hand info database										
133	1.Hand info database										
134	1.Hand info database										
135	1.Hand info database										
136	1.Hand info database										
137	1.Hand info database										
138	1.Hand info database										
139	1.Hand info database										
140	1.Hand info database										
141	1.Hand info database										
142	1.Hand info database										
143	1.Hand info database										
144	1.Hand info database										
145	1.Hand info database										
146	1.Hand info database										
147	1.Hand info database										
148	1.Hand info database										
149	1.Hand info database										
150	1.Hand info database										
151	1.Hand info database										
152	1.Hand info database										
153	1.Hand info database										
154	1.Hand info database										
155	1.Hand info database										
156	1.Hand info database										
157	1.Hand info database										
158	1.Hand info database										
159	1.Hand info database										
160	1.Hand info database										
161	1.Hand info database										
162	1.Hand info database										
163	1.Hand info database										
164	1.Hand info database										
165	1.Hand info database										
166	1.Hand info database										
167	1.Hand info database										
168	1.Hand info database										
169	1.Hand info database										
170	1.Hand info database										
171	1.Hand info database										
172	1.Hand info database										
173	1.Hand info database										
174	1.Hand info database										
175	1.Hand info database										
176	1.Hand info database										
177	1.Hand info database										
178	1.Hand info database										
179	1.Hand info database										
180	1.Hand info database										
181	1.Hand info database										
182	1.Hand info database										
183	1.Hand info database										
184	1.Hand info database										
185	1.Hand info database										
186	1.Hand info database										
187	1.Hand info database										
188	1.Hand info database										
189	1.Hand info database										
190	1.Hand info database										
191	1.Hand info database										
192	1.Hand info database										
193	1.Hand info database										
194	1.Hand info database										
195	1.Hand info database										
196	1.Hand info database										
197	1.Hand info database										
198	1.Hand info database										
199	1.Hand info database										

ER-diagrams and sequence diagrams. This specificity of use might present some disadvantages related to less completeness of specification description. In fact, the concrete nature of Web process architecture recovers the loss from specificity because of two salient features:

- primitives of domain ontologies reduce the cost of database scheme design because the separation of roles from entities makes free us from construction of cooperation modules for distributed records;
- The Web page transition is almost unambiguous. Consequently, agreement formation among users and developers progresses smoothly.

In conclusion, from the standpoint of importance for humans to share agreement, to find consensus, about system development, we propose a software development method that combines agile software development methods to ontology technology. Our proposed ontologies enable users and developers to emphasize construction of domain models based on business ontologies on the early stage of development instead of building terminologies, domain models, workflow models and requirement specifications to facilitate software development. Through prototype development and verification experiments of Japanese local government, we confirmed that our proposed method supports us in various types of domains with Web processes. We are re-organizing our product to make it available to the public.

References

- [1] Business Process Management Tutorials, BPM Product, <http://bpmtutorial.com/BPM/BPM-Product-Directory.aspx?ref=aw>
- [2] Malu Casati, Mehmet Sayal, and Umeshwar Dayal: "Challenges in Business Process Analysis and Optimization", C.Bussler and M.C.Chan (Eds): TES 2005, LNCS 3811, pp.1-10,2006
- [3] J. Conallen: "Modeling Web Application Architectures with UML", Communications of the ACM (42:10), pp.63-70, 1999.
- [4] Jan L. G. Dietz: "Enterprise Ontology –Theory and Methodology-", Springer-Verlag, Berlin Heidelberg, 2006.
- [5] Digital Enterprise Research Institute: <http://www.deri.org/>
- [6] C. Fellbaum ed: "WordNet", The MIT Press, 1998.
- [7] Gellersen, HW., and M. Gaedke. (1999) "Object-oriented Web Application Development", *IEEE Internet Computing* (3:1), pp.60-68.
- [8] N.Izumi, T. Yamaguchi: "Integration of Heterogeneous Repositories Based on Ontologies for EC Applications Development", International Journal of Electronic Commerce Research and Applications, vol.1, no.1, pp.77-91, (2002)
- [9] Malhotra, Y. Enterprise Architecture: An Overview, URL <http://www.kmbook.com/enterarch.htm> (1996)
- [10] The MIT Process Handbook Project: <http://ccs.mit.edu/ph/>
- [11] Rossi, G., Schwabe, D., Lyardet, F. (1999) "Web Application Models are more than Conceptual Models", In Proceedings of the World Wide Web and Conceptual Modeling'99 Workshop, ER'99, pp.239-253, Springer, Paris.
- [12] Semantic Web Services: <http://www.daml.org/services/>
- [13] Rudi Studer, V. Richard Benjamins, and Dieter Fensel1. Knowledge Engineering: Principles and Methods, Data Knowledge Engineering, 25, 1-2, 161–197, 1998.
- [14] M.Ushold, et al: The Enterprise Ontology, Knowledge Engineering Review, Special Issue on Putting Ontologies, Vol.13
- [15] N.Suguri, Y.Shigeta, N.Fukuta, N.Izumi and T.Yamaguchi:Towards On-the-fly Ontology Construction - Focusing on Ontology Quality Improvement,1st European Semantic Web Symposium, LNCS3053, pp.1-15 , 2004.5
- [16] National Institute of Information and Communications Technology : EDR Electronic Dictionary Technical Guide ,URL http://www2.nict.go.jp/kk/e416/EDR/ENG/E_TG/E_TG.html

Description of Temporal Constraints Using Semantic Web in Role-Based Access Control

Kazushi Tanihira^{1,2}, Yusuke Sakamoto², Hiromi Kobayashi²

¹ Open System Technology Co.,Ltd.

² Department of Information Media Technology, Tokai University
1117 Kita-kaname, Hiratsuka 259-1292, Japan

Abstract. This paper presents a description of an access control list (ACL) used to conduct role-based access control (RBAC), with particular emphasis on temporal constraints using extensible markup language (XML) and Web ontology language (OWL). It is natural to use OWL for describing ACL because OWL is based on set theory. Set theory is useful in this case because RBAC maps privileges naturally to users from the viewpoint of users' functions. Comparison of XML and OWL in terms of their use as an ACL is described. A teaching management model is used to illustrate examples of these implementations.

1. Introduction

Access control management has become increasingly difficult because of the unified concept of distributed computing systems, which typically include heterogeneous platforms. To overcome this difficulty, RBAC has been proposed as a user-centric access control model [2–7]. It is an authorization model that is centered on the role of a user's business. For that reason, it can map authorization flexibly to the association of user-role and role-permission (or privilege). Consequently, RBAC is expected to solve security problems that arise from a mismatch of authorization mechanisms between required functions of users in business and an OS or DBMS.

For Web-based applications, XML [12] has garnered attention and has been studied to describe an access control list (ACL) for independence of a policy description component in RBAC. First, we present a description of ACL using XML. Particularly, we give a description and implementation of temporal constraint assignments using XML in RBAC because these have not been studied sufficiently [2–7]. Subsequently, we present a description of ACL that includes temporal constraints using OWL, which is a more expressive modeling language for processing Web content information [13–19]. It can express classes and relations between them based on set theory. A teaching management model is adopted to develop these prototype systems.

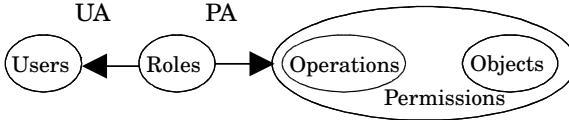


Figure 1. Core RBAC relationships

2. RBAC and Temporal Constraints

Core RBAC [2-3] is an essential component of the RBAC model. It consists of three administrative elements called users, roles, and permissions (or privileges). Moreover, permissions comprise operations and objects, as shown in Figure 1. The central concept to the RBAC is the role. Users and permissions do not connect directly; instead, they associate via roles as mediators. Users and permissions can be associated flexibly using roles of users. Constraints can be given not only to roles. User-role and role-permission assignments can also be constrained. For instance, permissions are not given to individual employees. They are given to roles: in a banking system, such roles would be a teller or accounting-manager. The Core RBAC model can be defined as follows, except for subjects [2].

Definition 1 Core RBAC model

Let $USERS$, $ROLES$, and $PRMS$ represent sets of users, roles, and permissions, respectively. In addition, let UA and PA represent the many-to-many user-to-role assignment and permission-to-role assignment, respectively. The assignment of users and roles is a many-to-many mapping that can be expressed as follows.

$$UA \subseteq USERS \times ROLES$$

Mapping role r onto a set of users is

$$\text{assigned_users}(r : ROLE) \rightarrow 2^{USERS}.$$

Therefore, it is expressed as follows.

$$\text{assigned_users}(r) = \{u \in USERS \mid (u, r) \in UA\}.$$

The assignment of permissions and roles is a many-to-many mapping that can be expressed as follows.

$$PA \subseteq PRMS \times ROLES$$

Mapping role r onto a set of permissions is

$$\text{assigned_permissions}(r : ROLE) \rightarrow 2^{PRMS}.$$

Consequently, it is expressed as follows.

$$\text{assigned_permissions}(r) = \{p \in PRMS \mid (p, r) \in PA\}$$

When dividing $PRMS$ into OPS and OBJ , denoting sets of operations and objects, respectively, then $PRMS = 2^{OPS \times OBS}$.

Description and implementation of temporal constraints have been studied in several fields [8-11]. Nevertheless, these have not been studied sufficiently in RBAC yet. We use the following three patterns that are commonly used in daily business in this paper. Note that we don't intend to propose temporal patterns

but to study description and assignment of temporal constraints using XML and OWL in RBAC.

Pattern 1 $[T_1, T_2]$

Pattern 2 $([T_1, T_2] t_1 - t_2)$

Pattern 3 $([T_1, T_2] W\{Weeks\} t_1 - t_2$

Plus $\{\{Days\} t_3 - t_4\}$

Minus $\{\{Days\}\}$)

where T_1 denotes an enabling start time, T_2 denotes a disabling stop time, t_1 denotes a service duration start time, t_2 denotes a service duration stop time, and *Weeks* denotes one or more of $|SU|MO|TU|WE|TH|FR|SA|$, representing Sunday through Saturday.

Take an example, the period from 09:00 to 17:00 on every Monday and Wednesday between January 1, 2004 and July 31, 2004, except January 3 and January 5, and that from 13:00 to 16:00 January 2 and from 13:00 to 17:00 January 4 are represented as follows.

$$([Jan - 1 - 2004, July - 31 - 2004] \\ W\{MO, WE\} 09 : 00 - 17 : 00 \\ Plus\{\{Jan - 2 - 2004\}13 : 00 - 16 : 00 \\ \quad \{Jan - 4 - 2004\}13 : 00 - 17 : 00\} \\ Minus\{Jan - 3 - 2004, Jan - 5 - 2004\})$$

3. Teaching Management Model

We describe briefly a teaching management model using the Internet for ease of later explanation.

“The teaching operation is performed through roles of persons called lecturers, teaching assistants, registrants, and educational staff. First, course names are registered by educational staff. Next, syllabi and contents, including exercises, are set by lecturers along with examinations. The progress of a course in a certain period of time is dependent upon respective registrants. Submission of exercises and examinations are also dependent upon respective registrants. Checking of exercises is performed by teaching assistants; marking of examinations and grading of results are performed by lecturers. Each registrant can see only their own grade. In addition, checking the status of setting syllabi, contents, and grading are performed by educational staff as the occasion demands”.

One of the advantages of RBAC is that users can easily join in access control analysis or design phases because their business terminology can be used as roles and abstract permissions.

Roles must first be defined in terms of RBAC. Because roles can possess different temporal constraints according to courses, each role is needed on every course, except educational staff, who are common to all operations. Roles are classified into

- course lecturer(s),
- course teaching assistant(s) (TA),
- course registrant(s), and
- educational staff.

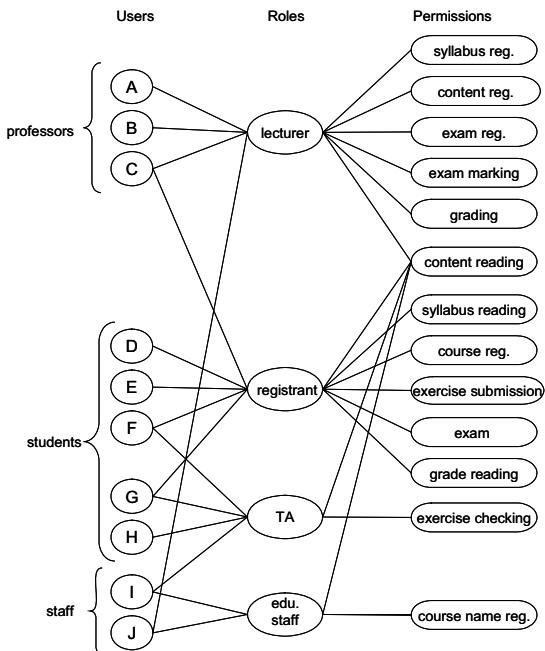


Figure 2. A part of user-role-permission assignment in teaching management model

Note that lecturers above are not representatives of a position in this school; they are lecturers in terms of their functions in teaching.

Figure 2 shows a part of an example of the association of users, roles, and permissions where many-to-many mapping is formed. Let a member of the faculty staff and a part-time professor become a lecturer of a course, and a staff member (except for a professor) and a student become a teaching assistant (TA) of a course. In addition, let a student and a member of the faculty staff become a registrant of a course.

4. Description Using XML

First, we developed this teaching management model using XML. RBAC is a role-centric access control [1]. An ACL is composed of description of roles, users, permissions, user-role assignments, and role-permission assignments. Let → denotes mapping, user-role assignments and role-permission assignments can be described *roles* → *users* and *roles* → *permissions*, respectively.

The description of these is composed of

- (1) user names in every role
- (2) permission names in every role

in ACL. The following is an example of user-role assignment.

Table 1. Tags in BTML

tag name	explanation
TimeConstraints	area of temporal constraints
enablestart	role enable start time
enablestop	role enable stop time
durationstart	duration of service start time
durationstop	duration of service stop time
weekpattern	week(s) applied for weekpattern
plus	additional day(s) [and time] to weekpattern
minus	subtraction day(s) from weekpattern

```

< UserRoleAssignment >
  < role > PCL < /role >
  < user > T0001 < /user >
  < user > T0002 < /user >
  < user > T0003 < /user >
< /UserRoleAssignment >

```

A checking order of input data with a ACL is dependent on the easiness of searching in the ACL; therefore, it is described as follows.

(1) role, (2)permission, (3) user, (4) role → user, (5) role → permission.

An application can be transacted automatically using XML tags, each of which is capable of having any name with a text-based description. Temporal constraints can be denoted using tags called basic temporal markup language (BTML) which includes a minimum set of tags to represent above temporal patterns is defined in Table 1. These tags are defined to express temporal constraints. Every temporal constraint is described between *< TimeConstraints >* and *< /TimeConstraints >*. Pattern 2 can be described as follows.

Pattern 2 ($[T_1, T_2]$ $t_1 - t_2$)

```

< TimeConstraints >
  < enablestart > T1 < /enablestart >
  < enablestop > T2 < /enablestop >
  < durationstart > t1 < /durationstart >
  < durationstop > t2 < /durationstop >
< /TimeConstraints >

```

BTML is used to describe ACL for independence of a policy description component. Java and a document object model (DOM) [12] as an application programming interface (API) are used to construct a checker (or, parser) for an access enforcement. Figure 3 shows the RBAC composition concept. ACL is described using BTML. Access control is conducted based on the description of ACL.

A part of the ACL description for a teaching management model using BTML is shown in Figure 4. In the first line, role ID *TAL* is of “teaching assistant of course Algorithm.” In the 11th line, role *LCP* is an abbreviation of “lecturer of course C programming,” and in the 13th line, privilege *SYR* is an abbreviation of “syllabus registration.”

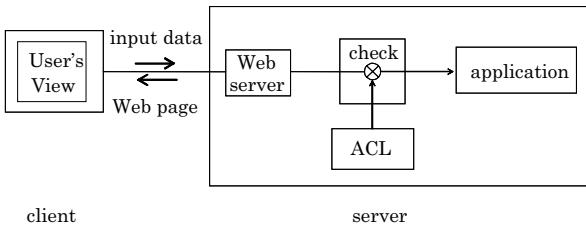


Figure 3. RBAC composition

```

<role roleID="TAL" rolename="Algorithm TA">
    <TimeConstraints>
        <enablestart>2004-04-01T08:00</enablestart>
        <enablestop>2004-07-31T18:00</enablestop>
            <durationstart>09:00</durationstart>
            <durationstop>17:00</durationstop>
        </TimeConstraints>
    </role>
    .....
    <RolePrivilegeAssignment>
        <role>LCP</role>
        <privilege>
            SYR
            <TimeConstraints>
                <enablestart>2004-02-01T08:00</enablestart>
                <enablestop>2004-03-20T08:00</enablestop>
            </TimeConstraints>
        </privilege>
        .....
        <privilege>
        .....
    </RolePrivilegeAssignment>

```

Figure 4. Example of ACL in teaching management model

5. Description Using OWL

Next, we developed a prototype of this teaching management model using OWL. An ontology description language, OWL is used for automated processing of Web content information using machines [13–19]. The inclusion relationship of resource sets can be represented explicitly using OWL. In addition, we used XML, which is a universal meta-language for defining markup to documents. However, XML does not provide meaning by itself; each application interprets its meaning. A resource description framework (RDF) and RDF schema (RDFS) were developed to express data models and vocabulary to overcome this problem. However, the expressivity of RDF/RDFS is limited. Therefore, OWL was developed as an extension of RDFS for more expressive power and efficient reasoning support.

The OWL syntax is usually represented as an RDF/XML style using tags. The following is an example of a user-role assignment with temporal constraints.

```

<owl:ObjectProperty rdf:id = "RoletoUser"/>
<owl:ObjectProperty rdf:id = "RoletoUserProfessor">
  <rdfs:domain rdf:resource = "#ProgrammingLecturer"/>
  <rdfs:range rdf:resource = "#Professor"/>
  <rdfs:subPropertyOf rdf:resource = "#RoletoUser"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource = "#hasTemporalConstraints"/>
      <owl:hasValue rdf:resource = "#TempConstraint2"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:ObjectProperty>

<owl:Class rdf:id = "TempPattern1">
  <owl:intersectionOf rdf:parseType = "Collection">
    <owl:Restriction>
      <owl:onProperty rdf:resource = "#enablestartTime"/>
      <owl:allValuesFrom rdf:resource = "&xsd;dateTime"/>
    </owl:Restriction>
    :
  </owl:intersectionOf>
</owl:Class>

<TempPattern1 rdf:id = "TempConstraint2">
  <enablestartTime rdf:resource = "2005-04-05T08:00:00"/>
  :
</TempPattern1>

```

A relationship of Class ProgrammingLecturer (a subsubclass of Role) and Class Professor (a subclass of User) is represented using ObjectProperty denoted by RoletoUserProfessor, which is a subproperty of RoletoUser. ObjectProperty RoletoUserProfessor contains Class ProgrammingLecturer as a domain and Class Professor as a range. In addition, this class has a temporal constraint value set called TempConstraint2 as a property. Class TempPattern1 consists of an enabling start time and other information. TempConstraint2 is an instance of Class TempPattern1.

Figure 5 shows part of a user-role-permission relationship example in a teaching management model. The OWL syntax is usually represented as an RDF/XML style using tags. However, a functional style syntax called abstract syntax [16] is occasionally used for compact writing convenience. A role to user assignment can be represented using an abstract syntax, as shown in Fig. 5. Part of this description can be visualized using an RDF graph visualization tool called RDF Gravity [20], as shown in Fig. 6. Note that URI resource nodes are omitted in this figure for simplicity.

We developed a checker of this teaching management system in which OWL is used to describe ACL, as mentioned above. Jena [21] and Pellet OWL Reasoner [22] are used to construct this checker. Jena is used as an API and Pellet OWL Reasoner as an external reasoner that is imported in Jena to construct this checker. Using this checker, an ACL is examined to determine whether or not user-role-permission data are matched. Temporal reasoning has not been

```

Class( User complete unionOf( User1 User2 ) )
Class( User1 complete oneOf( T001 T002 ) )
Class( Role )
Class( Lecturer complete
      unionOf( ProgrammingLecturer LiteracyLecturer )
      Role )
Class( ProgrammingLecturer partial
      restriction( hasTemporalConstraints value(TempConstraint1) ) )
Class( TempPattern1 complete
      intersectionOf(
          restriction( enableStartTime allValuesFrom( &xsd;dateTime ) )
          restriction( enablestopTime allValuesFrom( &xsd;dateTime ) ) ) )
ObjectProperty( RoletoUser )
ObjectProperty( RoletoUser1
    domain( ProgrammingLecturer )
    ranged( User1 )
    subPropertyOf( RoletoUser ) partial
    restriction( hasTemporalConstraints value(TempConstraint2) ) )

```

Figure 5. ACL example using abstract syntax of OWL

supported in Pellet. However, temporal constraints are only used for checking whether input data satisfy temporal constraints using the date-time function of Java. A user can access an application when checking is successful.

6. Concluding Remarks

This paper presented a description of ACL using XML and OWL, especially emphasizing temporal constraint assignments in RBAC. A teaching management model was used to present examples of description and implementation. First, an ACL was described using XML; a checker was developed using DOM as a parser. Second, an ACL was described using OWL and a checker was constructed using Jena and the Pellet OWL Reasoner as a reasoning engine. The merit of OWL description is its ease of temporal constraint inclusion. A temporal constraint can be added easily as restrictions in plural classes or object properties that represent relationships among classes. Moreover, the same temporal pattern with different values can be represented as a class. Patterns can be included easily using OWL. In addition, a semantic update history of ACL can be retained using OWL. Another merit of OWL description is its strict definition of data structure and utility of a common vocabulary. It is easy to understand roles based on set theory within a company or institution from the viewpoint of users. Strict definition of RBAC data structure reduces errors in ACL. Maintenance of the ACL is frequently needed in actual systems. Maintainability can be improved using a common vocabulary. We described ACL using XML and OWL manually; however, an input support tool is needed for actual application development. Therefore, further work is needed to produce an input support tool for ACL in RBAC.

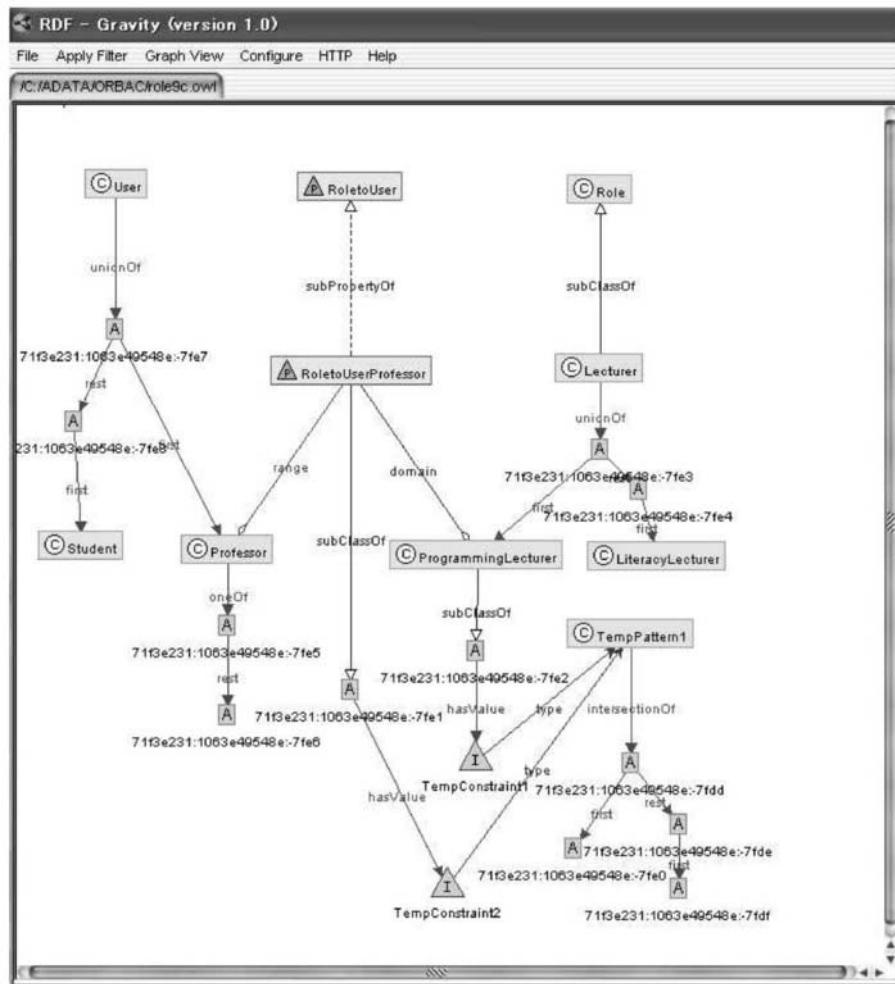


Figure 6. RDF Graph Visualization

References

- [1] K. Tanihira and H. Kobayashi, Properties of Role-Based Access Control in a Teaching Management System, *IEICE Trans. Information and Systems*, E-88-D(10), pp.2417-2421, 2005.
- [2] D.F. Ferraiolo, D.R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, Artech House, London, 2003.
- [3] R.S. Sandhu, E.J. Coyne, H.L. Feistein, and C.E. Youman, Role-Based Access Control Models, *IEEE Computer*, 29(2), pp.38-47, 1996.
- [4] M. Hauf, J. Schwarz, and A. Polze, Role-Based Security for Configurable Distributed Control Systems, *Proc. IEEE Object-Oriented Real-Time Dependable Systems (WORDS) 2001*, pp.111-118, 2001.
- [5] E. Bertino, P.A. Bonatti, and E. Ferrani, TRBAC: A Temporal Role-Based Access Control Model, *ACM Trans. on Information and System Security*, 4(3), pp.191-223, 2001.
- [6] J.B.D. Joshi, E. Bertino, and A. Ghafoor, Temporal Hierarchies and Inheritance Semantics for GTRBAC, *Proc. SACMAT'02*, pp.74-83, 2002.
- [7] R. Bhatti, E. Bertino, A. Ghafoor, and J.B.D. Joshi, XML-Based Specification for Web Services Document Security, *IEEE Computer*, 37(4), pp.41-49, 2004.
- [8] D.M. Gabbay, I. Hokinson, and M. Reynolds, *Temporal Logic: Mathematical Foundations and Computational Aspects Vol.1*, Oxford Science Publications, 1994.
- [9] *Proc. Temporal Representations and Reasoning (Time-01)*, IEEE Computer Society, 2001.
- [10] R.T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*, Morgan Kaufman, 2000.
- [11] C. Bettini, S. Jajodia, and S.X. Wang, *Time Granularites in Databases, Data Mining, and Temporal Reasoning*, Springer, 2000.
- [12] XML Homepage: <http://www.w3c.org/XML>
- [13] OWL Overview: <http://www.w3c.org/TR/owl-features/>
- [14] OWL Guide: <http://www.w3c.org/TR/owl-guide/>
- [15] OWL Reference: <http://www.w3c.org/TR/owl-ref/>
- [16] OWL Semantics and Abstract Syntax: <http://www.w3c.org/TR/owl-absyn/>
- [17] A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering*, Springer, 2004.
- [18] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce: Second Edition*, Springer, 2004.
- [19] G. Antoniou and F. van Harmelen, *A Semantic Web Primer*, MIT Press, 2004.
- [20] RDF Gravity Homepage: <http://semweb.salzburgresearch.at/apps/rdf-gravity/> index.html
- [21] Jena Homepage: <http://jena.sourceforge.net/>
- [22] Mindswap Pellet Homepage: <http://www.mindswap.org/2003/pellet/demo.shtml>

Development and Analysis of a Problem Domain Knowledge Base Oriented to PLA Specifications

Henrikas PRANEVICIUS¹, Germanas BUDNIKAS

Business Informatics Department, Kaunas University of Technology, Lithuania

Abstract. The paper presents a technique that applies knowledge engineering techniques for development and analysis of PLA specifications. Problem domain knowledge base (KB) is created using the knowledge acquisition technique joined with a piece-linear aggregate (PLA) model. The production rules of the KB are represented by decision tables, and static properties of the KB are checked in Prologa system. Further, the KB is combined with the defined KB of validated properties and validation method, and KB dynamic properties are checked in the expert system in CLIPS. Analysed KB is used defining a framework of PLA specification using Praxis editor and supplementing Praxis generated framework with the application functional description. The technique is illustrated with an example of a hard disk drive cache.

Keywords. PLA specification, knowledge base, decision table, expert system, validation of static and dynamic properties, CLIPS, Prologa.

Introduction

One of the most important parts in the development of distributed systems is specification and analysis of problem domain descriptions. Possible errors at this stage should be corrected as early as possible to minimise cost of the development. Formal methods and specification languages are widely used for design of the distributed systems. The most popular formal specification languages being used for description of distributed systems are SDL, Lotos, Estelle [1, 2].

Specification language Estelle/Ag is based on Aggregate method [3]. There are some differences between Estelle/Ag and Estelle—the piece-linear aggregate (PLA) model is used in Estelle/Ag. The use of such a model instead of a finite-state automaton, which is the formal background of standard Estelle, enables to create models both for validation and simulation. This is possible due to the special structure of the PLA. Further in the paper we use notion of PLA specification as the one written in Estelle/Ag language.

A construction of specifications in this language is performed in two phases. Specification editor Praxis helps to define the specification framework that describes an interaction of specified system aggregates, their states and conditions of state change.

¹ Corresponding Author: Henrikas Pranevicius, Business Informatics Department, Kaunas University of Technology, Studentu 56-301, LT-51424 Kaunas, Lithuania; E-mail: hepran@if.ktu.lt.

Next, the framework is heuristically supplemented with the knowledge from a problem domain about behaviour of an analysed system.

However, usually application domain experts find difficult to understand and learn formal specification languages. Representation of knowledge used in knowledge-based systems (KBS) is close to the way of thinking by a human. The works where problem domain KBSs are used for creation of (formal) specifications include: acquisition of requirements and incremental evolution of specifications [4]; definition of requirements for transition from an informal problem description to formal specification in VDM language [5]; generation of PLA specification structure [3]; use of knowledge-based meta-model for integrating characteristics of software development models [6]; formulation of requirement specifications in domain concepts [7]; use of formalisation step for problem knowledge description [8]; data-base design by creating its specification [9] and definition of system requirements and analysis [10]. General parts of such techniques/tools are:

- The knowledge acquisition that may be realised using text translators (e.g. [4, 7, 9]), or assisting by Graphic User Interface (e.g. [3, 5, 10]);
- Transformation of intermediate discourse representation structures (like Modified ER diagrams [5]; restricted English [7], etc.) to a well-defined model (e.g. object-oriented model [6, 9]; first order predicates [7]) or target specification (e.g. PLA [3], VDM [5]).

A key feature of these techniques is the analysis of correctness that is performed both during the construction of the initial knowledge description (e.g. [8, 9]) and while a target model or KB has been created (e.g. [4, 6, 7, 10]).

This paper presents an approach that utilises techniques of knowledge representation as well as knowledge engineering for building up a knowledge base (KB) aimed at development of PLA specification and validation² of the KB static and dynamic properties.

The static properties are characteristics of a KBS that can be evaluated without its execution. Such an evaluation is often referred to as static verification. During static verification, a KB is checked for anomalies [12]. Preece and Shinghal [13] present a classification of the anomalies that may be present in rule-based systems. It is necessary to note the difference between an anomaly and error. The anomaly indicates the existence of a possible error. The dynamic properties are those characteristics of a rule-based system that can be evaluated only by examining how the system operates at a run time. The most common techniques of validation and verification that have been developed for use on KBS are identified in [14]. A detailed review of specific methods and supporting tools can be found in [13].

Many authors (e.g. Bruynooghe *et al.* [15], Schreiber *et al.* [16]) believe that the declarative style of description that is used in KBs is more understandable and acceptable than the procedural style (the latter is used in PLA specifications). This is because a problem is described at the knowledge level "at which the knowledge engineer specifies expertise during knowledge acquisition" [17]. Therefore, the knowledge description is presented not using strict mathematic notation but concepts of a problem domain that is natural. Due to this reason, we suppose that the creation of PLA specifications using KB is more attractive in that sense. Moreover, as it will be shown later, verification of general static properties at the knowledge level permits to

² In the paper, for the sake of brevity, sometimes we refer to validation having in mind both validation and verification because "validation subsumes verification" [11].

detect errors that were not detected when using validation techniques for PLA specifications.

Until now, knowledge-based techniques were not used for the creation of PLA specifications except Praxis system. However, this system generates structure of the specification only and does not define the behaviour of an application. Our approach does it by using KB.

An applicability of our technique is defined by the applicability of the PLA method and specifications. The technique is oriented to creation of PLA specifications which primary domain of usage is communication protocols and business systems. It has been successfully applied for such applications [18]: network of queuing system, alternating-bit protocol, Internet cache protocol (version 2.0). Effectiveness of the proposed technique is stipulated by these factors: ability to be applied for real-sized applications as well as check for additional properties at the knowledge level before PLA specification is created. The scalability of the proposed technique is limited by software tools that are used in creating the specification. The limitation requirements for these tools are defined in [10, 19].

The paper is structured as follows. Part 1 describes construction of the problem domain KB oriented to PLA specifications and analysis of its static and dynamic properties. It also provides a brief explanation how the specifications are created. Part 2 presents illustration of the technique. Relation of the proposed approach to some other works are given in part 3. Conclusions sum up the proposed approach.

1. Development and Analysis of KB Aimed at Creation of PLA Specifications

Such a knowledge base (to be referred to as KB_{Ag}) represents problem domain knowledge using production rules. This representation is chosen due to its similarity to PLA specification language constructions describing conditions for state change: when *condition* begin ... end. Since most of the common verification and validation problems in rule-based systems can be solved using decision tables (DT) [20] and the tabular verification method is computerised in Prologa system [21], in our approach static verification will be performed using this method. Therefore, in order to perform the static verification of the KB_{Ag} , its production rules have to be transformed to Prologa DTs. Moreover, as stated in [20], a DT is equivalent to a set of production rules.

Because the KB_{Ag} will be used for the creation of PLA specifications, it has to contain knowledge about the PLA model. To acquire this knowledge, we applied the knowledge acquisition technique [22].

The predicates and production rules are used for description of concepts and relations of PLA model. Applying our approach for specification of an application one has to use the defined predicates and production rules for problem representation. Thus, the KB_{Ag} is created by manually filling in the knowledge about the application as predicates and production rules of the defined form. Next, we present examples of predicates and productions of KB_{Ag} .

$\text{ArrivalOfInputSignal} (an_i, x_i^j, x_j^1, \dots, x_j^{c_{ic}})$, where an_i – symbolic name of the i th aggregate; $x_j^1, \dots, x_j^{c_{ic}}$ – components of signal x_i^j .

State ($an_i, w_i^1, \dots, w_i^{c_{cc}}, d_i^1, \dots, d_i^{c_{dc}}$), where $w_i^1, \dots, w_i^{c_{cc}}$ and $d_i^1, \dots, d_i^{c_{dc}}$ are the coordinates of continuous and discrete state components.

The production rule that describes state change and signal output after occurrence of an external event has the following general form:

IF ArrivalOfInputSignal ($an_i, x_i^j, x_j^1, \dots, x_j^{c_{ic}}$)
 and State ($an_i, w_i^1, \dots, w_i^{c_{cc}}, d_i^1, \dots, d_i^{c_{dc}}$)
 and Aux ($an_i, w_i^1, \dots, w_i^{c_{cc}}, d_i^1, \dots, d_i^{c_{dc}}$)
 THEN State ($an_i, w_i^{l*}, \dots, w_i^{c_{cc}*}, d_i^{l*}, \dots, d_i^{c_{dc}*}$)
 and OutputSignal ($an_i, x_i^j, x_j^1, \dots, x_j^{c_{jc}}$)

where predicate Aux describes auxiliary conditions that check values of coordinates of state components.

1.1. Static Verification of the KB_{Ag}

Most of the validation problems in rule-based systems like redundant, ambivalent, categorised, cyclic or missing rules, redundant conditions, unused action parts may be resolved using DTs [10].

Our technique uses DT representation for static verification of KB_{Ag} since DTs clearly demonstrate incompleteness and inconsistency of knowledge [21] and software tool Prolog that assists verification process. Anomalies in DTs have direct correspondence to anomalies in production rules, which make the KB_{Ag} .

Static Anomalies in Rule Bases and Decision Tables

The decision table DT is defined [21]:

$$DT: CS_1 \times CS_2 \times \dots \times CS_m \rightarrow AV_1 \times AV_2 \times \dots \times AV_n,$$

where CS_i is a set of condition states, AV_j is a set of action values.

The following anomalies are distinguished: *intra*-tabular, which occur in a single DT, and *inter*-tabular anomalies, that originate from interactions between several DTs [23]. A relation between *intra*-tabular anomalies and anomalies in KB_{Ag} rule base is presented in [18]. A classification of *inter*-tabular anomalies can be found in [24].

Transformation of KB_{Ag} Productions to Single Hit Decision Tables

In our technique, static verification of KB_{Ag} is performed in Prolog system. The system uses single hit DT representation. In a single hit table each possible combination of a condition can be found in exactly one and only one column.

The transformation to Prolog DTs is specific with respect to the PLA model and employs some of its concepts—aggregates, internal and external events, input and output signals, discrete state component coordinate, etc. Production rules are transformed to the DTs of certain groups thus enabling to fully exploit advantages of tabular representation to perform static verification. Next, we present an outline of suggested steps of transformation of KB_{Ag} productions to single hit DTs.

1. Productions that describe certain types of events are transformed to corresponding event tables;
2. Predicates of antecedent (consequent) part of a rule are written in a form of condition (action) subjects in a DT;

3. Decrease of a coordinate of discrete state component by a constant value $Const$ is marked with not ($d_i^j = d_i^j + Const$). This notation is used while ambivalence property is being checked.

Technique for Detection of Intra- and Inter-tabular Anomalies in KB_{Ag}

This technique is mostly based on works of [10, 21] and operating in Prologa system. Below we present a portion of this technique.

A *subsumed column pair* (a case of intra-tabular redundancy), which definition is as presented below [23]:

A DT contains a *subsumed column pair*, if and only if it includes a pair of columns $(CS_j; AS_j)$, $(CS_k; AS_k)$ ($1 \leq j, k \leq t$, $j \neq k$ for which: $CS_j \subseteq CS_k$ and $AS_j \supseteq AS_k$,

in a single hit DT is represented by single column that corresponds to several KB_{Ag} productions. Thus, the anomaly of the subsumed column pair is detected if several KB_{Ag} productions correspond to the same DT column.

Full description of the technique for anomaly checking in KB_{Ag} as well as transformation steps of KB_{Ag} productions to single hit DTs are presented in [18, 25].

1.2. Functional Validation of KB_{Ag}

Functional validation of the problem domain KB is carried out using the expert system in CLIPS (C Language Integrated Production System). CLIPS is a tool for productive development and delivery of expert systems [19]. It is built by combining already verified problem domain KB with a KB of validated properties and validation method (KB VPVM) [26].

The defined KB VPVM, which is implemented in CLIPS, may be used for various kinds of applications. The instances of dynamic properties whose validation is implemented in the KB VPVM are the absence of static deadlocks, final state reachability, boundedness, and completeness. The reachable state validation method is realised in the KB VPVM. In a view of analysis of the execution paths, this method is similar to functional validation method suggested by Preece *et al.* [11] that analyses the sequences of rules that must fire to achieve a goal.

When performing the join of KB_{Ag} with KB VPVM, the needed adaptation changes are minor—they are an adaptation of description of validated properties for a specific application. For instance, in order to check the boundedness property, individual bounds on discrete state component coordinates have to be defined. Having combined the KB VPVM with the KB_{Ag} , the expert system (ES) in CLIPS is built. The ES uses CLIPS inference engine that is based on the forward chaining strategy.

In order to perform the functional validation using the ES, the initial and the final states of an analysed application are defined. An application model operates according to the reachable states method. If the validated properties are violated, the expert system generates a corresponding report and a designer corrects the KB accordingly.

1.3. Creation of PLA Specifications Using KB_{Ag}

In the first step, specification framework is constructed during a session with the specification editor Praxis [3] using the knowledge extracted from the validated and verified problem domain KB. At the next step, the generated by Praxis specification

Table 1. KB_{Ag} fragment of HDD cache.

Description fragment	KB _{Ag} predicates and rules
State of the aggregate <i>Cache</i> is characterised by predicate State:	
File search in cache, read from a disk, read from a cache, partial read from the disk and cache operations, search result and sought file size.	State(Cache, search, read_disk, read_cache, pread_disk, pread_cache, search_res, file_size)
For the briefness, afore defined predicate will be denoted as State_C.	
A program requests a file. At the first, the file is searched in cache memory.	IF ArrivalOfInputSignal (Cache, Request, FileName) and State_C THEN search* = True and State_C
If the search fails, the file is read from a disk.	IF EndOfOperation (Cache, Search) and State_C and search_res = False THEN read_disk* = True and State_C
If the search succeed, the file is read from the cache: if a size of the requested file is greater than the cache size, partial read operations (from the cache and then from the disk) are requested.	IF EndOfOperation (Cache, Search) and State_C and search_res = True and file_size > 1024 THEN pread_cache* = True and State_C IF EndOfOperation (Cache, PRead_Cache) and State_C THEN pread_disk* = True and State_C

framework is filled by rules of validated and verified KB_{Ag} that are written using Estelle/Ag syntax. A mapping between KB_{Ag} predicates and Estelle/Ag operators has been defined in [18]. It can be used for computerised translation to derive PLA specifications.

2. Illustration of the Proposed Technique

We illustrate creation of KB_{Ag} fragment of a hard disk drive (HDD) cache (see Table 1). A program requests a file. At the first, the file is searched in cache memory. If the search fails, the file is read from a disk; otherwise the file is read from the cache. If a size of the requested file is greater than the cache size, partial read operations (from the cache and then from the disk) are requested. Cache size is 1024 Kb. A portion of a formed PLA specification using KB_{Ag} of HDD cache queuing system is presented too.

Static Verification of HDD Cache KB_{Ag}

Two productions describing *end of search in cache* are represented by a decision table (see Table 2).

Table 2. Decision table describing *end of search in cache*.

1. EndOfOperation (<i>Cache, Search</i>)	True			False	-
2. ^State_C	True			False	-
3. search_res	True			-	-
4. file_size	≤ 1024	> 1024	.	-	-
1. read_disk [*] = True	.	.	x	.	.
2. pread_cache [*] = True	.	x	.	.	.
3. State_C	.	x	x	.	.
	1	2	3	4	5

The decision table shows that no rules correspond to the first row. Thus, a deficiency anomaly – missed rule presents in the KB. Therefore, the KB has to be supplemented with a corresponding rule.

A Fragment of a Formed PLA Specification

A fragment of the created PLA specification of the HDD cache example using our technique is presented below. Rules describing the cache behaviour after the end of the search operation have been used to supplement when *eop.Search* construction of PLA specification framework. The Praxis generated code is written in typewriter font, the added code—in *italic*.

```
when eop.Search
begin
  if(search_res = False) then
    Start read_disk;
  if((search_res = True) and (file_size > 1024)) then
    Start pread_cache;
  if((search_res = True) and (file_size <= 1024)) then
    Start read_cache;
end;
```

Static Verification and Functional Validation Experiments

Experiments were carried out using modified KB_{Ag} of the analysed example. Modifications were made by introducing individual cases of the anomalies, which are listed in the table 3. The aim of the experiments was to know what are the additional general properties could be checked using the proposed static verification technique for PLA specifications being developed?

As shown in the table 3, the proposed technique permits to check additional general properties. Additional experiments have been carried out: cases of the anomalies, which are listed in the table above, have been introduced to the developed PLA specification of the analysed example. Validation was carried out using Protocol analysis system PRANAS-2 [3]. Results of the experiments have shown that property *absence of redundancy – unsatisfiable rule condition* (unsatisfiable condition in *if...then* sentence in PLA specification) can be detected using the system PRANAS-2. Another cases of anomalies were not detected by the system.

Table 3. Results of validation experiment comparison.

Property	Detected during KB_{Ag} static verification	Detected during KB_{Ag} functional validation
Absence of redundancy – absorbed rules	+	-
Absence of redundancy – duplicated rules	+	-
Absence of redundancy – unsatisfiable rule condition	+	+
Absence of redundancy – irrelevant condition	+	-
Absence of ambivalence – ambivalent rules	+	-
Absence of deficiency	+	-

3. Related Works

Our technique is similar to the one proposed by Fuchs and Schwitter [7], Johnson *et al.* [4] in such a way that they also offer transformation of problem domain description to representation structures and then to an executable language. However, our technique checks general properties during validation and verification while Fuchs and Schwitter [7] technique checks specific invariant properties. Our approach is similar to that of [27] since they both offer the use of declarative languages for a description of the user needs. We use PLA model concepts for representation of the object structure, while Specht [27] use “object as theory” model. In our approach, in contrast to the compared one, we emphasise validation of the created declarative description. A transformation to target representation is used in both approaches. Sen *et al.* [28] present a technique for building the declarative descriptions in a unified way. We solve the same problem. Bergeron *et al.* [29] state that static and dynamic analyses are complementary. They propose to use static analysis first. In our work at the beginning of the analysis we use static verification technique that complements the dynamic validation - additional properties are detected. Our approach, as another ones (e.g. [6, 30, 31]), uses static and dynamic analyses in a pair in order to comprehensively analyse the application being developed. General structures of the predicates and rules for KBs of application and validation are defined in both techniques too. Our approach is similar to [32] since they all use the decision table (DT) for representation of state-based systems. Mappings between rules and DTs in order to elaborate advantages of tabular representation are used in our and [33] approaches. Our approach as well as many others, example of which is [34], exploit advantages of tabular representation in order to perform verification by transforming certain representation to DTs. However, our approach is specific in the sense that verification is performed on knowledge base that is oriented at creation of PLA specifications. In [35], already created PLA specifications are validated using first order predicate logic and Prolog. While in our work we use knowledge techniques for analysis of the specifications to be created using knowledge base. Sellini *et al.* [8] and we use intra- and inter- validation for analysis of the acquired knowledge. They also use an intermediate formalised description (application KB in our case) for construction of a knowledge model (specification in our case). While performing static verification of an application KB, we use results of Vanthienen *et al.* [10, 21], whereas when analysing the dynamic properties we use the reachable

state method that is similar to the functional validation method suggested by Preece *et al.* [11] in a view of analysis of execution paths.

4. Conclusions

The technique proposed in the paper allows to validate general static and dynamic properties of PLA specifications during development process and to develop these specifications using knowledge bases. Advantages of the technique are evaluated using the following criteria:

- *preliminary validation* - possibility to perform validation and verification experiments at the initial specification development stages when a knowledge base of the specification is being developed;
- *check for additional properties* - properties, that have not been examined during functional validation of PLA specifications, are analysed during static verification.

References

- [1] C. Facchi, M. Haubner, U. Hinkel, The SDL Specification of the Sliding Window Protocol Revisited. Technical Report TUM-I9614, *Technische Universit at Munchen*, 1996.
- [2] P. Spirakis, B. Tampakas, K. Antonis, K. Hatzis, G. Pentaris, Specification Languages of Distributed and Communication Systems: State of the Art, ESPRIT Long Term Research project Nr.20244 report, 1996.
- [3] H. Pranavicius, V. Pilkauskas, A. Chmieliauskas, Aggregate Approach for Specification and Analysis of Computer Network Protocols, Technologija, Kaunas, 1994.
- [4] W.L. Johnson, M.S. Feather, D.R. Harris, The KBSA Requirements/Specification Facet: ARIES. *Proc. of the Conference Knowledge-Based Software Engineering* (1991), 48-56.
- [5] J. d'Almeida, R. Achuthan, T. Radhakrishnam, V.S. Alagar, Transformation of semi-formal specifications to VDM. *Proc. of the Conference Knowledge-Based Software Engineering* (1992), 40-49.
- [6] P. Mi, W. Scacchi, A Knowledge-based Environment for Modeling and Simulating Software Engineering Processes. *IEEE Trans. on Knowledge and Data Engineering* 2(3) (1995), 283-294.
- [7] N.E. Fuchs, R. Schwitter, Attempto Controlled English. *Proc. of the CLAW 96, The First International Workshop on Controlled Language Applications* (1996), 124-136.
- [8] Sellini, F., C. Vargas, P.-A. Yvars, Considerations about validation of knowledge models in KBE systems. *Proc. of the 4th European Symposium on the Validation and Verification of Knowledge Based Systems*. Katholieke Universiteit Leuven (1997), 83-93.
- [9] S.A. Noah, M. Williams, Exploring and Validating the Contributions of Real-World Knowledge to the Diagnostic Performance of Automated Database Design Tools. *Automated Software Engineering* (2000), 177-186.
- [10] J. Vanthienen, Prolog v.5 User's manual, Katholieke Universiteit Leuven, 2000.
- [11] A. Preece, C. Grossner, T. Radhakrishnan, Validating Dynamic Properties of Rule-Based Systems. *International Journal of Human-Computer Studies* 44 (1996), 145-169.
- [12] P. Meseguer, A. Preece, Assessing the Role of Formal Specifications in Verification and Validation of Knowledge Based Systems. *Proc. of the 3rd IFIP International Conference on Achieving Quality in Software* (1996), 317-328.
- [13] A. Preece, R. Shinghal, Foundation and Application of Knowledge Base Verification, *International Journal of Intelligent Systems* 9 (1994), 683-702.
- [14] A. Preece, Evaluating Verification and Validation Methods in Knowledge Engineering, *Micro-Level Knowledge Management* (2001), 123-145.
- [15] M. Bruynooghe, N. Pelov, M. Denecker, Towards a more declarative language for solving finite domain problems. *Proc. of the ERCIM/ COMPULOG Workshop on Constraints* (1999), 1-14.
- [16] A.Th. Schreiber, B. J. Wielinga, J.A. Breuker, KADS: A Principled Approach to Knowledge-Based System Development, Academic Press, London, 1993.

- [17] W.V. Velde, A. Aamodt, Machine learning issues in CommonKADS, KADS-II/ TII.4.3/ TR/ VUB/ 002/ 3.0, 1994.
- [18] G. Budnikas, Development and analysis of aggregate specifications using knowledge bases. PhD dissertation. Kaunas University of Technology, 2004.
- [19] CLIPS Reference Manual, Basic Programming Guide, 2003.
- [20] J. Vanthienen, G. Wets, Integration of the Decision Table Formalism with a Relational Database Environment, *Information Systems* **20**(7) (1995), 595-616.
- [21] J. Vanthienen, C. Mues, G. Wets, Inter-tabular Verification in an Interactive Environment. *Proc. of the 4th European Symposium on the Validation and Verification of Knowledge Based Systems*, Katholieke Universiteit Leuven (1997), 155-165.
- [22] S. Russel, P. Norvig, Artificial Intelligence—a Modern Approach. Prentice Hall, 1995.
- [23] C. Mues, On the Use of Decision Tables and Diagrams in Knowledge Modeling and Verification. PhD dissertation, Katholieke Universiteit Leuven, 2002.
- [24] J. Vanthienen, C. Mues, A. Aerts, An illustration of verification and validation in the modelling phase of KBS development, *Data & Knowledge Engineering* **27** (1998), 337-352.
- [25] H. Pranavicius, G. Budnikas, Static Verification of Aggregate Specifications, *Information technology and control* **4**(29) (2003), 39-44.
- [26] H. Pranavicius, G. Budnikas, Creation of Estelle/Ag Specifications Using Knowledge Bases. *Informatica* **14**(1) (2003), 63-74.
- [27] G. Specht, O!-LOLA - Extending the Deductive Database System LOLA by Object-Oriented Logic Programming, *Informatica* **9**(1) (1998), 107-118.
- [28] M. Sen, J. Minambres, A. Garrido, A. Almansa, Logical Formal Description of Expert Systems. *Informatica* **13**(2) (2002), 177-208.
- [29] J. Bergeron, M. Debbabi, J. Desharnais, M.M. Erhioui, Y. Lavoie, N. Tawbi, Static detection of malicious code in executable programs. *International Journal of Requirement Engineering* (2001), 1-9.
- [30] B. Regnell, P. Runeson, Combining Scenario-based Requirements with Static Verification and Dynamic Testing, *Proceedings of the 4th International Workshop on Requirements Engineering: Foundation for Software Quality*, Pisa, Italy (1998), 1-12.
- [31] T.-H. Wang, T. Edsall, Practical FSM Analysis for Verilog, *Proc. IEEE Int'l Verilog HDL Conf. and VHDL Users Forum*, IEEE Computer Soc. Press, Los Alamitos, Calif. (1998), 52-58.
- [32] T.A. Arentze, A.W.J. Borgers, H.J.F. Timmermans, The integration of expert knowledge in decision support systems for facility location planning, *Computers Environment and Urban Systems* **19**(4) (1995), 227-247.
- [33] T.L. Chambers, A.R. Parkinson, Knowledge representation and conversion for hybrid expert systems, *Journal of Mechanical Design*, **120**(3) (1998), 468-474.
- [34] J. Degelder, M. Steenhuis, A knowledge-based system approach for code-checking of steel structures according to Eurocode 3. *Computers and Structures* **67**(5) (1998), 347-355.
- [35] H. Pranavicius, R. Miseviciene, V. Miliute, Use of Aggregate Specification and Logic Programming for Knowledge Base Validation and Verification, *Proc. of International Conference “Modelling and Simulation of Business Systems”*, Technologija (2003), 203-208.

Extracting Opinions Relating to Consumer Electronic Goods from Web Pages

Taichi NAKAMURA and Hiroshi MARUYAMA
Tokyo University of Technology

Abstract: This paper describes a means of extracting the highly credible opinion ratings from Web pages. The principle behind our proposal is based on extracting the co-occurrence relation between a word representing a feature in the specification of a product or service produced by a company and a word showing a consumer's evaluation and then enlarging the dictionaries by registering additional words or phrases that appear in the co-occurrence relation.

Keywords. Elicitation, Web pages, Co-occurrence, Text Mining

Introduction

Only 4% of customers who are not satisfied with the goods or services they have purchased directly appeal to a company. Some of the rest records their unhappiness with the items up on a bulletin board on the Web. Companies need to remember that customers intending to purchase an article make a decision only after receiving advice from others. We propose a means of extracting the highly credible evaluations from the staggering volume of information that customers post on Web pages. The principle of our proposal is based on extracting the co-occurrence relation between a word indicating a specification feature and a word indicating a user's evaluation of a product or a service. Section 1 describes current research works. Section 2 introduces our proposed methodology for eliciting the evaluation information and describes the sentences which represented the review evaluations of a cellular phone, posted on Web pages. The experimental results are discussed in Section 3, while Section 4 outlines proposed future works.

1. Related studies

The methods of analyzing opinion-related sentences on the Web which have been proposed in existing research studies are as follows: unsupervised learning algorithms for classifying reviews as "thumbs up" or "thumbs down" [1] and [2], a method for

quantitatively representing the degree of satisfaction with commercial goods and the degree of confidence in the results for analyzing opinion-related sentences [3], and a method for extracting a customer's opinion from a sentence in which the character string includes a subject, an item from the specification and an evaluation comment [4].

The methodologies proposed by Kobayashi or Fujimura may be able to provide an evaluation of specified features of a product. However with these methodologies it is necessary to manually up-date a dictionary. Even though significant efforts are made to register the words describing the specified features of a product in various domains of the dictionary, the words in the dictionary may become obsolete in even a short time.

2. Overall Research Program

2.1 Evaluation Information Data set Model

The data set model is illustrated in Figure 1. Set A is the pages retrieved on the Web by specifying the key words relating to the product or service. Set H is set of reviews extracted manually from Set A. Set S is the pages extracted automatically by the text mining system. The product set of Set H and Set S is Set T, so $T = H \cap S$. The recall ratio (R) and the precision ratio (P) are defined, respectively, as $R = |Set\ T| / |Set\ H|$ and $P = |Set\ T| / |Set\ S|$.

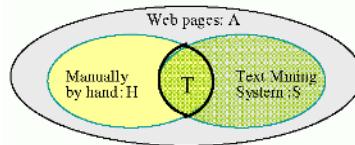


Figure 1. Evaluation Data Set Model

2.2 Evaluation Elicitation System

- (1) The elicitation system has a search engine, which are used to collect the Web pages relating to a product or service as depicted in Figure 2.
- (2) The block for extracting reviews extracts the patterns in which there is co-occurrence a specification requirement word and an evaluation word. The noun and the unknown word are registered into the specification dictionary. Similarly the ending words are added on the evaluation expression dictionary.
- (3) The block for analyzing might have different functions according to the way of utilizing reviews required by a customer. This block is not focused in our research.

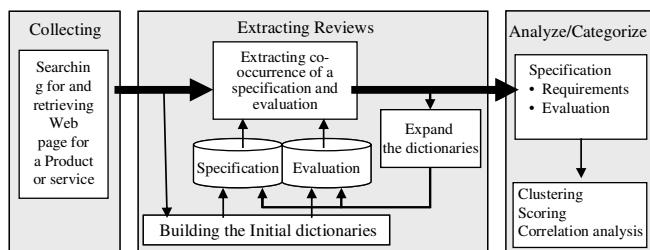


Figure 2. Evaluation Elicitation System

2.3 Initial Condition and Updating of the Dictionary

In order to examine the performance of our proposed system, we examined, as the review corpus for Set A, 2,000 sentences concerning cellular phones on a bulletin board on the Web. 1,131 sentences, which describe the specification features, and the evaluations were extracted manually, as the correct set of evaluation sentences, corresponding to Set H.

In order to ensure that for only appropriate sentences are extracted, the initial condition of the dictionary should be prepared in a way that can give a high precision ratio (P). Items relating to the functional specification of a product or service are entered as the relevant word followed by the suffix, “KINOU,” meaning “function” and added to the end of a word, while those relating to evaluation comments are entered as the relevant word followed by the suffix, “SEI,” which is an element added to the end of a word in order to express the characteristics of a product. The parts of speech which may represent a specification feature and those that may represent an evaluation may be expressed as follows:

Specification item: < verbal noun that functions as a verb when an affix is appended | nominal adjectival stem | compound noun >

Evaluation item: < verbal noun that functions as a verb when an affix is appended | nominal adjectival stem | adjective | verb phrase >.

First the words definitely representing the specified features are registered in the specification dictionary.

Then the co-occurrence pattern of the < [Specification item] + “Ga” or “Wa” + [Evaluation item]> is extracted. The candidate words of the evaluation which appear in the extracted character strings are then registered into the evaluation expression dictionary as the words correctly representing the evaluation.

After the initially setting of two dictionaries, the co-occurrence pattern is extracted and the candidate words of the specification and the evaluation expression are registered into the dictionaries respectively and so forth.

3. Experimental Results

The relationships between the recall ratio (R) and the precision ratio (P) are presented in Figure 3. In the case where the dictionaries are automatically updated, the precision ratio (P) is no more than 73% even at the initial state.

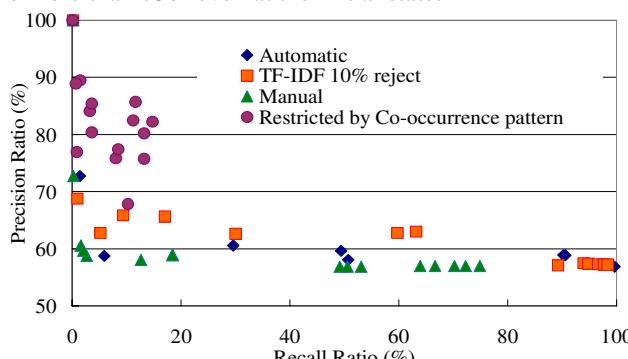


Figure 3. Relation between Recall and Precision

The reason for the low precision ratio (P) at the initial state may be that the specification dictionary initially contains the words such as “capability” and “copyright” which don’t represent a specification feature and also the words such as “enough” and “setup” which are not evaluation expressions. These apparently incorrect words are noise.

In order to eliminate these noise words, the 10% of the words which scored the lowest TF-IDF value were removed, the precision ratio (P) reached 100%.

Moreover, when the apparently incorrect words for the specification requirements and the evaluation were removed manually from the dictionary, the precision ratio (P) reached 100% at the initial state. Since the manual operation removed too many words, the recall ratio (R) could not reach near 100% in this case. According to the results of experiments, it seems to be difficult to achieve high values for both recall ratio and precision ratio only by expanding the dictionary. Therefore the strongly restrictive co-occurrence pattern of the specification feature and evaluation word is applied to extract the evaluation information. The precision ratio (P) is increased, however the recall ratio (R) is still low as illustrated in Figure 3.

4. Discussion

This paper describes the first step towards understanding and controlling the mechanism which elicits the evaluations on Web pages. The results of the experiment on extracting reviews relating to a cellular phone show that the proposed framework and methodology open up possibilities for improving the performance in eliciting the product evaluations on the Web. However the performance of the proposed system was not able to reach the target performance using this implementation. In order to reach the target performance, the following technical issues as follows need to be solved.

- (1) It is necessary to improve the accuracy in extracting the character strings defined in the sequence [specification item] [“Ga” or “Wa” in Japanese] [evaluation item] by using text mining process in a way which takes account of the characteristics of the subject to be analyzed.
- (2) Those words closely linked to target to be analyzed should be registered in the dictionary in order to reduce the number of incorrect words which act as noise in the extracted corpus.
- (3) The data collected from the Web pages needs to be cleaned to eliminate URLs, emotion characters and icons before running the extraction process.
- (4) Before extracting the sentences manually, as the correct set of evaluation sentences, corresponding to Set H. the person carrying out the task needs to be carefully trained.

References

- [1] Peter D. Turney, “Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews”, In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002, pp. 417-424.
- [2] Bo Pang, Lillian Lee, Shivakumar Vaithyanathan, “Thumbs up? Sentiment Classification using Machine Learning Techniques” Empirical Methods in Natural Language Processing(EMNLP2002), 2002, pp.76-86.
- [3] Kenji Tateishi, Yoshihide Ishiguro, Toshikazu Fukushima, “Opinion Information Retrieval From the Internet”, IPSJ NL-144-11 (in Japanese), 2001, pp.75-82.
- [4] Nozomi Kobayashi, Kentaro Inui, Yuji Matumoto, Kenji Tateishi, Toshikazu Fukushima, “Collecting Evaluative Expressions by A Text Mining Technique”, JPSJ NL-154-12 (in Japanese), 2003, pp.77-84.

This page intentionally left blank

3. Intelligent Agents

This page intentionally left blank

A Correlation-Driven Reactive Layer for Situation-Aware BDI Agent Architecture

Gabriel JAKOBSON^{a, 1}, John BUFORD^a, Lundy LEWIS^b

^a*Altusys Corp. Princeton, NJ, USA*

^b*Southern New Hampshire University, Manchester, NH, USA*

Abstract. This paper proposes a new model of situation management in distributed dynamic systems, such as telecommunication networks, modern battlespaces, and disaster relief operations. The proposed model is based a multi-agent systems approach, where the Belief-Desire-Intension agent architecture has been expanded by an advanced interpretation cycle. The critical feature of the advancement is the replacement of the traditional event-plan paradigm with a new event-situation-plan paradigm, which allows recognition of more complex operational situations.

Keywords: Situation management, multi-agent systems, BDI agent, event correlation

Introduction

This paper proposes a model of situation management (SM) for distributed dynamic systems, such as telecommunication networks, modern battlespaces, and disaster relief operations. Structurally those systems contain multiple components, which are engaged into fairly complex topological, spatial, causal, and other domain-specific relations. Dynamically, they act in time, are sensitive to various events, and their behavior could be characterized by evolving situations. The focus of the paper will be on a high-level situation management, i.e. on a situation management level, which is abstracted above the signal and data processing levels.

Due to the large scale of the managed systems, the natural distribution of decision-making power among local system components, and the desire to match the significant autonomy and mobility requirements of the components, the Multi-Agent Systems (MAS) [1] approach has been widely used for managing large-scale dynamic systems. Several different architectures of MAS have been proposed, most notably the architecture of Belief-Desire-Intention (BDI) agents [2], which is based on the principles of rational reasoning. Since its introduction, the BDI model has experienced several functional advancements; however recent attention to operations of large-scale distributed dynamic systems have revealed a weakness of the BDI model, namely the lack of an adequate capability to cope with fast moving, unpredictable and complex operational situations. The major reason for this weakness of the BDI model is the use

¹ Corresponding Author: Altusys Corp., P.O. Box 1274, Princeton, NJ 08542, USA;
Email: jakobson@altusystems.com.

of a simple paradigm of the agent's reaction to events, where agent's plan deliberation process is invoked by a single event (EP paradigm). In this work we propose an advanced paradigm for the agent's reactive layer called the Event-Situation-Plan (ESP) paradigm. According to the ESP paradigm plans are invoked by recognition of complex situations. We will show how the technology of event correlation could be used to implement the processes of situation recognition.

The rest of the paper is organized as follows. The Section 1 reviews the basics of situation management. The Section 2 gives a brief overview of the multi-agent approach to situation management. The Section 3 introduces the key concept of situation-driven plan invocation and shows how the traditional BDI model could be extended with situation awareness. The Section 4 describes the event correlation technology used for recognition of operational situation. The Section 5 presents an example of situation recognition in a SS7 Signaling Network, and the Section 6 concludes the paper with issues and future research directions

1. Situation Management

1.1. Situation Management - Basic Notions

SM is understood as a synergistic goal-directed process of information collection and sensing; recognition, monitoring, and prediction of situations happening in the system; reasoning and planning system actions, and implementation of those actions so that the system reaches a goal situation within the pre-defined constraints.

Depending on the goals of SM, one can distinguish between three types of SM: (a) diagnostic, (b) control and (c) predictive types of SM. It is useful to see these types mapped to the time axis. The diagnostic type SM is a retro-perspective task of determining why a certain situation happened. The control type of SM aims to change or keep the current situation, while the predictive SM plans to project possible future situations, e.g. threat situations. For example, finding a root of a packet transmission failure in a telecommunication network is an example of a diagnostic type SM; moving a tank unit from the area of direct hostile fire is a control type SM; and a projection of a potential terrorist attack on a critical infrastructure element is an example of a predictive SM.

1.2 On Situation Modeling

We assume that there exists a world, real or imaginary that could be sensed, perceived, reasoned about and affected. The world is populated with entities, which are engaged in different class, structural, causal, spatial, temporal and other relations. The entities possess distinguishable attributes with certain attribute values. Some entities are active, they change their state in space and time, and some of the entities can interact with other entities forming entity systems. A world situation is an aggregated state of entities and inter-entity relations of the world, which are observable at a specific time period. A world situation model is a mental representation of situations happening in the world. More formally a situation is defined as follows.

Let e be an entity $e \in E_i$, where $E_i \subseteq U$ is a subclass of all entities of the universe U . Each entity e is represented by its arguments $\{a_1, a_2, \dots, a_p\}$ with a corresponding set of

argument value domains $\{v_1, v_2, \dots, v_p\}$. Among all entity arguments we will define a subset called situational arguments. The semantics of situational arguments is declared or computed depending on the particular operational context of the application. Importantly, entities are time-dependent objects with their time of creation t , lifespan $\delta=(t, t')$, and time of elimination t' . Consequently, the attribute value of an entity is defined only during the existence of the entity, i.e. $a_i(t)$, $t \in \delta$. Now, the situation S_e on entity e during d is defined in the following inductive way:

$$\begin{aligned} 1. \quad S_e(d) = & \{ \langle a_1(t), a_2(t), \dots, a_p(t) \rangle \in v_1 \times v_2 \times \dots \times v_p / \forall (t, t') \in d \\ & [\langle a_1(t), a_2(t), \dots, a_p(t) \rangle = \langle a_1(t'), a_2(t'), \dots, a_p(t') \rangle] \}, \end{aligned}$$

where $d \subseteq \delta$ is the duration of the situation. We will call $B=\{e\}$ the base of the situation.

2. If $S_{B_1}(d_1)$ and $S_{B_2}(d_2)$ are two situations, where $B_1, B_2 \subseteq U$ and d_1, d_2 are subsets of common lifespans of all entities in B_1, B_2 , correspondingly, then,

$S_B(d) = S_{B_1}(d) \cup S_{B_2}(d)$ and $S_B(d) = S_{B_1}(d) \cap S_{B_2}(d)$ are situations, where, correspondingly

$$d = d_1 \cap d_2 \text{ and } B = B_1 \cup B_2 \text{ and } d = d_1 \cup d_2 \text{ and } B = B_1 \cap B_2$$

Let's us provide some explanations. First, we intentionally omitted to mention relations $R_j \subseteq E_1 \times \dots \times E_m = \{(e_1, \dots, e_m) / e_1 \in E_1, \dots, e_m \in E_m\}$, since we will handle a relationship and corresponding entities as a higher-level entity due to the use of an aggregation operation. Second, while considering situations, we look on a subset of all entities, called active entities. Like situational arguments, the semantics of active entities is declared or computed depending on the specific operational context. Due to our notions of active entities and situational arguments, multiple different situations can be defined on the same set of entities.

Different situation definitions can be found in the literature. For example, McCarthy's situation calculus [3] uses a state-based approach, where a situation is considered a snapshot of a complete world state at a particular time. Reiter [4] defined a situation as a sequence of actions enabling calculation of the current state knowing the initial state and the sequence of actions transforming the initial state. Reiter's definition was introduced with planning tasks in mind, e.g. robot actions planning. Yet another definition uses fluents [5], i.e. situation-dependent functions to define the situations. Our definition of a situation is closer to McCarthy's definition but assumes active entities and situational arguments as the primary constructs.

2. MAS Approach for Situation Management

2.1 Rational Reasoning in Multi-Agent Systems

Multi-agent systems are widely used for modeling complex distributed systems due to such features as capability to act independently in a persistent manner, rational reasoning, interaction with the world, and mobility. One can see three layers of

rational reasoning: the reactive, deliberative, and reflective layers. In the context of situation management we see the reactive layer performing the following two major functions: (a) sensing and perceiving world events and (b) detecting multi-event patterns and recognizing situations happening in the world

The deliberative layer performs different reasoning functions related to various cognitive acts such as planning, control, prediction, and explanation. The interplay between the deliberative and reactive layers contain a feedback from the deliberative layer to the reactive layer, e.g. supply facts, hypothesis and meta-control information to guide the situation recognition process performed at the reactive layer. The reflective layer contains processes for inferencing about the events happening in the reactive and the deliberative layers, and how well they proceed with the goals set for the agent.

2.2 The Basic BDI Agent Model

The Belief-Desire-Intension (BDI) model was conceived as a relatively simple rational model of human cognition [2]. It operates with three main mental attitudes: beliefs, desires and intentions. Rao and Georgeff [6] replaced the declarative notion of intentions with a procedural specification of instantiated and executable plans. Among many BDI agent models, the dMARS formalism serves as a well-recognized reference model for BDI agents [7]. Below is a brief sketch of the basic notions related to the BDI model.

An agent's beliefs are the facts about the World that the agent possesses and believes to be true. Beliefs could be specifications of the World entities, their attributes, relations between entities, states of the entities, and relations. In many cases, the agent's beliefs include the knowledge about other agents as well models of the agent itself, i.e. the reflective component of the beliefs. Desires are an agent's motivations for actions. Two kinds of activities are associated with desires: (a) to achieve a desire, or (b) to prove a desire. In the first case, by applying a sequence of actions the agent wants to reach a state of the world where the corresponding desire formula becomes true; in the second case, the agent wants to prove that the world is or isn't in a particular state by proving that the corresponding belief formula is true or not. Often desires are called goals or tasks.

Plans are operational specifications for an agent to act. An agent's plan is invoked by a trigger event, e.g. the acquisition of a new belief, removal of a belief, or receipt of a message. Actions could be external, i.e. essentially procedure calls or method invocations; or they could be internal such as adding and removing beliefs. Abstract plans are stored in the agent's plan library. During agent operations certain abstract plans are selected from the library and instantiated depending on variable bindings, substitutions and unifications. An agent's intention is understood as a sequence of instantiated plans that an agent is committed to execute. In response to a triggering external event, an agent invokes a plan from the plan library, instantiates it, and pushes it onto a newly created stack of intentions.

3. ESP Paradigm Extension to the BDI Agent Model

In many applications found in telecommunications, military battlefields, homeland security, and other domains, decisions are often made on the basis of multiple events.

Thus, we have extended the existing BDI agent model with the capability to respond to multiple events. Event correlation [8] is used to analyze those events and recognize emerging situations, while capturing an operational context of the world. The act of recognition of a situation invokes a plan or triggers an automatic generation of a plan from a specification, which is embedded in the situation. Feedback from recognized situations is used as contextual information to control the event correlation process. The architecture of the interpreter of a BDI agent extended with the ESP paradigm is illustrated in Figure 1. The architecture depicts the following main steps of the interpretation process: (a) Event Correlation, (b) Situation Composition, (c) Plan Deliberation, (d) Plan Instantiation, and (e) Intention Execution.

The process of construction of the full operational situation is performed by two preprocesses, at first, the component (local) situations are discovered by the event correlation process, and then the component correlations are combined into a full situational model by the situation composition process. The event correlation process is a fairly complex time-dependent process, which involves a variety of different kinds of information, including knowledge about the entity classes, correlation rules and situation classes. The event correlation process is aware about the current world entities and relations between them. The relations could be very different in nature, including basic class, structural, connectivity, temporal, and other relations specific to the application domain.

We distinguish between abstract situations, i.e. situations, which describe certain behavioral stereotypes of entities, and operational situations, which are instantiations of abstract situations. An important aspect of our approach is that a situation contains references to the plans associated with the situations.

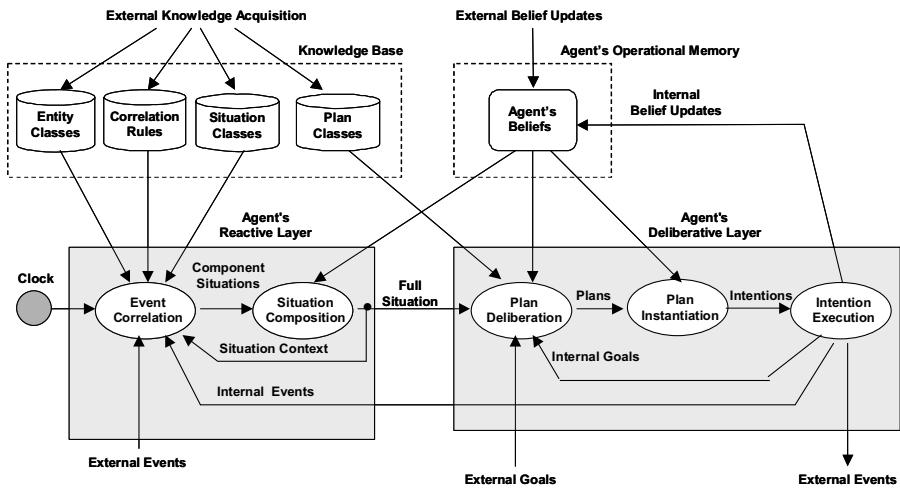


Figure 1. Architecture of the ESP-Based BDI agent interpreter

The invocation of a plan from a situation allows one to model complex plan triggering conditions, including triggering pre-conditions and maintenance conditions. Since operational situations are the result of the event correlation and situation recognition processes, the plan invocation conditions ultimately depend on temporal, structural, causal and other domain specific relations. The event correlation process is a time-

dependent process using the input from a clock. The rest of the interpretation loop is the same as that described in the traditional BDI model [6].

3. Event Correlation-Driven Situation Recognition

3.1. A Two-Step Process of Situation Recognition

Changes in the entities and relations among the entities create fairly complex situational pictures of the world. Many different local situations happening with certain subsets of entities in the real world create multiple component situation models that need to be combined into an overall full situational model. The component situations can be related to simple events like entity parameter changes, to more complex events of appearance or disappearance of entities or entity relationships, to significantly more complex events relating multiple entities and relationships.

The distribution of the entities in the world along spatial, structural, functional and other coordinates creates component situations along the same coordinates. Consequently, there is a certain degree of freedom how particular situation management architecture is designed: in a distributed manner following the above-mentioned coordinates, or in a more centralized manner. Per our definition of situations given in Section 1.2, where we defined a situation as a collection of time dependent states of situation attributes of certain entities, we are able to consider multiple concurrent local situations happening in the world. Such an interpretation of the situation management process allows an identification of two major steps of dynamic situation recognition: (a) recognition of local component situations using event correlation technology and (b) constructing a new full situational picture based on the current situation and the recognized local component situations.

One can see multiple correlation processes producing different component situations that used for construction of a full operational situation as shown in Figure 2.

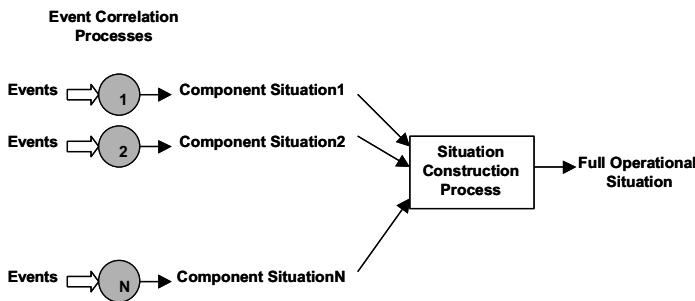


Figure 2. Multiple event correlation processes

In practice, the construction of a new situational picture translates into adding new situational facts into the agent's beliefs maintained in the agent's operational memory, or deleting the old ones.

3.2. Event Correlation

Event correlation is a real-time event analysis procedure, which assigns a new meaning to a set of events. Our approach to event correlation uses the principles of model-based reasoning originally described in [9] for troubleshooting electronic circuit boards. The idea of the model-based approach is to reason about the system from a representation of its structure and functional behavior. The model of extending this approach to real-time event correlation was proposed in [8].

A simplified version of the event correlation and situation recognition process is depicted in the diagram shown in Figure 3. The event correlation agent possesses the beliefs (i.e. facts) about the structure of a particular application, i.e. all the entities and inter-entity relations, and beliefs about the dynamic situations in the application domain. Corresponding models are built and maintained by the agent as it accesses the components of domain knowledge bases such as Domain Ontology (e.g. classes of entities and relations), Situation Ontology (classes of typical situations), and Domain Constraints (e.g. constraints representing the semantics of the entities, relations, and entity parameters).

Situations occurring in the world are recognized by applying some situation recognition rule and by matching the pattern in the rule's condition part C against the agent's beliefs. The condition C is constructed as an FOL predicate over terms representing entities, relations, and existing situations. The pattern matching and variable unification process can be based on different models, e.g. the Rete network [10] as it was demonstrated in the GRACE system [11].

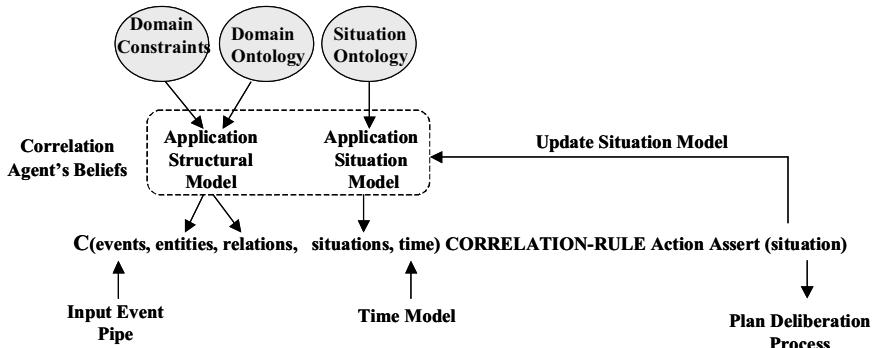


Figure 3 Event correlation and situation recognition process

During the event correlation process several time-dependent functions are performed, including: (a) time-dependent counting of event occurrences, (b) monitoring the duration of the situations and lifespan of the entities, (c) monitoring the correlation time window; (d) scheduling the time dependent actions, and (e) managing temporal relations between events.

4. Example: Situation Recognition by Event Correlation

To illustrate how a situation can be invoked at the reactive layer by a situation-aware agent, let's consider a fragment of a SS7 signaling network [12] shown in Figure 4. The signaling network is a packet-switched network used to control the circuit-switched telecommunication network. The signaling network is used for various functions, including call establishment, routing, and intelligent call services. Since very high standards are set for the signaling network reliability and availability, each signaling transfer point (STP) is duplicated using STP-pairs and Quads linked by communication lines serviced by different inter-exchange carriers (IXC). Interface from the signaling network to the switching network is established by service switching points (SSP), which are switches. So-called A-links connect STPs and SSPs.

For the sake of our example, assume that there are 3 inter-exchange carriers, IXC-1 (e.g. MCI), IXC-2 (e.g. AT&T) and IXC-3 (e.g. Sprint), where IXC-1 provides links Link-1, Link-2 and Link-4; IXC-2 provides link Link-3; and IXC-3 provides links Link-5 and link-6. The situation that we want to recognize is the IXC-Failure-Situation, which takes place if during less than 60 seconds four or more links failed at a particular STP pair so that 75 % of more of the failed links are provided by the same IXC. Let's assume that entity classes and the ontology of the signaling network domain have been already described, as well the ontology of typical situation classes.

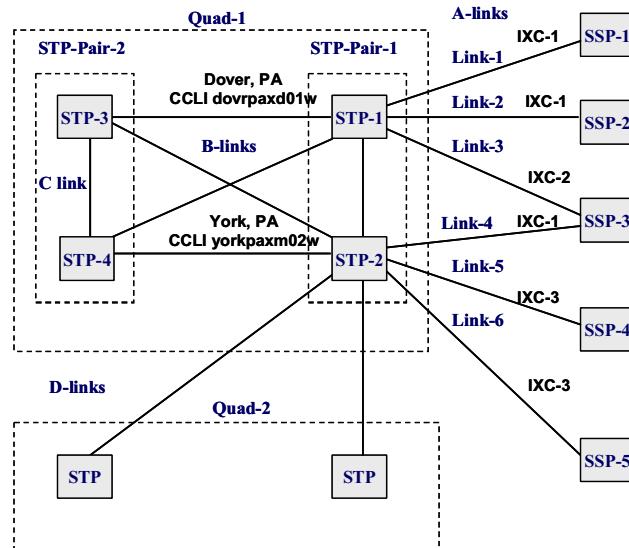


Figure 4. A fragment of a SS7 signaling network

Table 1 illustrates description of STP-1, STP-Pair-1, IXC-1 and Link-1 in a simple pseudo notation. In these specifications CLLI denotes a special 11-character code adopted in the telecommunication industry to uniquely identify a location and type of telecommunication equipment. In the STP-1 description CLLI refers to a digital cross-connect system (DCS), which is located in Dover, PA, and it belongs to the STP-Pair-1. Both STP-Pair-1 and IXC-1 contain variables LINK-DWN-CNT to count the number of down links. The LINK-1 specification includes the link state variable, which is

always set to “Down” when an alarm message LINK-1 failed is received by the event correlation process.

Table 1. Specification of entities STP-1, STP-Pair-1, IXC-1 and Link-1

STP-1 Parent STP-Class Slot NE-CLLI “dovrpaxd01w” Slot NE-Name “Dover STP” Slot NE-Type “dcstp” Slot LOCATION “Dover, PA” Slot STP-Pair “STP-Pair-1” Slot NUM-ASSOCIATED-LINKS “7” Slot SATRT-TIME Slot CURRENT-TIME	STP-Pair-1 Parent STP-Pair-Class Slot STP-Pair-Name “STP-Pair-1” Slot STP-CLLI-1 “dovrpaxd01w” Slot STP-CLLI-2 “yorkpaxm02w” Slot TOTAL-LINKS “14” Slot LINK-DWN-CNT “0”	LINK-1 Parent SS7-LINK-Class Slot LINK-ID “75823” Slot LINK-Type “A” Slot CLLI-1 “dovrpaxd01w” Slot CLLI-2 “sayrpaxsds0” Slot STP-Pair-IXC-Name “IXC-1” Slot STP-Pair-IXC Slot LINK-STATE “Up”
IXC-1 Parent STP-Pair-IXC-Class Slot STP-Pair-IXC-Name “IXC-1” Slot IXC-NAME “MCI” Slot STP-Pair “STP-Pair-1” Slot NUM-SUPPORTED-LINKS “5” Slot LINK-DWN-CNT “0”		

Table 2 shows an IXC-Failure-Rule and corresponding situation IXC-FAILURE-SITUATION, which is asserted into the overall operational situation model if the rule succeeds. The condition part of the rule verifies the failure IXC failure condition described earlier, while the action part contains only one action of asserting the situation. As the situation is added to the overall situation memory, an internal event message is generated. This message is used by the agent’s correlation engine for future correlations or passed to other agents.

In addition to network management tasks illustrated above, the use of the situation-aware BDI agent architecture for natural disaster situation management has been described in [13] and urban transportation threat monitoring in [14].

Conclusions

In this paper we proposed a method how to make BDI agent architecture more responsive to events and situations happening in the world of their activity. In general this increase was achieved by making the agents reactive layer more sophisticated to recognize complex world situations. We also showed how event correlation technology could be used for this purpose.

Introduction of situations between events and plans in the agent’s interpretation cycle has additional benefits, which could be a subject for a future research. Situations carry aggregated information about the behavior of agents and the surrounding world, and as such provide a basis for prediction of future situations and situation learning.

Situations concerning agent's own behavior could be used to implement agent's reflective self-organizing capabilities.

Table 2. Rule IXC-Failure-Rule and situation IXC-Failure-Situation

Rule IXC-FAILURE-RULE	SituationClass IXC-FAILURE-SITUATION
Condition	Components
STP-PAIR ?stp-pair	STP-Pair ?stp-pair
STP-PAIR-NAME ?stp-pair-name	STP-CLLI-1 ?clli1
LINK-DWN-CNT ?link-down-cnt & ≥ 4	STP-CLLI-2 ?clli2
START-TIME ?start-time	LINK-DWN-CNT ?link-down-cnt
CURRENT-TIME ?current-time	TOTAL-LINKS ?total-links
STP-PAIR-IXC ?stp-pair-ixc	STP-PAIR-IXC ?stp-pair-ixc
STP-PAIR-NAME ?stp-pair-name	STP-PAIR-IXC-IXC-NAME ?ixc-name
IXC-DWN-CNT ?ixc-down-cnt	IXC-DWN-CNT ?ixc-down-cnt
((?ixc-down-cnt x 100) /	
?link-down-cnt) ≥ 75	
?current-time - ?start-time ≥ 60	
Action	PrintStatement
Assert IXC-FAILURE-SITUATION	"At STP-Pair ?clli1 ?clli2 from ?total-links total links ?total-down links are down. Out of total down links ?ixc-down-cnt (more than 75 percent) links belong to ?ixc-name"
STP-PAIR ?stp-pair	
STP-PAIR-IXC ?stp-pair-ixc	
Send IXC-FAILURE-MESSAGE	
CLASS IXC-Class	
Slot MSG-Time ?current-time	
Slot STP-PAIR-IXC ?stp-pair-ixc	

References

- [1] M. Wooldridge. *An Introduction to Multi-Agent Systems*. John Wiley and Sons, 2002.
- [2] M. Bratman, *Intension, Plans, and Practical Reason*. Harvard University Press, 1987.
- [3] J. McCarthy and P. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence. In D. Michie, editor, *Machine Intelligence 4*, American Elsevier, New York, NY, 1969.
- [4] F. Pirri and R. Reiter, Some Contributions to the Situation Calculus. *J. ACM*, 46(3): 325-364, 1999.
- [5] D. Wen-Yu, X. Ke and L. Meng-Xiang. A Situation Calculus-Based Approach to Model Ubiquitous Applications. arXiv, Cornell University e-print Archive, 2003.
- [6] A. Rao and M. Georgeff, BDI Agents: From Theory to Practice. In *Proceedings of the First International Conference on Multiagent Systems (ICMAS'95)*, 1995.
- [7] M. d'Inverno, M. Luck, M. Georgeff, D. Kinny, and M. Wooldridge, The dMARS Architecture: A Specification of the Distributed Multi-Agent Reasoning System, In *Journal of Autonomous Agents and Multi-Agent Systems*, 9(1-2):5-53, 2004.
- [8] G. Jakobson and M. Weissman, Real-Time Telecommunication Network Management: Extending Event Correlation with Temporal Constraints. *Integrated Network Management IV*, IEEE Press, 1995.
- [9] R. Davis, R. Shrobe and W. Hamscher, Diagnosis Based on Description of Structure and Function. Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, pp. 137-142, 1982.
- [10] C. Forgy, Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artificial Intelligence*, 19, pp. 17-37, 1982.
- [11] G. Jakobson, M. Weissman, L. Brenner, C. Lafond and C. Matheus, GRACE: Building Next Generation Event Correlation Services, *IEEE Network Operations and Management Symposiums*, 2000.
- [12] R. Dreher. *Signaling System 7 (SS7) Basics*. Althos Publishing, Varina, NC, 2003.
- [13] J. Buford, G. Jakobson, L. Lewis, N. Parameswaran, and P. Ray, D-AESOP: A Situation-Aware BDI Agent System for Disaster Situation Management, *First International Workshop on Agent Technology for Disaster Management*. Hakodate, Japan, May 2006.
- [14] J. Buford, G. Jakobson, L. Lewis, Case Study of Urban Transportation Threat Monitoring Using the AESOP Situation Manager, *Proceedings of the 2005 IEEE Conference on Technologies for Homeland Security*, Cambridge, MA. April 2005.

Building Agent-Based Appliances with Complementary Methodologies

Leon Sterling ^a, Kuldar Taveter ^{a,1} and The Daedalus Team ^a

^a *The University of Melbourne, Department of Computer Science and Software Engineering, Australia*

Abstract. This paper describes a part of the Intelligent Lifestyle project conducted at the University of Melbourne in cooperation with our industrial partner Adacel Technologies in 2004. The Intelligent Lifecycle project aimed to design and build a system of intelligent agent-based appliances. It further aimed to demonstrate that agent development methods were suitable for wide acceptance by software developers with object-oriented but no agent-oriented experience. In this paper we describe how initial requirements engineering activities undertaken using the ROADMAP methodology can be complemented by rapid prototyping performed by using the RAP/AOR methodology. Overall, this paper defines and explains the first two stages of a systematic approach for achieving systems of intelligent agents.

Introduction

A software agent is commonly understood as an *active* entity, possessing the features of *autonomy*, *proactiveness*, *responsiveness* and *social behaviour* [8]. Because of its distributed nature, a promising area of application for intelligent agents is in building a smart home where appliances interoperate seamlessly for the benefit of the home occupants. This application area can be naturally modelled as a society of interacting autonomous entities – agents.

The Intelligent Lifestyle project, conducted at the University of Melbourne in 2004, was part of an ongoing collaboration between Adacel Technologies and the Intelligent Agent Lab at the University of Melbourne, investigating how agent technologies can be effectively deployed in industrial and commercial settings. The collaboration includes an Industry Linkage Project to build a system of ‘invisibly intelligent’ appliances supported through the Australian Research Council, and a project to develop a methodology for developing agent-based systems for wide use supported by the Smart Internet Cooperative Research Centre. As part of this latter project, an Adacel engineer with no previous agent experience participated in application of the methodology. The Intelligent Lifestyle project aimed to design and build a system of intelligent agents, for the explicit purpose of performing field testing and providing demonstrations of intelligent agents.

Since no specific agent-oriented methodology supports all possible quality goals of a software system to be created, such as politeness and robustness, it is feasible to use a combination of two or more methodologies as has been proposed in [10]. In the Intelligent Lifestyle project, the ROADMAP methodology [1, 6] was used for

¹ Corresponding Author: The University of Melbourne, Department of Computer Science and Software Engineering, Victoria, 3010, Australia; E-mail: kuldar@cs.mu.oz.au.

requirements engineering and the RAP/AOR methodology [2] – for rapid prototyping. Additionally, object-oriented methods were used for design and implementation.

In this paper, we first explain the requirements engineering activities conducted for the Intelligent Lifestyle project by means of the ROADMAP methodology [1, 6]. We then dwell on rapid prototyping by using the RAP/AOR methodology [2] and describe the simulation by using the JADE (JADE, <http://jade.cselt.it/>) agent platform [7]. Finally, we analyse the results and draw conclusions. Since the design and implementation of the system of intelligent agents worked out in the Intelligent Lifestyle project and its field testing have been described in the report [9] that is available online, we do not address them in this paper. In addition to this, the video made of the field testing is available as <http://www.cs.mu.oz.au/~kuldar/final.swf> (select Scenario and Intruder Scenario).

1. Customizing agent-oriented methodologies by reusing features

In [10] is proposed a modular approach enabling developers to build customized project-specific methodologies from agent-oriented software engineering (AOSE) features. An *AOSE feature* is defined in [11] to encapsulate software engineering techniques, models, supporting Computer-Aided Software Engineering (CASE) tools and development knowledge such as design patterns. It is considered a stand-alone unit to perform part of a development phase, such as analysis or prototyping, while achieving a quality attribute such as privacy. Another method for building customized methodologies – OPEN [12] – includes the notions of Work Units, Work Products, Producers, Stages and Languages. We can define an *AOSE feature* in terms of these notions as a Work Unit performed by one or more Producers in support of a specific software engineering Stage resulting in one or more Work Products represented in the respective Languages. For example, we can identify the feature of goal and role modelling performed by a domain engineer in support of the requirements engineering stage and resulting in Goal and Role Diagrams and Role Schemas. Analogously, we can identify the feature of simulation performed by a domain engineer and system architect in support of the rapid prototyping stage and resulting in the executable constructs of the JADE agent platform represented in the Java language. Both of the features mentioned support the Quality Goals of adequacy and correctness of the requirements captured.

According to [12], a *work product* is any significant thing of value (e.g., document, diagram, model, class, or application) that is developed during a project. A *language* is the medium used to document a work product. A *producer* is anything that produces (i.e., creates, evaluates, iterates, or maintains), either directly or indirectly, versions of one or more work products. A *work unit* is defined as a functionally cohesive operation that is performed by a producer. A *stage* is a formally identified and managed duration of time. Table 1 presents the Stages, Work Units, Producers, Work Products and Languages of the Intelligent Lifestyle project. As we implied in the paragraph above, features are orthogonal to these notions of OPEN. Table 1 thus shows how AOSE features originating in different methodologies complement each other. However, differently from OPEN [12], we do not regard it as necessary to rely on the formal metamodel of features. As has been demonstrated by our earlier work [6, 10, 11], informal approach to methodology composition works equally well and is more likely to be adopted by industry.

Table 1. Stages, Work Units, Producers, Work Products and Languages of the Intelligent Lifestyle project.

Stage	Work Units	Producer(s)	Work Product(s)	Language(s)
Requirements engineering	Goal and role modelling, environment modelling, knowledge modelling	Domain engineer	Goal and Role Diagram, Role Hierarchy Diagram, Role Schema, Environment Model, Knowledge Model, System Overview Diagram	ROADMAP Graphical Modelling Language
Rapid prototyping	Process modelling, simulation	Domain engineer, system architect	External AOR Diagram, constructs of JADE agent platform	AOR Modelling Language, Java
Architectural design	Logical architecture modelling, hardware architecture modelling	System architect	Software Architecture Diagram, Hardware Architecture Diagram, UML Interaction Diagram	Based on UML or UML
Detailed design	Designing individual components (agents) of the system	Software engineer	Component Architecture Diagram, UML Class Diagram, UML Interaction Diagram, UML State Machine Diagram	Based on UML or UML
Implementation	Implementing agents	Software engineer, Deployment engineer	UML Deployment Diagram, constructs of JADE agent platform	Java

2. Requirements engineering

Initial requirements engineering for the Intelligent Lifestyle project was performed by means of the ROADMAP methodology [1, 6]. Figure 1 shows the models employed by the ROADMAP methodology. In the ROADMAP methodology, the models are divided vertically into Domain Specific Models, Application Specific Models and Reusable Services Models. The Environment Model and Knowledge Model represent information about a specific domain and belong to multiple phases in the software development lifecycle. The Goal Model, Role Model, Agent Model and Interaction Model are tied to the system being modelled. Generic and reusable components in the system are captured by the Social Model and Service Model. The models are also split horizontally by dotted horizontal lines according to the analysis and design phases so that the Environment Model, Knowledge Model, Goal Model, Role Model and Social Model are parts of the domain analysis phase, while the Agent Model, Interaction Model and Service Model form parts of the design phase.

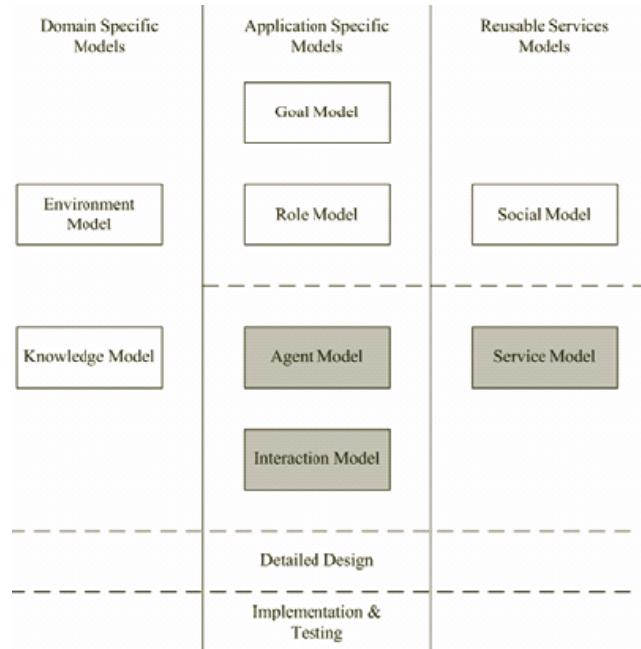


Figure 1. ROADMAP Analysis and Design Models.

2.1. The Environment Model and Knowledge Model

As the first step of requirements engineering, the Environment Model and Knowledge Model of the problem domain were created. The Environment Model consists of two high level environments – physical and conceptual – where each of them is further decomposed into specific zones. The conceptual environments are non-physical environments with which the system interacts, such as the Internet, Bluetooth and the SMS/GMS environments used in the Intelligent Lifestyle project.

The knowledge model is a way of formalizing the knowledge aspect of the system. It can be viewed as an *ontology* providing a common framework of knowledge for the agents of the problem domain. Figure 2 shows an example of a Knowledge Model component that was designed in order to enable the system to schedule activities and check users' schedules.

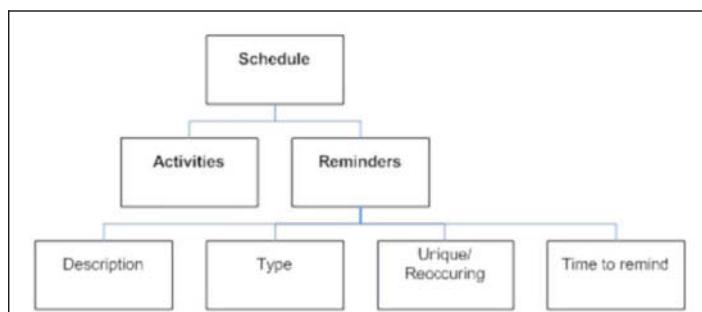


Figure 2. The Schedule Knowledge Component.

2.2. The Goal Models

The Goal Model provides a high level overview of the system requirements. Its main objective is to enable both domain experts and developers to pinpoint the *goals* of the system and thus the *roles* the system needs to fulfil in order to meet those goals. Implementation details are not described at all as they are of no concern in the analysis stage.

The Goal Model can be considered as a container of three components: Goals, Quality Goals and Roles. A Goal is a representation of a functional requirement of the system. A Quality Goal, as its name implies, is a non-functional or quality requirement of the system. We use the term *quality goal* instead of the artificial intelligence community's term of *soft goal* because we wish to emphasize the engineering aspects of the ROADMAP methodology that are necessary to ensure controlled delivery of desired outcomes. A *Role* is some capacity or position that the system requires in order to achieve its Goals. As Figure 3 reflects, Goals and Quality Goals can be further decomposed into smaller related sub-Goals and sub-Quality Goals.

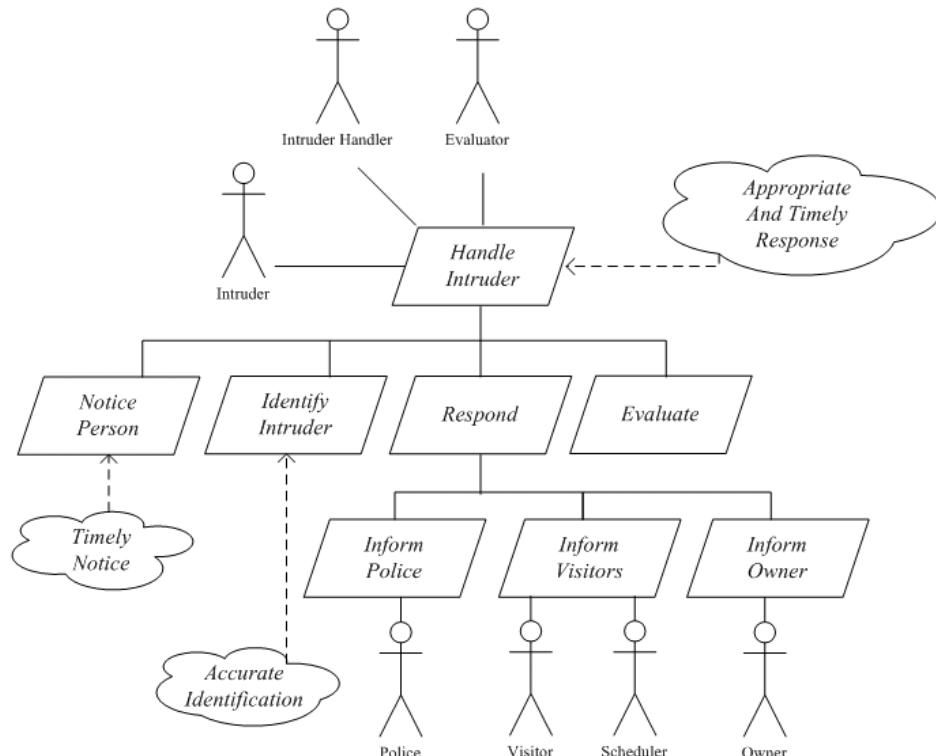


Figure 3. The Goal Model of intruder handling.

Figure 3 represents the Goal Model of the intruder handling scenario. In the diagram, the Role “Intruder Handler” has only one Goal which is to handle an intruder. This goal is characterized by the Quality Goal to provide an appropriate and timely response to a possible intruder detected. The Goal to handle an intruder can be decomposed into four sub-Goals – to notice person, identify intruder, respond and evaluate. There are the Quality Goals of timely notice and accurate identification pertaining to the sub-Goals to notice person and identify intruder, respectively. The sub-Goal to respond, in turn, has been divided into the sub-Goals to inform police, inform the visitors and inform the owner. To accomplish these, the additional Roles “Police”, “Visitor”, “Scheduler” and “Owner” are required. Please note that the order in which the sub-Goals are presented in Figure 3 does not per se imply any chronological order in which they are to be achieved.

2.3. The Role Models

The Role Model describes the properties of a Role. The term Role Schema is used interchangeably with Role Model. The Role Model consists of four elements to describe the Role:

Role Name: A name identifying the Role.

Description: A textual description of the Role.

Responsibilities: A list of duties (tasks) that the Role must perform in order for a set of Goals and/or Quality Goals to be achieved.

Constraints: A list of conditions that the Role must take into consideration when performing its responsibilities. These include guard conditions, safety conditions and quality conditions.

Table 2. Role Schema for the Intruder Handler.

Role Name	Intruder Handler
Description	Identifies and responds to the intruder detected.
Responsibilities	<p>Detect the presence of a person in the environment.</p> <p>Check the house schedule for strangers scheduled to be there.</p> <p>Take an image of the person.</p> <p>Compare the image against the database of known people.</p> <p>Contact the police and send the image to them.</p> <p>Check the house schedule for planned visitors.</p> <p>Send a message to stay away to each visitor expected that day.</p> <p>Inform the owner that the police are on their way and the visitors have been warned not to enter the house.</p>
Constraints	<p>The owner and each person pointed out by him/her needs to provide in advance personal information (face) to be recognised.</p> <p>A subject to be detected needs to be seen within the camera’s image area.</p> <p>The user must maintain the schedule.</p> <p>Visitors must be within the coverage area of mobile communication with their mobile access terminals switched on.</p>

Clearly, this is analogous to the delegation of work through the creation of positions in a human organisation. Every employee in the organisation holds a particular position in order to realise business functions. Different positions entail

different degrees of autonomy, decision making and responsibilities. Taking this analogy, a Role Schema is the “position description” for a particular Role. Table 2 shows the Role Schema created for the Role “Intruder Handler” shown in the Goal Model in Figure 3.

The Role and Goal Models, as well as the Environment Model and Knowledge Model, were created in close cooperation with the customer – our industry partner. This was facilitated by the use of the **Roadmap Editor Built for Easy deveLopment** (REBEL) tool [1] for building the Role and Goal Models.

3. Rapid prototyping by RAP/AOR

Input and feedback from customers is critical in successful software engineering, so in order to enable our industry partner to provide feedback about the system to be developed at as an early stage as possible, the Environment Model, Knowledge Model and the Goal and Role Models were turned into executable specifications by using the RAP/AOR methodology. The Radical Agent-Oriented Process / Agent-Object-Relationship (RAP/AOR) methodology of software engineering and simulation has been introduced in [2] and is based on [3] and [4]. Before introducing executable models of intruder handling, we will explain briefly the notation to be used. For further explanations, please refer to [2].

An *external* (that is, modelled from the perspective of an external observer) *AOR diagram* specified by Figure 4 enables the representation in a single diagram of the types and instances of institutional, human and artificial (for example, software) agents² of a problem domain, together with their internal agent types and instances and their beliefs about instances of “private” and external (“shared” with other agents) object types. There may be attributes and/or predicates defined for an object type and relationships (associations) among agent and/or object types. A predicate, which is visualized as depicted in Figure 4, may take parameters.

Figure 4 shows that the graphical notation of RAP/AOR distinguishes between an *action event* (an event that is created through the action of an agent, such as a physical move performed by an *Intruder*) *type* and a *non-action event type* (for example, types of temporal events or events created by natural forces). The graphical notation of RAP/AOR further distinguishes between a *communicative* action event (or *message*) type and a *non-communicative* (physical) action event type like providing another agent with a commodity.

The most important behaviour modelling elements of RAP/AOR are *reaction rules*. As is shown in Figure 4, a reaction rule is visualized as a circle with incoming and outgoing arrows drawn within the rectangle of the agent type or instance whose reaction pattern it represents. Each reaction rule has exactly one incoming arrow with a solid arrowhead that specifies the *triggering event type*. In addition, there may be ordinary incoming arrows representing *mental state conditions* (referring to corresponding instances of other object types or to the predicates defined for them). There are two kinds of outgoing arrows for specifying the performance of *epistemic*, *physical* and *communicative actions*. An outgoing arrow with a double arrowhead

² Agent type in RAP/AOR corresponds to Role in ROADMAP.

denotes an epistemic action (changing beliefs). An outgoing connector to an action event type denotes the performance of a physical or communicative action of that type.

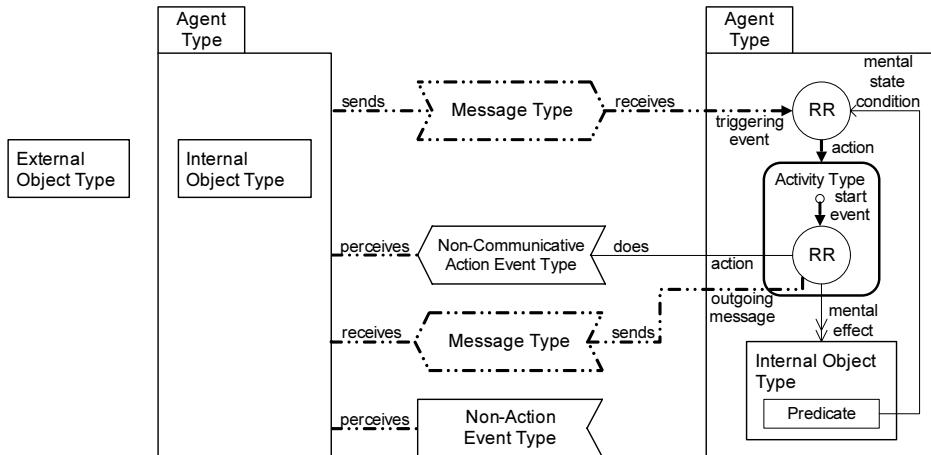


Figure 4. The core mental state structure and behaviour modelling elements of external AOR diagrams.

Reaction rules start activities. Each activity belongs to some *activity type* which we define as a prototypical job function in an organization that specifies a particular way of doing something by performing one or more elementary epistemic, physical and communicative actions in a non-zero span of time by an agent.

There are activity border events of the *start-of-activity* and *end-of-activity* types implicitly associated with the beginning and end of each activity. As Figure 4 reflects, the *start-of-activity* event type is graphically represented by an empty circle with the outgoing arrow to the symbol of the sub-activity type or internal reaction rule.

Figure 5 represents an external AOR diagram that models the scenario of intruder handling. The outermost activity of the scenario is started by reaction rule R1 that is triggered by an action event of the type *move(IntruderDescription)*. This action event is created by an *Intruder* and perceived by the *IntruderHandler*. The precise mechanism of perceiving is of no interest at this stage of a software engineering process. Note that the activity types modelled in the external AOR diagram in Figure 5 correspond to the Goals represented in the Goal Model in Figure 3. In other words, an activity of some type achieves a Goal of the respective type. For example, an activity of the type “Respond” achieves a Goal to respond to the detection of an *Intruder*. For the sake of clarity of Figure 5, the sub-activity type “Inform visitors”, which involves the agent type *Scheduler*, is not refined in the figure. Reaction rule R2 represents checking of the Boolean value of the predicate *isKnown* attached to the object type *Person*. If the predicate evaluates to *false*, that is, if the person described by the *IntruderDescription* is not recognized, an activity of the type “Respond” is started.

Please note that the real scenario of intruder detection is more complicated than the scenario created for simulation purposes that is represented in Figure 5. For example, the real scenario includes checking the house schedule to see if anyone, like a serviceman, has been scheduled to be in the house.

In [4] we have shown how external AOR diagrams can be straightforwardly transformed into the programming constructs of the Java Agent Development Environment (JADE, <http://jade.cselt.it/>) agent platform [7]. This enables rapid

prototyping by executable specifications. As a result of turning the external AOR diagram represented in Figure 5 into the implementation constructs of JADE, we obtained a dynamic model which enables to simulate the scenario of intruder handling and experiment with it. In the model, the agent types IntruderHandler, Police, Owner and Scheduler were represented as the respective software agent types of JADE. The shared object type IntruderDescription and the private object type Person were implemented as the corresponding Java object types and the predicate `isKnown` was represented in the form of a method attached to the object type Person.

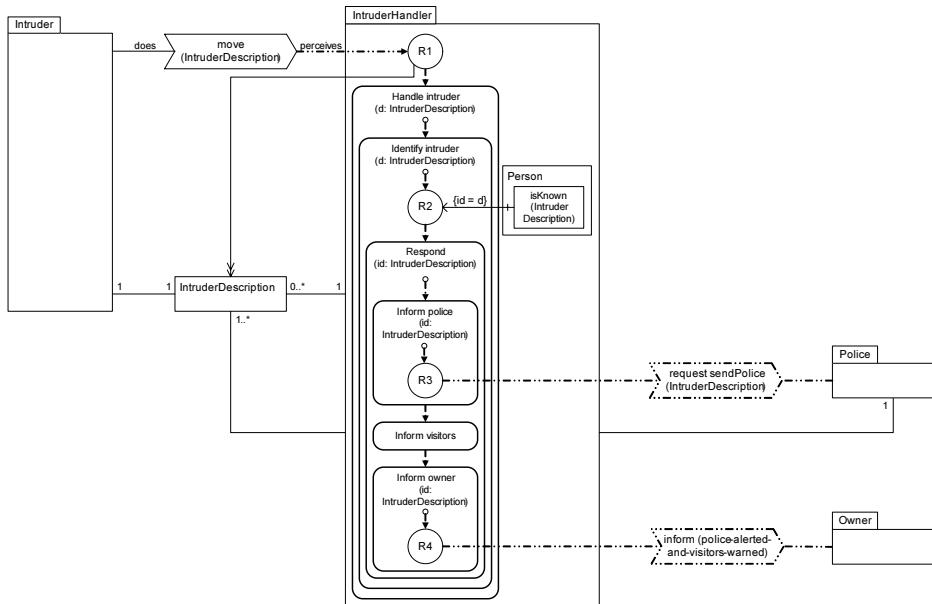


Figure 5. The external AOR diagram of intruder handling.

4. Conclusions and related work

The models produced at our requirements engineering and rapid prototyping stages proved useful for understanding the problem domain. Moreover, it was possible to simulate domain models which helped to understand how the system to be designed should look and function. However, at later stages we found that a real implementation required design decisions that could not be predicted from the requirements engineering phase. In other words, it appears to be hard to achieve the design and implementation of a robust, efficient software system by just extending the models of the problem domain. This seems to be due to software quality goals falling into two separate categories: qualities desired of the functionality of the system, such as timeliness, accuracy and politeness, and qualities desired of the construction of the system, such as robustness, efficiency and scalability. These categories are clearly related, but the relation is neither simple nor obvious. To tackle the problem, we plan to complement the ROADMAP methodology by other types of diagrams enabling to associate quality attributes of the latter type with specific software architectures and platforms. Also, we are currently working on adding features suitable for design from the Prometheus [13] methodology.

The Intelligent Lifestyle project demonstrated the value of rapid prototyping which enables receiving useful feedback from customers at an early stage of a software engineering process. This helps to avoid costly mistaken design decisions.

We believe that different tools are necessary for achieving automated agent-oriented software engineering, including rapid prototyping. This project served as a useful test case for developing the Roadmap Editor Built for Easy development (REBEL) tool [1] for building Goal Models and Role Models.

Acknowledgements

We are thankful to the Australian Research Council (Industry Linkage Project number 0348797), the Smart Internet Cooperative Research Centre (grant number SPA-07), the Adacel Technologies, Clarinox Pty Ltd and to the members of the Agentlab of UoM.

References

- [1] Kuan, P. P., Karunasakera, S., Sterling, L. Improving goal and role oriented analysis for agent based systems. In *Proceedings of the 16th Australian Software Engineering Conference (ASWEC 2005)*, 31 March - 1 April 2005, Brisbane, Australia. IEEE 2005, 40-47.
- [2] Taveter, K., Wagner, G. Towards radical agent-oriented software engineering processes based on AOR modelling. In B. Henderson-Sellers, P. Giorgini (Eds.), *Agent-oriented methodologies* (pp. 277-316). Hershey, PA: Idea Group, 2005.
- [3] Wagner, G. The agent-object-relationship meta-model: Towards a unified view of state and behavior. *Information Systems*, 28(5), 2003, 475-504.
- [4] Taveter, K. *A multi-perspective methodology for agent-oriented business modelling and simulation*. PhD thesis, Tallinn University of Technology, Estonia, 2004 (ISBN 9985-59-439-8).
- [5] Rao, A. S., Georgeff, M. P. BDI agents: From theory to practice. In Victor R. Lesser and Les Gasser (Eds.), *Proceedings of the First International Conference on Multiagent Systems*, June 12-14, 1995, San Francisco, California, USA (pp. 312-319). The MIT Press, 1995.
- [6] Juan, T., Sterling, L. The ROADMAP meta-model for intelligent adaptive multi-agent systems in open environments. In P. Giorgini, J. P. Muller, J. Odell (Eds.), *Agent-Oriented Software Engineering IV, 4th International Workshop, AOSE 2003, Melbourne, Australia, July 15, 2003, Revised Papers* (LNCS, Vol. 2935, pp. 826-837). Berlin: Springer-Verlag, 2003.
- [7] Bellifemine, F., Poggi, A. & Rimassa, G. (2001). Developing multi-agent systems with a FIPA-compliant agent framework. *Software – Practice and Experience*, 31 (2001), 103-128.
- [8] Jennings, N. R., Sycara, K., Wooldridge, M. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1), 7-38, 1998.
- [9] Sterling, L., Taveter, K. and the Daedalus Team. Experience from building industry strength agent-based appliances. *An industry experience report at the Australian Software Engineering Conference (ASWEC 2006)*, Sydney, Australia, April 18-21, 2006. Available as <http://www.cs.mu.oz.au/~kuldar/Daedalus-paper.pdf>.
- [10] Juan, T., Sterling, L., Winikoff, M. Assembling agent oriented software engineering methodologies from features. In F. Giunchiglia, J. Odell, G. Weiss (Eds.), *Agent-Oriented Software Engineering III, Third International Workshop, AOSE 2002, Bologna, Italy, July 15, 2002, Revised Papers and Invited Contributions* (LNCS, Vol. 2585, pp. 198-209). Berlin: Springer-Verlag, 2003.
- [11] Juan, T., Sterling, L., Martelli, M., Mascardi, V. Customizing AOSE methodologies by reusing AOSE features. In J. S. Rosenschein, T. Sandholm, M. Wooldridge, M. Yokoo (Eds.), *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*. Melbourne, Australia, July 14 - 18, 2003. ACM 2003, 113-120.
- [12] Firesmith, D. G., Henderson-Sellers, B. *The OPEN process framework: an introduction*. Sydney; London: Addison-Wesley, 2002.
- [13] Padgham, L., Winikoff, M. *Developing Intelligent Agent Systems*. John Wiley & Sons, 2004.

AOEX: An Agent-Based Exception Handling Framework for Building Reliable, Distributed, Open Software Systems

Susan Entwistle, Seng Loke, Shonali Krishnaswamy and Elizabeth Kendall
Faculty of Information Technology, Monash University, Melbourne, Australia

Abstract: Motivated by the lack of an exception handling mechanism that is adequate to design and implement reliable, distributed, open software systems we introduce our novel agent-based approach to exception handling. We present the basic concepts and state-of-the-art in exception handling research. Based on this a set of requirements are formulated that define the agent-based exception handling framework, known as AOEX. System architecture is proposed for these requirements.

Keywords: Exception, Fault Detection and Resolution, Intelligent Agents

1. Introduction

“The objectives of languages include reliability, readability, formality and even simplicity...have been sacrificed by a plethora of features...many of them unnecessary and some of them, like exception handling, even dangerous.” – [1].

Rapid advances in information technology and consumer demand have motivated the development of increasingly rich, complex software systems that are fundamental to modern society. Within these software systems, exceptions are inevitable and increasing due to the diverse range of potential points of deviations and errors, including hardware, software, and/or human error [2, 3]. Software systems that fail to adequately manage these exceptions often exhibit low dependability potentially resulting in economic losses and/or compromised safety. A critical challenge to creating reliable, distributed, open software systems is allowing them to operate effectively in environments where software exceptions occur.

Much of the exception handling research to date has concentrated on addressing the objectives of exception handling as a language design issue [2], or providing the software developer with exception usage design patterns and software frameworks [4-6]. Subsequently, software developers must anticipate exceptions during design and implement responses using language constructs. In these software systems, the exception handling related code is often numerous and complex [7]. This approach is error prone, time consuming, costly, and inflexible [4, 8, 9]. Consequently, researchers have concluded that an effective exception handling mechanism must be simple, user friendly, and provide a clear separation between an application’s normative and exceptional logic [2]. Drawing on the observation in the quotation by C.A.R Hoare that starts this section, we conclude that significant gains in software reliability can be

achieved through investigating approaches that do not view exception handling as a language design issue.

To address this challenge, we outline in this paper, our unique agent-based approach to exception handling that provides intelligent exception handling capabilities as an framework thus providing an alternative to language based exception handling mechanisms. [10] and [8] have expressed the importance of providing exception handling as a support service to ensure the survival of robust agents in open systems. Our proposed work rests on the same philosophical underpinnings but extends on it, as our focus is not restricted to applications executing on agent platforms but rather aims to intelligently manage exceptions in a non-agent-based application. We believe that an agent-based approach will provide the architecture and implementation to develop an exception handling system that is independent from the application normative processing, in which multiple interacting, intelligent agents can dynamically re-configure using varying co-ordination protocols, as required, to collaboratively anticipate, detect and resolve exceptions.

In this paper, we present our agent-based approach to exception handling. Section 2 of this paper discusses state-of-the-art exception handling research. Drawing on this research, the requirements of our AOEX system are formulated and presented in Section 3. Section 4 discusses the system architecture, focusing on the individual components' functionality and their interactions. Section 5 presents the conclusion and discusses future work.

2. Exception Handling

Dependable software systems exhibit the ability to respond appropriately to failures. This is achieved through the utilisation of an exception handling mechanism, which provides the capability to structure fault-tolerant activities into the software system to manage exceptions. Exceptions are infrequent, abnormal events that interrupt an application's normative processing. Typically, they are restricted to representing errors that result from a failure within the system.

In object-oriented languages such as Java [11], Eiffel [12], and C++ [13], exception handling models, language constructs, and runtime mechanisms are provided that define and support the interaction between the exception signaller, exception handler and client in the event of a failure [2, 14]. To effectively manage exceptional conditions requires detailed exception context information and the explicit declaration of external exceptions in an interfaces specification. [2] and [15] have found that these exception handling requirements often conflict with the object-oriented design goals of modularity, extensibility and encapsulation. This can potentially result in decreased applications reliability, maintainability, and robustness. [2] defines an ideal exception handling model to support the development of dependable, quality object-oriented applications but concluded that no existing language based exception handling mechanism fully addressed the models requirements.

Design patterns and software frameworks offer an alternative technique to enhance software reliability. A diverse range of exception patterns and frameworks [4-6] have been developed to assist the software developer in the design and implementation of robust systems. This approach captures expert exception handling knowledge and promotes design reuse but is limited in that it only covers a small subset of the potential

range of deviations and errors that may be encountered in a distributed, open software system.

[12] proposes the use of contracts to ensure that a software component behaves in accordance with its specification, this is known as Design by Contract (DBC). A contract specifies the pre, post and invariant conditions for a software component; violations of these conditions result in an exception being raised. Design by contract only addresses one of the two facets of software reliability, that being correctness. The robustness of the program is still reliant on the exception handling mechanisms to support software fault tolerance. Subsequently, this approach suffers from the same limitations as language based exception handling mechanisms.

The aspect-oriented paradigm [16] provides a new programming technique to clearly capture and modularise aspects of a design that crosscut a software system. [17] have explored the use of the aspect-oriented paradigm to model and implement exceptions as a separate concern. The assessment concluded that this approach significantly reduced the number of lines of code, and provided greater support for extensibility and design evolution. The identified limitations of this approach are insufficient support to reconstruct local effects, inadequate support for inheritance of semantic preconditions and inadequate support to express the delegation of the exception instance.

Research into agent-based exception handling has focused on the detection of agent commitment violations, known as social exceptions. Several lines of research [18] [8] into detecting these violations have explored the concept of citizen-based approaches to provide exception-handling services. This approach is based on the analogy with human societies, in which agents outsource exception functionality to specialised agents, known as a sentinel or guardian agent, that provides exception handling expertise. The sentinel agent monitors all agent communication and detects diagnoses and responds to commitment violations based on the monitored agent's behaviour and commitment model. Extending on this, [10] proposes that a grid based infrastructure coupled with a set of fundamental services, referred to as terraforming services, are required to ensure the survival of long lived agents, this includes legal services that enforce policies that outline an agents rights and responsibilities.

Investigations [19] [20] [21] [22] into improving fault-tolerance within multi-agent systems through replication have been inspired by natural systems, particularly insect societies [23], and approaches to hardware fault-tolerance. This approach is based on either a passive or active replication strategy, in which multiple redundant copies an agent's state and/or behaviour are replicated on two or more hosts. In the event of agent-failure, a replicate copy of the agent takes over.

Based on the evidence and discussions within this section, we conclude that no existing exception handling approach provides support for generic intelligent exception handling capabilities independent of application domain. We are directly addressing this problem by developing techniques that are based on decomposing exception handling into a problem of specification of behaviour and diagnostics. It is widely acknowledged that monitoring and diagnostics are one of the application areas for agent-based technologies [24]. A distributed multi-agent approach provides the support to design distributed generic, intelligent problem-solving entities. These entities can collaborate to monitor and reason about represented aspects of an application's observed behaviour in order to anticipate or detect deviations from its specified behaviour and respond accordingly. Thus an agent-based approach provides the tools and techniques required to develop a framework to support generic, intelligent

exception handling independent of application domain requiring only minimal configuration and instrumentation for an application.

3. Requirements

We envision a future in which intelligent exception handling capabilities will be provided for open, distributed systems. To deliver greater benefit to users compared to traditional approaches to exception handling, we need to provide the mechanism and tools that allow the intelligent anticipation, detection, diagnosis and resolution of exceptions independent of application-domain. That is, we need to provide support for the following functional requirements:

- ***Application Modeling:*** the means to express or derive an application's normative behaviour.
- ***Application Monitoring:*** mechanism to monitor an application to anticipate pending or detect exceptions based on application and/or system state.
- ***Exception Diagnosis:*** mechanism to diagnose exceptions based on the application and/or system state.
- ***Exception Resolution:*** mechanism to resolve exceptions with minimal software developer and/or end user intervention.

To realise our vision requires the design of a generic intelligent exception-handling framework that supports the following non-functional requirements:

- ***Ease of use:*** concepts, architecture, and the tools must be understandable and affordable.
- ***Portability:*** architecture and specification should support interoperability, and programming language and runtime independence.
- ***Extensibility:*** support extending and adapting of the exception handling requirements for heterogenous application domains.
- ***Reliability:*** respond appropriately to abnormal conditions that occur within the system.
- ***Resource Efficiency:*** protocols and algorithms should minimise use of computing resources e.g. power consumption, network traffic, etc.

4. System Architecture

The key design objective of AOEX is to provide an extensible, domain-independent framework that supports intelligent exception handling for an application. To achieve this we propose that our intelligent agents represent and reason about an application using a model-based approach and a set of general principles that capture reusable domain independent exception handling expertise. The model of the application provides an abstract representation of its normative behaviour by specifying the contractual characteristics, such as behavioural characteristics and affective relations, for its software components and services. Reasoning is then performed using a rules-based approach to manage software exceptions.

As illustrated in Figure 1 AOEX employs a multi-layered architecture, abstractions and key services facilitating the leveraging of existing technologies and promoting modularity. The first layer is concerned with the hosting environment that will provide

the runtime machinery and agent platform providing the runtime infrastructure for the AOEX framework. The second layer is concerned with the domain independent monitoring, diagnostic and resolution agents that utilise the exception handling knowledge base, and providing a toolkit to model software components and services supporting the creation of domain-specific application models. The final layer is concerned with the instrumentation of an application, and the application modeling that specifies the normative behaviour of its software components and services based on its exception handling requirements using the services provided by the lower layers.

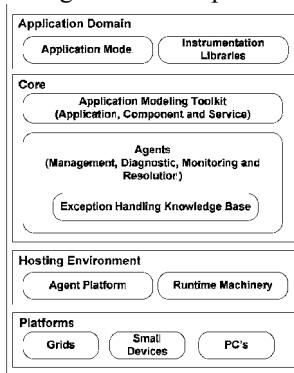


Figure 1. Overview of AOEX layered architecture

The core layer (layer two) is comprised of intelligent agents that monitor the behaviour of an application's components and services through sensors. The sensors provided information on the application's services and runtime components interfaces. Reflective programming techniques can be used to dynamically emit the methods that provide this information. The observed behaviour is compared to the application model of the expected behaviour, deciding what action to take through the agent's effectors, if any. This approach creates a closed control loop. Figure 2 illustrates AOEX multi-agent architecture to support this model.

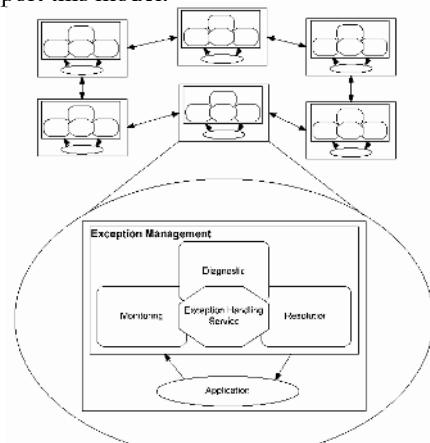


Figure 2. AOEX multi-agent architecture

The following sub sections discuss this architecture in further detail.

4.1. Application Model

The application model provides a structured mechanism to represent knowledge about the application to assist in managing exceptions. This knowledge is defined within an ontology thereby supporting a common vocabulary for the key concepts and relationships of the application model. This includes 1) a set of base classes (e.g. application), 2) software component and services behavioural characteristics (e.g. contracts, quality of service, synchronisation), 3) affective relations between the software components and services, and 4) exception resolution procedures.

The application is represented as the APPLICATION element:

- **PROCESS_NAME:** specifies the physical name of the application's process.
- **HARDWARE_REQUIREMENTS:** specifies a set of hardware requirements that this application requires to perform correctly.
- **HARDWARE_CONSTRAINTS:** specifies a set of hardware resource constraints for this application. This element includes attributes such as maximum memory, maximum CPU time, maximum disk capacity, etc.
- **SOFTWARE_REQUIREMENTS:** specifies the list of software that this application is dependent on to perform correctly.

The behavioral characteristics and affective relations between components and services of the application are represented within the COMPONENT element:

- **DEPENDENT_ON:** specifies the components and/or services that this component is dependent on to perform correctly.
- **PART_OF:** used to specify the parent component or service indicating a hierarchical relationship between components and/or services, if required.
- **BEHAVIOURS:** defines the functional behaviours of the component or service. This element includes attributes such as method parameters types, threshold values, expected return value, etc.
- **QUALITY_OF_SERVICE:** defines non-functional quality of service characteristics of a software component or service. This element includes attributes such as the maximum response time, transaction rate throughput, transaction integrity, security, reliability, etc.
- **SYNCHRONISATION:** defines the concurrency requirements of a software component or services. This element includes attributes such as mutual exclusion, locking strategy, etc.

A model of the application can be defined using the application modeling toolkit within AOEX.

4.2. Exception Handling Knowledge Base

We have developed an exception handling knowledge base that generically represents the exception for an application, their characteristic exceptions types, and handlers to resolve them. As illustrated in Figure 3 and 4 this has been achieved by focusing the exception domain taxonomies on the contractual aspects that cut across an application domain.



Figure 3. Overview of exception handling schema

Figure 4 illustrates a subset of the exception type's taxonomy for the contracts exception domain.

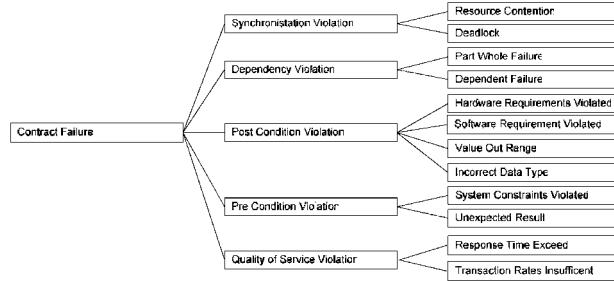


Figure 4. Overview of exception types taxonomy

Figure 5 illustrates a subset of the exception handler taxonomy for the contracts exception types.

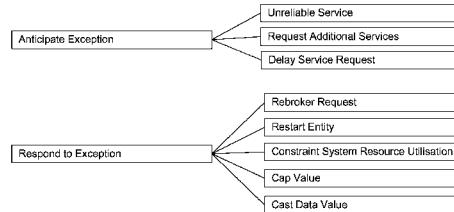


Figure 5. Overview of exception handler taxonomy

The exception handling schemas use of inheritance results in a simplified structure to identify, reuse and refine exception types and their associated handlers.

We can use monitoring and diagnostic rules that reason based on the application model using general principles. The following informally describes some of these rules:

1. When an application, software component or service depends on another application, software component or service that is not functional then it is also not functional.
2. When an application, software component or service is a part of another application, software component or service and it is not functional then the parent is not functional.
3. When an application, software component or service fails to adhere to a pre-condition the service provider (e.g. application software component or service) will not function.
4. When an application, software component or service fails to achieve its post-condition the service request (e.g. application software component or service) will not function.

The informal general principles presented above can be precisely represented as a mathematical notion for reasoning about the state of application. Where Entity denotes an application, component or service.

$$\forall x,y:\text{Entity} \bullet (x,y) \wedge \text{functional}(y, \text{false}) \vee \text{functional}(x, \text{false});$$

4.3. Agent Organisation

The agents within the AOEX framework are organised into societies that collaborate to solve an exception for a particular subsystem of the application. This approach enables the agent organisation to map to federated and hierarchically structured applications. Communication between societies is enabled through the use of directory services that provide the required information to facilitate communication between this and other societies. All agents within AOEX framework co-operate via asynchronous event based messaging to solve exceptions, typically based on a rules-based approach, storing the result in the standard exception object.

4.4. Monitoring Agent

The monitoring agent provides the interface between the application and the AOEX framework. When an application using the AOEX framework commences execution a monitoring agent is assigned to a group of interdependent software components and services as defined in the application model. Its role is to perform measurements and/or mediate communication in order to detect exceptions.

The monitoring agent includes a repository of exception handling expertise and relevant information from the application model for the monitored group of software components or services. Software components and services, for their part, must implement an interface enabling the monitoring agent to intercept dispatched method invocations and responses. This enables the development of a model of the interactions that the software component and/or service is involved in. This model is then used to compare the actual and expected behaviour to anticipate and detect exceptions.

If the monitoring agent detects an exception it raises an event to notify the diagnostic agent. The event encapsulates an exception object providing relevant exception context information to the diagnostic agent.

4.5. Diagnostic Agents

The diagnostic agent is required to determine the root cause of an exception. When an application commences execution the diagnostic agent registers to receive notifications of exceptions from a set of monitoring agents. The diagnostic agent determines the root cause of an exception based on the information contained within the exception object, the application state information, and a set of diagnostic tools. Domain specific diagnostic agent will need to be developed to extend AOEX dependent on the application's reliability requirements. For example, diagnostic agents could be developed to manage database errors, security errors, errors within a specific industry domain (e.g. finance, healthcare, etc).

Each diagnostic agent is comprised of three primary components: a world model, a goal-based rule set, and a set of diagnostic tools. The agent's world model contains knowledge about the application's exception history and inaccessible application state information, which is provided by the distributed diagnostic agents. This multi-agent approach enables the diagnostic agent to develop a broad view of the application, which can be used as input into and verification of the proposed diagnosis. The goal-based rule set uses a classifier approach to diagnostics. That is a hypothesis on the root cause of the exception is generated based on the diagnostic agents' exception handling expertise, the exception object and its world model. This process is repeated until either

the cause of the exception is determined or all potential causes have been investigated. Upon completion of this process, the diagnostic agent updates the exception object with the diagnostic results and raises an event to notify the resolution agent.

4.6. Resolution Agent

The resolution agent is responsible for responding to diagnosed exceptions. That is it acts on the applications' components and/or services in order to restore it to a valid state. The AOEX framework supports two types of resolution agents. They are the domain independent and domain dependent resolution agents. These agents register to receive notifications of specific diagnosed exception types when the application commences execution.

The resolution agent can be based on either a resumption or termination exception handling approach. If a restart approach is used the software components or services, for their part, must provide a mechanism to restore its state information. Regardless of the exception handling approach the resolution agent interfaces into the application through a standard interface defined by the AOEX framework. This interface provides the mechanism for the agents to modify the application state or take the necessary action to resolve the exception.

The AOEX framework provides the domain independent resolution agents. They provide standard response for the various exception types, such as restart service, broker service request, etc. In contrast the domain dependent resolution agent provides the capability to plug in customised exception responses tailored specifically to the application domains exception handling requirements.

5. Conclusions and Future Work

We have presented in this paper an agent-based framework, called AOEX, for intelligently managing exceptions independent of the application domain. Requirements for AOEX were described and system architecture developed.

Future work for this research includes the following categories:

- Implementation and evaluation of a prototype of the AOEX system architecture.
- Further development of the exception handling knowledge base to support a wider range of exception scenarios, including concurrent exceptions, anticipating exceptions, etc.
- Investigate approaches to simplify the creation of the application model. This could potentially be achieved through automatically building the application model based on a static analysis of the application's source code and its observed behaviour at run-time.
- Investigation of additional support services to assist in diagnosis, such as a reliability module that maintains and reports on software components and/or services reliability.

References

- [1] C. A. R. Hoare, "The emperor's old clothes," *Communications of the ACM*, vol. 24, pp. 75--83, 1981.
- [2] A. F. Garcia, C. M. F. Rubira, A. Romanovsky, and J. Xu, "A Comparative Study of Exception Handling Mechanisms for Building Dependable Object-Oriented Software," *The Journal of Systems and Software*, vol. 59, pp. 197--222, 2001.
- [3] D. A. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, and N. Treuhaft, "Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies," UC Berkeley Computer Science Technical Report UCB//CSD-02-1175 March 15 2002 2002.
- [4] D. Reimer and H. Srinivasan, "Analyzing Exception Usage in Large Java Applications," *Exception Handling in Object-Oriented Systems Workshop at ECOOP 2003*, 2003.
- [5] T. Anderson, M. Feng, S. Riddle, and A. Romanovsky, "Error Recovery for a Boiler System with OTS PID Controller," *Exception Handling in Object-Oriented Systems Workshop at ECOOP 2003*, 2003.
- [6] A. F. Garcia and C. M. F. Rubira, "An Exception Handling Software Architecture for Developing Robust Software," *2th Exception Handling in Object-Oriented Systems Workshop at ECOOP'2000*, 2000.
- [7] F. Cristian, "Exception Handling," *Dependability of Resilient Computers*, pp. 68-97, 1989.
- [8] M. Klein and C. Dellarocas, "Domain-Independent Exception Handling Services That Increase Robustness in Open Multi-Agent Systems," *M. Klein and C. Dellarocas, Domain-Independent Exception Handling Services That Increase Robustness in Open Multi-Agent Systems, Massachusetts Institute of Technology, Cambridge MA USA, Center for Coordination Science Working Paper CCS-WP-211, <http://ccs.mit.edu/papers/pdf/wp211.pdf>, May 2000 2000*, 2000.
- [9] C. Howell and G. Vecellio, "Experiences with Error Handling in Critical Systems," vol. Advances in Exception Handling Techniques in Object-Oriented Systems Workshop at ECOOP 2000, 2000.
- [10] J. M. Bradshaw, "Terraforming cyberspace," presented at Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium on, 2001.
- [11] K. Arnold and J. Gosling, *The Java Programming Language*, Second Edition ed: Addison Wesley, 1998.
- [12] B. Meyer, *Object-oriented software construction*. New York: Prentice-Hall, 1988.
- [13] B. Stroustrup, *The C++ Programming Language*, Third Edition ed: Addison-Wesley, 1997.
- [14] P. A. Buhr and W. Y. R. Mok, "Advanced Exception Handling Mechanisms," *Software Engineering, IEEE Transactions on*, vol. 26, pp. 820-836, 2000.
- [15] R. Miller and A. Tripathi, "Issues with Exception Handling in Object-Oriented Systems," *Lecture Notes in Computer Science*, vol. 1241, pp. 85--??, 1997.
- [16] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-Oriented Programming," presented at European Conference for Object-Oriented Programming, Jyväskylä, Finland, 1997.
- [17] M. Lippert and C. V. Lopes, "A study on exception detection and handling using aspect-oriented programming," in *Proceedings of the 22nd international conference on Software engineering*: ACM Press, 2000, pp. 418--427.
- [18] S. Hagg, "A Sentinel Approach to Fault Handling in Multi-Agent Systems," *Hägg S., A Sentinel Approach to Fault Handling in Multi-Agent Systems, in Proceedings of the Second Australian Workshop on Distributed AI, in conjunction with the Fourth Pacific Rim International Conference on Artificial Intelligence (PRICAI'96), Cairns, Australia, August 27, 1996.*, 1996.
- [19] A. Fedoruk and R. Deters, "Improving Fault-tolerance in MAS with Dynamic Proxy Replicate Groups," presented at International Conference on Intelligent Agent Technology, Halifax, Canada, 2003.
- [20] Z. Guessoum, J. P. Briot, S. Charpentier, O. Marin, and P. Sens, "A fault-tolerant multi-agent framework," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*: ACM Press, 2002, pp. 672--673.
- [21] S. Mishra and Y. Huang, "Fault Tolerance in Agent-Based Computing Systems," presented at 13th ISCA International Conference on Parallel and Distributed Computing Systems, Las Vegas, NV, USA, 2000.
- [22] F. B. Schneider, "Towards Fault-tolerant and Secure Agentry," presented at 11th Int.~Workshop on Distributed Algorithms ({WDAG} '97), 1997.
- [23] E. Bonabeau and G. Théraulaz, "Swarm Smarts," *Scientific American*, pp. 72-79, 2000.
- [24] I. A. Letia, F. Craciun, Z. Kop, and A. Netin, "Distributed Diagnosis by BDI Agents," presented at IASTED International Conference Applied Informatics, Innsbruck, Austria, 2000.

Agent-based modeling and simulation of network softbots' competition

Igor KOTENKO¹ and Alexander ULANOV¹
SPIRAS, Computer Security Research Group, Russia

Abstract. The research devoted to design and implementation of new knowledge-based methods and tools for verification and validation of complex software systems is now an important direction of scientific investigations. The paper describes the approach and software environment developed for agent-based modeling and simulation of defense against coordinated distributed attacks in the Internet. According to this approach, the cybernetic opposition of malefactors and defense systems in the Internet is represented by competition of antagonistic softbots' teams. The possibility of the approach application is analyzed by testing the defense mechanisms against Distribute Denial of Service attacks.

Keywords. Intelligent agents and softbots, knowledge-based methods and tools for testing, verification and validation, computer network security

Introduction

Nowadays one of very important research directions is connected with *development of new knowledge-based methods and tools for comprehensive investigation of complex systems (including testing, verification and validation of software components)*. It is especially true for the systems which operate in *distributed competitive noisy environments*, where the large groups of various defense and offense components counteract and collaborate (for example, in such domains as information warfare, competition in business, computer network assurance, etc.).

One of the solutions of this problem can be based on the *investigative modeling and simulation* using the family of various approaches (system dynamics, discrete-event and agent-based simulation, etc.) and different models (from analytical to scaled-down and full-scale). The choice of specific approaches and models depends on the goals of investigation, the complexity of subject domain, and the necessary fidelity and scalability of models. Our interest is related to the competition processes taking place in the Internet consisting in interaction of large number of cooperating and antagonistic software agents or softbots. Analytical models let imitate global processes (including viral epidemic), but describe the ones only on the abstract level. Packet-level simulation of network processes gives the opportunities to improve the fidelity of simulation, and can represent attack and defense actions as packet exchange. Those models can precisely specify these actions on the data link, network, transport and application layers. The greatest fidelity is archived with hardware testbeds. But these testbeds succeed in simulation of sufficiently limited fragments of interactions.

¹ St. Petersburg Institute for Informatics and Automation, 39, 14th Liniya, St. Petersburg, 199178, Russia;
E-mails: ivkote@iias.spb.su, ulanov@iias.spb.su

The paper describes an *integrated agent-oriented and packet-level approach* for simulation of competition processes in the Internet elaborated by authors. We suggest using this approach for investigation of distributed defense mechanisms which can be deployed in the Internet for protection from distributed coordinated network attacks. The possibility of the approach application is analyzed by testing the defense mechanisms against one of the most dangerous classes of network attacks – DDoS (Distribute Denial of Service) [22]. According to the approach suggested, the cybernetic counteraction of “bad guys” and security systems is represented by the *interaction of different softbots’ teams*. The aggregated system behavior becomes apparent by means of the local interactions of particular softbots in dynamic environment that is defined by the model of computer network. We distinguish at least the team of malefactors and the defense team. The softbots from the same team collaborate to achieve the joint intention (to realize the threat or to defense the network).

The main basis for the research is the agent teamwork theory. There are three well-known approaches to the formalization of the agent teamwork – joint intentions theory [5], shared plans theory [12] and the hybrid approaches [14, 26] which use the combination of joint intentions and shared plans theories. A lot of teamwork approaches are implemented in various multi-agent software, e.g. GRATE*, OAA, CAST, RETSINA-MAS, COGNET/BATON, Team-Soar, etc.

Another fundamental component of the research is represented by the studies on reasoning systems about opponent intentions and plans on the basis of current situation estimation [3, 16, 29, 30]. There were published the studies on determining the malefactor’s plans during the intrusion detection [8, 10]. It is proposed to use the ideas of agent plans recognition on the basis of stochastic formal grammar recovery algorithms [11]. The important components in this research are the methods of reflexive processes theory [20], game theory [25] and control in conflict situations [6]. Authors used the methods of agent actions scenario specification based on the stochastic attributive formal grammars [11]. They correlated with colonies of cooperative distributed grammars and grammar models of multi-agent systems [17]. As teams are to adapt to reconfiguration, traffic changes and new types of defense and attacks on the basis of past experience it is important to take into account the present studies in the area of adaptation and self-learning [1, 13].

The rest of the paper is structured as follows. *Sections 1 and 2* outline suggested approach for modeling and simulation. *Section 3* describes the software environment developed for simulation. *Section 4* presents one of simulation scenarios fulfilled. *Conclusion* outlines the main results of the paper and future work directions.

1. Modeling and Simulation Approach

It is assumed the competing softbots gather information from different sources, operate with fuzzy (or probabilistic) knowledge, forecast the intentions and actions of opponent, estimate the possible risks and try to deceive each other, and react on opponent’s actions. The choice of behavior for each team depends on the chosen goal of functioning. The choice of every step of a team behavior is defined dynamically depending on the opposite team actions and the state of environment.

Each team acts in the conditions of limited information. Every team member might have different information about actions done by other team members. Therefore the model of behavior must be able to represent the incompleteness of information and the

possibility of accidental factors. The softbots are to foresee, what each softbot knows, what task has to be solved and to which one it must address its request if it is outside of its competence. The messages of one softbot are to be represented in such terms that are understandable by others.

The use of ontologies is the one of the most perspective approaches to structure the distributed knowledge. As for every application domain the information security ontology represents the partially normalized set of notions that are to be used by other softbots. Besides the relation of partial order the nodes of this structure have other relations peculiar to the application domain. The ontology defines the subset of notions, that various softbots use to solve tasks stated. Each softbot uses a fragment of application ontology. Each softbot specialization is represented by a subset of ontology nodes. Some of nodes can be shared by several softbots. Usually only one of them has the detailed description of a node. This softbot is the owner of the corresponding fragment of knowledge base. At the same time some part of ontological knowledge base is shared for all team. This part is the fragment that is to be the shared context.

The team of malefactors evolves with the aid of generation of new instances and types of attacks and attack scenarios. The defense team adapts by changing the security policy, forming new instances of defense methods and security profiles.

The softbots' counteraction model includes: (1) Ontology of application domain containing application notions and relations between them; (2) protocols of teamwork (for malefactors' and defense team); (3) Models of individual, group and team behavior; (4) Communication component for message exchange; (5) Models of environment (computer network), including topological and functional components.

The approach for teamwork proposed in the paper is based on the joint use of the elements of joint intentions theory, shared plans theory and hybrid approach. The teamwork is assumed to be organized due to the shared (group) plan of actions [19]. The structure of team is described in terms of group and individual roles hierarchy. The leaves of hierarchy correspond to the roles of individual softbots, the intermediate nodes – to the group roles. The specification of action plans hierarchy is made for every role. The following elements are defined for every plan: initial conditions, when the plan is offered for fulfillment; the conditions with which the plan stops being fulfilled; actions executed on the team level as the part of the shared plan. The joint activity is obviously expressed for the group plans. Softbots can create the “snapshots” of mental state of the whole team due to forming of joint intentions on the different abstract levels. The mechanisms of softbot interaction and coordination are based on the three groups of procedures [26, 19]: (1) the providing acts consistency; (2) softbots' functionality monitoring and recovery; and (3) communication selectivity support (to choose the most “useful” communication acts).

2. Attacks and Defense Softbots

The idea of *DDoS attack* consists in reaching the global goal – the denial of service of some resource – due to joint efforts of many components that are acting on attack side. In that way the initial goal is divided into more simple sub-goals. They are given to particular softbots. At the same time the goal on the top level stays shared between softbots. On the low level, the local goals are formed. Their achievement is targeted on solving the shared task. The softbots interact with each other to coordinate local solutions. This is necessary to reach the shared goal “denial of service”.

Generally, the components of attack system are the programs which have the following features: autonomy; initial knowledge about itself, interacting entities and environment; knowledge (or hard-coded algorithm) that allows to process the external data from environment; the presence of a goal and a list of actions to reach this goal; the communication and interaction mechanisms (protocols) to reach the shared goal.

Analyzing the present methods of DDoS realization it is possible to determine at least two types of attack softbots: “Daemon” – it executes the attack directly; “Master” – it coordinates the actions of other system components. So the attack team is a two-level system. Masters act on the higher level directly fulfilling the malefactor’s tasks. They make decisions: when to start the attack, what target to attack, what is the attack intensity. Masters coordinate daemons’ actions by sending commands. Daemons act on lower level. After receiving the messages from masters, they start or finish sending the attack packets or change the attack intensity.

On the preliminary stage the master and daemons are deployed on available (compromised) hosts in the Internet. Then the attack team is established: daemons send to master the messages saying they are alive and ready to work. Master stores the information about team members and their state. The malefactor sets the common goal of team – to perform DDoS attack. Master receives attack parameters. Its goal is to distribute these parameters among all available daemons. Then daemons act. Their local goal is to execute the master command. To fulfill attack they send the attack packets to the given host. After this it is believed that the goal is reached. Master asks daemons periodically to find out that they are alive and ready to work. Receiving the messages from daemons the master manages the given rate of attack. If there is no any message from one of the daemons the master makes the decision to change the attack parameters. For example, it can send to some or all daemons the commands to change the attack rate. Daemons can execute the attack in various modes. This feature affects on the potentialities of defense team. Daemons can use different types of attacks, send attack packets with various rates, spoof source IP address and do it with various rates. Malefactor can stop the attack by sending to master the command “stop the attack”. Then master distributes this command among all daemons, and they stop the attack.

The main task of *defense systems against DDoS* is to accurately detect attacks and quickly respond to them [31]. It is equally important to recognize the legitimate traffic that shares the attack signature and deliver it reliably to the victim [21]. *Traditional defense* include detection and reaction mechanisms. Different network characteristics are used for detection of malicious actions (for example, source IP address [24], traffic volume [9], and packet content [23]). To detect abnormal network characteristics, many methods can be applied (for instance, statistical, cumulative sum, pattern matching, etc.). As a rule, the reaction mechanisms include filtering, congestion control and traceback. But, as a result of several reasons (detection of DDoS attack is most accurate close to the victim, separation of legitimate is most successful close to the sources, etc.), adequate victim protection to constrain attack traffic can only be achieved by *cooperation of different distributed components* [21]. So, the DDoS problem requires a distributed cooperative solution [2, 4, 23, 18, 32, 31, 21, etc.].

The analysis of present DDoS defense systems shows the following their features: The defense systems are built of basic components which have some local meaning but serve together for common shared goal; The number and functionality of defense system components depend on the place of their deployment; As a rule, the defense systems have a hierarchical structure, where different levels serve for particular sub-tasks of the complex defense goal. The general approach to the DDoS defense is the

following. The information about normal traffic is collected from different network sensors. Then the analyzer-component compares in real-time the current traffic with the normal traffic. The system tries to trace back the source of anomalies (due to “traceback” mechanisms) and generates the recommendations how to cut off them or how to lower the quantity of these anomalies. Depending on security administrator’s choice, the system applies some countermeasure. We set the following defense softbot classes: “Sensor” – for initial information processing; “Sampler” – the network data collector that forms the traffic model; “Detector” – for attack detection; “Filter” – for attack traffic filtering; “Investigator” – for attack investigation. *Sensor* processes information about network packets and collects statistic data on traffic for defended host. *Sensor* determines the size of overall traffic (*BPS – bit per seconds*) and the addresses of n hosts that make the greatest traffic (in developed prototype – all hosts). Its local goal is to give these parameters to detector every k seconds. *Samplers* are deployed in the defended subnet to collect the data on its normal functioning. Using this data they can detect anomalies. The examples of methods which can be realized by sampler are Hop counts Filtering (HCF) [15], Source IP address monitoring (SIPM) [28], Bit per Second (BPS), etc. *Detector* local goal is to make the decision if the attack happens. It sends its decision and N addresses to filter and to investigator. *Filter* local goal is to filter the traffic on the basis of data from detector. If it was determined that the network is under attack, filter begins to filter the packets from the given hosts. The goal of *investigator* is to identify and defeat the attack softbots. When investigator receives the message from detector it examines the given addresses on the presence of attack softbots and tries to defeat identified softbots.

3. Simulation Environment

The simulation environment architecture consists of the following components (Fig.1): OMNeT++ Framework, INET Framework, Multi-agent & DDoS Framework. Agent-based simulation is implemented in Multi-agent Framework that uses the library of attack and defense mechanisms called DoS Framework. INET Framework is used to simulate the IP nodes. It is an OMNeT++ model itself.

OMNeT++ Framework [27] is a discrete event simulator. Its primary application area is the simulation of computer networks and other distributed systems. Simulation models are composed of hierarchically nested modules that interact due to message passing (Fig.1, OMNeT++ Framework: simulation model & component library). Module functionality is programmed using C++, while the model structure is defined by the special topology description language. INET Framework and Multi-agent DDoS Framework are the OMNeT++ models. The exchange of messages between modules happens due to channels (modules are connected with them by the gates) or directly by gates. A gate can be incoming or outgoing to receive or to send messages accordingly. Channel is characterized by propagation delay, bit error rate and transmission data rate. OMNeT++ INET Framework is the OMNeT++ modular simulation suite with a realistic simulation of Internet nodes and protocols. The highest IP simulation abstraction level is the network itself, consists of IP nodes. IP node can represent router or host. IP node in INET Framework corresponds to the computer representation of Internet Protocol (Fig.1, INET Framework). Multi-agent & DDoS Framework is the INET Framework modular suite aimed to simulate the DDoS attack and defense mechanisms on the basis of team counteraction (Fig.1, Multi-agent DDoS Framework).

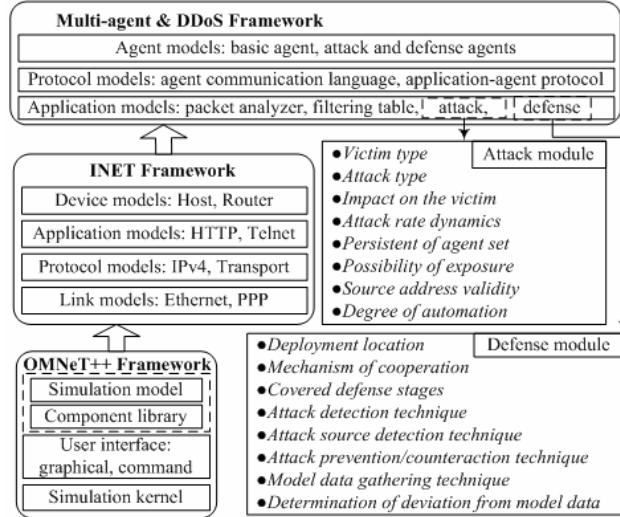


Figure 1. Simulation environment architecture

DDoS Framework suite consists of DDoS attack and defense modules (Fig.1, Attack module, Defense module) and the modules that expand IP node from INET: the filtering table and the packet analyzer. Attack and defense modules are the applications and are deployed on the network layer of IP node. To set the DDoS attack conditions it is necessary to define the corresponding *input parameters*, including victim type (host), attack rate dynamics (function of attack packets sending rate), spoofing technique (no spoofing, random, subnet), etc. Also one need to set up the defense parameters, including deployment location (defended, intermediate, source subnet), detection technique, model data gathering technique and its parameters (time interval and time shift of data collection), etc. The examples of *output parameters* used to estimate the defense are as follows: Time of attack detection; Time of attack reaction (time from detection to counteraction); Percent of false positives; Percent of false negatives; Percent of normal traffic filtration; Computational complexity, etc.

Agent Framework consists of modules representing softbots implemented as applications. There were used the elements of abstract FIPA architecture [7] during softbot modules design and implementation. Agent communication language is implemented for softbot interactions. The message passing happens above the TCP protocol (transport layer). Softbot can control the other modules (including DDoS Framework modules) due to messages. Softbots are deployed on the hosts in the simulation environment. Their installation is fulfilled by connecting to the modules serving transport and network layers of protocol stack simulated in INET Framework.

The example of multi-window user interface of the simulation environment is depicted in Fig.2. The window for simulation management (at the bottom of Fig.2, at right) allows looking through and changing simulation parameters. It is important that you can see the events which are very valuable for understanding attack and defense mechanisms on time scale. Corresponding status windows (on top of Fig.2, in the middle and at left) show the current status of teams. It is possible to open different windows which characterize functioning of particular hosts, protocols and softbots, for example, at the bottom left of Fig.2, the window of one of the hosts is displayed.

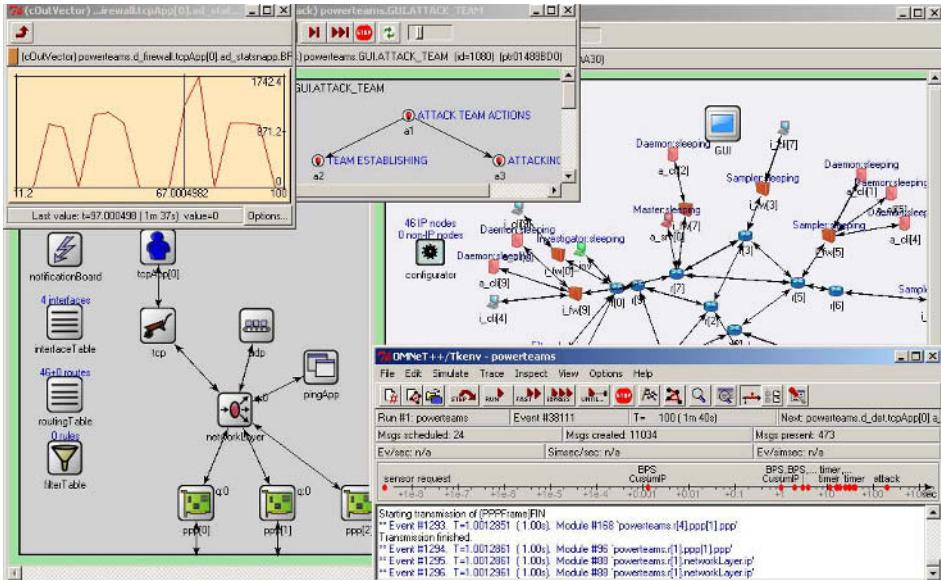


Figure 2. Common representation of simulation environment

At the basic window of visualization (Fig.2, at upper right), a simulated computer network is displayed. The network represents a set of hosts and channels. Hosts can fulfill different functionality depending on their parameters or a set of internal modules. The routers are labeled with the sign “”. Attack softbots are deployed on the hosts marked with red color. Defense softbots are located on the hosts marked with green color. Above the colored hosts there are the strings that indicate the corresponding state of deployed softbots. The other hosts are the standard hosts that generate generic traffic.

Each network for simulation consists of three sub-networks: (1) the subnet of defense where the defense team is deployed; (2) the intermediate subnet where the standard hosts are deployed. They produce the generic traffic in the network including the traffic to defended host; (3) the subnet of attack where the attack team is deployed.

4. Simulation Example

Learning mode. The main task of learning mode is to create the model of generic traffic. The clients send the requests to the server and it replies. At this time sampler analyses requests and uses them to form the models and parameters. Fig.3 depicts the change of new addresses amount for sampler during first 300 seconds of learning. Fig.4 represents the graph of change of maximum BPS (for interval 10 seconds and shift 3 seconds) after 300 seconds from the beginning of learning.

Decision making and acting. Simulation scenario is realized on the same configuration as was used during learning. The only difference – the attack team is engaged. Attack initial parameters are as follows: target of attack is server d_srv; intensity of attack – 5 packets per sec); no IP spoofing is used.

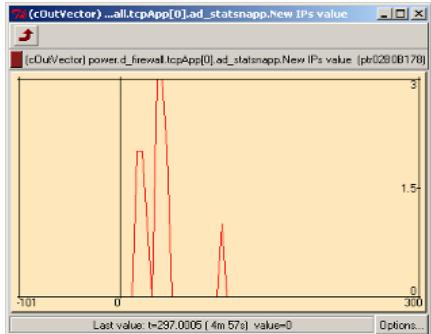


Figure 3. Change of new IP addresses amount

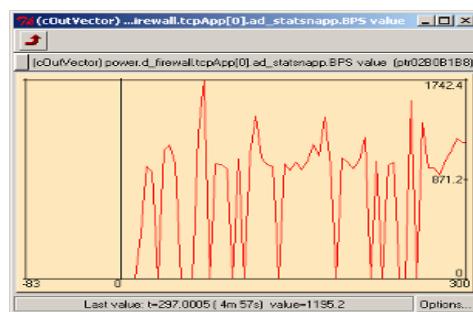


Figure 4. Change of BPS parameter

Fig.5 represents the graphs of channel throughput (bits/s to sec) on the entrance to the defended network before (red) and after (blue) filter. After simulation start the clients begin to send requests to the server and it replies. This is the way the generation of generic network traffic takes place (Fig.5, interval 0 – 300 sec). After establishing the defense team begins to function. Sampler collects traffic data and compares it with the model data that it acquired during learning mode. The addresses that are the source of anomalies are sent to detector. Detector makes the decision about the attack and sends to filter and investigator the addresses of suspicious hosts.

After 300 sec from simulation start the attack team begins attack actions. When daemons receive the attack command they begin to send the attack packets (Fig.5, timestamp 300 sec). After a while, sampler determines the suspicious hosts with the use of BPS method. The BPS parameter of these hosts exceeds normal. Detector receives the addresses of these hosts from sampler and sends them to filter and investigator. Filter sets the filtering rules and the packets from the given hosts begin being dropped (Fig.5, timestamps 400 – 600 seconds, blue graph).

Investigator tries to inspect the given hosts and to defeat the attack softbots deployed there. It succeeds in defeating of 4 daemons. However the other daemons continue the attack (Fig.5, after 400 seconds, red graph). Master makes the decision to redistribute the intensity of attack to keep the overall intensity on the given level. Also it decides to change the method of IP spoofing to complicate the detection and defeating of attack softbots by defense team. Master sends to alive daemons the command: target – d_srv, target port – 2001, intensity – $5/(10-4)=0.83$, IP spoofing method – “random”. When daemons receive the command they continue to send the attack packets having applied the new parameters (Fig.5, timestamp 600 sec).

Detector sees that the input channel throughput has noticeably lowered since the traffic from attack team has raised (Fig.5, after 600 sec). Detector does not receive the anomaly report from sampler though. This is because the method BPS used by sampler does not work fine when attacker changes the sender address in every packet. That is the reason detector fails to confront some address with the big traffic. Therefore detector decides to apply another DDoS defense method – SIPM. The large amount of new IP addresses for sampler will lead to attack detection and dropping of malicious packets. This method however does not allow tracing the source of attack and investigator will fail to defeat attack softbots. But the attack packets will be filtered and the traffic in the subnet of defended host will return to normal state.

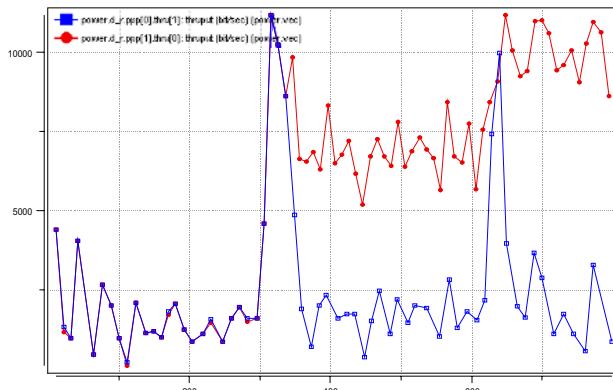


Figure 5. Graphs of channel throughput

5. Conclusion

The main results of the paper consist in developing a new agent-based approach for comprehensive investigation of defense mechanisms against distributed coordinated attacks in the Internet and implementing the software environment (written in C++ and OMNeT++) intended for simulation of DDoS attacks. We tried to model and simulate counteraction between malefactors and defense systems in the Internet as competition between teams of softbots. The main attention was drawn to the application of packet-based imitation of softbots' interaction which provides the acceptable fidelity and scalability of computer attack and defense mechanisms representation. The goal of the paper is not to present an investigation of new defense methods, but to show the possibilities of the approach suggested and the knowledge-based simulation tool developed. One of the features of this tool is the possibility to insert new attack and defense methods and investigate them. So the approach suggested and the environment implemented can be extended for investigation of other classes of attacks.

The approach was examined on the example of "Distributed Denial of Service" attacks and defense simulation. We considered different phases of operations of antagonistic softbots' teams – learning, decision making and counteracting, including adaptation of one team to the actions of opposite team. Various experiments with this environment have been fulfilled. These experiments include the investigation of attack scenarios and protection mechanisms for the networks with different structures and security policies. One of the scenarios was demonstrated in the paper.

Future work is connected with developing an expressive formal framework for specification of softbots' competition and collaboration in the Internet, building a more powerful simulation environment, investigating new defense mechanisms, and conducting experiments to both evaluate new large-scale network defense mechanisms and analyze the efficiency and effectiveness of different security policies which can be implemented for practical security solutions in the Internet.

This research is being supported by grant of Russian Foundation of Basic Research (№ 04-01-00167), grant of the Department for Informational Technologies and Computation Systems of the Russian Academy of Sciences (contract №3.2/03) and partly funded by the EC as part of the POSITIF project (contract IST-2002-002314).

References

- [1] T.Back, D.B.Fogel, Z.Michalewicz, *Evolutionary computation. Vol. 1. Basic algorithms and operators*, Institute of Physics Publishing. (2000).
- [2] R.Canonica, D.Cotroneo, L.Peluso, S.P.Romano, G.Ventre, Programming routers to improve network security. *Proceedings of the OPENSIG Workshop*. (2001).
- [3] E.Charniak, R.P.Goldman, A Bayesian Model of Plan recognition. *Artificial Intelligence*, V.64, N 1. (1993).
- [4] S.Chen, Q.Song, Perimeter-Based Defense against High Bandwidth DDoS Attacks. *IEEE Transactions on Parallel and Distributed Systems*, Vol.16, No.7. (2005).
- [5] P.R.Cohen, H.J.Levesque, Teamwork. *Nous*, Vol.25, No.4. (1991).
- [6] V.V.Druzhinin, D.S.Kontorov, M.D.Kontorov, *Introduction into conflict theory*. Moscow, Radio i svyaz' (1989) (in Russian).
- [7] FIPA. <http://www.fipa.org>
- [8] C.W.Geib, R.P.Goldman, Plan recognition in intrusion detection systems. *DARPA Information Survivability Conference and Exposition*, DARPA and the IEEE Computer Society. (2001).
- [9] T.M.Gil, M.Poletto, MULTOPS: a data-structure for bandwidth attack detection. *Proceedings of 10th Usenix Security Symposium*. (2001).
- [10] R.P.Goldman, C.W.Geib, C.A.Miller, A New Model of Plan Recognition. *Proceedings of the 1999 Conference on Uncertainty in Artificial Intelligence*. (1999).
- [11] V.Gorodetski, I. Kotenko, Attacks against Computer Network: Formal Grammar-based Framework and Simulation Tool. *Lecture Notes in Computer Science*, V.2516. (2002).
- [12] B.Grosz, S.Kraus, Collaborative plans for complex group actions. *Artificial Intelligence*, Vol.86. (1996).
- [13] D.Gu, E.Yang, Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey, *Technical Report of the Department of Computer Science, University of Essex*, CSM-404. (2004).
- [14] N.R.Jennings, Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, Vol.75, No.2. (1995).
- [15] C.Jin, H.Wang, K.G.Shin, Hop-count filtering: An effective defense against spoofed DDoS traffic. *Proceedings of the 10th ACM Conference on Computer and Communications Security*. (2003).
- [16] H. Kautz, J.F.Allen, Generalized plan recognition. *Proceedings of the Fifth National Conference on Artificial Intelligence*. (1986).
- [17] J.Kelemen, Colonies: grammars of reactive systems. *Proceedings of AICRS'97*. (1997).
- [18] A.D.Keromytis, V.Misra, D.Rubenstein, SOS: An architecture for mitigating DDoS attacks. *Journal on Selected Areas in Communications*, Vol. 21. (2003).
- [19] I.Kotenko, A.Ulanov, Multiagent modeling and simulation of agents' competition for network resources availability. *Second International Workshop on Safety and Security in Multiagent Systems, Utrecht, The Netherlands*. (2005).
- [20] V.A.Lefevre, *Reflexion*. Moscow, "Kognito-Center" (2003) (in Russian).
- [21] J.Mirkovic, M.Robinson, P.Reiher, G.Oikonomou, Distributed Defense Against DDOS Attacks. *University of Delaware. CIS Department. Technical Report CIS-TR-2005-02*. (2005).
- [22] J.Mirkovic, S.Dietrich, D.Dittrich, P.Reiher, Internet Denial of Service: Attack and Defense Mechanisms. Prentice Hall PTR. (2004).
- [23] C.Papadopoulos, R.Lindell, I.Mehring, A.Hussain, R.Govindan, Cossack: Coordinated suppression of simultaneous attacks. *Proceedings of DISCEX III*. (2003).
- [24] T.Peng, L.Christopher, R.Kotagiri, Protection from Distributed Denial of Service Attack Using History-based IP Filtering. *IEEE International Conference on Communications*. (2003).
- [25] A.A. Stognii, A.I. Kondrat'ev *Game theory information modeling in decision making systems*. Kiev: Naukova dumka. (1986) (in Russian).
- [26] M.Tambe, Towards flexible teamwork. *Journal of AI Research*, Vol. 7. (1997).
- [27] OMNeT++ homepage. <http://www.omnetpp.org/>
- [28] T. Peng, C. Leckie, and R. Kotagiri, Proactively Detecting DDoS Attack Using Source IP Address Monitoring, *Networking 2004*, Athens, Greece. (2004).
- [29] M.Vilain, Getting Serious about Parsing Plans: A Grammatical Analysis of Plan Recognition. *Proceedings of the Eighth National Conference on Artificial Intelligence*, Cambridge, MA. (1990).
- [30] M.P.Wellman, D.V.Pynadath, *Plan Recognition under Uncertainty*, Unpublished web page. (1997).
- [31] Y.Xiang, W.Zhou, An Active Distributed Defense System to Protect Web Applications from DDoS Attacks. *The Sixth International Conference on Information Integration and Web Based Application & Services*. (2004).
- [32] D.Xuan, R.Bettati, W.Zhao, A gateway-based defense system for distributed dos attacks in high-speed networks. *IEEE Transactions on Systems, Man, and Cybernetics*. (2002).

Flexible and Efficient Use of Robot Resources Using Higher-Order Mobile Agents

Mayu MIZUNO^a, Masato KURIO^b, Munehiro TAKIMOTO^a,
and Yasushi KAMBAYASHI^c

^a*Department of Information Sciences, Tokyo University of Science, Japan*

^b*NEC Software Ltd., Japan*

^c*Department of Computer and Information Engineering, Nippon Institute of Technology, Japan*

Abstract. This paper presents a framework for controlling intelligent robots connected by communication networks. This framework provides novel methods to control coordinated systems using higher-order mobile agents. Higher-order mobile agents are hierarchically structured agents that can contain other mobile agents. The combination of the higher-order mobile agent and mobile multi-robots open a new horizon of efficient use of mobile robot resources. Instead of physical movement of multi-robot, mobile software agents can migrate from one robot to another so that they can find the most suitably equipped and/or the most suitably located robots to perform their task. Thus the framework presented here provides efficient use of robot resources as well as scalability and flexibility.

Keywords. Mobile agent, Dynamic software composition, Intelligent robot control

Introduction

In the last decade, robot systems have made rapid progress in not only their behaviors but also in the way they are controlled. In particular, control systems based on multi-agent methodologies, enable a controlled robot to learn to adapt to the circumstances around it through its own interactions. Furthermore, multi-agent systems introduced modularity, reconfigurability and extensibility to control systems, which had been traditionally monolithic. It has made easier the development of control systems on distributed environments such as multi-robot systems.

On the other hand, excessive interactions among agents in the multi-agent system may cause problems in the multi-robot environment. Consider a multi-robot system where each robot is controlled by an agent, and interactions among robots are achieved through a communication network such as a wireless LAN. Since the circumstance around the robot changes as the robots move, the condition of each connection among the various robots also changes. In this environment, once some of the connections in the network are disabled, the system may not be able to maintain consistency among the states in the robots. Additionally, such a problem has a tendency to increase, as the number of interactions increases.

In order to lessen the problems of excessive communication, mobile agent methodologies have been developed for distributed environments. In the mobile agent system, each agent can actively migrate from one site to another site. Since a mobile agent can bring the necessary functionalities with it and perform its tasks autonomously, it can reduce the necessity for interaction with other sites. In the minimal case, a mobile agent requires that the connection is established only when it performs migration [1]. This property is useful for controlling robots that have to work in a remote site with unreliable communication or intermittent communication. The concept of a mobile agent also creates the possibility that new functions and knowledge can be introduced to the entire multi-agent system from a host or controller outside the system via a single accessible member of the multi-robot system.

The model of our system is cooperative work by a pool of heterogeneous multi-robots [2]. The property of inter-robot movement of the mobile agent contributes to the flexible and efficient use of the robot resources. Suppose a multi-robot system that consists of small heterogeneous mobile robots that are scattered in a field. Each robot has some general capability such as mobility, and some specific capability such as an infrared cameras, ultrasound system or long arms. The mobile agents can integrate such heterogeneous mobile robots to achieve a specific goal. For example, a mobile agent with a specific purpose can migrate to each robot one by one to find the most physically suitable one to perform the purpose. Alternatively, if any robot cannot achieve the purpose alone, the agent can migrate to several robots to make them cooperate to perform the goal. This means that if the current robot system lacks some capabilities, we can complement it by adding some robots with new capabilities and coordinating mobile agents. Thus the mobile agents enable robot systems to be more scalable. An additional aspect of mobile agents is that a mobile agent can migrate to the robot that is most conveniently located to a given task, e.g. closest robot to a physical object such as soccer ball.

Since agent migration is much easier than robot motion, it contributes to save power-consumption. Here, notice that any agents on a robot can be killed as soon as they finish their tasks. If each agent has a policy choosing idle robots rather than busy ones in addition to these migration strategies, it would result in more efficient use of robot resources.

Based on the observations mentioned above, we have developed a framework for constructing intelligent multi-robots controlled by higher-order mobile agents where the higher-order property of the mobile agents means that they can be organized hierarchically and dynamically. Each mobile agent can be a container of other mobile agents and can migrate to other agents. This migration to an agent enables the migrated agent to acquire new functions as well as to permit the robot that receives the migrated agent to acquire new functions.

This higher-order property gives significant advantages to control systems based on mobile agents. Consider robots moving around in a situation where there are obstacles. In this case, avoiding collision with obstacles or other robots can be regarded as common behavior for every robot. Such common behavior can be achieved by migration of the agent that rotates a robot in response to signals from collision-sensors. After that, in order to realize complete behavior, an agent for the main behavior has only to migrate to the agent with the common behavior. This means that the main behavior can be described without considering obstacles, and therefore, the higher-order property contributes to not only increasing scalability due to dynamic extension but also decreasing development-cost due to separation of functions.

Moreover, since the extended agent behaves as a single agent, and it can migrate to another agent with the containing agents, communications through remote connections can also be reduced further. We implemented an example to prove our framework's effectiveness. Although it is a toy program, it provides intuitive understanding leading to practical use.

The structure of the balance of this chapter is as follows. In the first section we describe the background. The second section describes the higher-order mobile agents with dynamic extension. Dynamic extension is the key feature that supports the ability to add new functionalities to intelligent robots in action. The third section shows an example of intelligent robot systems in which robots search for an object cooperatively. Through this example, we demonstrate how the higher-order mobile agents with dynamic extension can contribute efficient use of robot resources. Finally, the fourth section discusses future works and conclusions.

1. Background

The traditional structure for the construction of intelligent robots is to make large, often monolithic, artificial intelligence software systems. The ALVINN autonomous driving system is one of the most successful such developments [3]. Putting intelligence into robots is, however, not an easy task. An intelligent robot that is able to work in the real world needs a large-scale knowledge base. The ALVINN system employs neural networks to acquire the knowledge semi-automatically [4]. One of the limitations of neural networks is that it is assumed that the system is used in the same environment as that in which it was trained. When the intelligent robot is expected to work in an unknown space or an extremely dynamic environment, it is not realistic to assume that the neural network is appropriately trained or can acquire additional knowledge with sufficient rapidity. Indeed, many intelligent robots lack a mechanism to adapt to a previously unknown environment.

On the other hand, multi-agent robotic systems are recently becoming popular in RoboCup or MIROSOT [5]. In traditional multi-agent systems, robots communicate with each other to achieve cooperative behaviors. The ALLIANCE architecture, developed in Oak Ridge National Laboratory, showed that cooperative intelligent systems could be achieved [6]. The architecture is, however, mainly designed to support self-adaptability. The robots in the system are expected to behave without external interference, and they show some intelligent behaviors. The observed intelligence, however, is limited due to the simple mechanism called *motivation*. Robots' behaviors are regulated by only two rules *robot impatience* and *robot acquiescence*. These rules are initially defined and do not evolve. In contrast, the goal of our system is to introduce intelligence and knowledge into the robots after they start to work [2]. Therefore, our system does not have any learning mechanism or knowledge acquiring mechanism. All the necessary knowledge is sent as mobile agents from other robots or the host computer.

An interesting research work of multi-agent robot control system was conducted at Tokyo University of Science [7]. Their work focused on the language aspect of robot control systems using multi-agents. They employed a hierarchical model of robot agents where the root agent indirectly manages all the agents. The lowest agents are physical devices and each has only one supervisor agent. Communications are

performed through super-agent channels and sub-agent channels. Each robot has a hierarchically structured set of agents and the structure is rigidly constructed at the initial time. Therefore the structure of the control software is predetermined, and there is no concept of dynamic configuration of the structure of agents. The framework we present in this chapter provides dynamic re-structuring of the set of agents and provides more flexibility in the hierarchically structured control software.

For the communication aspect, they employ agent negotiation. In contrast, we employ agent migration so that our model can more suitably fit in a realistic multi-robot environment where the communication should be expected to be intermittent.

One notable feature of their system is the description language, called Multiagent Robot Language (MRL). This language is based on the committed-choice concurrent logic programming language and compiled into the guarded Horn clauses [8][9]. This feature has advantages of transparency of the agent descriptions over our framework that is based on Java. The efficiency problem of logic programming is overcome by re-compiling into C language. We also implement a descriptive language based on functional languages, Objective Caml and Scheme, in order to achieve the transparency of the agent descriptions [10][11].

The work most closely related to ours is the distributed Port-Based Adaptable Agent Architecture developed at Carnegie Mellon University [12]. The Port-Based Agents (PBAs) are mobile software modules that have input ports and output ports. All PBAs have the map of the port addresses so that they can move other robots and combine themselves with other PBAs to compose larger modules. The usefulness of PBA architecture is demonstrated by the Millibot project also at Carnegie Mellon University [13]. In a robot mapping application, PBA is used to control the mapping robots, and when the working robot has a hardware failure, the PBA on the robot detects it and moves to an idle robot.

Software composition is clearly possible using port-based modules. The dynamic extension capability of our mobile agent control system, however, is another strategy for the composition of larger software.

The PBA is derived from the concept of port-based objects, designed for real-time control applications [14]. Therefore it may have advantages as a robot control mechanism. The framework we present in this chapter is an exploration of the applications of mobile agents and software compositions through mobility and extensibility. Constructing robot control software by mobile agents and its dynamic extension is not only novel but also flexible due to dynamic inheritance. It may be superior for extensibility of working software.

2. Higher-Order Mobile Agent with Dynamic Extension

The mobile agent system we have used to control robots is based on a mobile agent system, called MobileSpaces, developed by I. Satoh [15]. MobileSpaces is also based on the mobile ambients computational model proposed by L. Cardelli and A. D. Gordon [16]. MobileSpaces provide the basic framework for mobile agents. It is built on the Java virtual machine, and agents are supposed to be programmed in Java language.

Mobile agents can migrate from place to place. When they migrate, not only the program code of the agent but also the state of the agent can be transferred to the

destination. The higher-order mobile agents are mobile agents whose destination can be other mobile agents as well as places in traditional agent systems.

Two unique features are worth mentioning for our robot control system. 1) Each mobile agent can contain one or more mobile agents (hierarchical construction), and 2) Each mobile agent can migrate to any other mobile agent (inter-agent migration). Thus migration to another agent results in a nesting structure of agents. Agents in the other agent are still autonomous agents that can behave independently. Figure 1 illustrates the situation that agent C migrates from agent A to agent B, and agent D that is contained in agent C also migrate from agent A to agent B.

In order to enhance the intelligent robot control system in action, we have added the dynamic extension feature to customize functions of robots while they are running [2].

Suppose an agent A is working somewhere and we want to extend its capability. One way is to replace that agent with a new agent B. On the other hand in our system, we only need to send an agent A' with the new feature to the agent A. While the agent A' is included in A, the agent A behaves with the extended feature. If the agent A' leaves the agent A, the agent A behaves with the original feature. All the other agents do not have to be aware of the change of the agent A. In Figure 2, after an agent A' migrates to an agent A, the other agent B still communicates to the agent A without knowing the migration of A'. The agents A and A' behave just as a single agent for the agent B.

In order to extend the agent A, the agent A' only needs to have the new feature to be added. If the agents A and A' have methods with the same signature, the method in agent A' overrides the method with the same signature in the agent A. The signature is an extended type of a function. When it is said that two functions have the same signature, it means that they have the same name, the same number of the same type of formal parameters, and the same return type. The agent migration achieves the same semantics as dynamic inheritance [17].

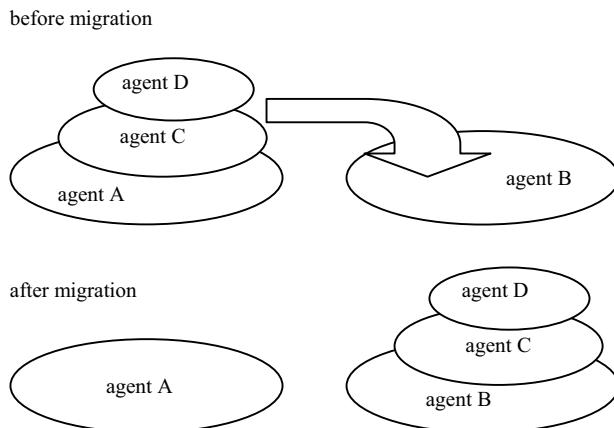


Figure 1. When agent C migrates from agent A to agent B, the contained agent D also migrates from A to B

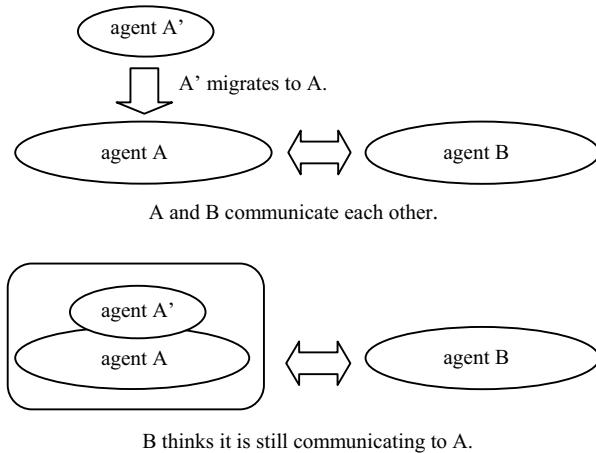


Figure 2. Dynamic extension by migration of agent with new features

3. Robot Controller Agents for Target Searcher

In this section, we demonstrate that the model of higher-order mobile agents with dynamic extension is suitable for the composition of software to control an intelligent robot. For this purpose, we show an example of cooperative target search that can be employed for practical applications.

Intelligent multi-robots are expected to work in distributed environments where communication is relatively unstable and therefore where fully remote control is hard to achieve. Also we cannot expect that we know everything in the environment beforehand. Therefore intelligent robot control software needs to have the following features: 1) It should be autonomous to some extent. 2) It should be extensible to accommodate the working environment. 3) It should be replaceable while in action. Our higher-order mobile agent with dynamic extension satisfies all these functional requirements.

Our control software consists of autonomous mobile agents. Once an agent migrates to a remote site, it requires minimal communication to the original site. Mobile agents are higher-order so that the user can construct a larger agent by hierarchical composition of smaller agents. Finally, if we find that the constructed software is not satisfactory, we can replace the unsatisfactory component (an agent) with new component (another agent) by using agent migrations.

We employed the ER1 Personal Robot Platform Kit by Evolution Robotics Inc. (information available at <http://www.evolution.com/>) as the platform for our prototype system. Each robot has two servomotors with tires. The power is supplied by rechargeable battery. It has a servo motor controller board that accepts RS-232 serial data from a host computer. Each robot holds one notebook computer as its host computer. Our control agents migrate to these host computers by wireless LAN.

Let us consider how to program robots to find a target. For such a task, the most efficient solution would be to make all robots search for the target simultaneously. If

the targets were comparatively fewer than the robots, however, most robots would move around in vain, consuming power without finding anything. This problem can be more serious in our model where any robots can be shared by any agents, because the robots to which an agent is going to migrate may be already occupied with another task. Thus, this searching strategy could result in increasing the total costs for the aggregate of the multi-robots.

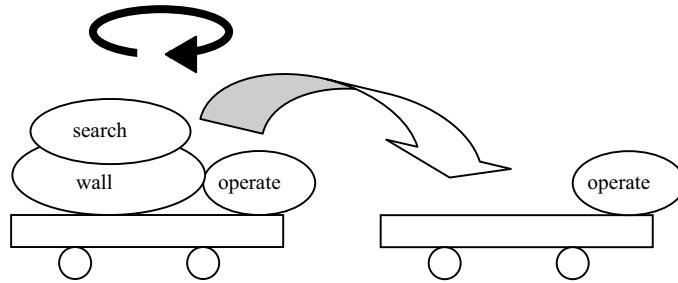
On the other hand, our strategy of using higher-order mobile agents achieves power-preserving search without sacrificing efficiency, i.e. only one robot is in action. The core of our idea is finding the nearest robot to the target by using agent migration. Initially, an agent is dispatched from the host machine to a nearby robot in the multi-robots system. Then, the agent hops around the robots one by one and checks the robot's vision in order to locate the target until it reaches the robot that is closest to the target. Until this stage, robots in the multi-robot system do not move; only the mobile agent migrates around so that robots can save power.

Once the agent finds the target, it migrates to the closest robot and makes the robot move toward the target. This strategy is obviously not as efficient as that of having all robots search for a target simultaneously. But the migration time is negligible comparing to robot motion and the efficiency of power-preservation offsets the slight search inefficiency. In our multi-robot case, the robots' vision does not cover 360 degrees; so that a robot has to rotate to check its circumstance. Rotating a robot at its current position may capture the target and another robot closer to the target. Then the agent migrates to the more conveniently located robot. Take note that the rotation requires much less movement of the wheels than exploration, and it contributes the power saving.

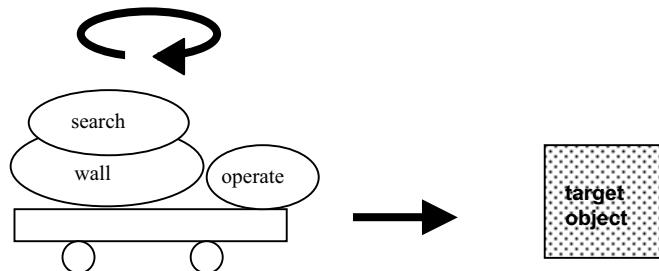
Details of our searching algorithm are the followings: 1) an agent chooses an arbitrary robot to which it migrates, and performs the migration, 2) as the agent arrives on the robot, it makes that robot rotate as shown by Figure 3 (a), 3) if the target is found, the agent make the robot move to that direction as shown in Figure 3 (b); otherwise, go back to the step 1, and 4) at this stage, if all robots have been tried without finding the target, the agent makes the last robot do random-walk until it can find a target as shown by Figure 3 (c).

This algorithm can be implemented as a migrating *search* agent. The *search* agent migrates on the other mobile agent, *wall*. The sole task of *wall* agent is to avoid collisions. If the *search* agent can control the *wall* agent, all that the *search* agent has to do against the *wall* agent is to make it hibernate until reaching step 4, and then it wakes the *wall* agent. We achieve this implementation by designing the *search* agent as an included agent of *wall* agent. The *search* agent migrates to the *wall* agent, so that these two agents are composed into one agent. After that, the combined agent traverses each robot. The core part of the *search* agent is the method *behavior()*. It implements the searching algorithm. Appendix shows the pseudo-code of the method *behavior()*.

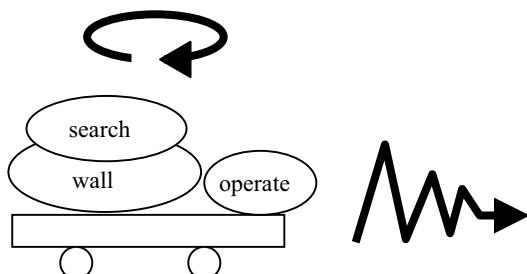
In principle, we can make the *wall* agents stationary; the *search* agent does not have to take the *wall* agent to other robots, since all the robots have the *wall* agents. But the current system is a prototype and the cooperation with other tasks is yet to be perfected; thus the *search* agents migrate with the *wall* agent.



(a) The search agent rotates a robot. At this time, if it cannot find the target, it migrates to another robot.



(b) If a target is found as a result of rotation, the search agent makes the robot move toward it.



(c) If the last robot cannot find the target after the rotation, the search agent makes the wall agent active to make the robot perform random-move.

Figure 3. The *search* agent's behaviors

4. Conclusions and Future Directions

We have presented a novel framework for controlling intelligent multi-robots. The framework helps users to construct intelligent robot control software by migration of mobile agents. Since the migrating agents are higher-order, the control software can be hierarchically assembled while they are running. Dynamically extending control

software by the migration of mobile agents enables us to make base control software relatively simple, and to add functionalities one by one as we know the working environment. Thus we do not have to make the intelligent robot smart from the beginning or make the robot learn by itself. We can send intelligence later as new agents.

We have implemented a team of cooperative search robots to show the effectiveness of our framework. Even though our example is a toy program, our framework is scalable. The creation of a practical system is accomplished by adding more mobile agents to the base system. We are conducting numerical experiments to measure how much power preserving we can accomplish.

Our future directions for research will include the addition of security features, refinement of the implementation of dynamic extension, additional proof of concept for dynamic addition of new functionality, and additional work on scheduling of conflicting tasks, such as introducing self-adaptability via PBA.

The current intelligent robot control system is entirely implemented in Java language. Even though Java is an excellent platform, it lacks of transparency as a description language that MRL provides. It is also hard to interact with existing knowledge databases implemented in Lisp languages. Therefore, we are reimplementing the entire system in functional settings so that the control software obtains transparency in description as well as interfaces with existing knowledge databases.

References

- [1] W.J. Binder, G. Hulaas, and A. Villazon, Portable Resource Control in the J-SEAL2 Mobile Agent System, *Proceedings of International Conference on Autonomous Agents*, (2001), 222-223.
- [2] Y. Kambayashi and M. Takimoto, Higher-Order Mobile Agents for Controlling Intelligent Robots, *International Journal of Intelligent Information Technologies*, 1(2), (2005), 28-42.
- [3] D.A. Pomerleau, Defense and Civilian Applications of the ALVINN Robot Driving System, *Proceedings of 1994 Government Microcircuit Applications Conference*, (1994), 358-362.
- [4] D.A. Pomerleau, ALVINN: An Autonomous Land Vehicle in a Neural Network, *Advances in Neural Information Processing System 1*. Morgan Kaufmann, San Fransisco, (1989), 305-313.
- [5] R.R. Murphy, *Introduction to AI robotics*, MIT Press, Cambridge, 2000.
- [6] L.E. Parker, ALIANCE: An Architecture for Fault Tolerant Multirobot Cooperation, *IEEE Transaction on Robotics and Automation*, 14(2), (1998), 220-240.
- [7] H. Nishiyama, H. Ohwada, and F. Mizoguchi, A Multiagent Robot Language for Communication and Concurrency Control, *Proceedings of International Conference on Multi-Agent Systems*, (1998), 206-213.
- [8] E. Shapiro, *Concurrent Prolog: Collected Papers*, MIT Press, Cambridge 1987.
- [9] K. Ueda, *Guarded Horn Clauses*, PhD Thesis, University of Tokyo, 1986.
- [10] Y. Kambayashi and M. Takimoto, Functional Language for Mobile Agents with Dynamic Extension. *Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Artificial Intelligence 3214*, Springer-Verlag, Berlin Heidelberg New York, (2004), 1010-1017.
- [11] Y. Kambayashi and M. Takimoto, Scheme Implementation of the Functional Language for Mobile Agents with Dynamic Extension, *Proceedings of IEEE International Conference on Intelligent Engineering Systems*, (2005), 151-156.
- [12] T.Q. Pham, K. R. Dixon, J.R. Jackson and P.K. Khosla, Software Systems Facilitating Self-Adaptive Control Software, *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, (2000), 1094-1100.
- [13] R. Grabowski, L.E. Navarro-Serment, C.J.J. Paredis and P.K. Khosla, Heterogeneous Teams of Modular Robots for Mapping and Exploration, *Autonomous Robots*, 8(3), (2000), 293-308.
- [14] D.B. Stewart and P.K. Khosla, The Chimera Methodology: Designing Dynamically Reconfigurable and Reusable Real-Time Software Using Port-Based Objects, *International Journal of Software Engineering and Knowledge Engineering*, 6(2), (1996), 249-277.

- [15] I. Satoh, MobileSpaces: A Framework for Building Adaptive Distributed Applications using a Hierarchical Mobile Agent System, *Proceedings of IEEE International Conference on Distributed Computing Systems*, (2000), 161-168.
- [16] L. Cardelli and A.D. Gordon, Mobile Ambients, *Foundations of Software Science and Computational Structures, Lecture Notes in Computer Science 1378*, Springer-Verlag, Berlin Heidelberg New York, (1998), 140-155.
- [17] M. Abadi and L. Cardelli, *A Theory of Objects*, Springer-Verlag, Berlin Heidelberg New York, 1996.

Appendix

```

public void behavior() {
    synchronized (t) {
        eventHandler.behavior(t);
    }
    // Rotate the residing robot in order to find the target labeled "ar".
    String resMessage =
    sendCommandAndwaitFor("move rotate toward object ar'\n", "move");
    String objectRlt = waitForObject("ar");
    // Check whether it finds the target.
    if (objectRlt == null) {      // the case in which the target is not found.
        // Check the IP address, and if it is the last one, perform the random work.
        if (myIP.equals(the last IP)) {
            randomWalk();
        }
        else{      // <<jump to next Robot>>
            // An array has the sequence of robot addresses.
            // Currently, the sequence is fixed.
            // Retrieve the next IP address from the array and set it to machineIP.
            ctxt.move(new AgentURL(":///Controller/Range/Search/"),
                      new AgentURL(the next IP));
        }
    }
    else{      // The case in which the target is found.
        // By using the tokenizer, it gets the distance to the target. It is in the fifth token.
        // And then, it moves forward that distance.
        for (int i = 0 ; i < 5 && objectSt.hasMoreTokens());
        objectStr = objectSt.nextToken(), i++) {
        objectEvent[i] = (new Double(objectStr)).doubleValue();
        objectDist = objectEvent[4];
        sendCommandAndwaitFor("move "+objectDist+" c\n", "move");
    }
}
}

```

System Support of Diabetic Self-treatments Using Mobile Phones

Takuya YOSHIHIRO[†], Kazuhiro IKEMOTO[†], Kumiko MORI[‡], Sachiko KAGAWA[‡], Yasuhisa YAMAMOTO[‡] and Masaru NAKAGAWA[†]

[†] Faculty of Systems Engineering, Wakayama University, Japan

[‡] Wakayama Rosai Hospital, Japan

Abstract. We developed a system which helps diabetic people making self-treatments in their home, by exchanging treatment information and advices with their doctors and nurses in the hospital. To cure diabetes, it is essential to make daily treatment such as improving meals, doing exercises, taking insulin, or taking medicines to keep blood glucose level in a proper range. However, since many of diabetic people do not go to see their doctor frequently, say, about once a month, they tend to be lazy about self-treatments during that long period of time. As a result, many people are suffering from diabetes long time. To improve this situation, we developed the system to help self-treatments. In our system, patients report their concrete treatment activities to their doctors using mobile phones. The doctors watch the reports and make proper encouraging advices to the patients. We expect that such kind of advices can raise patients' motivation for their self-treatments. In designing the system, we tried to have doctors understand the concrete activities of patients' self-treatments through patients' daily reports, so that doctors can make proper and effective advices and encouragements. In this paper, we introduce the new diabetic self-treatment supporting system and report the evaluation results through a short-term test operation with several patients.

Keywords. Diabetes, Mobile Phones, Databases, System Support, Medical Care

1. Introduction

Diabetes is one of the serious diseases in the world. The symptom of diabetes is that blood glucose levels are kept high and if it becomes severe, several complications occur at heart, kidneys, eyes, feet, and so on. The big problem of this disease is that the population of diabetes is continuously increasing, and now, lots of people are suffering from this disease. In United States, according to the National Diabetes Fact Sheet [1], the total number of diabetes in U.S. is estimated as 20.8 million, which is about 7.0% of the whole population. To say more, the number of pre-diabetes, whose blood glucose levels are higher than normal but not yet high enough to be diagnosed as diabetes, is estimated as much as 41 million. In Japan, according to the report of Japanese government [2], the total number of diabetes is estimated as 7.4 million, which is about 5.8% of the whole population. Also, the number of pre-diabetes is estimated as 16.2 million.

To cure diabetes, a typical patient comes to see a doctor periodically, say, about once a month, and in each time, the doctor instruct how to make self-treatment to cure effectively, then the patients perform self-treatments at their home by themselves. Self-treatments consist of three types of treatments: nutrition improvement, physical

exercise, and medicines. And they all require continuous effort. In fact, less people can perform those treatments completely, and thus many people, who are a little lazy about those daily treatments, take very long time to cure, or can not recover from diabetes. This might be one reason of recent increasing of diabetes.

To improve this situation, several computer-aided systems have been developed. One is web-based medical interviewing system "Wagamaman[3][4]," developed in Hiroshima City University and St. Marianna University. This system is aimed to raise patients' motivation for self-treatments by sending medical questions periodically, say, once a week. Patients answer the questions on the web page, and then receive advices which is automatically created by the knowledge-database server. The advice messages also include encouragement, and then patients' motivation is raised to make better self-treatments. At the same time, doctors can understand patients' self-treatment conditions so that better medical treatments can be expected when they come to see the doctor.

Another approach is performed by the easy-to-operate blood-glucose monitoring system e-SMGB [5]. In this system, patients use a mobile phone that can measure blood glucose levels. By using this, patients can easily send daily blood glucose levels to the server on the Internet. Then, both patients and doctors can watch their daily transition of the blood glucose levels on its web page, and understand whether patients' self-treatments are working effective or not. As additional functions, patients can report their self-treatment conditions by writing comments on the web page, and doctors can see it to make effective advices to the patients.

Both approach mentioned above seems to work effective and that will be true, but they both do not handle their detail activities of diabetic self-treatments, and this is one of the limitations of them. Concretely, by those systems, doctors are not able to know, for example, what kind of foods patients eat, or how insulin is taken and whether it works effectively or not, or other detail conditions. For diabetic treatments, to watch those individual activities is important because the pile of wrong or lazy treatments actually reduce the effects of their self-treatments.

Our approach is aimed to raise effect of medical treatments by reporting such detail information of self-treatments to doctors. For this purpose, we designed and implemented a system to send enough information from patients to doctors using mobile phones' web browsers. The main features of our system are as follows: First, the set of reporting items are determined to satisfy the condition that the number of reporting items is small enough and also by which doctors can make enough understanding about the patients' self-treatment conditions. Second, treatment plan of each patient is registered in the database and by comparing the plan with patients' reports, the system automatically displays the wrong treatments in the screen. Doctors can find patients' wrong treatments easily and do not overlook them. Further, to report menu of meals, we use photo images taken with mobile phones' attached cameras. Patients can send photo image of meals, and doctors can see them to grasp the concrete condition of meal treatments. Also, doctors can evaluate daily meals, and then patients can see the evaluation result on their mobile phones. All the above functions are implemented in easy-to-operate user interfaces designed carefully.

By using our system, patients can report detail activities of their self-treatments at any time and place, and doctors can make effective and encouraging advices under the enough understanding of patients' self-treatment activities. In this paper, we explain about our attempt to develop the system and show the evaluation result through a short-term test operation of the system.

This paper is organized as follows: In Section 2, we describe requirements for our system and direction of our system design. In Section 3, we explain about the whole functions and user interfaces of our system. In Section 4, we show our evaluation results through our test operation. Finally, we give the concluding remarks in Section 5.

2. Requirements and System Design

2.1. Our Trials in Designing Systems

Our trial with this system is summarized into following three points.

1. How doctors can understand the concrete conditions of patients' self-treatments using mobile phones.
2. How doctors can reduce their time to understand patients' conditions and make effective advices.
3. How patients can raise their motivations to continue their self-treatments

Those points are all very important for diabetic treatments. The main point is the first one. It is directly essential for doctors to make effective advices and to improve the effect of self-treatments. However, if we take too much care to report concretely, the labor of patients will be increased. It surely reduces the points 2 and 3. Therefore, to develop the really usable system, the design to keep the balance of them is important.

2.2. Determining the Set of Reporting Items

For well-balanced design, we first determined the set of reporting items to send from patients to doctors, where it is important to satisfy the condition that the number of items is not many while they are enough for doctors to understand the patients' condition. To determine the item set, we should understand typical diabetic treatments. Diabetic self-treatments are classified into three types, described as follows:

Improving meals: patients must improve their daily meals to save calories and take well-balanced nutrition.

Physical exercise: moderate exercise will effective to lose weight and improve metabolic rate.

Insulin and medicines: a doctor sometimes prescribes insulin or medicines to lower their blood glucose levels on some particular timing. (Those are used only when it is required; meals and exercises are considered as main treatments for diabetes.)

In the diabetic treatments, to continue those three daily treatments are essential. Since those treatments affect blood glucose levels, to do those treatments with monitoring them is effective. Therefore, reporting items are categorized into four: a) meals, b) exercises, c) insulin and medicines, and d) blood glucose levels. For each category, patients should report i) what kind of activities he/she did and ii) when he/she did it, as common items among every categories. In addition, we decided to report iii) the timing related with meals (e.g., before breakfast, after dinner, etc.) to understand the sequence of several treatments.

For each category, particular items exist. As for meals, to grasp concretely what the patient eat, patients take photographs with their mobile phones (which have cameras) and send them via e-mail. Photo images have far more information than text so that doctors can easily understand what the patients ate. Also, we add a feeling of fullness is helpful, which is useful to imagine the amount of meals. As for exercise, patients' subjective impression of hardness is also helpful to imagine the real amount of exercise. In summary, the set of patients' reporting items are as follows:

meals: time, amount(subjective statement), photo-images.

exercise: time, sort of exercise, amount(subjective statement), timing.

insulin and medicines: time, name of insulin or medicines, amount, timing.

blood glucose levels: time, measured blood glucose levels, timing.

2.3. Reducing Time to Understand Patients' Conditions

We designed the system to make doctors spend less time to use the system and to understand the patients' conditions concretely. For this purpose, we implement a function of indicating "alerts." The alert function is a function to notify doctors if patients perform wrong self-treatments. When a doctor is too busy to watch the patients' reports, the doctor only have to check the alert as a minimum work. This function also prevents doctors from overlooking wrong self-treatment activities of patients. To implement the alert function, our system memorizes the instructions of treatment as "treatment plan," then, compares the plan with his/her self-treatment reports. If some differences are found, alert messages are displayed to notify the doctors of it.

To reduce the time for doctors to understand patients' conditions, we also designed the user interface carefully. To say the major feature, we prepared a "home" page of each patient. From this page, we can access all the information about the patient within one hyperlink so that doctors can easily grasp the patients' information or update them. Also, important information are directly seen in this page, e.g., the calendar view is prepared to see which day the patient made reports, and the list of alerts are also seen. About those, see Section 3 for details.

2.4. Raising Motivation of Patients

To raise patients' motivation is very important for diabetic self-treatments using our system. In other words, some effective method is required to have patients continue to make reports of their self-treatments, because daily operation of mobile phone is so stressful. In this system, we implement two such methods.

First is evaluation of daily meals by dietitian. In our system, patients can send photo-images of their daily meals, and doctors can make advices via e-mails. We consider, however, the effect of text-format advices have a limitation, thus we determined the evaluation format and implemented to show the evaluation result visually on mobile phones. The evaluation is done with six classified nutrition scores and social advices and encouragements are given by dietitians.

Second, we prepare the transition graph of reported blood glucose levels, which can be displayed on mobile phones. Using this function, patients can check the transition by themselves. Since the effects of their daily self-treatments are reflected into this graph, patients will try to keep low levels with this graph.

3. A System to Support Self-Treatments of Patients

3.1. Overview

The framework of our system is illustrated in Figure 1. The system consists of mobile phones used by patients, personal computers in the hospital, and a database server at a data center, which are all connected to the Internet. Namely, HTML based web pages should be displayed and e-mail should be exchanged on both mobile phones and PCs. The database server at a data center is a server computer that manages our database and also plays a role of web server. In detail, we use PostgreSQL as its DBMS, Apache as its web server, and the system is implemented with PHP. To send photo images of meals, we use e-mail and thus Postfix is installed.

Our system supports several kinds of communications. First, doctors register the profiles of patients. Then, patients send the self-treatment reports, which are sent from patients with mobile phones to the server. Next, doctors check the reporting data on their PCs in the hospital. If doctors found something wrong with the reports, they can send advices to the patients via e-mails. Also, patients can reply to them. Doctors or dietitian in the hospital can make a evaluation of daily meals of patients. Patients can see the evaluation results and the transition graph of reported blood glucose levels, on their mobile phones.

In the following subsections, details of those communication functions are described.

3.2. Management of Patients

Figure 2 is the view on which all the information of one patient is managed. In the upper-side of the view, we found the operation menu; from those hyperlinks we can

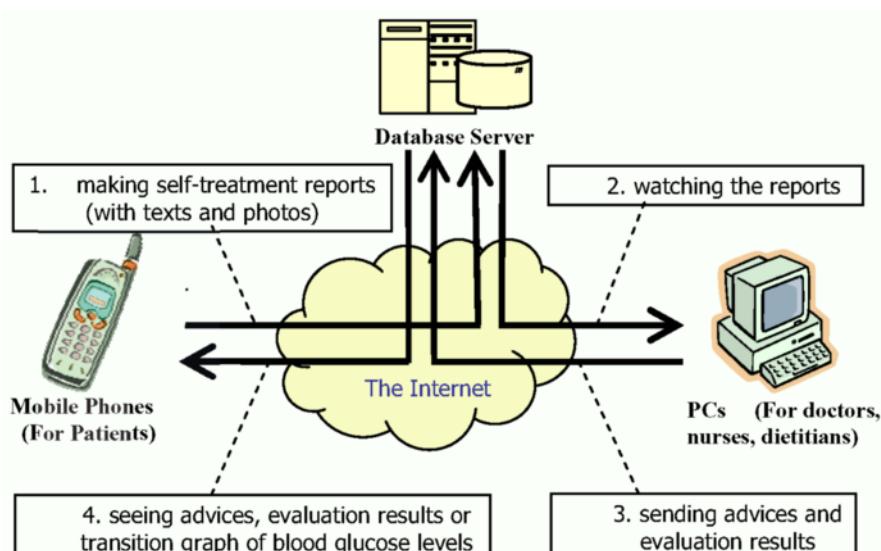


Figure 1. System Framework

configure patients' basic information (e.g., id number, age, sex, e-mail address, etc.), configure treatment plans (exercises to perform, medicines to take, etc.), and send e-mails to patients. On the down-side of the menu, we found the patient's basic information. And its right-side, the history of alerts is displayed. The treatment plan is compared with the patient's self-treatment reports and if they are different from the plan, "alert" arises here. At the bottom-side, we can see the calendar view where we can see the days on which some patient's reports are sent. If some reports exist, hyperlink is set on that day by which we can jump to the report view (described in Section 3.4). The patient's home view is used as a patient's "home page," where doctors can access all the information of him/her, and can perform every action to him/her, within one hyperlink from this page.

3.3. Making Self-treatment Reports

Patients make daily self-treatment reports via mobile phones. Figure 3 shows the examples of reporting interfaces on mobile phones to report about exercises, medicines, and blood glucose levels. Figure 3(a) is the view for reporting exercises. Here, in addition to date and time, sort of exercises and hardness should be reported. Here, choices of the exercise sort are previously registered by doctors for each patient, and

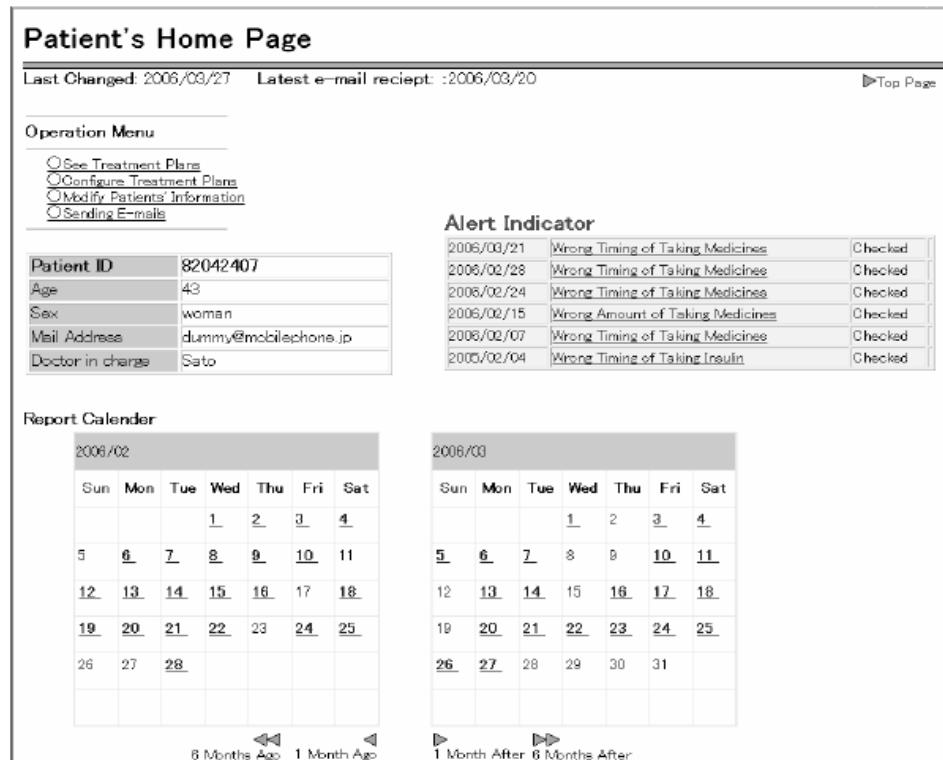


Figure 2. The Patient Home View

(a) Reporting Excises

Report of exercises

- Time of starting exercises

Date: 2006 / 3 / 25

Time: 0 : 35

Duration: 30 min.

- Sort of exercises:
 - Taking a walk
 - Walking
 - Taking a Bath
 - Running
 - Swimming
- Hardness:
 - very easy
 - a little easy
 - normal
 - a little hard
 - very hard
- comments:

→ Send the report

(b) Reporting Medicines

Report of insulin and medicines

Date: 2006 / 3 / 25

Time: 0 : 37

- Informations of medicines
 - Merbir(250mg)
 - Amount: 1 tablets
 - Timings: before breakfast
- glufast(10mg)
- Amount: 1 tablets
- Timings: after breakfast
- Comments:

→ Send the report

(c) Reporting Blood Glucose Levels

Report of blood glucose levels

Date: 2006 / 3 / 25

- Input measured time and values

[measuring 1]

Time: 12 : 30

Blood glucose levels: 120

Timings: after lunch

[measuring 2]

Time: 0 : 00

Blood glucose levels:

Timings: -----

- Comments:

→ Send the report

Figure 3. User Interface of Reporting Self-treatments on Mobile Phones

then the patient can select one of them. Figure 3(b) is the interface for reporting measured blood glucose levels. Since a patient measures the value several times in a day, a patient can report several values of the same day at the same time. Figure 3(c) is the interface for reporting taking medicines. Here, patients check the name of medicines, and input amount and timing taken. Particularly, timing is important because it affects blood glucose levels. Two selecting boxes are prepared for timings; one is to select meals, e.g., break fast, lunch, dinner, before sleep, etc., and another is to select timings, e.g., just before meal, just after meal, 2 hours before meal, etc. Reporting meals are also performed in the similar interfaces. The only difference is the method of sending photo images of meals; it is sent via e-mails.

3.4. Viewing Patients' Reports

Doctors view patients' reports to find something wrong with their reports. The system automatically compares the treatment plan with the patient's reports, and then indicates the difference as alerts. Nevertheless, view of detail reports is required to grasp patients' self-treatment activities. Our report view is shown in Figure 4. Here, in one screen, doctors can see every detail of a patient's reports in a day; reports are shown in a tabulated style. Reports are classified by three rows (e.g., morning, after noon, and night) and four columns (e.g., exercises, meals, insulin and medicines, and blood glucose levels). In each cell, corresponding report items are shown, where doctors can

Patient's Reports

[Patient's top page](#) [Top page](#)

2006/02/23 Patient ID: 82042308; women

	exercises(7.3 units)	meals→evaluation	insulin and medicines	blood glucose levels
Morning	•08:00~09:00 3 units(60 min) Types: Walking (level: easy) hardness: very easy	•07:23~07:37 (14 min.) amount: moderate  07:23	•07:23(medicines) type: melein(250mg) amount: 1 tablet timing: just before breakfast comments: (none)	•06:24 value: 80 timings: just before dinner •07:30 value: 123 timing: after dinner
	•09:15~10:15 3 units(60 min) Types: Swimming (level: hard) hardness: a little hard		•07:23(medicines) type: glufast(10mg) amount: 1 tablets timing: just before breakfast comments: (none)	
afternoon	•15:00~15:40 1.3 units(40 min.) types: Taking a walk hardness: a little hard	•12:18~12:38 (20 min.) amount: full  12:18	•12:24(medicine) type: glufast(10mg) amount: 1 tablets timing: just before lunch comments: (none)	•12:24 value: 112 timings: just before dinner •12:24 value: 130 timing: after dinner
night		•19:00~19:15 (15 min.) amount: full comments: I have little appetite  18:58	•18:59(medicine) type: melein(250mg) amount: 1 tablets timing: just before dinner comments: (none)	•18:50 value: 115 timings: just before dinner •12:24(medicine) value: 135 timing: after dinner

[Top page](#) ▲

Figure 4. Viewing Patients' Reports

see their time, timing compared with meals, and so on. Photo images of meals are also shown with its sending time.

3.5. Self-check of Treatment Conditions

To raise patients' motivation, our system supports two types of self-check function of the patients' treatment conditions. One is evaluation of meals of every day, namely, doctors or dietitians evaluate the reported meal information, and give some advices to the patient. Figure 5(a) is an evaluated result shown on mobile phones. Here, several evaluation parameters are shown in radar-chart style, and an advice is also shown. Such advices are very important to raise motivation of patients since it can be precious communication to encourage them to continue their self-treatments.

Another one is the transition graph of blood glucose levels shown in Figure 5(b). This view is also shown on mobile phones of patients and they can check the transition at any time and place by themselves. From our experience, when a patient is lazy in treatments, then it immediately reflects on the blood glucose levels. Therefore, it is expected that patients check the graph to make a self-examination, which will result in better treatments.

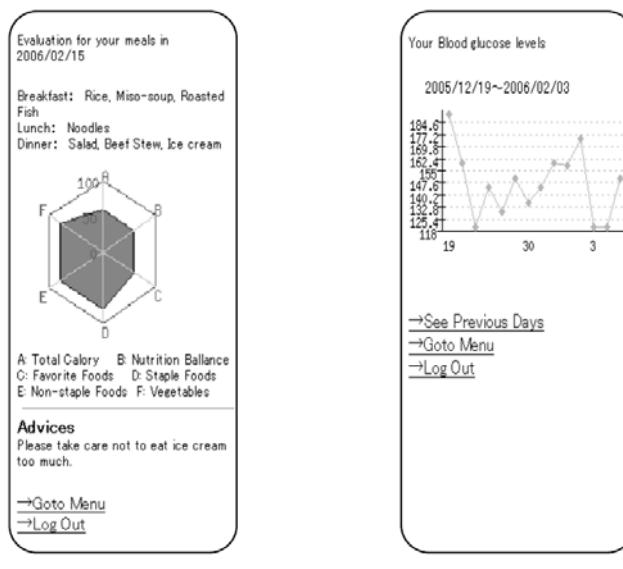
4. Evaluation

To evaluate the effect of our system, we performed a short (about three months) test operation with several patients and had interviews for doctors, nurses, and dietitians. All of them said that the system is useful to improve diabetic treatments, but at the same time, several problems arose.

Through the test operation, we found that wrong treatments about medicines occur more than expected. Even in our short-term operation, we found two of the cases through alert indication, and correct them. As another concrete result, one patient has lost weight by 1kg per week. This result is mainly from the meal evaluation function; certain evaluation and advices really effect well in this case.

As for each function, the function for meals is particularly evaluated highly. Namely, sending photo images and evaluation of them are said to be very effective. One big reason is that the size of photo image sent this time is 640*480 so that detail of the food menu can be understood. Then dietitians can make proper advices and encouragements that really effect. Also, the alert indication is evaluated well. By this function, doctors actually found wrong treatments without taking time and also doctors did not overlook them.

On the other hand, several problems to be solved also arose. One is the problem of operating mobile phones. To do data communication using mobile phones, patients must be accustomed to doing that. However, many of people are not accustomed with mobile phones. Therefore, some solutions such as making lectures of reporting are desired. Also, daily reports can be considerable labor for patients. It is desirable to reduce the number of reporting items or reporting days according to the patient's diabetic condition or mental character.



(a) Evaluation Results

(b) Blood Glucose Levels

Figure 5. Self-checking of Treatment Conditions on Mobile Phones

As other problems, although sending photo-images works effective, there were several cases that dietitians could not understand what the patient ate. One reason is that some foods cannot be distinguished only by outward appearance, but in many cases, the problem comes from the technique to take photos, e.g., out of focus, bad angle, or hardness to tell the size of foods. It is expected that the latter case can be solved by making some instruction for patients.

Group treatment policy is also important. In the hospital, a group of doctors and nurses perform treatments of the same patient. Thus, to treat patients in the same way, we have made a guideline of treating patients with this system.

5. Conclusion

We designed and implemented the new diabetic self-treatment supporting system, and evaluated it through the short-term test operation. Our trial is how patients can report the concrete condition of their self-treatment activities, and how doctors can make a careful distance help of the self-treatments. For this purpose, we designed the system carefully and made a test operation to know how the system works in the scene of medical treatments. As a result, we found that several functions work well and actually made good effects. Particularly, alert indication and photo evaluation of meals got great marks. Nevertheless, we also found several problems which should be solved, such as operation ability, labor of operating mobile phones, operation policy, and so on. Anyway, it is worth trying to solve those problems and make good use of the system by applying further ideas to raise patients' and doctors' motivation to use this system.

Acknowledgements

The authors would like to thank Daisuke Morihama and Izumi Kawamura who made great effort to design and implement the system. We also would like to thank doctors, nurses and dietitians who gave us many beneficial ideas and advices on designing it.

References

- [1] National Diabetes Fact Sheet, Department of Health and Human Services of U.S.A, 2005.
- [2] Fact-finding Survey on Diabetes in Japan, Ministry of Health, Labour and Welfare of Japan, 2002 (In Japanese).
- [3] Takumi Ichimura, Machi Suka, Akihiro Sugihara, Kazunari Harada, "Health support Intelligent System for Diabetic Patient by Mobile Phone," Proc. of the 9th International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies(KES2005), Vol.3, pp.1260-1265, 2005.
- [4] Diabetic Treatment Supporting System "Wagaman," <http://ichimura.chi.its.hiroshima-cu.ac.jp/hsisd/> (In Japanese).
- [5] e-SMGB, ARKRAY Inc., <http://www.ekenko.ne.jp/esmbg/>.

4. Special Techniques and Applications

This page intentionally left blank

Parallel Preprocessing for Classification Problems in Knowledge Discovery Systems

N.R. AKCHURINA¹, V.N. VAGIN²

¹*International Graduate School of Dynamic Intelligent Systems, Paderborn, Germany*

²*Moscow Power Engineering Institute, Krasnokazarmennaya 14, 111250, Moscow, Russia*

Abstract. A new algorithm for solving the actual problem in machine learning of joint preprocessing of qualitative and quantitative attributes with missing values is proposed. A parallel version of the algorithm developed by the authors is also presented. In thorough tests on 55 databases from the UC Irvine Repository specially designed from real databases of various fields for testing and comparing generalization algorithms, usage of the proposed algorithms has allowed to increase the classification accuracy (the main criterion of learning process) of the well-known classification algorithms: ID3, C4.5, Naïve Bayes, table majority, instance based algorithm almost in all the cases. In case of resources being available the parallel version of the algorithm allows to speed up preprocessing efficiently.

Keywords. Table preprocessing for knowledge discovery in databases, parallel table preprocessing for knowledge discovery in databases, discretization, attribute values grouping, data mining, rough sets

Introduction

In modern society, the amount of information stored in databases is constantly growing. The commonality of these data is that they contain a large number of hidden regularities that are very important for making strategic decisions.

To solve this problem, a new generation of intelligent tools for automated data mining and knowledge discovery is needed. One of the main problem solved by knowledge discovery and arising in such fields as marketing (a common use of mining), investment analysis, detecting insurance or banking fraud, analyzing Supreme Court decisions, discovering patterns in health care, and analyzing sequences in the human genetic makeup [1,2] is classification problem.

Classification [3] is learning a function that maps (classifies) a data item into one of several predefined classes.

The main characteristic of the learning process is the accuracy shown by the classifier on the unclassified objects.

The Knowledge Discovery in Databases process [3] can involve a significant iteration and may contain loops among data selection, data preprocessing, data transformation, data mining, and interpretation of mined patterns. The most complex

steps in this process are data preprocessing and data transformation which include the following operations: discretization of continuous-valued attributes, grouping values of symbolic attributes and imputing missing values.

The last operation is necessary because most of the algorithms can't directly process tables with some data missing.

There are different reasons for using discretization and grouping [4,5]:

1. Many supervised machine learning algorithms require a discrete feature space.
2. It usually increases the speed of induction algorithms.
3. It can exceed the classification accuracy.

Discretization and grouping methods can be classified [4,5] as global vs. local, supervised vs. unsupervised.

Discretization / grouping methods that do not make use of instance labels in the discretization / grouping process in analogy to supervised versus unsupervised learning methods are called unsupervised discretization / grouping methods. In contrast, discretization / grouping methods that utilize the class labels are referred to as supervised discretization / grouping methods.

Methods of discretization / grouping restricted to single attributes are called local, while methods that simultaneously convert all attributes are called global.

It was shown [5] that usage of global discretization and symbolic grouping methods allows to increase classification accuracy more than usage of local ones, as well as results shown by induction algorithms on datasets preprocessed by supervised discretization and symbolic grouping algorithms surpass the results on datasets preprocessed by unsupervised ones.

In connection with it a very interesting phenomenon should be mentioned. The performance of the C4.5 induction algorithm [5] can be significantly improved if the attributes were discretized in advance in spite of the fact that C4.5 is capable of locally discretizing features.

Basing on these notes and the idea that extreme globalization is processing all available attributes, the objective of this paper is to generalize global and supervised value grouping problem for simultaneous processing purely qualitative, scaled qualitative, continuous numerical, and discrete numerical attributes and to develop an algorithm that solves the stated problem under conditions with certain attribute values missing, as well as a parallel version of the algorithm.

1. Basic Concepts of Rough Set Theory

An *information system* is a pair $A = (U, \Lambda)$, where U is a non-empty, finite set called the *universe* and Λ is non-empty, finite set of *attributes*, i.e. $a : U \rightarrow V_a$ for $a \in \Lambda$, where V_a is called the *value set* of a . Elements of U are called *objects*.

Any information system of the form $A = (U, \Lambda \cup \{d\})$ is called *decision system* (*decision table*) where $d \notin \Lambda$ is called *decision attribute* and the elements of Λ are called *conditional attributes* or simply *conditions*.

The decision d determines a partition $CLASS_A(d) = \{X_A^1, \dots, X_A^{r(d)}\}$ of the universe U , where $X_A^k = \{x \in U \mid d(x) = v_d^k\}$ for $1 \leq k \leq r(d)$. The set X_A^i is called the *i-th decision class* of A .

With any $B \subseteq \Lambda$, there is associated an equivalence relation $IND_A(B)$ called the B -indiscernibility relation: $IND_A(B) = \{(x, x') \in U^2 \mid \forall a \in B \quad a(x) = a(x')\}$

For any $B \subseteq \Lambda$ and $X \subseteq U$ we denote the restriction of equivalence relation $IND_A(B)$ to X by $IND^X(B)$, and the family of all equivalence classes of $IND^X(B)$ by $[IND^X(B)]$.

Generalized decision in A is the function $\partial_A(x) = \{i \mid \exists x' \in U \quad x' IND(\Lambda)x \quad d(x') = i\}$. A decision table A is called *consistent (deterministic)* if $|\partial_A(x)| = 1$ for any $x \in U$, otherwise A is *inconsistent (non-deterministic)*.

2. Generalized Attribute Value Partition Problem

It is necessary to transform the source decision system, so that pairs of objects that belong to different decision classes should be discernible by conditional attributes in the resulting decision system iff they are discernible by conditional attributes in the source system, and, in addition, so that the number of the values that can be taken by conditional attributes of the resulting system should be minimal.

Let us present a formal definition of the generalized value partition problem [6,7].

Let $A = (U, \Lambda \cup \{d\})$ be a decision table where $\Lambda = \{a_i : U \rightarrow V_{a_i}\}$, for $i \in \{1, \dots, k\}$. Any function $P_{a_i} : V_{a_i} \rightarrow \{1, \dots, m_i\}$ is called a *partition of V_{a_i}* . The *rank of P_{a_i}* is the value $rank(P_{a_i}) = card(P_{a_i}(V_{a_i}))$. The family of partitions $P = \{P_a\}_{a \in \Lambda}$ defines from A a new decision table $A^P = (U, \Lambda^P \cup \{d\})$, where $\Lambda^P = \{a^P : a \in \Lambda\}$ and $a_i^P(u) = P_{a_i}(a_i(u))$ for any $u \in U$, $i \in \{1, \dots, k\}$.

The family of partitions P_A - *consistent*, if $\partial_A = \partial_{A^P}$, where ∂_A and ∂_{A^P} are generalized decisions of A и A^P . A - consistent family of partitions P^{opt} is called *A-optimal*, if $\prod_{a \in \Lambda} rank(P_a^{opt}) \leq \prod_{a \in \Lambda} rank(P_a)$ for any A -consistent family of partitions P .

Generalized value partition problem consists in finding an *A-optimal* family of partitions P^{opt} and obtaining the decision table $A^{P^{opt}}$.

It is obvious that in order to ensure that a pair of objects u_j, u_k that is discernible by conditional attributes in the source system is discernible by conditional attributes in the resulting system, it is necessary that the values of at least one discerning attribute of the objects u_j, u_k turn into different values in the resulting decision system.

To save the information about the values of this attribute for the objects u_j, u_k in the source decision system, we introduce the concept of *chain*.

Let $A = (U, \Lambda \cup \{d\})$ be a decision system, where $U = \{x_1, x_2, \dots, x_n\}$, $\Lambda = \{a_1, \dots, a_k\}$ and $d : U \rightarrow \{1, \dots, r\}$ and $V_{a_i} = \{v_1^{a_i}, v_2^{a_i}, \dots, v_{n_i}^{a_i}\}$. A triple $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ is called a *chain*, where $a_i \in \Lambda$ for $i = 1, \dots, k$ and $i_1, i_2 \in \{1, \dots, n_i\}$.

It is necessary to find the least number of chains C that discern all pairs of objects from different classes that are discernible by at least one condition attribute in the source system. In the resulting decision system, for each chain $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ from the obtained set, the values $v_{i_1}^{a_i}, v_{i_2}^{a_i}$ should turn into different values of the attribute $a_i^{P^{opt}}$. This problem clearly can be reduced to the problem of coloring n graphs constructed for the attributes a_i , $i = 1..n$, such that the set of its nodes is the value set of the attribute a_i , and the set of its arcs is the set of all chains in C of the attribute a_i .

To find C at each iteration it is necessary to choose a chain that discerns as many pairs of objects from different decision classes as possible indiscernible by the chains found at the previous iteration steps.

To calculate these pairs of objects, we propose concept of *d - relative discernibility matrix for chains* based on concept of *d - relative discernibility matrix*.

d - relative discernibility matrix is a symmetric matrix with entries $c_{ij} = \{a \in \Lambda \mid a(x_i) \neq a(x_j) \wedge d(x_i) \neq d(x_j)\}$.

d - relative discernibility matrix for chains is a symmetric matrix with entries $c_{ij} = \{(a, v_1, v_2) \mid a(x_i) = v_1 \wedge a(x_j) = v_2 \wedge v_1 \neq v_2 \wedge d(x_i) \neq d(x_j)\}$.

It is obvious that the problem of choosing a chain that discerns as many pairs of objects from different decision classes as possible could be reduced to the problem of finding the most frequently met chain in the elements of *d - relative discernibility matrix for chains* $M(A)$.

On the basis of these concepts an algorithm proposing a solution to the above formulated problem was developed. The speciality of this algorithm is its ability to work directly with decision tables which have some data missing. The strategy of obtaining missing values bases on the idea of consistency of the table (when we are finding a feature for some object we exclude those attribute values which lead to inconsistency basing on the object's features known).

The algorithm operates under the assumption that the source decision system is consistent.

2.1. Algorithm for Generalized Attribute Value Partition Problem (VG)

Step 1.

Perform a naive discretization that consists in dividing the continuous attributes value domains into a given number of equal intervals.

Step 2.

Each missing value is replaced by a unique value. Denote by NV_i the set of unique values of the attribute $a_i \in \Lambda$, $i \in \{1,..,k\}$.

Step 3.

Find *d - relative discernibility matrix for chains* $M(A)$.

Step 4.

Find the most frequently met chain $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ among the elements of *d - relative discernibility matrix for chains* $M(A)$.

Step 5.

The chain $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ is added to the chain set C.

Step 6.

All the elements of d - relative discernibility matrix for chains $M(A)$ that contain chain the $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ are substituted for empty sets.

Step 7.

If all the elements of d - relative discernibility matrix for chains $M(A)$ are empty sets, then go to step 8, else go to step 4.

Step 8.

For any $a_i \in \Lambda$, we construct a graph $\Gamma_{a_i} = \langle V_{a_i}, E_{a_i} \rangle$, where V_{a_i} is the value set of the attribute a_i and E_{a_i} is the set of all chains in C of the attribute a_i . Apply the efficient strategy of coloring the graph and obtain a function $f_i : V_{a_i} \rightarrow \{1, \dots, m_i\}$, $i = 1, \dots, k$.

Step 9.

$F = \{f_i\}_{a_i \in \Lambda}$ determines from $A = (U, \Lambda \cup \{d\})$ a new decision system $A^F = (U, \Lambda^F \cup \{d\})$, where $\Lambda^F = \{a^F : a \in \Lambda\}$ and $a_i^F(u) = f_i(a_i(u))$ for any object $u \in U$, $i = 1, \dots, k$.

Step 10.

Divide numerical attributes into intervals, and the scaled qualitative attributes into subsets.

Step 11.

Stop.

2.2. Parallel Version of the Algorithm for Generalized Attribute Value Partition Problem

Steps 1-2 copy those of algorithm 1.

Step 3

For each process PRank of PNum

Form matrix $M[PRank]$ from k th ($k \bmod PNum = PRank$) column of d - relative discernibility matrix for chains.

Step 4.

Find the most frequently met chain $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ among the elements of $M[PRank]$.

Step 5.

The chain $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ is added to the chain set $C[PRank]$.

Step 6.

All the elements of $M[PRank]$ that contain the chain $(a_i, v_{i_1}^{a_i}, v_{i_2}^{a_i})$ are substituted for empty sets.

Step 7.

If all the elements of $M[PRank]$ are empty sets, then go to step 8,
else go to step 4.

Step 8.

Form chain set S from the most frequently met chains in $C[PRank]$.

Step 9.

For each process $PRank$ of $PNum$

Check up whether S discerns all objects from different decision classes using matrices $M[PRank]$.

If it does then go to step 10, else add the most frequently met chain not yet added to S and go to step 9.

Step 10 - 12. copy steps 9-11 of algorithm 1.

3. Experimental Results

The presented algorithms have been implemented in Assembler and tested on databases of the UC Irvine Repository, which was specially designed from real databases of various fields for testing and comparing classification algorithms [8]. Table 1 shows the increase of the classification accuracy reached by the well-known algorithms:

- ID3 [9]
- C4.5 [9]
- Naïve Bayes [10]
- Table Majority [11]
- Instance Based [12]

on 55 data sets of this repository preprocessed by the developed algorithms (columns $ID3/C4.5/NB/TM/IB$ – classification accuracy of ID3/C4.5/Naïve Bayes/Table Majority/Instance Based classifiers; $VG \rightarrow ID3/VG \rightarrow C4.5/NB \rightarrow ID3/TM \rightarrow ID3/IB \rightarrow ID3$ – classification accuracy of ID3/C4.5/Naïve Bayes/Table Majority/Instance Based classifiers on the tables preliminarily preprocessed by the proposed algorithms for generalized attribute value partition problem (VG)) and the characteristics of the databases:

train/test – the number of objects in train/test data set;

C – marked in case the corresponding data set contains quantitative attributes;

Q – marked in case the corresponding data set contains qualitative attributes;

M – marked in case some values are missing in the corresponding data set.

Cells were shaded when classification accuracy hasn't degraded on the data sets preprocessed by the presented algorithms.

4. Analysis of the Results

It is worth noting that almost in all the cases usage of the proposed algorithms allows to increase or leave invariable the classification accuracy of the well-known classification algorithms.

But how can one know for certain that the use of the proposed methods will yield good results?

Table 1. Classification accuracy of ID3, C4.5, NB, TM, IB with/without preliminary preprocessing by the algorithm for generalized attribute value partition problem (VG).

		Train	Test	C	Q	M	ID3 (%)	VG → ID3 (%)	C4.5 (%)	VG → C4.5 (%)	NB (%)	VG → NB (%)	TM (%)	VG → TM (%)	IB (%)	VG → IB (%)
1	Audiology	151	77			√	73,68	68,42	76,32	73,68	53,95	69,74	26,32	26,32	71,05	67,11
2	Australian	461	231	√	√		81,30	76,96	86,96	88,70	77,83	87,83	60,00	58,70	80,87	76,52
3	balance-scale	417	210	√			78,47	72,73	77,03	69,86	89,47	88,04	44,98	44,98	66,03	66,03
4	Banding	138	100	√	√	√	82,00	64,00	77,00	67,00	86,00	81,00	63,00	63,00	66,00	61,00
5	Breast	466	233	√		√	94,42	93,99	93,99	91,42	96,57	94,85	70,39	74,25	95,28	92,27
6	breast-cancer	191	95		√	√	71,58	66,32	74,74	73,68	74,74	76,84	74,74	73,68	57,89	61,05
7	Cars1	262	132	√			81,68	73,28	74,81	72,52	63,36	62,60	63,36	62,60	71,76	63,36
8	Chess	2130	1066		√		98,69	96,62	99,53	97,37	87,15	86,77	53,19	68,29	90,43	90,53
9	Cleve	202	101	√	√	√	64,36	77,23	76,24	80,20	82,18	84,16	62,38	65,35	71,29	71,29
10	Corral	34	128		√		87,50	100,00	81,25	100,00	90,63	87,50	82,81	100,00	86,72	100,00
11	Crx	490	200	√	√	√	72,50	76,50	83,00	81,50	76,50	81,00	55,00	57,50	74,00	73,00
12	DNA-nominal	2000	1186		√		90,30	91,32	92,41	92,58	94,60	94,10	57,76	63,74	74,20	87,27
13	Echocardiogram	88	45	√	√	√	63,64	59,09	63,64	65,91	68,18	75,00	65,91	61,36	54,55	61,36
14	Flare	711	357	√	√		81,46	82,30	85,11	85,11	81,18	72,19	75,84	84,55	75,28	83,99
15	German	667	335	√	√		66,77	73,95	73,05	73,95	77,55	73,95	72,75	73,35	67,96	65,87
16	german-org	667	335	√	√		71,56	71,56	74,55	72,46	74,85	72,46	72,75	72,75	69,16	69,16
17	Glass	142	72	√			62,50	65,28	62,50	61,11	50,00	59,72	30,56	40,28	65,28	61,11

18	glass2	108	55	✓		69,09	85,46	69,09	78,18	65,46	78,18	58,18	80,00	50,91	78,18
19	hayes-roth	132	28	✓		82,14	92,86	82,14	92,86	64,29	89,29	60,71	82,14	75,00	85,71
20	Heart	181	91	✓		76,67	80,00	83,33	80,00	85,56	85,56	57,78	57,78	76,67	74,44
21	Hepatitis	103	52	✓	✓	78,85	80,77	71,15	75,00	76,92	78,85	86,54	86,54	84,62	82,69
22	Ionosphere	235	118	✓		91,45	91,45	88,03	88,89	84,62	91,45	60,68	65,81	75,21	88,03
23	Iris	100	50	✓		94,00	94,00	92,00	92,00	94,00	96,00	30,00	92,00	78,00	96,00
24	labor-neg	40	17	✓	✓	94,12	82,35	76,47	82,35	88,24	88,24	64,71	64,71	88,24	94,12
25	led24	200	3000		✓	55,33	33,87	65,57	39,63	64,10	42,37	9,83	11,33	36,93	28,20
26	led7	200	3000		✓	66,53	61,20	67,40	43,37	68,93	66,33	56,80	41,13	60,90	46,97
27	Lenses	17	9		✓	62,50	62,50	62,50	62,50	37,50	62,50	37,50	62,50	75,00	62,50
28	lenses-full	24	24		✓	100,00	100,00	91,67	91,67	95,83	95,83	100,00	100,00	100,00	100,00
29	Liver-disorder	231	116	✓		53,04	61,74	60,87	63,48	55,65	59,13	62,61	64,35	62,61	62,61
30	Lung-cancer	29	5		✓	50,00	50,00	50,00	50,00	50,00	50,00	25,00	50,00	75,00	75,00
31	lymphography	98	50	✓	✓	78,00	72,00	74,00	76,00	82,00	84,00	60,00	72,00	66,00	70,00
32	mofn-3-7-10	301	1025		✓	91,02	100,00	85,55	91,41	86,43	85,94	83,20	99,22	89,06	100,00
33	monk1	124	432		✓	81,02	100,00	75,69	100,00	71,30	75,00	64,35	100,00	78,70	100,00
34	monk2	169	432		✓	69,91	95,60	64,58	88,43	61,57	56,02	81,94	100,00	73,84	98,15
35	monk3	122	432		✓	91,67	95,37	97,22	97,22	97,22	97,22	60,65	77,78	82,87	89,35
36	parity5+5	100	1024		✓	50,78	100,00	50,00	81,25	50,00	50,00	54,39	100,00	53,03	100,00
37	Post-operative	61	31	✓	✓	60,00	60,00	66,67	66,67	60,00	60,00	66,67	56,67	53,33	50,00
38	primary-org	226	113		✓	40,71	35,40	45,13	39,82	46,02	46,90	23,89	27,43	32,74	41,59

39	primary-tumor	227	114		✓		40,71	35,40	45,13	39,82	46,02	46,90	23,89	27,43	32,74	41,59
40	Solar	216	109	✓	✓		72,22	70,37	69,44	74,07	63,89	66,67	50,00	58,33	61,11	64,81
41	Sonar	139	71	✓			78,57	62,86	74,29	60,00	70,00	68,57	55,71	55,71	48,57	52,86
42	soybean-small	31	16	✓			100,00	100,00	100,00	100,00	100,00	100,00	31,25	100,00	100,00	100,00
43	Splice	2127	1065		✓		90,04	91,73	93,33	93,05	94,64	93,70	60,15	63,91	73,50	87,22
44	threeOf9	428	86		✓		100,00	100,00	98,82	98,82	76,47	76,47	60,00	60,00	76,47	76,47
45	tic-tac-toe	639	321		✓		80,94	91,56	81,88	87,19	71,25	69,69	63,44	79,69	78,13	84,69
46	tokyo1	480	481	✓			88,33	85,00	90,42	85,42	78,75	85,63	62,08	65,21	77,50	83,54
47	Tutorial	13	16		✓		80,00	100,00	86,67	86,67	80,00	86,67	93,33	100,00	100,00	100,00
48	Vehicle	565	283	✓			69,86	64,54	68,09	62,06	39,36	58,87	23,05	25,89	60,64	62,77
49	Vote	300	135		✓		94,07	95,56	97,04	97,04	91,85	94,82	78,52	87,41	94,81	97,04
50	Vote1	300	135		✓		89,63	89,63	92,59	90,37	89,63	89,63	78,52	81,48	88,89	89,63
51	Vote1-irvine	290	145		✓		86,90	86,90	88,28	87,59	88,97	88,28	80,00	84,83	89,66	86,90
52	Vote-irvine	290	145		✓		93,79	92,41	95,17	95,17	89,66	93,10	80,00	90,34	92,41	94,48
53	Wine	119	61	✓			85,00	78,33	85,00	81,67	95,00	83,33	43,33	46,67	53,33	76,67
54	xd6	461	512		✓		100,00	100,00	99,02	99,02	81,84	81,84	96,88	96,88	98,63	98,63
55	Zoo	67	34		✓		97,06	94,12	79,41	82,35	64,71	94,12	58,82	94,12	88,24	94,12

As there is no universal generalization algorithm, that will perform better on all tables than its rivals [1], there is also no answer. The best decision is to implement various algorithms based on different approaches and divide the model set into three (not two as was proposed earlier) sets:

- the *training* set – to train all algorithms and with and without preprocessing;
- the *validation* set – to choose the best variant;
- the *test* set – to predict accuracy on unseen data.

Tests have shown that the parallel version of the algorithm allows to reach approximately the same classification accuracy but in case of resources being available permits to speed up preprocessing efficiently.

5. Conclusions

In this paper, a solution to a relevant problem [6,7] of preprocessing decision tables that contain both qualitative and numerical attributes with missing values was proposed. A parallel version of the algorithm was also presented. The results of thorough testing of the proposed algorithms on 55 databases from the UC Irvine Repository, which is specially designed for testing and comparing generalization algorithms [8], were also presented, which demonstrated that the proposed algorithms usage allows to increase the classification accuracy of the well-known classification algorithms: ID3 [9], C4.5 [9], Naïve Bayes [10], table majority [11], instance based algorithm [12] almost in all the cases. Tests also showed that the parallel version of the algorithm allows to reach approximately the same classification accuracy but in case of resources being available permits to speed up preprocessing efficiently.

References

- [1] M.J.A. Berry, G.S. Linoff, Data Mining Techniques: for Marketing, Sales, and Customer Relationship Management. Second ed. Wiley-Interscience, New York, 2004.
- [2] P. Langley, Applications of Machine Learning and Rule Induction / Communications of the ACM 38(11) (1995), ACM Press, New York.
- [3] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uturusamy (eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Cambridge, MA, 1996.
- [4] D. Pyle, Data Preparation for Data Mining, Morgan Kaufmann Publishers, New York, 1999.
- [5] J. Dougherty, R. Kohavi, M. Sahami, Supervised and Unsupervised Discretization of Continuous Features / Machine Learning: Proceedings of the Twelfth International Conference // Armand Prieditis A., Russell S. (eds.). Morgan Kaufmann Publishers, San Francisco, 1995.
- [6] N.R. Akchurina, V.N. Vagin, Generalized Value Partition Problem: A Rough Set Approach / Journal of Computer and Systems Sciences International, 43(2) (2004), 223-238.
- [7] V.N. Vagin, N.R. Akchurina, New Techniques for Handling Missing Data and Feature Extraction for Knowledge Discovery / Knowledge-Based Software Engineering // Stefanuk V. and Kajjiri K. (eds.), IOS Press, 2004, 169-176.
- [8] <http://www.ics.uci.edu/~mlearn/MLRepository.html>
- [9] J.R. Quinlan, C 4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, California, 1993.
- [10] P. Langley, W. Iba, K. Thompson, An Analysis of Bayesian Classifiers / Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press and MIT Press, 1992, 223-228.
- [11] R. Kohavi, The Power of Decision Tables / Proceedings of the European Conference on Machine Learning, Lecture Notes in Artificial Intelligence 914 // Lavrac N., Wrobel S. (eds.), Springer Verlag, New York, 1995, 174-189.
- [12] D. Wettschereck, A Study of Distance-Based Machine Learning Algorithms, PhD thesis, Oregon State University, 1994.

Categories for Description of Dynamic Production Systems¹

Vadim STEFANUK, Alexander ZHOZHIKASHVILI
Institute for Information Transmission Problems
Russian Academy of Sciences
Bolshoy Karetny per., 19
127994 Moscow GSP-4, Russia
stefanuk@iitp.ru; zhozhik@iitp.ru

Abstract. Based on the categorical formalism developed for the production type knowledge-based Intelligent System, the paper describes a scheme, which allows an easy conversion of a category, build for a static production system, into a category, suitable for description of a dynamic knowledge system. The scheme is shown to be applicable for an arbitrary category. For instance, the presented examples demonstrate that the scheme might describe many different ways of introduction of dynamic properties into an intelligent system. Besides the pure dynamic intellectual systems the proposed scheme may be applied to other knowledge-based systems, where it is important and essential to store the information on the way the system inferences have been made.

1. Introduction

The theoretical results presented in the paper follow from our original approach to the knowledge-based AI, which has been used in our group for many years. The central element of this approach is a knowledge base of rules, or the production knowledge base. However, the productions used are understood in a essentially more general way than in it is done in the most approaches found in the literature on AI and Expert Systems. Actually each production is being treated as a certain entity describing some elementary operation intended for the use in AI systems. The central property of this elementary operation is the ability to recognize the situation, which AI system comes across with, and then to find in the system memory a description of this situation in order to produce (in accordance with the found information) from the given situation a new one.

To study the productions understood in so wide sense the present authors have developed a special abstract language allowing the decision making to be independent

¹ The research has been supported by the Russian Fund for Basic Research, project 05-01-00382, and by the project 2.24 issued by the Russian Academy of Science.

on the concrete implementation of situations in a concrete AI system. This language operates with so-called CT-productions. It is based on the mathematical formalism of Category Theory [1].

Basic notions for CT-productions are situations, patterns – generalizations of the situations, and matching operations. For each subject domain a category must be described, corresponding to those three notions. Then productions in this category may be represented as pairs of morphisms, the knowledge base being a set of pairs of morphisms of certain type, the data base may be represented as a set of some morphisms. Then the inference in the system consists in finding new morphisms [2, 3].

In the present paper we propose some development of the language in order to include into our theory the area of dynamic production systems. In the latter systems it is possible to reconsider of data already obtained by the AI system, which might lead to change of some inferences made in the system before and to a possible replacement them with some new inferences. For the dynamic case there should be some changes in the developed language made in order to allow the category morphism to store some information, allowing to decide on either usage of the morphism or on its rejection.

The present authors proposed a concept of *Dynamic Expert System* [4, 5, and 6] and have implemented it by a simple conversion of an ordinary (static) ES into a dynamic one. The conversion was based on the introduction of data of two different classes, static and dynamic, and allowing the Expert System to reconsider its past inferences due to the dynamic data. Such a Dynamic Expert System "Seismo" was successfully applied in the area of Seismology Events Prediction [4].

In development of the new TC language to cover Dynamic Expert System a similar problem was formulated: having a category for a certain problem domain with a certain collection of situations, patterns, and matching operations to convert it, introducing minimal changes, into a category, describing a dynamic system intended for the same problem domain.

Along these line a notion of *layered category* has been introduced which will be described below.

2. The Language of CT-productions

The basic notion in this language is the notion of situation. Let us assume that our AI system deals with a certain problem. Then the *situation* describes those evidences concerning the problem to be solved, which are available to the AI system up to a certain stage in its reasoning process. After new evidences arrive the situation changes.

In order to describe a set of similar situations, which require from the AI system to make similar actions despite of the difference of these situations in some details, the notion of a *pattern* is introduced. The main operation of the production system is the *matching* of the situation against the pattern, which shows whether the situation matches the pattern, and besides produces some important side-information, describing peculiarity of the given situation. The production is understood as a pair of patterns, the first describing the situation for which the production may be applied and the second – the situation obtained after its application.

In our language of CT-productions to make a system of patterns means to make a category, among morphisms of which we distinguish certain class of morphisms referred to as the situations, provided that if the morphisms α , β , and ϕ are

connected with the relation $\alpha = \beta\varphi$, then the morphism presents a situation if and only if morphism β does present some situation. We call the triple (X, S, φ) a pattern, where X, S - are objects of the category, and $\varphi : X \rightarrow S$ presents a morphism.

The situation $\alpha : I \rightarrow S$ is said to be matching the pattern $\varphi : X \rightarrow S$, if there is morphism $\beta : I \rightarrow X$, satisfying the condition $\alpha = \beta\varphi$. The pair of patterns (X, S, φ) and (X, T, ψ) , where S and T are the objects, is referred to as the production, i.e. the production is defined as the pair of morphisms $\varphi : X \rightarrow S$ and $\psi : X \rightarrow T$. The production is considered applicable to the situation $\alpha : I \rightarrow S$ when for some morphism $\beta : I \rightarrow X$ one has $\alpha = \beta\varphi$. The result of application of the production to the situation α will be $\beta\psi : I \rightarrow T$.

In order to be able to consider the concrete problem to be solved with the intellectual system within this scheme, it is necessary to define what kind of the situations may arise during solution, what patterns may be reasonably found to describe sets of similar situations and decide how the matching procedure should look like. After this one may build a category to describe such situations and patterns. Now the knowledge base may be represented as a certain set of morphisms which are involved in productions, while the current state of the system inference is defined with a situational morphism.

3. Context Lattice

An elementary fragment of knowledge, obtained by the system in the inference process will be referred to as *fact*. (In our Expert System Shell "Znatok" it is for instance an assertion ATTRIBUTE=VALUE.) In our terminology the facts may be either static or dynamic. While the truth value of a static fact is absolute, the truth value of the dynamic fact is context dependent.

In order to keep our exposition general enough we will not define the notion of context at this stage. The context may include information on time, on validity of some conditions, on arising of certain events and etc.

Let P be the set of all possible contexts. When $p \in P, q \in P$, we will assume $p = q$ if any dynamic fact, which is true in the context p is true in the context q and *vice versa*. If any dynamic fact which is true in the context p is also true in the context q , we will write $p \leq q$. It is easily verified that the binary relation \leq , which was just defined, is the relation of order, and hence the set P becomes an ordered set. It is possible that for $p \in P, q \in P$ there are exist dynamic facts, which are true in the context p , but false in the context q and in the same time there may exist facts, which are true in the context q , but false in the context p . In this case the contexts p и q are not comparable. Thus, the introduced order is partial order, and is not linear one.

Let now one fact is true in the context p and another one is true in the context $q, p \in P, q \in P$. When $r \in P, p \leq r, q \leq r$, in accordance with our definition, both facts are true in the context r . It is frequently considered important to find a *minimal context* r . In order to provide its existence let us require that in this partially ordered set of elements each pair of elements does have its least upper bound, i.e. the set P to be a lattice [7].

This upper bound we will denote as it usually done in the theory of lattices, with the symbol $p \vee q$. We will request in addition that the set P would have the smallest element, which we will denote with the symbol 0. The minimal context means that we may assert only the truth of static factors and there are no arguments in favor of any of dynamic ones. Below we will refer to such a lattice as the *context lattice*.

4. Layered Category

Let now \mathbf{C} be an arbitrary and P a context lattice. Let us build a new category \mathbf{C}_P in the following way. The objects of the category \mathbf{C}_P will be the same as the objects of the category \mathbf{C} . Let X, Y be some objects of category \mathbf{C} (and consequently of the category \mathbf{C}_P), $\varphi \in \mathbf{C}(X, Y)$. We will define the set of morphisms from X to Y in the category \mathbf{C}_P with the formulae $\mathbf{C}_P(X, Y) = \mathbf{C}(X, Y) \times P$. For brevity we will denote the element (φ, p) of the set $\mathbf{C}(X, Y) \times P$ with the symbol φ_p . Let $\varphi_p : X \rightarrow Y, \psi_q : Y \rightarrow Z$. Let us define the morphism $\varphi_p \circ \psi_q : X \rightarrow Z$ with the formulae $\varphi_p \circ \psi_q = (\varphi \circ \psi)_{p \vee q}$. The meaning of the above definition is quite transparent: a composition of two morphisms is defined in such a context, in which each of constituting morphisms are defined. In order the definition of the category \mathbf{C}_P would be correct it is necessary to check if the composition is associative one and that in the category the unit morphisms do exists. The first assertion is proved with the following chain of equalities: if $\varphi_p : X \rightarrow Y, \psi_q : Y \rightarrow Z, \tau_r : Z \rightarrow W$, then

$$\begin{aligned} (\varphi_p \psi_q) \tau_r &= (\varphi \psi)_{p \vee q} \tau_r = ((\varphi \psi) \tau)_{(p \vee q) \vee r} = (\varphi(\psi \tau))_{p \vee (q \vee r)} = \\ &= \varphi_p (\psi \tau)_{q \vee r} = \varphi_p (\psi_q \tau_r) \end{aligned}$$

The role of the unit morphism of the object X of the category \mathbf{C}_P is played by the morphism 1_0 , where the symbol 1 denotes the unit morphism of the object X of the category \mathbf{C} , and with the symbol 0 the smallest element of context lattice is denoted. Indeed, if $\varphi_p : X \rightarrow Y$, then $1_0 \varphi_p = (1\varphi)_{0 \vee p} = \varphi_p$, and if $\psi_q : Y \rightarrow X$, then $\psi_q 1_0 = (\psi 1)_{q \vee 0} = \psi_q$.

On the base of an arbitrary category \mathbf{C} and the context lattice P the provided procedure allows to build a new category, which will be called the category layered with help of a context lattice. As if a separate layer with its morphisms corresponds to each context. The layers are (partially) ordered and the everything presented in a lower layer preserved also in the above layer.

A functor $F : \mathbf{C}_p \rightarrow \mathbf{C}$ may be naturally defined, assuming $F(X) = X$ for each object X , and $F(\varphi_p) = \varphi$ for each morphism φ_p of the category \mathbf{C}_p . This functor is a version of "forgetting" functor, which forgets about layered structure of the category. The correctness of the functor definition follows from the next chain of equalities: $F(\varphi_p \psi_q) = F((\varphi \psi)_{pq}) = \varphi \psi = F(\varphi_p)F(\psi_q)$.

5. Versions of Context Lattices

The context lattice may be defined in a variety of ways. Let us consider some of the possibilities.

The context providing the truthfulness of some fact may consists in fulfillment of a number of conditions. Let M is the set of all conditions possible for consideration. The set of all the subset of M may be taken as P . Now when $p \in P$, $q \in P$, i.e. $p \subset M$, $q \subset M$, then the condition $p \leq q$ means that $p \subset q$. In this case we have an obvious formulae $p \vee q = p \cup q$. In other words, if for the truthfulness of some dynamic fact it is requested the truthfulness of some set of conditions, and for the truthfulness of another fact another set of conditions are to be true, then both facts will be true provided that all the above conditions are jointly fulfilled.

Some logical expressions may be considered as context. In this case the inequality $p \leq q$ means that an implication $q \Rightarrow p$ takes place. The operation of least upper bound becomes the following logical disjunction $p \vee q = p \& q$.

For some applications the next example context lattice is important. Let us assume that some established dynamic fact is true up to a certain moment in time, and after this moment its truthfulness should be verified. To each context $p \in P$ we will attach its corresponding moment t_p . Then P may be considered as the set of all real numbers. The condition $p \leq q$ means now that $t_p \geq t_q$. Indeed, if some fact is true in the context p , i.e. up to the moment t_p , where $t_q \leq t_p$, then it will be true up to the moment t_q , i.e. it is true in the context q , and hence $p \leq q$. Under such a definition of the inequality the set P is ordered linearly, and hence we may write $p \vee q = \max(p, q)$. In other words, $t_{p \vee q} = \min(t_p, t_q)$.

We will give another very simple and very important for us example of the context lattice. Let $P = \{0,1\}$, and as inequality an ordinary number inequality is taken. Here

the number 0 means "static", the number 1 means "dynamic". Thus, if φ is some morphism of the category \mathbf{C} , then the morphism φ_0 of the layered category \mathbf{C}_P may be used to describe static situations and patterns, while the morphism φ_1 may be used for description of the dynamic ones. As the ordering is linear one we again obtain that $p \vee q = \max(p, q)$. The operation to find the least upper bound has now the following meaning: when $p \in P$, $q \in P$, then $p \vee q = 0$ iff $p = 0$ and $q = 0$, i.e. $p \vee q$ corresponds to the state "static" if and only if both p and q correspond to this state. Thus, if $\varphi_p : X \rightarrow Y$ и $\psi_q : Y \rightarrow Z$ are morphisms of the category \mathbf{C}_P , then the morphism $\varphi_p \psi_q$ is dynamic one if at least one of its constituting morphisms is dynamic.

For dynamic systems it is necessary to change the mode of performance of the inference engine of the production system. Usually the result of the inference is a finite situation, which is added to the knowledge base. In case of dynamic system it is reasonable to store all the set of situations, obtained in the intermediate inference steps, i.e. the chain of morphisms, whose composition brings the result. If the inference is repeated then the system should not analyze the set of productions anew, as it may go along the same chain and to obtain the required result instantly. However in the layered category the stored morphisms obtained during the inference are also referred to the different layers. For the inference only those morphisms are admitted, which belong to the layer, corresponding to the current context. If the context changed, some of the morphisms might disappear. The latter means that the system will not be able to obtain the result and has to restart the inference on the base of productions from the very beginning. From the other side, those morphisms, which correspond to the given layer, will be successfully used and the repetition of the inference would not be needed.

One of the simplest way, which is also very effective, way of organizing of dynamic system is based on the last example of layered category, when one has only two layers: 0 – static and 1 – dynamic. In this version the system is restarted in certain time intervals. In this arrangement the layer 1 is removed, and hence all the dynamic fact are reconsidered. The static facts are stored in the knowledge base. The authors of this paper used such a quasistatic scheme to design an intelligent system for seismology phenomena prediction [5, 6].

6. The Other Applications of the Layered Categories

The idea of connecting to each morphism some information on the possibility of its use may find some applications beyond the area of dynamic systems. Let, for example, a number of users are allowed to work with a given system. Each user may have a knowledge base of his own and may enter the system his personal data. In the same time the system should distinguish the inferences, which are obtained due to the information from various users.

To implement such a model let us consider the set of users P . We will introduce on it a certain order that is a degeneration of the partial order, namely, we will consider

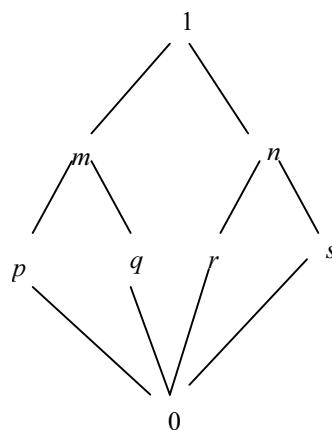
each two elements of the set P to be incomparable, i.e. the assertion $p \leq q$ will not be fulfilled for any two elements p and q of the set P . Such a set may not be considered as a lattice. We will add to the set two more elements $\{0, 1\}$ and will connect them with the rest of the set P by the conditions $0 \leq p \leq 1$ for all $p \in P$.

Now the set P became a lattice, which we will consider to be a context lattice of a layered category. In such a set the upper bound of the elements p and q , corresponding to two different users, is equal to 1. The meaning of the corresponding context is quite transparent. The context, corresponding to the user p , mentions the facts, which are true from the point of view of this user. The context 1 corresponds to the facts recognized with at least one of the users, i.e. something like a general knowledge. The 0 corresponds to the set of facts, which are intersection of the sets of facts recognized by some users.

Let now \mathbf{C} is some category, which will be used to build a layered category \mathbf{C}_P .

In accordance with the above definition of morphisms of the layered category, if φ_p and ψ_p are morphisms "belonging" to one of the users p , the composition of the morphisms will be the following one $\varphi_p \circ \psi_p = (\varphi \circ \psi)_{p \vee p} = (\varphi \circ \psi)_p$. Thus, each user works in the category \mathbf{C} , but any of the generated by him morphisms carries an information on this user. From the other side, if the morphisms φ_p and ψ_q are referred to different user, then the composition formulae gives the following $\varphi_p \circ \psi_q = (\varphi \circ \psi)_{p \vee q} = (\varphi \circ \psi)_1$. Thus, resulting morphism contains the information that it was obtained as a result of a collective efforts of various users.

If necessary, one may build more sophisticated models of user interaction. It is possible to distinguish several groups of users keeping close users in one group. Let the users p and q belong to one school, which we denote with the name m , yet the users r and s - belong to the school n . Let us build a context lattice with the help of the following graph:



Now the upper bound of elements p and q will be element m , hence by creating a composition of morphisms $\varphi_p \circ \psi_q = (\varphi \circ \psi)_{p \vee q} = (\varphi \circ \psi)_m$ we obtain a morphism "belongs to the school m ", while the morphism $\varphi_p \circ \psi_r = (\varphi \circ \psi)_{p \vee r} = (\varphi \circ \psi)_l$ will be considered still as referring to the activity of all users, as it was obtained in the result of activity of specialists belonging to different schools.

In conclusion we will show how to build a system which is the dynamic system for one of the users and in the same time is static one for the other user. Let first user deals with the dynamic system described with the layered category \mathbf{C}_p . Let us add to its context lattice one more element, which is less than any of its elements. This element will be denoted with the symbol -1 to emphasize that this element is less than the element 0 existing in P . It is easy to verify that the obtained set $P' = P \cup \{-1\}$ will be also a lattice. As before we will associate the context $p \in P$ with the morphisms of the first user, while as the morphisms of the second user will be considered morphisms of the type p_{-1} . Then the part of the category, which the first user deals with, coincides with \mathbf{C}_p ; i.e. he works with the dynamic system. While the second user in fact deals with category \mathbf{C} , i.e. his category is static one. When the joint use of the system is admitted, we obtain $\varphi_p \circ \psi_{-1} = (\varphi \circ \psi)_p$, i.e. composition of morphisms will be dynamic one and the data, obtained in the result of activity of both users, should be revised only when the information supplied by the first user needs to be reconsidered.

7. Conclusion

We described a scheme, which allows an easy conversion of a category, used to describe a static production system, into a category, suitable for description of a dynamic knowledge system. The scheme is applicable for an arbitrary category.

The presented examples showed that the scheme might describe many ways of introduction of the dynamic properties into a system. Besides the study of proper dynamic intellectual systems the similar scheme may be applied to other production systems, where it is important to store the information on the way the system inferences have been produced. One of the future theoretical researches would be to study, which properties of original category are inherited in the corresponding layered one.

References

- [1] Mac Lane, S. Categories for the working mathematician, 2nd edition, Springer 1998.
- [2] Zhozhikashvili, A., Stefanuk V.L. Categorical Patterns for Artificial Intelligence Problems. Proceedings of RAN: Theory and Control Systems, 5, 1999, pp. 5-16.
- [3] Stefanuk V.L., Zhozhikashvili A.V. Productions and rules in artificial intelligence, KYBERNETES, The International Journal of Systems & Cybernetics. MCB University Press: 2002, P. 817-826.
- [4] Stefanuk V.L. Dynamic expert system, KYBERNETES, The International Journal of Systems & Cybernetics. Vol. 29, 5/6, MCB University Press: 2000, P. 702-709.

- [5] Stefanuk V.L., Zhozhikashvili A.V. Design and debugging of Knowledge Base in a Dynamic Expert System. Earth Physics Institute of Russian Academy of Sciences, Report on the topic "Seismichnost", 1992, preprint, 26 P. (in Russian)
- [6] Stefanuk V.L. The Behavior of Quasistatic Shell in Changing Fuzzy Environment, The Proceedings of the IY National Artificial Intelligence conference KII'94, Rybinsk, 1994, V.1, pp.199-203 (in Russian)
- [7] Birkhoff G. Lattice theory. AMS Colloquium Publications, Vol. 25, 3rd Ed. Providence: American Mathematical Society 1967.

Linguistic Knowledge for Search Relevance Improvement

Gennady OSIPOV, Ivan SMIRNOV, Ilya TIKHOMIROV,
 Olga VYBORNOVA and Olga ZAVJALOVA

Institute for System Analysis, Russia

Abstract. The paper presents methods and software implementation for semantically relevant search. We mainly focus on the usage of linguistic knowledge for improvement of semantic relevance in search engines. We state the effectiveness of information retrieval systems based on semantic search involving refined linguistic processing tools. Advantages of semantic search in comparison with traditional search engines are discussed.

Keywords. Knowledge-based semantic search, linguistic semantics, natural language processing, meta-search.

Introduction

The main part of information now is presented as web-documents and Internet/Intranet-distributed databases. However, the process of rapid accumulation of information on the Internet was rather chaotic, therefore a paradox occurred: the volume of accumulated information increases much faster than the ability to search within that information. Fast and efficient search for necessary information is a common problem of modern society.

When we use modern search engines, the first documents returned as the result of the query processing are often occupied by links to non-relevant (in the context of the query) documents, although their relevance is rated as 100 per cent by search engines.

There are three reasons for that:

- search engines use algorithms based on various statistical scores of document texts, rather than on “understanding” the meaning of queries and the contents of informational resources;
- natural language query is usually ambiguous and it depends on the situation context;
- the user is not always able to formulate his/her query exactly, and search engines are of no help in this task.

The problem of relevance is usually solved by means of keyword-based linear search with the help of statistical and some kinds of linguistic methods. Some systems proclaim the possibility of semantic search, of natural-language query input, of

question-answering. However, they use inadequate linguistic and software tools. The output results of such systems are large arrays of documents, but only a lesser portion of documents in the array is relevant.

Improvement of search precision is possible if the following tools are employed:

- tools for natural language processing;
- linguistic knowledge base;
- tools for applied semantic analysis, comparison and ranking. [1][2][3][4].

So the principal motivations we have in mind while developing an intelligent information search engine are:

- to extend standard keyword search engine mechanisms;
- to provide support for search queries in natural language;
- to improve search precision;
- to preserve the speed of standard search methods.

1. Linguistic processing

Most of the existing search methods employ search by key words, they do not take into account the possibility of semantic search containing language processing tools and being an alternative type of search.

Search by key words often does not satisfy the main requirement of the user, namely the requirement of semantic relevance of the found documents to the query, even in spite of the fact that all key words of the query are present in these found documents. However a document can be semantically relevant to the query even if it does not contain any key words of this query. For example, if we have a query "speech of the President of the Russian Federation", then the relevant documents might contain the following variants: speech of the President of the Russian Federation, report of the President of the Russian Federation, speech of the leader of the Russian Federation, report of the President of the Russian Federation, speech of the President of Russia, report of the leader of Russia, etc.

One more example. Query: "Visit of the president to Brussels". The same idea can be expressed differently, say "The president went to Brussels." To process the query we introduce a semantic relation DIR, which denotes the fact that Y is a destination of X. Then we search for DIR(president, Brussels).

Of course natural language copes with the task of increase of the search engine work relevance far better than search by key words. That is why queries in the form of natural language discourse should be allowed. From this follows the necessity of semantic analysis of the query text itself and of the required documents.

A disadvantage of the traditional search by key words is that using a simple set of key words it is impossible to convey the semantic direction of the query, impossible to "focus" it. In other words it is impossible to formulate the search need of the user. A query is not just a set of words. It is a sentence/phrase in which words are not just stuck together, but are connected with each other according to certain syntactic rules.

When forming a discourse, syntax deals above all with meaningful units bearing not only their individual lexical meaning, but also generalized categorical meaning in constructions of various complexity. These units are called syntaxemes (minimal indivisible semantic-syntactic structures of language [5] [6]. Syntaxemes are detected taking into account: a) categorical semantics of the word; b) morphological form; c) function in the sentence (according to the constructive capabilities within a sentence there are 3 functional types of syntaxemes: free/conditional/bound).

Categorical semantics is a generalized meaning characterizing words that belong to the same categorical class (for instance, to the class of people, things, attributes for nouns).

Let us consider some examples of different syntaxemes that are similar in form, but different in meaning:

1. The mother brings her son ***to school***.
2. Laziness brings researchers ***to trouble***.

In (1) “***to school***” is a spatial noun with a directive meaning (direction of movement), it is a free syntaxeme, since its meaning does not depend on the position in a sentence. In (2) “***to trouble***” is an attributive noun meaning logical consequence. It is a conditional (stipulated by a syntactic construction) syntaxeme, since its meaning is realized only in a certain complicative model in the position of a semi-predicative complicant of the model.

In a particular discourse, in a particular sentence of a query a word performs as a syntaxeme, i.e. has a certain syntactical function, in a certain grammatical form, it realizes only one of the possible meanings which this word can take in this sentence/phrase. The main task of the semantic analysis is to reveal semantic relations between syntaxemes.

Semantic analysis essentially enhances the precision and recall of the search and decreases the number of irrelevant documents returned in the result of the search.

Following this general overview let's now consider the linguistic analysis process in our system in more detail.

The main idea of semantic search is semantic processing of user query texts and of returned documents. Semantic processing involves generation of semantic images of documents and match of the obtained images. In the result the additional types of relevance are calculated that allow to remove the documents obviously not corresponding to the search query semantics.

Semantic processing of the discourse is made for three stages: morphological, syntactical and semantic analysis itself. [7], [8], [9]. Each stage is fulfilled by a separate analyzer with its input and output data and its own settings.

Interpretation of utterances implies the following milestones:

1. Morphological and syntactical analysis of the sentence/phrase.
2. Situational analysis of the sentence/phrase (building a subcategorization frame, correlation of the sentence/phrase situation with the situation described in the knowledge base of the system, selection of an adequate action).
3. Generation of the system answer.

1.1. Morphological analysis

At the morphological analysis stage words and separators (in case available) are recognized in the discourse. Based on the word morphology the list of all possible grammatical forms for each word are defined [10]. Word forms corresponding to the same normal dictionary form of the word and to the same part of speech and those in the same number (singular or plural) (for parts of speech that can change the number) will be unified into groups which will be further called lexemes (though they are not lexemes in the strict linguistic sense).

Obviously, several such lexemes can correspond to the same word. In order to decrease the number of the resulting variants of the sentence, the morphological analyzer has a filter: it can be defined for every part of speech whether it will be taken into account in the further analysis. The settings allow to ignore interjections and particles by default if there are variants of this word belonging to other parts of speech.

At the output we get a list of sentences each of which is a list of words. Each of the word in its turn is a list of lexemes.

1.2. Syntactical analysis

The main task of the syntactical analysis is to establish syntactical dependencies between lexemes revealed at the previous stage. In particular, syntactical analysis is made to detect the boundaries of minimal semantic-syntactic structures (syntaxemes).

The first two tasks can be solved within one sentence. Compound sentences are split into simple clauses which are further processed as separate sentences. A list of variants is composed for all sentences acquired at the output of morphological analysis so that in each sentence variant each word has only one lexeme. Since the number of sentence variants is equal to the product of the number of lexemes for each word, the task of limiting the number of variants becomes apparent. For this heuristics allowing to reject obviously incorrect variants are applied. Besides, a maximum allowable number of variants can be prescribed in the syntactical analyzer settings.

After that the algorithm for subordinating syntactical relations discovery is applied for each variant. In the result lexemes are unified into dependency trees: the lexeme at the parent node governs all child node lexemes. A minimal semantic-syntactic structure (syntaxeme) is a tree the root of which is a noun or a preposition that governs the noun. It should be noted that proper names also belong to nouns. A noun phrase (NP) is any syntaxeme subtree into which the root noun is included.

Besides searching for syntactical dependencies the syntactical analysis detects homogeneous parts. Thus at this stage two types of relations can be detected between lexemes: government and homogeneity. Every time when the program detects some syntactical relation, the weight of the sentence variant is increased. At the end of the sentence analysis only variants with maximal weight are kept. Sentences with zero weight are deleted by default (this option can be modified using the syntactical analyzer settings).

Thus the syntactical analysis input is a sentence acquired at the output of the morphological analysis. The output is a sentence in the form of a list of variants, each of which is a list of dependency trees.

1.3. Semantic analysis

The main task of the semantic analysis or, to be more exact, the kind of semantic analysis described below (which can become deeper in case of properly composed knowledge base of the domain), is to reveal semantic meanings of syntaxemes and relations on the set of syntaxemes. In general a semantic relation is understood as relation of concepts in the conceptual system of the domain. Representatives of semantic relations in lexis are predicate words, i.e. lexemes representing predicates. The main place here is occupied by verbs that have, as a rule, the central position in the semantic structure of the sentence and that decisively influence NPs and sentences. The information about syntactical compatibility of every verb is put to special tables of relations. The tables of relations for each verb indicate types of relations between its arguments.

Semantic search image consists of an ordered map of triples: <relation, arg1, arg2>, where <relation> stands for semantic relation type, and <args> are dependency trees for a corresponding NP or PP.

In those cases when we come across syntaxemes the meaning of which can be realized only in the given syntactic position, we can speak about bound syntaxemes. Tables of roles and semantic relations are compiled for such a type of syntaxemes.

Free syntaxemes are those capable of independent isolated usage and the meaning of which does not depend on their syntactic position. A special heuristic algorithm is developed for free syntaxemes, and for those syntaxemes that have interrelations with each other avoiding verbs. It should be noted that participles and adverbial participles as special verb forms, as well as deverbalives, can act as predicate words.

To do the semantic analysis, first of all it is necessary to extract predicate words (predicators). At the present stage predicators are only verbs or deverbalives. Then all types of syntaxemes surrounding the predicator are extracted. For this purpose the system has a dictionary of verbs (and deverbalives) containing, firstly, the verb, secondly, syntaxemes that can be used with this verb, thirdly, the information necessary for the syntaxemes identification: morphological form and categorical semantics of nouns. Note that the verb itself imposes constraints on possible readings of syntaxemes, for example, ***to give a book to the friend*** (a syntaxeme with the meaning of addressee) - ***to listen to the friend*** (a syntaxeme with the meaning of object). Besides, the algorithm of the linguistic analysis module work has a set of rules that allow to determine the meaning of a syntaxeme (for example for those cases when two or more nouns within a sentence have the same morphological forms and categorical semantics as well as for structural semantic modifications of the sentence).

Besides, the dictionary indicates how NPs are interrelated within role structures. A set of binary relations in the set of roles is also specific for each type of predicate words and is defined a priori.

The total number of NPs, roles and binary relations is presented in the form of a semantic graph describing the situation in the neighborhood of one predicator. This graph is a fragment of a semantic network describing the semantic image of the whole text.

Every time when a syntaxeme fills up the predicator role or when two roles correspond to a semantic relation, the program increases the weight of the sentence variant. Hence in case of simultaneous syntactical and semantic analysis the “heaviest” variants from the point of view of both syntactical and semantic relations are kept. That is why simultaneous analysis is not equal to sequential analysis: in the latter case

variants with the greatest number of syntactical dependencies are first selected and then those ones among them are chosen in which the role structure of predicates is filled up in the best way and more semantic relations are found. If a verb is polysemantic (i.e. there are several entries for one word in the dictionary), then all variants one by one are considered. Those variant(s) are finally selected on the further stages of analysis, where syntaxemes meanings for a greater number of NPs of the fragment are found, and where categorical semantics attributes worked most frequently. If there are a few equivalent variants again, then the variant with the maximum ratio of the number of syntaxemes found in the sentence to the total number of syntaxemes described in the given dictionary entry should be chosen (i.e. the variant with the best (complete) verb role structure filling).

Participial phrases and adverbial participial phrases are processed after the corresponding main clauses. The subject of the main clause becomes the subject of an adverbial participial phrase. Candidates for the subject/object of a participial phrase are the nearest NPs, roots of which agree with the participle in the gender, number and case.

Syntaxemes are also searched for interrogative words like *who*, *what*, *where*, *why*, *when*, *how*, *how much*, *what for*, *at what time*. A special attribute is given to the found meanings (roles in the pairs <role, NP> representing semantics of the query and the document) and then to relations in triples <relation, NP1, NP2>. When comparing the query with the document (during relevance calculation), this attribute will allow to consider NP, NP1, NP2 of the query that coincide with any NPs of the document corresponding to the given interrogative words in categorical semantics.

At present the system works with Russian language, and its English version is under development. One of the further directions of work is to expand the system for handling queries in other languages. Semantic roles and relations are universal, so the main concepts of the system can be preserved.

1.4. Linguistic knowledge base

The linguistic knowledge base in our system consists of two components:

- thesaurus containing a set of words and NPs associated with each other, like synonymy, partonymy, semantic likelihood, etc.
- predicate dictionary. This dictionary describes the ways of syntactic context realization of different types of semantic relations between concepts. The list of predicates, say, for Russian language contains verbs and various verb derivative forms. The predicate dictionary also has the table of relations between the set of syntaxemes and the set of predicator roles.

The thesaurus is used for query pre-processing and the predicate dictionary is used for semantic analysis.

Let's consider the structure of the linguistic knowledge base on an example.

Verb= love

Role=subject – syntaxeme meaning

Facultative=No

Soul=Undefined

S=+case – syntaxeme expression form

Role=object
 Facultative>No
 Soul=Undefined
 $S=+$ case – syntaxeme expression form
 Role=causative
 Facultative>No
 Soul=Undefined
 $S=$ case – syntaxeme expression form
 Relation=subject object PTN – semantic relation
 Relation=subject causative CAUS

2. Meta-search engine software architecture

The described methods are implemented in the meta-search engine. The component model architecture of the engine is:

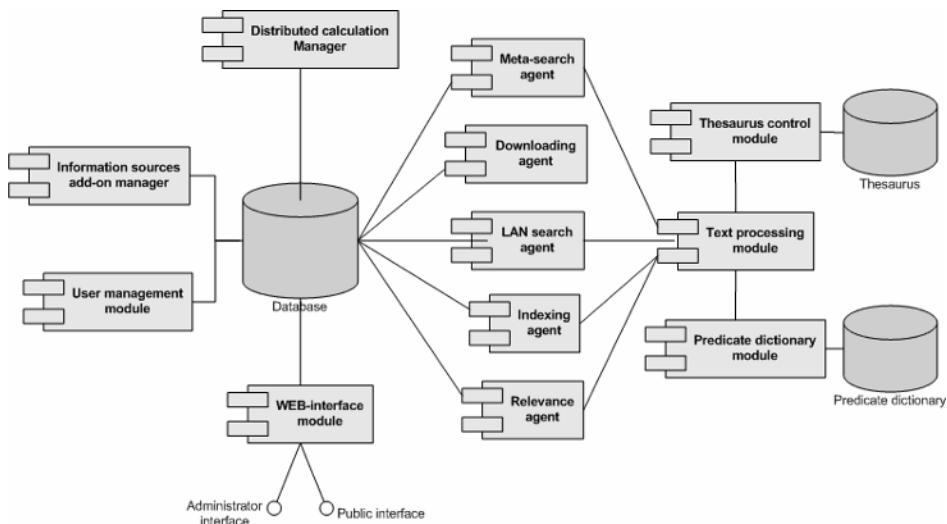


Figure 1. Meta-search engine component model diagram.

Meta-search engine consists of the following components:

1. Multi-agent environment provides distributed search queries processing by redirecting functionally independent tasks to agent modules located on different computers of the local network [7].
2. Database stores indexed documents and multi-engine environment support data.
3. Text processing module makes part-of-speech tagging and semantic analysis.
4. Agent modules make possible all search tasks solving (search query processing, documents downloading and indexing, relevance calculation).
5. Thesaurus for search query preprocessing.

6. Predicate dictionary stores linguistics knowledge.
7. Search engine modules support web-interface, users management and so on.

3. Semantic filtering

Semantic relevance improvements can be achieved using additional semantics filtering [2], [3], [4]. The dataflow diagram gives explanation of the main idea of additional document processing using linguistics analysis.

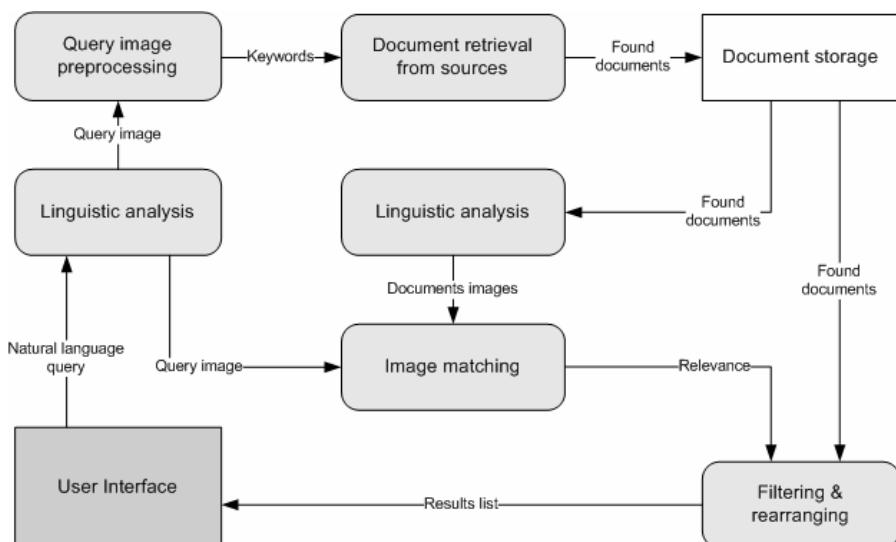


Figure 2. Semantic search process dataflow diagram.

The typical scenario of user search query processing includes the following steps:

1. Linguistic analysis of the query text generating its semantic search image.
2. Extraction of keywords and creation of an alternative search image for external search engines.
3. Redirection of keywords search image to currently added-on external search engines.
4. Getting and merging results lists from each search engine.
5. For each document in the joint results list:
 - 5.1. Extraction of excerpts of its text with matching keywords.
 - 5.2. Linguistic analysis of the excerpts and generation of a semantic search image.
 - 5.3. Match of the semantic search image of the document against that of the query (Calculating semantic relevance).
6. Rearrangement of the results list according to semantic ranking, removing false drops, i.e. documents ranked below some pre-specified threshold.

The overall process can thus be generally referred to as "semantic filtering" of the results list.

4. Conclusion and future work

We presented methods for knowledge-based semantic search elaborated within an intelligent system for semantic search and analysis of information on heterogeneous informational resources and services (Internet/Intranet networks, local/distributed databases), along with solution of the problem of semantically relevant search being the central concept of the system.

We are developing new algorithms for recognition and analysis of various models of sentences, parametric algorithms of relevance calculation, algorithms for machine learning adapted for work with text information and enrichment of the system predicate dictionary.

One more subject of thorough further research is methods of implicit relations revelation as well as anaphora resolution in case of multiple variants requiring semantic filtering of syntaxemes. It will be necessary for selection of the best variant of resulting interpretation using preference criteria (individual criteria for every type of phenomena is needed). The preference criteria that have weight coefficients may include: categorical semantics and morphological forms of candidates, results of syntaxemes search, word order and other factors.

At present we are also developing the English version of the system and considering the possibility of multilingual search and translation within the system. The Russian prototype of the semantic meta-search engine is available at www.exactus.ru.

References

- [1] A. Klimovski, G. Osipov, I. Smirnov, I. Tikhomirov, O. Vybornova, O. Zavjalova. Usability evaluation methods for search engines. // to appear in Proceedings of international conference IAI'2006, Kiev, 2006. (in Russian)
- [2] A. Klimovski, I. Kuznetsov, G. Osipov, I. Smirnov, I. Tikhomirov, O. Zavjalova. Problems of providing search precision and recall: Solutions in the intelligent meta-search engine Sirius // Proceedings of international conference Dialog'2005, Moscow, Nauka, 2005. (in Russian)
- [3] G. Osipov, I. Smirnov, I. Tikhomirov, O. Zavjalova. Intelligent semantic search employing meta-search tools. // Proceedings of international conference IAI'2005, Kiev, 2005. (in Russian)
- [4] I. Kuznetsov, G. Osipov, I. Tikhomirov, I. Smirnov, O. Zavjalova. Sirius - engine for intelligent semantic search in local and global area networks and databases. // 9th National conference on artificial intelligence CAI-2004, b.3. Moscow: Fizmatl, 2004. (in Russian).
- [5] O. Zavjalova. About principles of verb dictionary generation for the tasks of automatic text processing. // Proceedings of international conference Dialog'2004, Moscow, Nauka, 2004. (in Russian)
- [6] G. Zolotova, N. Onipenko, M. Sidorova. Communicative grammar of Russian language. Moscow, 2004. (in Russian)
- [7] G. Osipov. Interactive synthesis of knowledge-based configuration. //Proceedings of 5 Second Joint Knowledge Base Conference on Software Engineering. Sozopol, Bulgaria, 1996.
- [8] G. Osipov. Semantic types of natural language statements. A method of representation. //10th IEEE International Symposium on Intelligent Control 1995, Monterey, California, USA, Aug. 1995.
- [9] G. Osipov. Methods for extracting semantic types of natural language statements from texts. //10th IEEE International Symposium on Intelligent Control 1995, Monterey, California, USA, Aug. 1995.
- [10] Morphology modules from <http://www.aot.ru/download.shtml>

Reasoning by structural analogy in intelligent decision support systems

A.P. EREMEEV¹, P.R. VARSHAVSKY²

¹*Moscow Power Engineering Institute, Krasnokazarmennaya 14, 111250, Moscow, Russia, e-mail: eremeev@apmsun.mpei.ac.ru*

²*Moscow Power Engineering Institute, Krasnokazarmennaya 14, 111250, Moscow, Russia, e-mail: VarshavskyPR@mpei.ru*

Abstract. The problem of modeling human reasoning (common sense reasoning) and, in particular, reasoning by analogy in intelligent decision support systems is considered. Special attention is drawn to modeling reasoning by structural analogy taking the context into account. This work was supported by RFBR.

Keywords. Analogy, common sense reasoning, reasoning by analogy, intelligent decision support system

Introduction

Investigation of mechanisms that are involved in the analogous reasoning process is an important problem for experts in artificial intelligence (AI). The great interest in this problem is caused by the necessity of modeling human reasoning (common sense reasoning) in AI systems and, in particular, in intelligent decision support systems (IDSS) [1].

The analogy used for solving various problems [2, 3], e.g., for solving problems of automated theorem proving, for generation of hypotheses about an unknown subject domain, for generalizing the experience in the form of an abstract scheme, etc.

Notwithstanding the fact that the method of analogy is intuitively clear to everyone and is actively used by humans in everyday life, the notion of analogy does not allow for complete formal definition and, hence one cannot uniquely describe the mechanism of reasoning by analogy.

At the present time, there are a great number of various models, schemes, and methods that describe mechanisms of reasoning by analogy [3].

Unfortunately, there are not such modern systems of reasoning by analogy, which are oriented to use in IDSS and, in particular, in real-time IDSS (RT IDSS).

In this paper, we consider approaches and methods of reasoning by structural analogy, which are oriented towards use in IDSS. The use of the respective methods in IDSS broadens the possibilities of IDSS and increases the efficiency of making decisions in various problem (abnormal) situations.

Special attention in this paper will be paid to the most efficient inference method on the basis of structural analogy that takes into account the context.

We use semantic networks (SNs) as a model of the knowledge representation.

1. Knowledge representation in the form of a semantic network

The choice of an SN for knowledge representation possesses an important advantage, which distinguishes it from other models, such as natural representation of structural information and fairly simple updating in a relatively homogenous environment. The latter property is very important for RT IDSSs oriented towards open and dynamical problem domains.

A semantic network is a graph $\langle V, E \rangle$ with labelled vertices and arcs, where V and E are sets of vertices and arcs, respectively. The vertices can represent objects (concepts, events, actions, etc.) of the problem domain, and the arcs represent the relation between them.

We consider the structure of the SN in more detail using an example from power engineering - operation control of the nuclear power unit (see figure 1).

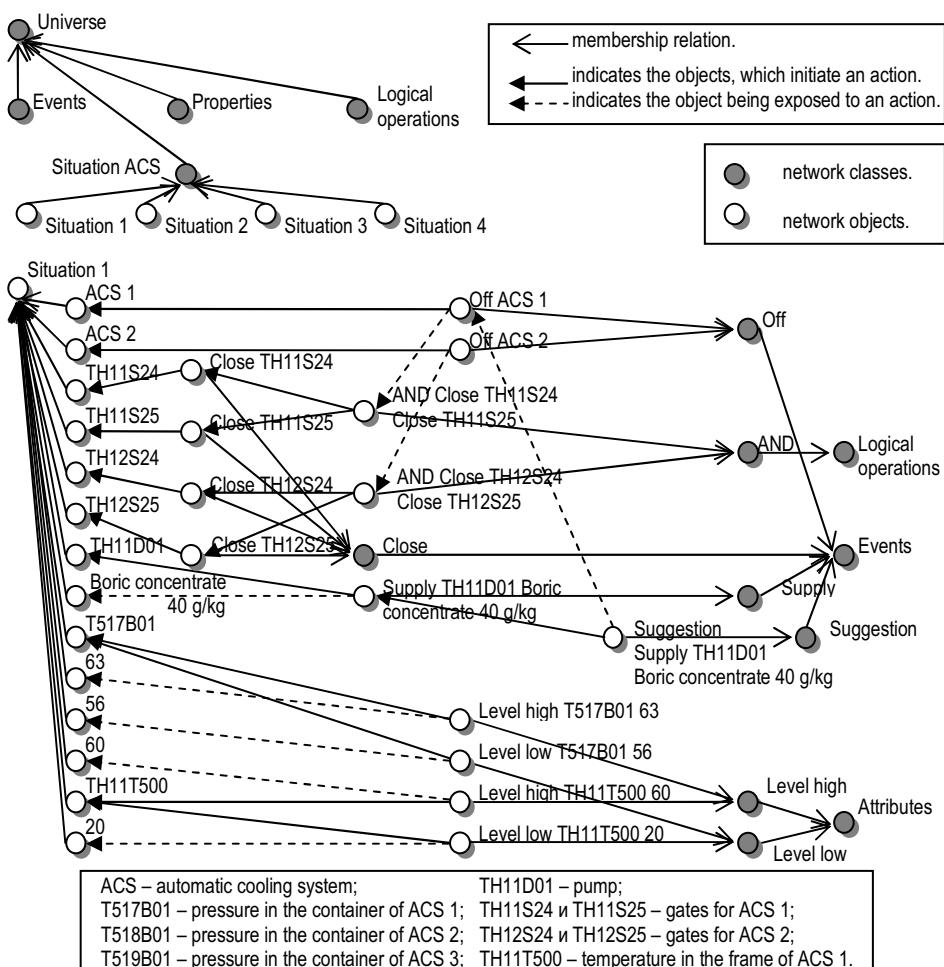


Figure 1. A fragment of the semantic network that represents the metalevel and the Situation 1 that was formed in the course of ACS functioning

We give a semantic interpretation of the information given in the SN for Situation 1 represented in fig. 1:

- It is recommended to supply the pump TH11D01 with boric concentrate 40g/kg caused by switching off automatic cooling system ACS 1 due to closing the gates TH11S24 and TH11S25;
- The upper setting T517B01 is equal to 63;
- The lower setting T517B01 is equal to 56;
- The upper setting TH11T500 is equal to 60;
- The lower setting TH11T500 is equal to 20.

2. Reasoning by structural analogy taking into account the context

In [4] it was proposed to consider an analogy as a quadruple $A = \langle O, C, R, P \rangle$, where O and R are the source object and the receiver object and C is the intersection object, i.e., the object that structurally is intersected with the object source and object receiver, and has a larger cardinality of the set of properties in the comparison with these objects. In other words, the analogy between the source object and receiver object is considered in the context of the intersection C , and P is the property for definition of the original context. The structure of this analogy is represented in fig. 2.

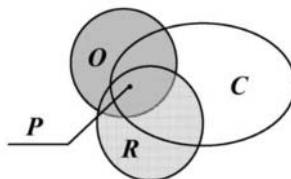


Figure 2. Structure of analogy using the context

Using the described structure of the analogy, the authors of [4] propose the algorithm for the problem solution that is based on an analogy of the properties. An SN with information about the problem domain, a receiver R , and the property for defining the original context P provide input data for this algorithm.

The algorithm for the problem solution on the basis of an analogy taking into account the context consists of the following stages.

Stage 1. Determine all objects of the SN, except for receiver R , that have property P . If there are no objects of this kind, then the search for a solution fails (without finding an analogy), otherwise, go to stage 2.

Stage 2. For the objects found in stage 1, determine all possible intersections of C with R taking into account P . If there are no intersections of C with R , the first search for a solution fails, otherwise, go to stage 3.

Stage 3. From the objects extracted in stage 1, determine all possible sources O for analogies with the receiver and the intersection taking into account P . In the case of success (possible analogies for R are defined), go to stage 4, otherwise, the search for a solution fails.

Stage 4. From the analogies extracted in stage 3, choose the most appropriate ones (taking into account the requirements of the decision making person (DMP)). In the case of success, go to stage 5; otherwise, the search for a solution fails.

Stage 5. The analogies obtained in stage 4 (which could be previously compared with each other taking into account the context) are given to the DMP, which means successful termination of the algorithm.

Having obtained analogies, the DMP may then make the final choice of the best ones. On the basis of these facts, the facts (properties) that hold for the source O are transferred to the receiver R .

Consider a modified algorithm for a problem solution that uses the structural analogy based on the modified structure of an analogy and the algorithm for the search of minimal intersections [3, 5]. The modification consists in the fact that P is considered not as a unique property, but as a set of properties that determine the original context of the analogy.

As compared with the base variant, one of the main advantages of this modified algorithm is the possibility of implementing the search for a solution on the basis of an analogy without refining the original context, since in the result of the search for the minimal intersection, one can easily distinguish all possible contexts for the analogy. Another important advantage of the modified algorithm is the possibility of a more detailed refinement of the original context for the determination of analogies. Moreover, in the modified algorithm there is a possibility to construct an analogy taking into account the context between well-known objects, the source and the receiver.

3. Conclusion

The method of reasoning by analogy on the basis of a structural analogy was considered from the aspect of its application in modern IDSS, in particular, for a solution of problems of real-time diagnostics and forecasting. The example of an algorithm for the solution search on the basis of an analogy of properties that takes into account the context was proposed. The more efficient algorithm, in the sense of the solution quality, is proposed. It uses a modified structure of an analogy that is capable of taking into account not one property (as in the base algorithm), but a set of properties. These properties determine the original context of the analogy and transfer from the source to the receiver only those facts that are relevant in the context of the constructed analogy.

The presented method was applied at implementation of a prototype of a RT IDSS on the basis of nonclassical logics for monitoring and control of complex objects like power units.

References

- [1] Vagin V.N., Eremeev A.P. Certain Basic Principles of Designing Real-Time Intelligent Decision Systems // Journal of Computer and Systems Sciences International, v. 40(6), 2001.
- [2] Pospelov D.A. Reasoning modeling. -M.: Radio and communication, 1989. (in Russian).
- [3] Varshavskii P.R., Eremeev A.P. Analogy-Based Search for Solutions in Intelligent Systems of Decision Support // Journal of Computer and Systems Sciences International, v. 44(1), 2005, p. 90–101.
- [4] Long D., Gariglano R. Reasoning by analogy and causality: a model and application // Ellis Horwood Series in Artificial Intelligence, 1994.
- [5] Eremeev A., Varshavsky P. Analogous Reasoning for Intelligent Decision Support Systems // Proceedings of the XI-th International Conference “Knowledge-Dialogue-Solution” – Varna, v.1, 2005, p. 272-279.

The heuristic methods of obtaining the effective measurement in diagnostic systems

V.N. VAGIN¹, P.V. OSKIN²

¹*Moscow Power Engineering Institute, Krasnokazarmennaya 14, 111250, Moscow, Russia, e-mail: vagin@apmsun.mpei.ac.ru*

²*Moscow Power Engineering Institute, Krasnokazarmennaya 14, 111250, Moscow, Russia, e-mail: ospavel@newmail.ru*

Abstract. The problem of finding the best current measurement point in model-based device diagnostics with assumption-based truth maintenance systems is viewed. The new heuristic approaches of current measurement point choosing on the basis of supporting and inconsistent environments are presented.

Keywords. Model-based diagnosis, measuring point choosing, assumption-based truth maintenance system

Introduction

The diagnostic systems are one of the most actively used systems in technical areas: electronics engineering, motor industry, robotics, space vehicles, thermal and atomic power stations and many others. Many diagnostics problems require building the behaviour prognoses, the work with contradictions and defaults, effective treatment of new facts and assumptions.

The typical problem of diagnostics is to find a fault of a diagnosed device on the base of some set of observations.

1. Model-based Diagnosis

The generalized problem of diagnostic can be formulated as follows. There is a device exhibiting an incorrect behaviour. The device will consist of components, one or several of which are not working properly what is the reason of incorrect behaviour. There is a structure of connections between components and a possibility to get measurements on their inputs and outputs. It is necessary to determine what of components are faulty with minimal resource expenses.

At present two main approaches to a solution of the given problem are viewed [1-3].

The first approach is heuristic diagnostics [1]. The base of this approach is the knowledge extraction from an expert and building fault determining rules in the form of “symptoms→faults”.

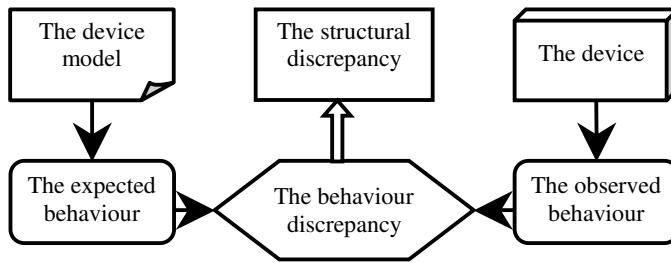


Figure 1.

Drawbacks of the given approach are:

- a complexity of multiple fault diagnostics;
- the problems in unexpected faults recognition;
- a complexity of expert knowledge formalizing;
- difficulties using the knowledge bases for other diagnostic problems;
- a rigid dependence from a device structure;
- a difficulty of explanations.

The second approach is model-based diagnostics [2, 3]. This approach is based on the knowledge of device component functionality.

The model of a device is a description of its physical structure, plus the models for each of its components. A compound component is a generalized notion including simple components, processes and even logical inference stages.

Model-based diagnosis process is the comparison of predicted device behaviour with its observed behaviour (see figure 1).

It is supposed, that the model is correct, and all differences between device behaviour and a device model indicate availability of broken components.

Main advantages of the model-based approach:

- diagnosing the multiple faults;
- unexpected fault recognition;
- a precision of a component model description does not depend on the expert experience;
- a possibility of new device diagnosing;
- multiple using the models;
- detailed explanations.

2. Assumption-based Truth Maintenance Systems

For building a prognosis network, a component behaviour model, finding minimal conflicts characterizing uncorrespondence of observations with prognoses and minimal candidates for a faulty, it is profitable to use possibilities given by assumption-based truth maintenance systems (ATMS) [4,5].

The truth maintenance systems (TMS) are the systems dealing with the support of a coherence in databases. They save the assertions transmitted to them by a problem solver and are responsible for maintaining their consistency. Each assertion has the

justification describing what kind of premises and assumptions this justification was obtained. The environment is a set of assumption.

The inference of an inconsistency characterizes assumption incompatibility within the presuppositions of which this conclusion was made. Also there is introduced the environment set which contains some inconsistency [4]. The sets of inconsistency environments E_1, E_2, \dots, E_m are Nogood= $\{E_1, E_2, \dots, E_m\}$. A consistent ATMS environment is not Nogood.

There are the following correspondences between ATMS and the model-based diagnosis approach:

- ATMS premises – an observed device behaviour;
- ATMS assumptions – components of a device;
- inferred ATMS nodes – predictions of an diagnostic system;
- Nogood - the difference between predicted and observed device behaviour.

3. The Current Measurement Point Determination

One of the key aspects of the model-based fault search algorithm is to determine the optimal current measurement in a diagnosed device [2]. Efficiency of the current measurement choosing allows essentially reducing a decision search space while the inefficiency of choice will increase an operating time, the space of a searching algorithm, and also require additional resource spends to implement a measurement.

The best measurement point in a diagnosed device is a place (point) of measuring a value giving the largest information promoting the detection of a set of fault components at minimal resource spending.

One of the best procedures for reducing resource expenses is to produce the measuring giving the maximal information concerning predictions made on the basis of the current information on a system.

3.1. Heuristic Methods of Choosing a Measurement Point

The purpose of the best choosing a measurement point is to derive the maximal component state information. After each measuring there is a confirmation or refutation of prediction values in a point of measurement. So, it is possible to use the following aspects:

- Knowledge about environments that support predicted values in the measurement points which can be confirmed or refuted.
- Knowledge about inconsistent environments.
- Knowledge about coincided assumptions of the inconsistent environments.

3.2. Knowledge About Supporting Environments

The diagnostic procedure constructs predictions of values for each device point with the list of environments in which the given prediction is held. The list of environments represents assumption sets about correctness of corresponding device components.

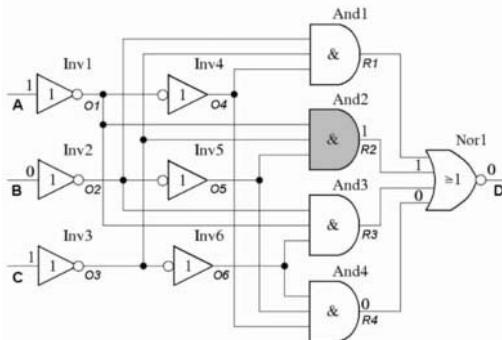


Figure 2.

Let's consider a sample of the 3-bit parity checker (see figure 2) [6].

The vector $(1, 0, 1)$ is input to device components (Inv1, Inv2, Inv3). Let the following measurements be obtained: $D=0$, $R4=0$, $R2=1$. The mismatch between observations and predictions speaks about a fault in a device. Based on measured observations additional predictions of values are formed. In general it is obtained the following set of predictions with appropriate environments:

```

<R3=0, {{And3,Inv1}, {And2,And3,Inv6}, {And2,And3,Inv5}}>;
<R1=0, {{And1,Inv3}, {And1,And2,Inv4}, {And1,And2,Inv5}}>;
<O6=0, {{And2,Inv6}}>;
<O6=1, {{Inv3,Inv6}}>;
<O5=1, {{And2}}>;
<O5=0, {{Inv2,Inv5}, {And4,Inv1,Inv3,Inv4,Inv6}}>;
<O4=0, {{And2,Inv4}}>;
<O4=1, {{Inv1,Inv4}}>;
<O3=1, {{And2}}>;
<O3=0, {{Inv3}}>;
<O2=0, {{And2,Inv5}}>;
<O2=1, {{Inv2}, {And4,Inv1,Inv3,Inv4,Inv5,Inv6}}>;
<O1=1, {{And2}}>;
<O1=0, {{Inv1}}>.

```

As we are interested with a measurement point with the greatest information on failure the point is selected from a quantity of assumptions.

Designate an environment set as $Env(x)$. So, for the considered example, $Env(OI)=\{\{And2\}, \{Inv1\}\}$. Let's introduce the function $Quan(x)$, by which we will designate the information quantity obtained at measuring values in the point x .

If the environment J represents a unique assumption, then obviously the set cardinality will be equal 1: $|J| = 1$. The information quantity obtained from such environment is equal to 1. If the environment consists more than one component the information quantity obtained at confirming or refuting a value is less because we have knowledge not about a concrete valid / fault component but about a component set among of which are faulty. Therefore the information quantity obtained from a environment consisting of more than one assumption, we heuristically accept equal to half of set cardinality. Thus the function $Quan(x)$ is:

$$Quan(x) = \sum_{\substack{J_i \in Env(x) \\ |J_i|=1}} |J_i| + \sum_{\substack{J_j \in Env(x) \\ |J_j|>1}} \frac{|J_j|}{2} \quad (1)$$

Summing is produced on all possible values in the point x .

Then for the presented list of considered example predictions there are:

$$Quan(R3) = (2/2 + 3/2 + 3/2) = 4$$

$$Quan(R1) = (2/2 + 3/2 + 3/2) = 4$$

$$Quan(O6) = (2/2) + (2/2) = 2$$

$$Quan(O5) = (1) + (2/2 + 5/2) = 4.5$$

$$Quan(O4) = (2/2) + (2/2) = 2$$

$$Quan(O3) = (1) + (1) = 2$$

$$Quan(O2) = (2/2) + (1 + 6/2) = 5$$

$$Quan(O1) = (1) + (1) = 2$$

Points with the greatest value of the function $Quan(x)$ have the greatest priority of a choice. We will call the given method of choosing a measurement point as SEH (Supporting Environment Heuristics).

In our example the point giving the greatest information quantity is O2.

3.3. Knowledge about the Sets of Inconsistent Environment

As a result of each measurement there is a confirmation or refutation of some prediction. The environments E_1, E_2, \dots, E_m of refuted prediction form the set Nogood = $\{E_1, E_2, \dots, E_m\}$. It can be used for directional searching for more precise definition what kind of components from Nogood is broken.

Let we have the set Nogood = $\{\{And2, Inv1\}, \{And2, Inv2, Inv5\}, \{And2, Inv3\}\}$.

Obviously the more of the components from Nogood are specified by measuring a value in some device point the more the information about which components of Nogood are broken will be obtained. For using this possibility, it is necessary to take the intersection of each environment from $Env(x)$ with each set from Nogood:

$$Env(x) \cap Nogood = \{A \cap B : A \in Env(x), B \in Nogood\}$$

Continuing the example at fig. 2, we obtain the following:

$$<R3, \{\{Inv1\}, \{And2\}, \{And2, Inv5\}\}>$$

$$<R1, \{\{Inv3\}, \{And2\}, \{And2, Inv5\}\}>$$

$$<O6, \{\{And2\}, \{Inv3\}\}>$$

$$<O5, \{\{And2\}, \{Inv2, Inv5\}, \{Inv1\}, \{Inv3\}\}>$$

$$<O4, \{\{And2\}, \{Inv1\}\}>$$

$$<O3, \{\{And2\}, \{Inv3\}\}>$$

$$<O2, \{\{And2\}, \{And2, Inv5\}, \{Inv2\}, \{Inv1\}, \{Inv5\}, \{Inv3\}\}>$$

$$<O1, \{\{And2\}, \{Inv1\}\}>$$

For this approach the equation (1) can be changed as follows:

$$QuanN(x) = \sum_{\substack{J_i \in Env(x) \cap Nogood \\ |J_i|=1}} |J_i| + \sum_{\substack{J_j \in Env(x) \cap Nogood \\ |J_j|>1}} \frac{|J_j|}{2} \quad (2)$$

For each of these points we calculate the information quantity as follows:

$$QuanN(R3) = 1 + 1 + 2/2 = 3$$

$$\begin{aligned}
 \text{QuanN(R1)} &= 1 + 1 + 2/2 = 3 \\
 \text{QuanN(O6)} &= 1 + 1 = 2 \\
 \text{QuanN(O5)} &= 1 + 2/2 + 1 + 1 = 4 \\
 \text{QuanN(O4)} &= 1 + 1 = 2 \\
 \text{QuanN(O3)} &= 1 + 1 = 2 \\
 \text{QuanN(O2)} &= 1 + 2/2 + 1 + 1 + 1 + 1 = 6 \\
 \text{QuanN(O1)} &= 1 + 1 = 2
 \end{aligned}$$

Points with the greatest value of function $\text{QuanN}(x)$ have the greatest priority of a choice. We will call the given method of choosing a measuring point as SIEH (Supporting and Inconsistent Environment Heuristics).

One can see that the point O2 as the most informative is again offered. And in the given approach the difference in information quantity between various points is expressed more brightly than without Nogood usage.

3.4. Knowledge about Coincided Assumptions of the Inconsistent Environments

During diagnostics of faulty devices as a result of confirmations and refutations of some predictions there is a modification of a set of inconsistent environments Nogood.

In each component set from Nogood one or more components are broken what was a reason of including a supporting set into the inconsistent environments Nogood. Taking the intersection of all sets of the inconsistent environments, we receive a set of components which enter into each of them, so their fault can be a reason explaining an inconsistency of each set holding in Nogood. Thus, we obtain the list of components a state of which is recommended to test first of all, i.e. the most probable candidates on faultiness.

The set intersection of inconsistent environments is expressed by the following equation:

$$\text{SingleNogood} = \bigcap_{E_i \in \text{Nogood}} E_i$$

For $\text{Nogood} = \{\{\text{And2}, \text{Inv1}\}, \{\text{And2}, \text{Inv2}, \text{Inv5}\}, \{\text{And2}, \text{Inv3}\}\}$ the set of the most probable candidates will be the following: $\text{SingleNogood} = \{\text{And2}\}$.

If $\text{SingleNogood} = \emptyset$, it means that there are some disconnected faults. In this case the given approach is inapplicable and it is necessary to define more precisely the further information by any other methods.

After obtaining a set $\text{SingleNogood} \neq \emptyset$, on the base of environments of value predictions in device points it is necessary to select those measurement points that allow to effectively test components to be faulted from SingleNogood .

For this purpose we will work with the sets obtained as a result of an intersection of each environment from $\text{Envs}(x)$ with SingleNogood :

$$\text{Envs}(x) \cap \text{SingleNogood} = \{J \cap \text{SingleNogood} : J \in \text{Envs}(x)\}.$$

The following versions are possible:

- $\exists J \in \text{Envs}(x) : J \equiv \text{SingleNogood}$. One of environments of the value prediction in the point x coincides with the set SingleNogood . The given version allows to test faulty components from the set SingleNogood most effectively so this measurement point x is selected with the most priority.

- b) $\exists J \in \text{Env}(x) : |J \cap \text{SingleNogood}| < |\text{SingleNogood}|$. The cardinality of *SingleNogood* is more than the cardinality of a set obtaining as result of an intersection *SingleNogood* with a set from *Env(x)*. We evaluate this version as $\max_{J \in \text{Env}(x)} |J \cap \text{SingleNogood}|$, i.e. the more of components from *SingleNogood* are intersected with any environment from *Env(x)*, the more priority of a choice of the given measurement point for the observation.
- c) $\exists J \in \text{Env}(x) : \text{SingleNogood} \subset J$. The *SingleNogood* includes in a set from *Env(x)*. We evaluate this version as $\min_{J \in \text{Env}(x)} (|J| - |\text{SingleNogood}|)$, i.e. the less a difference between *SingleNogood* and *Env(x)*, the more priority of a choice of the given measurement point for the current observation.
- d) $\forall J \in \text{Env}(x) : J \cap \text{SingleNogood} = \emptyset$, i.e. no one of the most probable faulty candidates does not include in environments *Env(x)* supporting predictions at the point *x*. We evaluate this version as the least priority choice, i.e. 0 in the numerical equivalent.

Also to the version D there are referred other methods of definition of current measurement point priorities which happen when *SingleNogood* = \emptyset . Thus in the estimations of a choice priority a numerical value returned as a result of call of other method is accepted. We call it by *ResultD(x)*.

At appearance of the greater priority choosing between versions B and C, heuristically we accept the version B as at this choice the refinement of faulty candidates is produced better.

Note for various supporting sets of the same *Env(x)*, the availability both the version B and the version C is also possible. In this case, as a resulting estimation for the given *Env(x)* the version B is also accepted.

Continuing the example at figure 2, we obtain the following:

$$\text{Nogood} = \{\{\text{And2}, \text{Inv1}\}, \{\text{And2}, \text{Inv2}, \text{Inv5}\}, \{\text{And2}, \text{Inv3}\}\}$$

$$\text{SingleNogood} = \{\text{And2}\}$$

$$< \text{R3, version C}, \min_{J \in \text{Env}(x)} (|J| - |\text{SingleNogood}|) = 2 >$$

$$< \text{R1, version C}, \min_{J \in \text{Env}(x)} (|J| - |\text{SingleNogood}|) = 2 >$$

$$< \text{O6, version C}, \min_{J \in \text{Env}(x)} (|J| - |\text{SingleNogood}|) = 1 >$$

$$< \text{O5, version A} >$$

$$< \text{O4, version C}, \min_{J \in \text{Env}(x)} (|J| - |\text{SingleNogood}|) = 1 >$$

$$< \text{O3, version A} >$$

$$< \text{O2, version C}, \min_{J \in \text{Env}(x)} (|J| - |\text{SingleNogood}|) = 1 >$$

$$< \text{O1, version A} >$$

Let's estimate the obtained results.

Designate by *maxD* the maximal numerical value among versions D for all assessed measurement points, and by *CompCount* a quantity of device components.

Accept in reviewing the following assessments:

1. $\max_{J \in \text{Env}(x)} |J \cap \text{SingleNogood}| < \text{CompCount}$. The quantity of components which are the intersection result is always less than the quantity of whole device components;

2. $\min_{J \in \text{Envs}(x)} (|J| - |\text{SingleNogood}|) < \text{CompCount}$. The quantity of components in the prediction environment is always less than the quantity of the device components.

Taking into account these assessments one can introduce a numerical assessment of the obtained results:

$$\text{QuanSNG}(x) = \begin{cases} 0, & \text{if } \forall J \in \text{Envs}(x) : J \cap \text{SingleNogood} = \emptyset \\ & \text{ResultD}(x), \\ & \text{if } \text{SingleNogood} = \emptyset \\ \text{maxD} + \text{CompCount} - \min_{J \in \text{Envs}(x)} (|J| - |\text{SingleNogood}|), & \text{if } \exists J \in \text{Envs}(x) : \text{SingleNogood} \subset J \\ \text{maxD} + \text{CompCount} + \max_{J \in \text{Envs}(x)} |J \cap \text{SingleNogood}|, & \text{if } \exists J \in \text{Envs}(x) : |J \cap \text{SingleNogood}| < |\text{SingleNogood}| \\ \text{maxD} + 2 * \text{CompCount}, & \text{if } \exists J \in \text{Envs}(x) : J \equiv \text{SingleNogood} \end{cases} \quad (3)$$

Accordingly, for the example in the figure 2:

$$\text{maxD} = 0;$$

$$\text{CompCount} = 11;$$

$$\text{QuanSNG(R3)} = 0 + 11 - 2 = 9$$

$$\text{QuanSNG(R1)} = 0 + 11 - 2 = 9$$

$$\text{QuanSNG(O6)} = 0 + 11 - 1 = 10$$

$$\text{QuanSNG(O5)} = 0 + 2 * 11 = 22$$

$$\text{QuanSNG(O4)} = 0 + 11 - 1 = 10$$

$$\text{QuanSNG(O3)} = 0 + 2 * 11 = 22$$

$$\text{QuanSNG(O2)} = 0 + 11 - 1 = 10$$

$$\text{QuanSNG(O1)} = 0 + 2 * 11 = 22$$

The points with the greatest value of function $\text{QuanSNG}(x)$ have the greatest priority of choice. We will call the given method as SCAIEH (Supporting and Coinciding Assumptions of Inconsistent Environment Heuristics).

One can see that the most preferable measurement points are O1, O3 and O5, one of environment assumption of which coincides with *SingleNogood*. It differs from guidelines of other choice methods, but at the same time for the given example the choice of any of these points allows to test the most probable faulty component And2.

The developed methods of heuristic choice of the best current measurement point are recommended to use for devices with a great quantity of components as quality of guidelines directly depends on the quantitative difference of environments.

4. Practical Results

Let's test the developed methods of the best measurement point choosing for the 9-bit parity checker [6] represented in figure 3.

For each experiment one of device components is supposed working incorrectly what is exhibited in a value on its output opposite predicted. A consequence of the

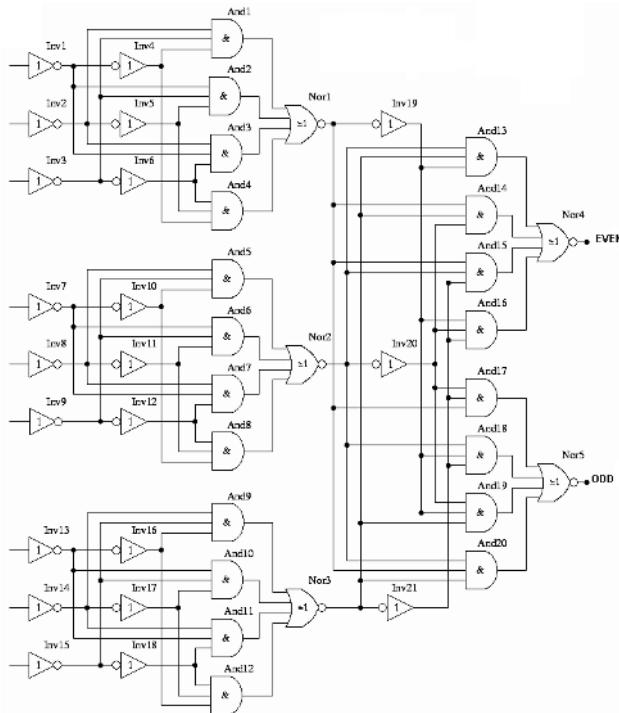


Figure 3.

incorrect component work is changing of outputs of those components which produce the results depending on values on the output of a faulty component. These changed results of component operations are transmitted to appropriate inquiries of a diagnostic system.

In the beginning of each experiment to inputs of components (Inv1, Inv2, Inv3, Inv7, Inv8, Inv9, Inv13, Inv14, Inv15) in a diagnostic complex the vector of values (1,0,1, 0,1,0, 1,0,1) enters. Then to the diagnostic system the value 0 retrieved from the output of the component Nor5 that depends on the work of a broken component and differs from predicted is transferred. It leads to the appearance of an inconsistency in the diagnostic system and starts the automatic process of testing.

In figure 4 the quantity of the stages required to each method for fault localization is shown. A method stage is a measurement point choosing. The smaller the quantity of method stages, the faster a fault is localized.

From the obtained results one can see that the method efficiency for different fault components is various and hardly depends on the device structure.

Let's estimate the method efficiency. The device consists of 46 components. The output values of 45 components are unknown (a value on the output of Nor5 is transmitted to the diagnostic system with input data together). So, the maximal stage quantity necessary for a fault definition is equal 45. Let's accept 45 stages as 100 %. For each experiment it is computed on how many percents each of the developed methods is more effective than exhaustive search of all values. Then define the average value of results. The evaluated results are represented in table 1.

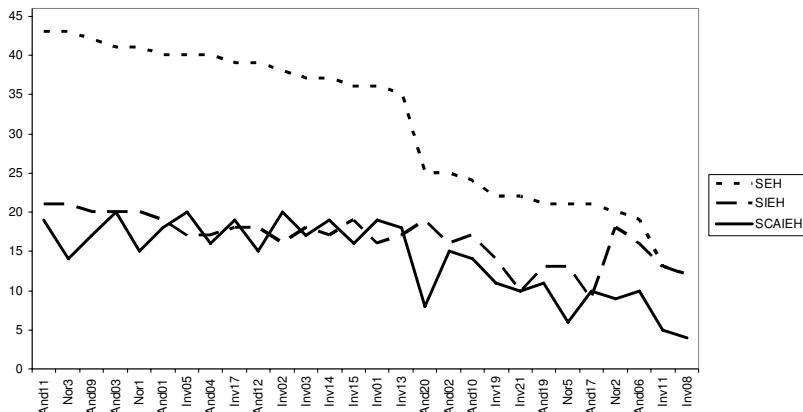


Figure 4.

Table 1.

The method	SEH	SIEH	SCAIEH
On how many percents the method is more effective, %	30,79	63,17	68,65

From table 1 one can see that the greatest efficiency of current measurement point choosing has the heuristic method based on the knowledge about coincided assumptions of the inconsistent environments SCAIEH.

5. Conclusion

In this paper the heuristic methods of finding the best current measurement point based on environments of device components work predictions are presented.

Practical experiments have confirmed the greatest efficiency of current measurement point choosing for the heuristic method based on the knowledge about coincided assumptions of the inconsistent environments SCAIEH.

Advantages of heuristic methods of the best current measurement point choosing is the simplicity of evaluations and lack of necessity to take into consideration the internal structure interconnections between components of the device.

References

- [1] Clancey W. Heuristic Classification. // Artificial Intelligence, 25(3), 1985, pp. 289-350.
- [2] de Kleer, J. and Williams, B.C. Diagnosing multiple faults. // Artificial Intelligence, 1987, v.32, pp. 97-130.
- [3] Forbus K.D., de Kleer, J. Building Problem Solver. // A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England, 1993.
- [4] de Kleer J. An Assumption-based TMS. // Artificial Intelligence, 1986, v.28, p.127-162.
- [5] Vagin V.N., Golovina E.Ju., Zagoryanskaya A.A., Fomina M.V. Exact and Plausible Inference in Intelligent Systems. // M.: Fizmatlit, 2004, - 704p. (in Russian).
- [6] Fröhlich P. DRUM-II Efficient Model-based Diagnosis of Technical Systems. //PhD thesis, University of Hannover, 1998.

An Advancement-Record System Using Knowledge Adaptable to Children's Developmental Stages

Kimio SHINTANI^a, Aki KONO^a, Masaya ASANO^b,
Hirohide HAGA^b and Shigeo KANEDA^b

^a*Early Childhood Education, Tokiwakai College*

^b*Graduate School of Engineering, Doshisha University*

Abstract. Recently, advancement-record systems for early childhood education have been commercially marketed in Japan. The academic advancement of children is recorded by using *knowledge* constructed from many *advancement-inspection items*. Such a record is required by law, and it's effective for third-party appraisal by the Japanese government. The knowledge conventionally used in such a system, however, has two major limitations. One is the subjective score selection. The advancement score is recorded in ambiguous expressions, such as "possible," "almost possible," and "impossible." The other limitation is that the items are fixed for each school year. If the highest score is selected at the starting point of one school year, the nursery teacher cannot read any developmental process from a constant item value for one year, even if the item is evaluated every month. To resolve these problems, this paper proposes an advancement-record system that uses a new type of knowledge. This knowledge has two key features: 1) Objective score selection based on Vygotsky's psychological theory of "The Zone of Proximal Development," and 2) Re-designed knowledge using items that can be chosen freely, according to the advancement stage of each child. The proposed system thus provides more detailed daily observation of child development as well as statistical analysis of advancement scores.

Keywords. ICT Application System, Advancement Record, Early Childhood Education, Nursery Schools, Nurture, Nursery Teachers, Kindergarten Schools

1. Introduction

Recently, advancement-record systems for early childhood education domains, such as nursery schools or kindergarten schools, have been commercially marketed in Japan [1][2][3]. Such record-making and compilation is required by law, and the recorded documents can be effective for third-party appraisal by the Ministry of Health, Labor and Welfare of the Japanese government.

This type of system uses *knowledge* for advancement inspection. The knowledge is constructed from many *advancement-inspection items*, each of which depends on the age of the child¹. This knowledge has five major categories: *Health*, *Human relations*,

¹ The age is divided into the following ranges: 1) from birth to less than six months, 2) from six months to less than twelve months, 3) from twelve months to less than 18 months, 4) from 18 months to less than 24 months, 5) two years old, 6) three years old, 7) four years old, and 8) five years old.

*Environment, Language, and Expression*², and each major category is divided into “sub-categories.” Finally, each sub-category has several advancement-inspection items. Conventional record systems, however, have two major limitations. One is the subjective score selection, because the advancement score is recorded in ambiguous expressions such as “possible,” “almost possible” and “impossible.” The other limitation is that the advancement-inspection items are fixed for one school year. If the highest score is selected at the starting point of one school year, nursery teachers cannot read any development process from a constant item value for one year, even if the item is evaluated every month.

To resolve these problems, this paper proposes an advancement-record system using a new type of knowledge. The proposed knowledge has two important features. One is its objective score selection based on Vygotsky’s psychological theory “ZPD: The Zone of Proximal Development.” The other is its re-designed knowledge with items that can be chosen freely, according to the advancement stage of each child. The proposed knowledge provides 1) more detailed observation of child development, 2) graphical output for easy observation, and 3) statistical analysis of the advancement scores.

In the following, Section 2 describes the details of conventional advancement knowledge and its problems. A new advancement-record system is proposed in Section 3. Section 4 gives the details of the graphical user interface. Finally, Section 5 concludes the paper.

2. Conventional Advancement Knowledge

Figure 1 shows a sample of conventional advancement *knowledge* used in evaluating a child’s developmental stage. The knowledge is large and written in Japanese. The knowledge consist of 334 items. Only a part of the knowledge is shown in Fig. 1. The knowledge has five *major categories*: Health, Human relations, Environment, Language and Expression, each of which is divided into *sub-categories*. The sub-category is constructed from many *advancement-inspection items*.

In Fig. 1, the major category is “Health,” which has the two sub-categories “Meal” and “Nap.” The sub-category Meal has 10 advancement-inspection items. Each advancement-inspection item has a *score*, assigned by nursery teachers. The score value is selected from cross, triangle, and circle, which correspond to “impossible,” “almost possible” and “possible,” respectively.

This type of conventional knowledge has the following major disadvantages.

1. Subjective score selection by nurturing persons: The advancement score is recorded in ambiguous expressions. Thus, two item values cannot be compared if different nursery teachers record them. Furthermore, it is not possible to compare item values if the nursery teacher records them at different times, even if the same nurturing person records the item values.
2. Items locked into age: The items are fixed for each school year. If the highest score is selected at the starting point of a school year, the nursery teacher cannot read any developmental process from the constant item value for that year, even if the item is evaluated every month.

² These major categories are selected in “Nursery School Nurture Guide” (in Japanese) [6].

3. No standardization: The sub-categories and items within the recorded advancement knowledge vary among nursery schools in Japan. Thus, if a child moves from one nursery school to another, the old record may have little or no meaning at the new nursery school.

Major-Category	Sub-Category	Evaluation Item	Recording Date		
			May-05	Oct-05	Feb-06
Health	Meal	Eats after greeting	△	△	
		Distinguishes his/her own food and the food of others.	○	○	
		Tries to eat alone, using spoon or fork.	△	△	
		Tries to eat at urging of nurturing person.	△	○	
		Bites and swallows very well.	×	×	
		Drinks without dropping while using a cup.	×	×	
		Does not play with the meal	△	△	
		Tries to eat with chopsticks	○	○	
	Nap	Wipes hands and face before and after lunch	×	△	
		Eat after removing packaging paper.	○	○	
	Sleeps alone	×	×	
		Sleeps during scheduled nap time.	×	△	
		Wakes up very well	○	○	
		Doesn't disturbance the other children after waking up.	×	×	
			
			

Figure 1. Example of Conventional Advancement Knowledge (*part of the Knowledge*)

A conventional record is written two or three times in one school year. The evaluation results are recorded in May, Oct. and Feb. as shown in Fig. 1. Thus the score can be input a few months after the child's actual advancement achievements. Furthermore, the nurturing persons are unable to refer to changes in scores for their daily guidance. Consequently, the conventional record becomes no more useful than as a mere "report for the government."

To resolve the above problems, the authors adopted three approaches: 1) Re-selected Items from authorized diagnostics methods that are widely used in Japan, 2) Items Not Locked into Age, 3) Objective Score based on Vygotsky's ZPD: The Zone of Proximal Development. The newly designed knowledge was implemented in a new advancement-record system, which is described in the following section.

3. New Advancement Record System

3.1 System Overview

The authors have developed a new advancement-record system. Figure 2 shows the system overview. The advancement information is shared among parents, a nursery teacher, a chief nursery teacher, a medical doctor, and a public-sector parenting support center under an access-control scheme. The advancement-inspection record should be a tool for daily nurturing. In order to achieve this goal, nursery teachers should evaluate the items *every month* in the proposed system. This interval is shorter than that of conventional systems.

Figure 3 shows a block diagram of the system, implemented by Xoops [9] and MySQL [10]. The system provides advancement-inspection interface screens for the nursery teachers. The nursery teachers can input and read the evaluation scores for each

child. The item score is merely a “number,” and the knowledge contains many inspection items. It is not easy to read the advancement stage of each child from raw data, and to solve this problem, the system offers two output features. One is a graphical-output sub-system and the other is a statistical-analysis sub-system, which employs the R system [11].

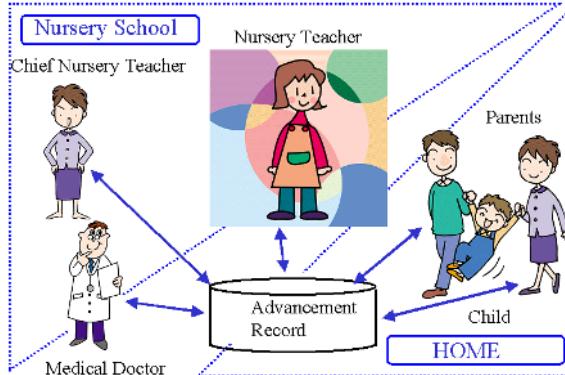


Figure 2. Proposed Advancement-record System

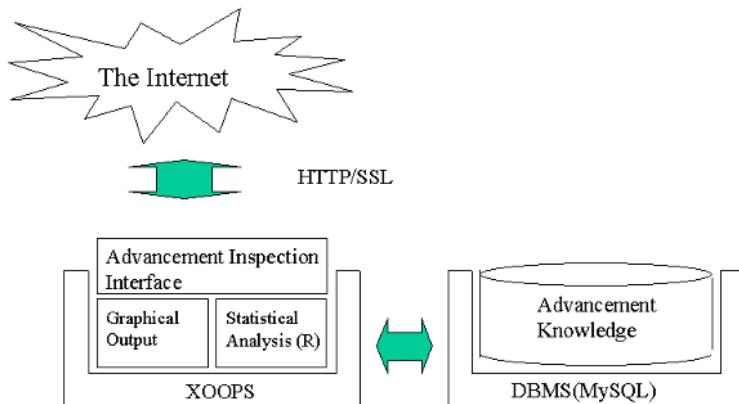


Figure 3. Block Diagram of Proposed System.

3.2 Advancement-inspection Knowledge

The advancement knowledge was completely re-designed for the proposed system. Here, it should be noted that there is no existing standard for advancement-inspection knowledge. One of our targets in knowledge design is to provide a candidate for the standard advancement-inspection knowledge. The knowledge used in the proposed system has three major features as described below.

Inspection Items Selected and Modified from Major Inspection Documents

Since there is no standard for advancement-inspection knowledge, the authors designed the knowledge by using three documents commonly used in Japan: 1) Nursery School Nurture Guide (in Japanese), published by The Ministry of Health, Labor and Welfare of the Japanese Government [6], 2) Infant Spiritual Advancement Diagnostics: from three

years old to seven years old (in Japanese), published by Makoto Tsumori and Keiko Isobe [4], and 3) Infant Advancement Inspection Method by Onjyouji (in Japanese), published by Munenori Onjyouji [5].

The Nursery School Nurture Guide is conceptual, so it is difficult to generate concrete items from this document. Therefore, many items were selected from the commonly used advancement-inspection tests in “Infant Spiritual Advancement Diagnostics” by Makoto Tsumori. This document has very concrete descriptions and thus is effective for item generation. However, it contains no items for children younger than 3 years old. Therefore, the third document, Infant Advancement Inspection Method by Onjyouji, was incorporated. Finally, 334 items in 32 sub-categories were selected as shown in Table 1.

Table 1 Numbers of Items in Five Major Categories

Age	Major Categories					Total
	Health	Human Relation	Environment	Language	Expression	
From birth to less than six months	14	6	0	1	0	21
From six months to less than twelve months	11	3	1	5	4	24
From twelve months to less than 18 months	18	8	4	8	2	40
From 18 months to less than 24 months	13	7	4	8	2	34
Two years old,	14	11	7	8	6	46
Three years old	13	12	10	7	8	50
Four years old	16	10	18	11	9	58
Five years old	13	10	18	11	9	61
TOTAL	112	67	58	57	40	334

Items Chosen Freely, according to Advancement Stage of the Child

The inspection item scores are inputted every month in the proposed system. Let's assume that each item is fixed for one school year. This scheme is widely employed in conventional knowledge-recording systems. If the highest score is selected at the starting point of one school year, the nursery teacher cannot read any developmental process from the constant item value for one year. For instance, if the score is 4 in April³, the record is 4,4,4,4,4,4,4,4,4,4 for one school year.

Figure 4 shows the relation between some items for one-year-olds and those for two-year-olds. The entire body of knowledge is very large. The number of one-year-old items is 40, and that of two-year-old is 34. Figure 4 shows a small part of this knowledge. The major categories are selected by the Japanese government, and the knowledge used for each age has identical categories.

The names of sub-categories, however, are not identical. In Figure 4, sub-category “Meal” of major category “Health” for one-year-olds is not included in the major category “Health” for two-year-olds. Thus, the sub-categories can never be described in a one-to-one mapping over plural years. The number of items for one sub-category varies with the child's age. In some cases, no identical sub-category name can be found in the knowledge sets for higher-age/lower-age comparison.

In the first stage of the authors' study, they tried to find a one-to-one mapping for all item names. However, this target is far away from the real world. Thus, as shown in

³ In Japan, the school year starts on 1 April.

Figure 4, the authors finally made each sub-category correspond to others in different age-years as much as possible. Consequently, a seamless one-to-one mapping between item names is not guaranteed in the proposed knowledge.

Objective Score Selection based on Vygotsky's Psychological Theory

For the sake of objective observation, the appraisal value must not contain any ambiguity. Vygotsky [8] maintained that a child follows the nurturing adult's example and gradually develops the ability to do certain tasks without help or guidance. He called the difference between what a child can do with help and what he or she can do without guidance the "zone of proximal development" (ZPD). From this point of view, the authors defined the following five evaluation scores for the inspection items used in the objective observation.

One-year-old			Two-year-old		
Major Category	Sub-Category	Item Contents	Major Category	Sub-Category	Item Contents
Health	Meal	Tries to eat with spoon by himself/herself.	Health	Clothes	Wishes to put on and remove clothing by himself/herself.
	Meal	Enjoys the time of the meal.		Clothes	Lays out the clothes by himself/herself if the clothes are simple.
	Clothes	Tries to wear or lay out clothes with interest.		Clothes	Puts on the clothes by himself/herself with the support of the nurturing person.
Human Relation	Sense of Relief	Relies on the nurturing person, he/she has a sense of relief.	(No Item found in the Two-year-old knowledge.)		
	-----	-----			

Figure 4. Relationships between Items for One-year-olds and Items for Two-year-olds.

Score 0: Completely impossible

Score 1: Possible with active support of nurturing persons

Score 2: Possible with indications by nurturing persons

Score 3: Possible if child is with friends who act simultaneously

Score 4: Possible through personally achieved advances

The above standard for score selection is general and does not depend on issues related to the particular item. Figure 5 shows the proposed system's interface screen for inputting the advancement-inspection item scores. Each of the above criteria is described in the left-center of the user interface. Each item is shown in the center of the interface screen, and the item score is selected by using a "radio button."

The removal of ambiguity in the items' evaluation provides several advantages. First, this scheme removes the influence of the evaluator's judgment style. Even if one class has two nursery teachers, the scores can be decided mutually between them. Second, the objective score absorbs the difference due to the fact that observation results may be inconsistent over time. Even if the observation judgment of a nursery teacher changes over time, the score difference is kept minimal. This means that the evaluated scores can be applied to statistical analysis or displayed by graphical output for easy observation.

4. Graphical User Interface

The proposed system has four major interface screens: 1) user interface for evaluation, 2) graphical output of the evaluation scores, 3) statistical analysis results, and 4) user management. The user management interface has features such as user account create/modify/delete options and user property controls, and it's accessed by the system administrator. The details of the administrator's interface are omitted in this paper.

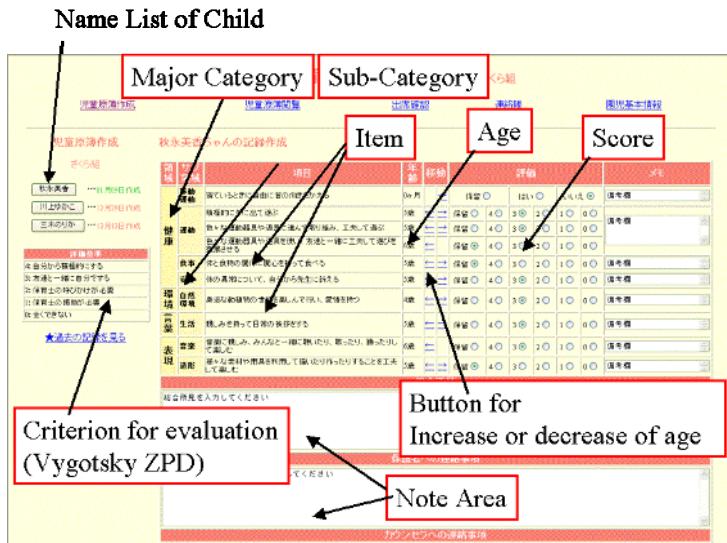


Figure 5. User Interface for Evaluation

User Interface for Evaluation

The user interface for evaluation shown in Fig. 5 has score input function for each inspection item. The items' score values are inputted through "radio buttons." There are many sub-categories and items, so the inspection-item area can be scrolled up and down. The evaluation screen also has two "Note Areas" for the nurturing person's comments.

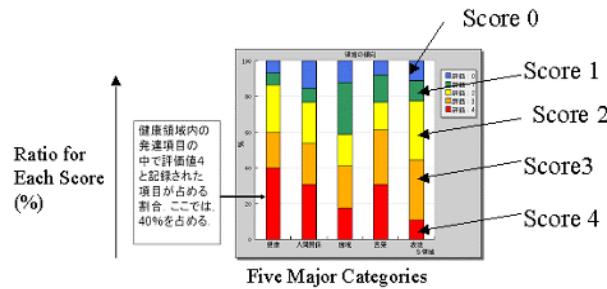
One of the most important interface features is the "button for increasing/decreasing age." In Fig. 5, the right-facing arrow indicates "increase the age for the item," and the left-facing arrow indicates "decrease the age for the item." The arrow does not appear if there is no identical sub-category name in the target knowledge. If the corresponding sub-category has plural items, an item selection sub-window appears in the screen for the nursery teacher to select the appropriate items.

Figures 6(A) and 6(B) show graphical outputs of the average values for each sub-category of one child. The child is selected from the list of names in the class. Figure 6(A) shows score value distribution for each sub-category. Figure 6(B) shows the change in score value distribution over time. The value of the sub-category is the average of all items in the sub-category.

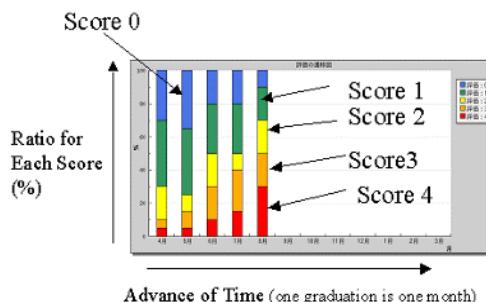
Figure 7 show a radar chart graph of all sub-category values for one child. The average score of all children in the class is shown by the solid red line in the graph. The nurturing person can read the tendency of advancement in each sub-category and compare the

values with the class as a whole. These charts enables users to grasp the balance of growth of the child development by this graph.

Figures 6 and 7 avoid the risks of showing values determined by subjective criteria. First, if the scores were selected at different times, the values could not be compared with each other in Fig. 6. Moreover, if one class has two nursery teachers, their averaged values would be reflected in Fig. 7. These graphical outputs are thus very convenient for daily observation, providing fruitful results from objective score selection based on Vygotsky's psychological theory.



(A) Score Value Percentage for Each Major Category



(B) Time Dependency of Five Score Values

Figure 6. Ratio for Each Score

Values for the Observed Child

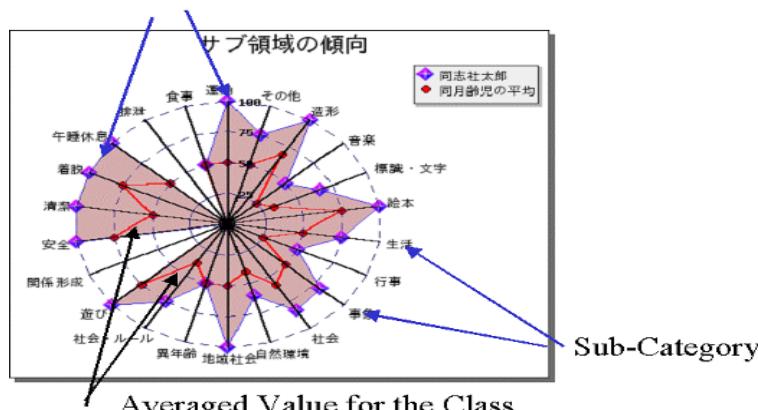


Figure 7. Values of Sub-categories and Averaged Value in the Class

Correspondence Analysis of the Score Values

The nurturing person can read only the statistical results by himself/herself in Figs. 6 and 7. These graphs are written from raw inspection-score data. To read more detailed information, statistical analysis can be applied. In the proposed system, “Correspondence Analysis” is employed for the objective observation.

Correspondence analysis enables us to analyze two-way tables that display some measurement of correspondence between the rows and columns. This method was originally developed in France by Jean-Paul Benzécri in the early 1960's and 1970's. Similar techniques have been developed independently in several countries. For instance, Chikio Hayashi of the Institute of Statistical Mathematics of Japan proposed a similar method in 1952 [7].

Figure 8 shows an example of correspondence analysis of item scores. The analysis tool is the “R” system, a free version of the S-plus statistical analysis package. The result shows two dimensions, first component and second component. The triangle mark shows the name of the sub-category. The black circle mark shows each child. Twenty-three children are plotted in Fig. 8. If a black circle mark (child) is located near a triangle mark, the child has a high score for the corresponding sub-category. If one black circle mark (child) is located away from the other children, the behavior is considered “unique,” and the nursery teacher should pay special attention to the child.

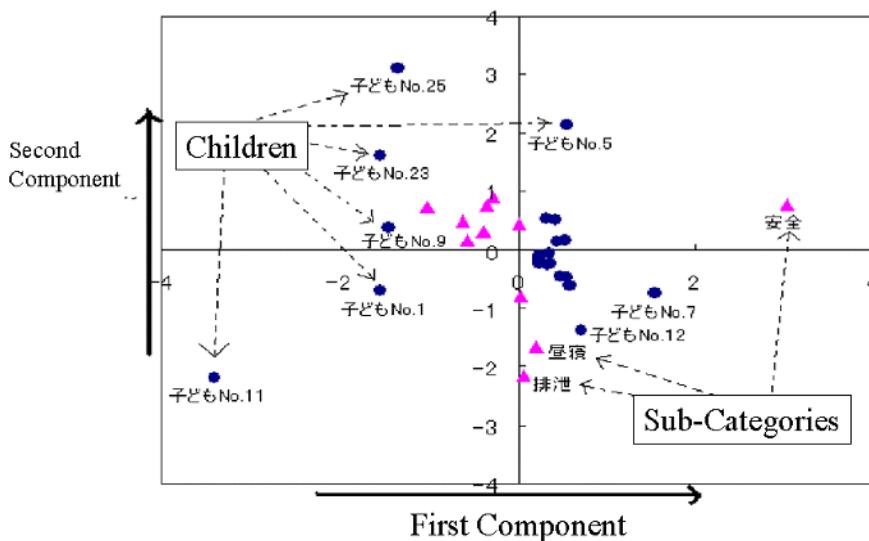


Figure 8. Correspondence Analysis of Score Values

5. Conclusion

This paper has proposed newly designed advancement-inspection knowledge for the early childhood education domain, specifically nursery schools in Japan. The proposed knowledge has three important features: 1) Advancement-inspection items selected and modified from commonly used inspection documents in Japan, 2) Items chosen freely, according to advancement stage of the child, and 3) Objective selection of advancement

scores based on Vygotsky's psychological theory. The totally re-designed knowledge and objective score-selection scheme enables the system to analyze scores statistically and to display graphical outputs as an effective feedback tool for daily nurture. The new knowledge is suitable for wide use and may become a candidate for a new standard of advancement-inspection knowledge.

Furthermore, a new advancement-inspection system was demonstrated in this paper. This system employs the above knowledge. The advancement information is shared among parents, a nursery teacher, a chief nursery teacher, a medical doctor, and a public-sector parenting support center. The objective score enables the nurturing persons to compare the inspection item scores among different classes and different observations over time. The system is implemented by using Xoops and MySQL. An evaluation trial will be started in a real nursery school in Osaka, Japan from April 2006.

References

- [1] Keijiro Sasada, Kimio Shintani, Hirohide Haga, and Shigeo Kaneda, "A Support System for Early Childhood Education: E-infant Education NET System," The 4th IASTED International Conference on Web-based Education, No. 461-808, Feb. 2005
- [2] Kimio Shintani, Kazuki Komai, Atsushi Yoshida, Shigeo Kaneda, and Hirohide Haga, "Proposal for a Framework of Community Support in Infant Education by using the Combination of Multiple Cameras and a Bulletin Board," The 4th IASTED International Conference on Web-based Education, No. 461-811, Feb. 2005
- [3] "Just Crayon," <http://www.justsystem.co.jp/crayon/> (In Japanese, 2006/03/05 Confirmed)
- [4] Makoto Tsumori and Keiko Isobe, "Infant Spiritual Advancement Diagnostics: from 3 years old to 7 years old (in Japanese)," Dainippon Publishing Ltd., 1995.
- [5] Munenori Onjyouji, "Infant Advancement Inspection Method by Onjyouji (in Japanese)," Keio Univ. Publishing, 2004.
- [6] The Ministry of Health, Labor and Welfare, the Japanese Government, "Nursery School Nurture Guide (in Japanese)," 1998.
- [7] C. Hayashi, K. Yajima, H.H. Bock, N. Ohsumi, Y. Tanaka, and Y. Baba (Eds.), "Data Science, Classification, and Related Methods Series: Studies in Classification, Data Analysis, and Knowledge Organization," Vol. XVI, Springer, 1998.
- [8] Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.
- [9] Xoops, <http://www.xoops.org/>
- [10] MySQL, <http://dev.mysql.com/>
- [11] The R Project for Statistical Computing, <http://www.r-project.org/>

A Database System of Buddhist Canons

Takehiko TANAKA^{a,1}, Yohei NINO^a Rong ZHANG^a Martin ROLLAND^b
Masaru NAKAGAWA^a Susumu AOKI^c Keigo UTSUNOMIYA^d and
Toshinori OCHIAI^c

^a Wakayama University, Japan

^b University of Technology of Belfort Montbéliard, France

^c International College of Postgraduate Buddhist Studies, Japan

^d Osaka Ohtani University, Japan

Abstract. Buddhist scriptures written in Chinese several hundred ago are in the possession of temples in Japan. To make use of the contents for Buddhist study and archeology, we have ever made the comparative tables with acknowledged catalogues and taken shot images of the scripture using digital cameras. Those shot images are sometimes unreadable due to degradation, or one takes lots of trouble with combining the images. In this paper, we focus on the database for assisting the reading. Since the correspondence of the scriptures between Buddhist canons or the possessions in temples is available in the database, we can easily use well-maintained text files for our shot images. We also introduce the processors for eliminating redundant part of shot images and text data, together with the viewer which enables us to read the seamless image and its text simultaneously. It is possible for the database to accept enormous image and text files, interested people, and agents for the reading support.

Keywords. digital archive, database system, image processing, reading support, Buddhist canon

1. Introduction

Buddhist scriptures were originally written in Sanskrit. As the Buddhism was propagated to the east, the versions translated in Chinese have been read, transcribed by handwriting, run off, and distributed around East Asia such as China, Korea, and Japan. To set thousands of those scriptures in order, there have been made catalogues. *Taisho Tripitaka* is a definitive one, and currently lots of researchers on Buddhist studies make use of it to identify the documents in temples or to write bibliographical information.

There exists many persons and projects that computerize the scriptures, and those data are on exhibit. CBETA[1] is an organization and a project in Taiwan which assembles text files made manually by typing the characters referring the Buddhist scriptures.

Taisho Tripitaka is quite popular but it is not all about the Buddhist canon written in Chinese. Taisho Tripitaka was basically compiled on the ground of Korean version of canon by block printing, most of which were made about 13th century. In contrast,

¹Faculty of Systems Engineering, Wakayama University, 930 Sakaedani Wakayama Japan, E-mail: takehiko@sys.wakayama-u.ac.jp

there exist temples in Japan which possess a suite of scriptures, called *Issaikyo*. That is often referred to as say “Kongoji-Issaikyo” or “Nanatsudera-Issaikyo” with the name of the temple prepended. It is known that those scriptures were transcribed between 8th and 12th centuries. We have extensively visited the temples with Issaikyo and made tables displaying whether each scripture exists or not in those temples, according to the orientation of another famous catalogue, to make sure that every Issaikyo overlaps with Taisho Tripitaka. With regard to the content, however, there were found several documents of which Japanese and Korean versions are quite different while the title is the same. This fact suggests that it is insufficient to only computerize the materials, but that a database system is strongly needed which enables one to compare the details of the scriptures among different catalogues and temples’ possessions.

For the researchers to put the images to practical use, we have to solve the problem on retrievability; those images are easy to view but it is quite hard to search the intended character or phrase from the image file. The best medium of retrievability is a text file. One can easily find the desired letter or word with the feature of some standard text editor. Moreover, by using full-text searching software, we can immediately find a text file including the specified keywords from numerous digital documents.

In this paper, we introduce a suite of programs for the purpose of enabling us to view both the shot images of Buddhist scriptures and the corresponding text data. Until now, image and text processors were implemented which eliminate redundant data of shot images and text files, respectively. In addition we attempted to construct a viewer displaying the image and the characters simultaneously, as a web application. In the days ahead, we have to define the sustainable database with enormous images and text data together with their correspondence, in consideration of researchers’ use. We describe the overview and several features of the database under discussion.

After our system is completed and the data is fulfilling, the researchers of Buddhist studies or study of Japanese or philology will have their full of real, lossless Buddhist scriptures to think of human relationships and propagation of Japanese Buddhism of those days by means of the massive literature. Moreover, this paper contributes to the communities of information communication technologies in the sense that it presents a solution of how to construct an effective searching and browsing system for tens of thousands of images.

2. Shot Images of Buddhist Scriptures

2.1. Shot Images

We have computerized the Buddhist canons with two parties separated. One is called the “imaging group” who go to the temple to digitize (make the shot images of) the scriptures with digital cameras. The other is the “processing group” who extract the information about letters from the image files to store into a database server. This paper mainly dedicates solely to the activities of the processing group. Here, we briefly explain the format of the image files, which are the input of our process as well as the works of the imaging group.

Most Buddhist scriptures of Kongoji-Issaikyo are rolled book. When one is spread out, it is tens of meters length while being as tall as a largish book. Each line is written



Figure 1. Shot Image

vertically, and so many lines from right to left construct a document. A scripture has about 8,000 Chinese characters on average. Since the number of characters per line is 17 normally, we can estimate that the scripture has about 470 lines. Every character was handwritten and allocated less orderly than that in epigraphs.

Since it is impossible to take only one picture of the scripture, the imaging group has taken several pieces of picture. There are typically constructed 20 pieces for a scripture. In earlier activities, we had used digital cameras with 5 mega pixel (2560x2048), and now with 12 mega pixel (3840x3072). Those shot images are so large in terms of file size as well that we reduce the images to the ones with 2.5 mega pixel and about 30,000 colors which are the input of image processors. Figure 1 shows an image, which is the beginning of a scripture.

We started photographing at Kongoji Temple in 2000. We have visited the temple once or more a month, took shot images of scriptures in a room of the temple. Until now, we have computerized more than 3,000 scriptures. Since the total amount is said to be 5,300, we will finish shooting in five years.

When shooting the images, the adjacent pieces hold several lines in common. Usually 5 or 6 lines are overlapped, although the last piece might overlap 20 lines. The human eyes can immediately recognize the overlapped lines when looking at two adjacent images, but the computer is not as good at doing that. Therefore we had to produce a program detecting the border lines which mean that the outside is discarded for the image composition.

2.2. Image Processing

For calculating several sorts of coordinates from an image file, we decompose the image processing to the following six steps; (1) converting the image format, (2) detecting and deleting the margins at the left, right, top and bottom, (3) making the image black and white, (4) detecting line regions, (5) detecting character regions for each line region, and (6) detecting discarded line regions. We actually implement every process using basic image data processing.

Throughout processing, we employ PNG (Portable Network Graphics) as the image format, because of its quality, property of compression, and tractability. The image pro-

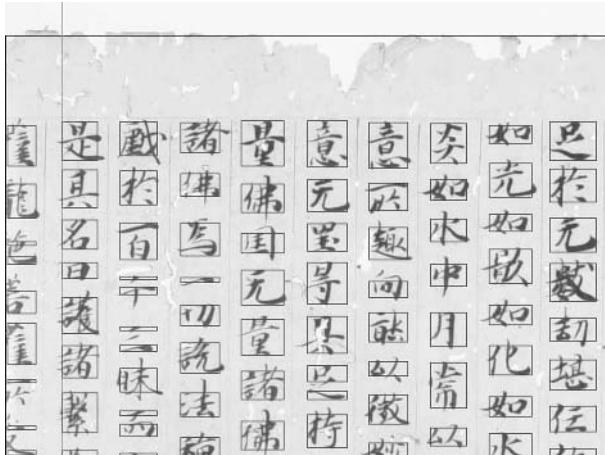


Figure 2. Recognized Character Regions

cessing programs had been developed with Ruby and the Imlib2 library in the beginning, and now, they are rewritten in Java.

We will make a brief introduction of image processing, leaving the details or algorithms to [2]. To detect the margin for a given shot image, we defined a function (a method in Java) which takes a pixel as input and judges whether it is closer to white or to the color of paper. Since it was already known that the color of paper is roughly gray, yellow or brown, we implemented three judgment methods to select one according to the shot image. Whether the input image is a full color or monochrome, our programs scan it in vertical or horizontal direction but not in combination with these directions. For example, connected components which would be gained by some advanced character recognition are not obtained. Our processors attempt to identify the regions of body texts, lines or characters as *rectangles*. We will expect to make our processors high performance in dealing with thousands of shot images, although this approach might recognize the regions a little less accurately. Figure 2 is a part of image with detected regions boxed. With regards to the accuracy, we have applied the programs to several images to find that the detection of margins and lines is no problem while there is room for improvement to that of characters.

The properties of the coordinates are as follows, which are convenient for storing the data in the database:

(1) A character region is expressed as a 5-tuple of a file name, x-coordinate, y-coordinate, width, and height, where the unit is pixel and the top-left is the origin.

(2) When character regions belong to a common line region, they have the same image file name, the same x-coordinate and the same width. This property helps the sort of character regions in a scripture; we can sort them where the file name, the x-coordinate and the y-coordinate are the key. Moreover, it saves the data size of database since we can register the data of the line region apart from the character-specific data (the y-coordinate and the height).

Finally, to detect discarded line regions in adjacent two images, we make use of recognized character regions. Each line region can be expressed as a vector with the coordinates of the character regions. And then, we attempt to compare the vector of the

second leftmost line on the right image with that of each line on the left, to find the position whose error is the minimum. You can find the vertical line in Fig. 2, which means that the left region is discarded.

3. Text data of Buddhist Scripture

We employ a CD-ROM distributed by CBETA as original text files. The disk includes 9,035 plain text files of Buddhist scriptures, XML (Extensible Markup Language) files structured with XML tags of the same number as plain text files, and data for external characters. The text file covering the passage seen in Fig. 1 was able to be found using the title. To extract the text data for our database, we started with the XML files but not the plain text files due to the completeness and the tractability of the XML documents. Note that so far as we have looked at pairs of images and the text files, the position of line feeds of images is nothing to do with that of text files.

We made a Ruby script with less than 100 steps to extract the text information from a given XML file. The details are described in [3]. Instead of applying an XSLT (Extensible Stylesheet Language Transformation) or any other XML transformer, we regard the XML documents as merely plain text, and implemented the extraction program which removes redundant elements (not only tags but the contents within the tags) and tags (preserves the contents within the tags), and decides the position of line feeds. In addition, when it finds an external character, the information is recorded in another file associated with the output text file, while the external reference is replaced with a special symbol which is never used in Buddhist canons, as old-fashioned typesetters did. It takes less than two seconds per XML file to convert the encodings (from Big5 to UTF-8) with iconv and extract text information.

After extracting 9,035 text files of Buddhist canon, we constructed a full-text search system using Hyper Estraier[4]. That supports the UTF-8 encoding and the N-gram method for indexing. In addition, Hyper Estraier supplies CGI programs for a web-based document retrieval. For these advantages, we have effectively made a full-text search environment. Note that it supports the query with regular expressions, but that the feature works only for ASCII words and has no advantage for us.

We conducted experiments on the performance of this full-text search system. In advance of the experiments, we got some shot images lined up, and for each shot image, we saw what text file covers it using the name, volume and number of the scripture. A single experiment starts with choosing a string in the given shot image. We type the letters in the input form. After pushing the submit button, we get a result window. From the output of the search engine, we do not only know how many files are hit but whether the text file which is picked out previously is included.

As a result of choosing several words in shot images of two scriptures, a word with at most three characters finds one hundred or more text files, while four-letter words or longer report that there are less than ten documents hit. Moreover, two or three words each of which has three characters gains good results just as a longer single word. In each case the corresponding text file is hit, as far as we correctly type the characters. The turnaround time is about 0.1 second in most cases, and we have not ever experienced the search over one second.

4. Design of Buddhist Canon Database

This section describes the design of the database for Buddhist canons, based on the image and text files above. We have already laid a plot in [5], and in this paper we put it into effect to deal with image files in combination to text data. The outline of the design is as follows:

1. The database holds the hierarchy among catalogues, scriptures, images, line regions and character regions, as well as those contents.
2. The database stores and correctly manages three sorts of data; (1) files outside the system, such as digital shot images, (2) data generated by the processors of images or text files, and (3) annotations made by users, mainly to correct the automatically-generated, misdetected coordinates or values.
3. The database does not preserve any image data or text files. Instead, their file-names are registered.

Actually, we defined the tables `catalogue`, `scripture`, `image`, `line_region` and `character_region` and linked adjacent two tables above with one-to-many relationships. Here we will clarify the five tables constructing the hierarchy:

Catalogue: A collection of Buddhist canon (scriptures). It hold true of Taisho Tripitaka, Kongoji-Issaikyo, Nanatsudera-Issaikyo, and so on.

Scripture: A document of Buddhist canon. Although most scriptures were made as scrolls, we do not use the “scroll” in our database since “scripture” is the more appropriate name.

Image: A shot piece of a scripture. In general, one scripture are constructed with about twenty images or more. Note that adjacent two images have overlapping lines, due to the traditional way of photographing. This means that we have to memory some information in the database for deleting redundant regions to put together a scripture images.

Line_region: A vertical line of text in an image. When an image file, an X coordinate and a width are given, the image of line region is easily obtained. At stated previously, we have implemented the recognizer which calculates pairs of X coordinate and width with few errors.

Character_region: A region of character of line region. When an image file, X and Y coordinate, and a width and a height are given, the image of character region is easily obtained. By adopting the rule that identifies the X coordinate and the width of all the character regions in a line region, we can save the data of coordinate in the database, by the sacrifice of the correctness. We have been developing the recognizer of character regions. The output seems basically good but mistake-filled in details.

As mentioned in the design outline, our database does not preserve any image data, not only shot image with digital camera but also subimage required for the viewer. Instead, we adopt the following rules; (1) Each shot image is located in a file tree which is appropriately sectioned by directories. It is properly named so that the processors and the viewer can find the file easily. (2) Subimages which are the output of the processors, such as line regions, character regions or status images for confirmation, are generated when there is a need for the viewer, and then, the files are saved in a cache directory. Those rules are helpful in making the database efficient in terms of disk size. It is true

that storing all the subimages in the database might save time, but those would occupy enormous amount of storage which most of them would be never or once used.

Only these five tables could not form a sustainable database. We have to additionally define the tables which behave as the lubricant. First of all, to cope with the fact that we can correspond some scriptures in Kongoji to Taisho Tripitaka and cannot the others, we set up the table of `scripture_correspondence`, which has two links to the scripture table for expressing the correspondences. This table helps for the correspondence not only between an Issaikyo and Taisho Tripitaka, but between the catalogues. In other words, we can store the published relationships investigated by the humanity researchers.

For an image, there may exist more than one way of splitting it into vertical lines. Some line regions might be generated by the image processor we have developed, or some specified or collected by a user. Since we would like to take in these candidates of line recognition but not to store only one in the database, two tables are introduced which serve as the intermediary between `image` and `line_region` tables. When a user choose a set of line regions, he or she would obtain the ordered pairs of the x-coordinate and the width which constitute of line regions. In a similar way, we introduced two tables between `line_region` and `character_region`.

As for the text data, we introduced two tables `line_annotation` and `character_annotation`, attached to `line_region` and `character_region` respectively. The record of `line_annotation` holds an annotation or a text expression of the corresponding line region. The table has the attribute to indicate the sort. The `line_region` is connected to `line_annotation` with a one-to-many relationship. The relationship between `character_region` and `character_annotation` is alike.

5. Applications for Reading Support

We implemented the viewer of the shot images of Buddhist scriptures in combination with their text data, before the construction of the database. Figure 3 shows a screenshot. The upper half of the window displays the image of a scripture, while on the lower half the texts line up in vertical writing. The scrollbar at the bottom of the figure is for the text data. On the other hand, if you grab the image and rock it to and fro, then the image will swing together. The feature of the scrollable image is based on AJAX (Asynchronous JavaScript + XML) — just as moving on the map using Google Maps, we can change the viewpoint of the Buddhist scripture with a drag-and-drop operation. That is why the parts of the image and the text are scrollable right and left independently. It is a future task to make the position of text part go along with that of image and vice versa.

The viewer is constructed as a web application where a browser connects the web server to retrieve the images and text data required for indication. The images are split in advance; we ran the image processors to obtain the values of coordinates, drew lines and boxes according to the coordinates, corrected the position manually, and cut the images by line for the viewers. So are the text data. Dividing images helps to reduce the communications traffic and enables us to render the scripture with a cheap personal computer, in comparison with the way that one composes a extremely wide image, sends it to another person, and the receiver takes a look at it. By the way, our approach needs a web server with a database server connected, although a server computer including



Figure 3. Viewer of Buddhist Scripture

the two capabilities is enough. In an operation phase, the database server would have a storage for much enormous split images.

The file size of the images should be mentioned. A combined, single image of scripture (29602x1049) reaches about 40 megabytes (MB), while the fragmented images (128x128) amounts to 47 MB. Another image of 87 MB is partitioned into the images of which the total size is 116 MB, i.e. about 30% larger than the single bay's size. Each piece occupies 20 kilobytes on average. Needless to say, the files can be reduced-size by shortening its width, height, and/or colors.

We performed experiments with the distant places to verify the practicality of this web application. The server which stores an HTML file and thousands of divided images located in Wakayama University, while the client is in International College of Postgraduate Buddhist Studies, Tokyo. Both end points rest on cities of Japan where the distance is about 500 km. The transmission speed measures about 1 Mbps. As a result of the test use, the user of the client was able to view the image, without feeling stressed when downloading fragmented subimages at short intervals. This suggests that the viewing environment where the server stores enormous amount of split images for the remote clients to get some of them is so effective.

We are planning other applications for supplying the features of reading support. To make the full-text search more usable, we have to attach an interface which allows us to input the Chinese character with a mouse operation only, e.g. by choosing one from displayed characters. We would like to cope with the change of letter shape.

As for the combination of images and text data, applications beyond the viewer will be required. At the moment, wrong coordinates are corrected by looking at a *status image* as Fig. 2 and then reading a text file consisting of numerical values. It is quite troublesome. If the graphical user interface where one can view the trimmed region, line regions and character regions on an original image and do a drag-and-drop operation to correct the position, then those who are not familiar with the database will be able to

modify. Moreover, we will have to present an interface for comparing the status images to select one or merge them.

6. Related Works and Discussions

There have been designed and developed various support systems for digital archives, in collaboration with experts of information communication technology and researchers of human studies. Here we are introducing several works.

Ishikawa et al.[6] proposed a data model and implemented a scrapbook system for managing digital images of historical documents. The data model is described by Relax schemata. Clipping character regions lies in the hands of the users. They attempted to apply the system to wood strips made over 1,000 years ago[7]. Kitadai et al.[8] also implemented a character recognition system for wood strip, which detects the region of solid ink using the color of wood together with a discriminant analysis. Each wood strip has at most few dozens of characters, which is about 1/100 as many as a typical Buddhist scripture.

Maeda et al.[9] investigated a digital library with multilingual text search and annotation, where a document written in 12th Century was set as the major matter. They attempted to recognize the characters using an OCR (optical character reader).

Tsuboi et al.[10] attempted to extract character regions from digital images of woodblock-printed books handwritten in Japanese. The processes seems to be complicated since there exist several sorts of Chinese and Japanese characters some of which are conjoined. The authors would like to put a high priority on processing enormous images rather than accuracy. As far as we know, other well-known recognizers for handwritten characters aim at smaller number of characters with high precision. That is why we implemented the image processors from scratch.

As far as text data of Buddhist canons in Chinese are concerned, CBETA seems to be most comprehensive, but we have felt dissatisfied with the feature of text search. Actually, we gave some keywords but failed to obtain the target document while succeeding using our search system. That is because the text files have symbols such as a period. Our system supports a smooth full-text search outside the database, while the bibliographical information is related with the content under our database.

Recent years, “folksonomy”[11] attracts attention on the net, as the notion in contrast to formal classification methods. It is not so easy for our system to adopt that notion immediately. We however take it into consideration that the envisioned users of our system are of great variety; some belong to archeology and others Buddhist studies; some are experts and others may be novices; of course, they live in various places. Therefore when the keywords and annotations by a broad range of people are accumulated in the database, the system extended for the feature of folksonomy will be able to combine those pieces of information to propose a new form of information to the users.

7. Conclusions

In this paper, we have reported our activities of the applications and database for supporting the reading of Buddhist scriptures. The database expresses the correspondence of

the scriptures between Buddhist canons or the possessions in temples. Several processors scrape redundant regions off images and text data so that we can use a web-based viewer to take a look at the image and its text simultaneously. As a result, our system is a good solution of the database for enormous pieces of information together with various sorts of users.

Here we are explaining a perspective on our database system. Taking the quality of the existing scriptures into account, it is not good to direct us to produce an automatic, high-precision character recognizer. We would rather leave the matching between images and texts to those who can read Chinese. To remove the burden of finding Chinese characters of the images one after another, we would like to support the workers by presenting the relevant text data from CBETA for example. If he or she chooses the strings and sometimes types the letters for specifying the text data of character regions, line regions or images, then the operation will be much effective. Our database registers the name of worker (which may be a recognizing program), which will form a sort of incentive. As a matter of course, this feature will contribute to a large-capacity, high-accuracy database of Buddhist canons.

References

- [1] Chinese Buddhist Electronic Text Association (CBETA), <http://www.cbeta.org/index.htm>.
- [2] R. Zhang, Y. Nino, T. Tanaka, M. Nakagawa, S. Aoki, K. Utsunomiya and T. Ochiai, *Image Processing for Buddhist Canon Database*, IPSJ SIG Technical Reports, 2006-CH-69 (2006), pp.25-32 (in Japanese).
- [3] T. Tanaka, Y. Nino and M. Nakagawa, *Text Processing for Buddhist Canon Database*, IPSJ SIG Technical Reports, 2006-CH-69 (2006), pp.33-40 (in Japanese).
- [4] Hyper Estraier: a full-text search system for communities, <http://hyperestraier.sourceforge.net/>.
- [5] Y. Nino, T. Tanaka, R. Zhang, M. Nakagawa, S. Aoki, K. Utsunomiya and T. Ochiai, *A Database System for the Picture Files of Buddhist Canons*, Journal of Japan Society of Information and Knowledge, Vol.15, No.2 (2005), pp.15-18 (in Japanese).
- [6] M. Ishikawa, K. Hatano, T. Amagasa, S. Ueyama and T. Katsumura, *An Electronic Scrapbook System for Historical Document Images*, IPSJ Transactions on Databases, Vol.44, No.SIG12 (TOD 19) (2003), pp.110-122 (in Japanese).
- [7] M. Ishikawa, K. Hatano, T. Amagasa, S. Ueyama and T. Katsumura, *A Design of an Electronic Scrapbook System for Historical Documents*, IPSJ Symposium Series, Vol.2003, No.21 (2003), pp.227-234 (in Japanese).
- [8] A. Kitadai, K. Saito, D. Hachiya, M. Nakagawa, H. Baba and A. Watanabe, *Design and Prototype of Assistant System for Decoding Scripts of Mokkan*, IPSJ Symposium Series, Vol.2004, No.17 (2004), pp.215-220 (in Japanese).
- [9] A. Maeda, A. Sako and T. Sugihashi, *Building a Digital Library of Kyoto Studies and its Multilingual Information Access*, IPSJ Symposium Series, Vol.2003, No.21 (2003), pp.195-202 (in Japanese).
- [10] A. Tsuboi, K. Hachimura and M. Yoshimura, *Segmentation of Characters for Historical Woodblock-Printed Books*, IPSJ SIG Technical Reports, 2005-CH-66 (2005), pp.53-60 (in Japanese).
- [11] Folksonomy - Wikipedia, the free encyclopedia, <http://en.wikipedia.org/wiki/Folksonomy>.

Author Index

Akchurina, N.R.	275	Labra, J.E.	167
Aman, H.	135	Lewis, L.	213
Aoki, S.	327	Loke, S.	233
Asano, M.	317	Lounis, H.	73
Barzdins, G.	157	Majtás, L.	139
Berrueta, D.	167	Mäkilä, T.	21
Budnikas, G.	196	Maruyama, H.	206
Buford, J.	213	Mise, T.	41, 115
Choinzon, M.	61	Mizuno, M.	253
Eessaar, E.	96	Modroiu, E.-R.	106
Entwistle, S.	233	Mori, K.	263
Eremeev, A.P.	303	Morita, T.	176
Fukaya, K.	31	Nakagawa, M.	263, 327
Fukazawa, Y.	31	Nakamura, K.	51
Grigorenko, P.	83	Nakamura, T.	206
Gruzitis, N.	157	Nakatani, T.	41, 115
Haga, H.	3, 317	Návrat, P.	143
Hashimoto, M.	41, 115	Nino, Y.	327
Hashiura, H.	51	Ochiai, T.	327
Hasida, K.	176	Ohnishi, A.	125
Hayashi, Y.	51	Okazaki, H.	135
Hoshii, S.	176	Osipov, G.	294
Ikemoto, K.	263	Oskin, P.V.	307
Izumi, N.	176	Polo, L.	167
Jakobson, G.	213	Pranevicius, H.	196
Jakubík, J.	143	Rolland, M.	327
Kagawa, S.	263	Saito, S.	151
Kambayashi, Y.	253	Sakamoto, Y.	186
Kametani, H.	115	Sasahira, T.	3
Kanazawa, N.	3	Schieferdecker, I.	106
Kaneda, S.	3, 317	Shintani, K.	317
Katamine, K.	41, 115	Shinyashiki, Y.	41, 115
Kendall, E.	233	Shirogane, J.	31
Kinoshita, D.	51	Smirnov, I.	294
Kobayashi, H.	186	Stefanuk, V.	285
Komiya, S.	51	Sterling, L.	223
Kondo, K.	176	Takimoto, M.	253
Kono, A.	317	Tanaka, T.	327
Kotenko, I.	243	Tanihira, K.	186
Krishnaswamy, S.	233	Taveter, K.	223
Kudins, R.	157	The Daedalus Team	223
Kurio, M.	253	Tikhomirov, I.	294
Kuusela, T.	21	Tourtoglou, K.	11

Tyugu, E.	v, 83	Yamada, H.	135
Ubayashi, N.	41, 115	Yamaguchi, T.	v, 176
Ueda, Y.	61	Yamamoto, S.	147, 151
Ulanov, A.	243	Yamamoto, Y.	263
Utsunomiya, K.	327	Yazid, H.	73
Vagin, V.N.	275, 307	Yoshihiro, T.	263
Varshavsky, P.R.	303	Zavjalova, O.	294
Virvou, M.	11	Zhang, R.	327
Vybornova, O.	294	Zhzhikashvili, A.	285
Yaegashi, R.	51		

This page intentionally left blank

This page intentionally left blank