Martin Atzmüller, Rainer Knauf (Eds.) and
Stephan Bode, Qurat-ul-ann Farooq, Matthias Riebisch (Eds.)

Proceedings

International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2010)

and

First International Workshop on Evolution Support for Model-Based Development and Testing (EMDT2010)

part of the umbrella conference
55th International Scientific Colloquium (IWK2010)

Ilmenau, Germany, September 13 and 16, 2010

EMDT2010 in Coorporation with the SIG OOSE of the German Computer Society GI and with ACM/ SIGSOFT

Editors' addresses:

Martin Atzmüller
University of Kassel
Knowledge and Data Engineering Group
Wilhelmshöher Allee 73
34121 Kassel, Germany
atzmueller@cs.uni-kassel.de

Rainer Knauf
Ilmenau University of Technology
Chair of Artificial Intelligence
P.O. Box 100565
98684 Ilmenau, Germany
rainer.knauf@tu-ilmenau.de

Stephan Bode, Qurat-Ul-Ann Farooq, Matthias Riebisch
Ilmenau University of Technology
Department of Software Systems/Process Informatics
P.O. Box 100565
98684 Ilmenau, Germany
{stephan.bode | qurat-ul-ann.farooq | matthias.riebisch}@tu-ilmenau.de

Contents

Part I

International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2010)

Preface

Welcome to the International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS 2010), September 13th, 2010, Ilmenau, Germany.

Evaluation, Verification, Validation and Refinement of Intelligent systems have been an important issue from the very beginning of their applications. These issues were an important research area and engineering aspect in the 80's and 90's. A number of conceptual approaches as well as practical tools were developed then.

With time the focus of research in the design of intelligent systems moved away from these topics, towards knowledge representation, discovery and processing, the Semantic Web technologies, and a number of other AI-inspired areas. However, recently a number of researchers have realized that the lack of systematic methods and formal techniques for the design, evaluation and refinement is often an important reason for limited applications of even mature intelligent systems. Therefore, there is a growing need to reconsider some of the basic issues in this field. Today, in fact, the classic approaches to the Design, Evaluation, Verification, Validation and Refinement have to be assessed from the new perspectives in order to transfer their principles to new approaches and application fields. The practical design issues are of prime importance. The integration of Intelligent Systems with mainstream technologies and related design approaches from other areas, e.g., from Software Engineering, from Machine Learning, or from the Social Sciences, is especially important. The quality issues need to be considered as early as possible during the Design phase of the system.

The goal of the workshop was to promote and further a community-wide discussion of ideas that will influence and foster continued research concerning the topics of Design, Evaluation, and Refinement, as well as attract new researchers to the field. The objective was to focus on the contributions in the above fields and to provide an environment for communicating different paradigms and approaches, thus hopefully stimulating future cooperation and synergistic activities.

DERIS2010

The proceedings contain the papers presented at DERIS 2010 held on September 13th, 2010 in Ilmenau, Germany. In total, we received 10 submissions, from which we were able to accept seven submissions based on a rigorous reviewing process, as regular research papers. Each submission was reviewed by at least 2 program committee members.

The topics of interest of the DERIS workshop series were mainly located in the area of Design, Evaluation, Verification, Validation, and Refinement and include but are not limited to:

- Principles in knowledge systems and ontology design
- Detecting and handling inconsistencies and other anomalies within knowledge bases
- Fundamentals and formal methods for verification of AI systems
- Fundamentals and formal methods and techniques of validity assessment of AI systems, AI principles, and intelligent behavior in general
- Special approaches to verify and/or validate certain kinds of AI systems: Rule-based, case-based
- Special approaches or tools to evaluate systems of a particular application field
- Knowledge base refinement by using the results of evaluation
- Development and evaluation of ontologies
- Maintenance and evolution of knowledge systems and ontologies
- Explanation in the context of evaluation and assessment
- Problems in system certification
- Ontology and knowledge capture
- Design and evaluation issues in automatic knowledge capture and knowledge discovery
- Design and evaluation of semantic web applications and systems
- Formal methods in verification and evaluation of intelligent systems
- Case studies in design and evaluation and the lessons learned

The organizers would like to thank all who contributed to the success of the workshop. We thank the authors for their submissions, and especially thank the Program Committee for their good work in carefully reviewing and collaboratively discussing the submissions. For the submission and reviewing process we used the Easy-Chair system, for which the organizers would like to thank Andrei Voronkov, the developer of the system.

September 2010

Martin Atzmüller
Rainer Knauf

Organizing Committee

Martin Atzmüller, Knowledge and Data Engineering Group, University of Kassel, Germany
Rainer Knauf, Artificial Intelligence Group, TU Ilmenau, Germany

Program Committee

M. Atzmüller, University of Kassel, Germany
J. Baumeister, University Würzburg, Germany
S. Gaudl, Fraunhofer IDMT, Germany
A. Gonzalez, University of Central Florida, Florida, USA
K. P. Jantke, Fraunhofer IDMT, Germany
R. Knauf, TU Ilmenau, Germany
A. Ligêza, AGH UST Krakow, Poland
G. J. Nalepa, AGH UST, Krakow, Poland
Th. Roth-Berghofer, DFKI GmbH, Germany
D. H. Sleeman, University of Aberdeen, United Kingdom

VALIDATION OF MIXED-STRUCTURED DATA USING PATTERN MINING AND INFORMATION EXTRACTION

Martin Atzmueller

University of Kassel
Knowledge and Data Engineering
Kassel, Germany
atzmueller@cs.uni-kassel.de

Stephanie Beer

University-Hospital of Würzburg
Gastroontologics Research Group
Würzburg, Germany
beer_s@klinik.uni-wuerzburg.de

ABSTRACT

For large-scale data mining utilizing data from ubiquitous and mixed-structured data sources, the appropriate extraction and integration into a comprehensive data-warehouse is of prime importance. Then, appropriate methods for validation and potential refinement are essential. This paper presents an approach applying data mining and information extraction methods for data validation: We apply subgroup discovery and (rule-based) information extraction for data integration and validation. The methods are integrated into an incremental process for continuous validation options. The results of a medical application demonstrate that subgroup discovery and the applied information extraction methods are well suited for mining, extracting and validating clinically relevant knowledge.

1. INTRODUCTION

Whenever data is continously collected, for example, using intelligent documentation systems [1], data mining and data analysis provide a broad range of options for scientific purposes. The mining and analysis step is often implemented using a data-warehouse [2, 3, 4]. For the data preprocessing and integration of several heterogenous sources, there exist standardized extract-transform-load (ETL) procedures, that need to incorporate suitable data schemas, and integration rules. Additionally, for unstructured or semi-structured textual data sources, the integration requires effective information extraction methods. For clinical discharge letters, for example, the structure of the letter is usually non-standardized, and thus dependent on different writing styles of different authors.

However, a prerequisite of data mining is the validation and the quality assurance of the integrated data. Especially concerning unreliable extraction and integration methods, the quality of the obtained data can vary significantly. If the data has been successfully validated, then the trust in the data mining results and their acceptance can be increased.

In this paper, we propose an approach for the validation of mixed-structured data using data mining and information extraction and propose appropriate refinement options. We focus on a data mining technique for mining *local patterns*, i.e., subgroup discovery, e.g., [5, 6, 7] that are especially suitable for the task: Local patterns consider local regularities (and irregularities) of the data and are therefore useful for spotting non-expected, contradicting, and otherwise unusual patterns potentially indicating problems and errors in the data.

Concerning the information extraction techniques, we consider popular methods implemented in the UIMA [8] and ClearTK [9] framework, and especially focus on the TEXTMARKER system, e.g., [10, 11] for rule-based information extraction. Rules are especially suitable for the proposed information extraction task since they allow a concise and declarative formalization of the relevant domain knowledge that is especially easy to acquire, to comprehend and to maintain. Furthermore, in the case of errors, the cause can easily be identified by tracing the application of the individual rules.

The combined approach enables data mining from heterogenous sources. The user can specify simple rules that consider features of the text, e.g., structural or syntactic features of the textual content. We focus on an incremental level-wise approach, such that both methods can complement each other in the validation and refinement setting. Furthermore, validation knowledge can be formalized in a knowledge base, for assessing known and expected relations in the data.

The approach has been implemented in a clinical application for mining data from clinical information systems, documentation systems, and clinical discharge letters. This application scenario concerns the data integration from heterogenous databases and the information extraction from textual documents. The experiences and results so far demonstrate the flexibility and effectiveness of the presented approach that make the data mining and information extraction methods suitable components in the mining, validation and refinement process.

2. BACKGROUND

In the following, we shortly summarize the methods for data mining and information extraction, subgroup discovery, and rule-based information extraction using TEXTMARKER.

2.1. Subgroup Discovery

Subgroup discovery is a flexible data mining method for discovering local patterns that can be utilized for global modeling in the context of exploratory data analysis, description, characterization and classification.

Subgroup discovery is applied for identifying relations between a (dependent) target concept and a set of explaining (independent) variables. Then, the goal is to describe subsets of the data, that have the most unusual characteristics with respect to the concept of interest given by the target variable [6]. For example, the risk of coronary heart disease (target variable) is significantly higher in the subgroup of smokers with a positive family history than in the general population.

In the context of the proposed validation approach, we consider certain gold-standard concepts as targets, as well as target concepts that are true, if and only if equivalent concepts from two different sources match. Then, we can identify combinations of factors that cause a mismatch between the concepts. These combinations can then indicate candidates for refinement.

2.2. Rule-based Information Extraction

Information extractions aims at extracting a set of *concepts*, *entities* and *relations* from a set of documents. TEXTMARKER [10, 11] is a robust system for rule-based information extraction. It can be applied very intuitively, since the used rules are especially easy to acquire and to comprehend. Using the extracted information, data records can be easily created in a post-processing step. Humans often apply a strategy according to a *highlighter metaphor* during 'manual' information extraction: First, top-level text blocks are considered and classified according to their content by coloring them with different highlighters. The contained elements of the annotated texts segments are then considered further. The TEXTMARKER [10, 11] system tries to imitate this manual extraction method by formalizing the appropriate actions using *matching rules*: The rules mark sequences of words, extract text segments or modify the input document depending on textual features.

TextMarker aims at supporting the knowledge engineer in the rapid prototyping of information extraction applications. The default input for the system is semi-structured text, but it can also process structured or free text. Technically, HTML is often the input format, since most word processing documents can be obtained in HTML format, or converted appropriately.

3. THE MINING AND VALIDATION PROCESS

Figure 1 depicts the process of validation and refinement of mixed-structured data using pattern mining and information extraction methods. The input of the process is given by data from heterogenous data sources, and by textual documents. The former are processed by appropriate data integration methods adapted to the different sources. The latter are handled by information extraction techniques, e.g., rule-based methods that utilize appropriate extraction rules for the extraction of concepts and relations from the documents. In general, a variety of methods can be applied.

The process supports arbitrary information extraction methods, e.g., automatic techniques like support-vector machines or conditional random fields as implemented in the ClearTK [9] toolkit for statistical natural language processing. However, the refinement capabilies vary for the different extraction approaches: While black-box methods like support vector machines or conditional random fields only allow an indirect refinement and adaptation of the model, i.e., based on adapting the input data and/or the method parameters for constructing the model, a white-box approach implemented using rules provides for a direct modification of its model, namely the provided rules. Therefore, we especially focus on rule-based methods due to their rich refinement capabilities.

After the integration and extraction of the data, the result is provided to the pattern mining system which obtains a set of validation patterns as output. This set is then checked both for internal consistency and compared to formalized background knowledge. In the case of discrepancies and errors, refinement are proposed for the data integration and/or the information extraction steps. After the rules have been refined, the process iterates with the updated schemas and models.

In the following we discuss exemplary results obtained from a medical project. We applied data collected by the SONOCONSULT system, a multifunctional knowledge system for sonography, which has been in routine use since 2002 documenting more than 12000 patients in two clinics. The system covers the entire field of abdominal ultrasound (liver, portal tract, gallbladder, spleen, kidneys, adrenal glands, pancreas, intestine, lymph nodes, abdominal aorta, cava inferior, prostate, and urinary bladder). The data was integrated with the SAP-based i.s.h.med system, and the information extraction techniques were applied for textual discharge letters from the respective patients; SONOCONSULT was used for documentation. By integrating different data sources into the warehouse it is possible to measure the conformity of sonographic results with other methods or inputs. In our evaluations, we applied computer-tomography diagnoses and additional billing diagnoses (from the hospital information system) as a gold-standard.

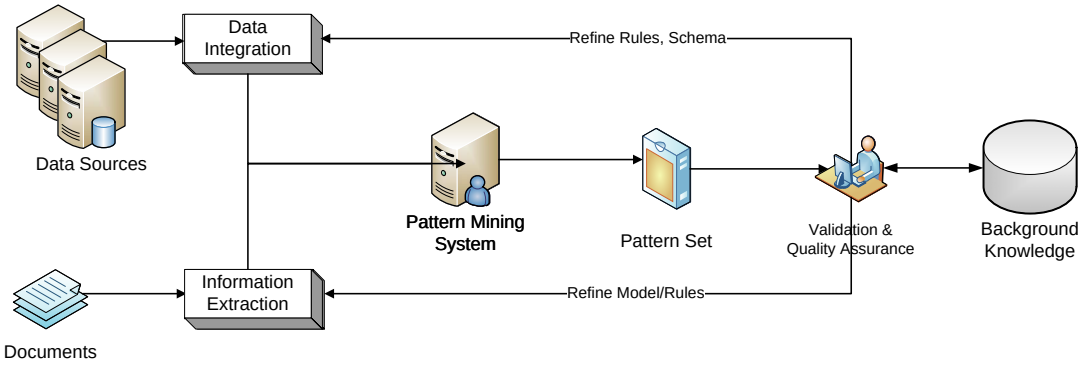


Fig. 1. Process Model: Validation of Mixed-Structured Data using Pattern Mining and Information Extraction

Table 1 shows the correlation of SONOCONSULT based diagnosis with CT/MR, diagnoses listed in the discharge letter and diagnoses contained in the hospital information system for a selection of cases from a certain examiner. It was quite interesting that the conformity between SONOCONSULT based diagnoses with the diagnoses contained in the hospital information system was relatively low. Evaluating this issue it was obvious that various diagnosis were not listed in the hospital information system because they were not revenue enhancing and not relevant for all clinical situations. Therefore, we looked at the accordance with the discharge letters which were found to be highly concordant at least for the diagnosis of liver metastasis. Liver cirrhosis is more awkward to detect using ultrasound and has to be in a more advanced stage. Therefore, some of the discharge diagnoses "liver cirrhosis" were only detected using histology or other methods.

In some cases, there are discrepancies with respect to the formalized background knowledge that still persist after refinement of the rules and checking the data sources. In such cases, explanation-aware mining and analysis components provide appropriate solutions for resolving conflicts and inconsistencies. By supporting the user with appropriate justifications and explanations, misleading patterns can be identified, and the background knowledge can be adapted. The decision whether the background knowledge needs to be adapted is performed by the domain specialist. As we have described in [12] there are several continuous explanation dimensions in the context of data mining and analysis, that can be utilized for improving the explanation capabilities. In the medical domain, for example, patterns are usually first assessed on the abstract level, before they are checked and verified on concrete patient records, i.e., on a very detailed level of abstraction. Then, discrepancies are modeled in the background knowledge, for example, certain exception conditions for certain subgroups of patients.

The validation phase is performed on several levels: On the first level, we can use a (partial) gold-standard both for checking the data integration and information extraction tasks. We only require a partial gold-standard, i.e., a sample of the correct relations, because we need to test the functional requirements of the data integration and extraction phases. On the next level, we can incrementally validate the integrated data using the extracted information, or vice versa, using the mined patterns. In the case of discrepancies, we can rely on the partial gold-standard data for verification, or we can identify potential causes and verify these on concrete cases. Therefore, the final decision for the refinements relies on the user, which reviews all proposed refinements in a semi-automatic approach.

For the refinement steps, we can either extend the (partial) gold-standard, or we perform a boot-strapping approach, using a small gold-standard sample of target concepts for validation, e.g., for validating and refining the information extraction approach, which is in turn used for the validation of the data sources. In the next step, the validation targets can be extended and the process for refinement is applied inversely. The boot-strapping approach for validation and refinement is thus similar to the idea of co-training, e.g., [13] in machine learning that also starts with a small labeled (correct) dataset and iteratively adapts the models using another co-trained dataset.

4. CONCLUSIONS

This paper presented an approach for the validation of mixed-structured data using information extraction and pattern mining methods. In an incremental approach, data can both be validated and refined with an increasing level of accuracy. The presented approach has been successfully implemented in a medical project targeted at integrating data from clinical information systems, documentation systems, and textual discharge letters.

The experiences and results so far demonstrate the flexibility and effectiveness of the pattern mining and information extraction methods for the presented validation and refinement approach.

Total Case Number	SONO CONSULT Diagnoses	SAP Diagnoses	% Conformity with SONO CONSULT	CT/MR Diagnoses	% Conformity with SONO CONSULT	Discharge Letter Diagnoses	% Conformity with SONO CONSULT
Liver cirrhosis							
16	12	6	20	1	33	9	50
Liver metastasis							
28	16	11	65	15	87	17	94

Table 1. Exemplary study for a selection of cases concerning liver examinations performed by a certain examiner: Conformity of system diagnoses with various sources of diagnosis input. The columns indicate the degree of correlation of the different sources with SONOCONSULT diagnoses measured by the number of covered cases.

5. REFERENCES

[1] Frank Puppe, Martin Atzmueller, Georg Buscher, Matthias Huettig, Hardi Lührs, and Hans-Peter Buscher, "Application and Evaluation of a Medical Knowledge-System in Sonography (SonoConsult)," in *Proc. 18th Europ. Conf. on Artificial Intelligence (ECAI 2008)*, 2008, pp. 683–687.

[2] Jonathan C. Prather, David F. Lobach, Linda K. Goodwin, Joseph W. Hales, Marvin L. Hage, and W. Edward Hammond, "Medical Data Mining: Knowledge Discovery in a Clinical Data Warehouse," in *Proc. AMIA Annual Fall Symposium (AIMA-1997)*, 1997, pp. 101–105.

[3] Rüdiger Wirth and Jochen Hipp, "CRISP-DM: Towards a Standard Process Model for Data Mining," in *Proc. 4th Intl. Conf. on the Practical Application of Knowledge Discovery and Data Mining*. 2000, pp. 29–39, Morgan Kaufmann.

[4] Martin Atzmueller, Stephanie Beer, and Frank Puppe, "A Data Warehouse-Based Approach for Quality Management, Evaluation and Analysis of Intelligent Systems using Subgroup Mining," in *Proc. 22nd International Florida Artificial Intelligence Research Society Conference (FLAIRS), accepted*. 2009, pp. 372–377, AAAI Press.

[5] Martin Atzmueller, Frank Puppe, and Hans-Peter Buscher, "Exploiting Background Knowledge for Knowledge-Intensive Subgroup Discovery," in *Proc. 19th Intl. Joint Conference on Artificial Intelligence (IJCAI-05)*, Edinburgh, Scotland, 2005, pp. 647–652.

[6] Stefan Wrobel, "An Algorithm for Multi-Relational Discovery of Subgroups," in *Proc. 1st European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, Berlin, 1997, pp. 78–87, Springer Verlag.

[7] Willi Klösgen, "Explora: A Multipattern and Multistrategy Discovery Assistant," in *Advances in Knowledge Discovery and Data Mining*, Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padraic Smyth, and Ramasamy Uthurusamy, Eds., pp. 249–271. AAAI Press, 1996.

[8] David Ferrucci and Adam Lally, "UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment," *Nat. Lang. Eng.*, vol. 10, no. 3-4, pp. 327–348, 2004.

[9] P. V. Ogren, P. G. Wetzler, and S. Bethard, "ClearTK: A UIMA Toolkit for Statistical Natural Language Processing," in *UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC)*, 2008.

[10] Martin Atzmueller, Peter Kluegl, and Frank Puppe, "Rule-Based Information Extraction for Structured Data Acquisition using TextMarker," in *Proc. of the LWA-2008, Special Track on Knowledge Discovery and Machine Learning*, 2008, pp. 1–7.

[11] Peter Kluegl, Martin Atzmueller, and Frank Puppe, "Textmarker: A tool for rule-based information extraction," in *Proc. Biennial GSCL Conference 2009, 2nd UIMA@GSCL Workshop*. 2009, pp. 233–240, Gunter Narr Verlag.

[12] Martin Atzmueller and Thomas Roth-Berghofer, "Ready for the MACE? The Mining and Analysis Continuum of Explaining Uncovered," in *AI-2010: 30th SGAI International Conference on Artificial Intelligence. Accepted.*

[13] Avrim Blum and Tom Mitchel, "Combining Labeled and Unlabeled Data with Co-Training," in *COLT: Proceedings of the Workshop on Computational Learning Theory*. 1998, pp. 92–100, Morgan Kaufmann.

VALIDATION OF KNOWLEDGE-BASED SYSTEMS THROUGH COMMONKADS

Feras Batarseh

Intelligent Systems Lab
School of Electrical Engineering and Computer Science
University of Central Florida (UCF)
4000 Central Florida Blvd.
Orlando, FL 32816

Avelino J. Gonzalez

Intelligent Systems Lab
School of Electrical Engineering and Computer Science
University of Central Florida (UCF)
4000 Central Florida Blvd.
Orlando, FL 32816

Rainer Knauf

Ilmenau University of Technology
Department of Computer Science and Automation
P.O.Box 100565
Ilmenau, Germany 98684

ABSTRACT

This paper defines a method that can be used for validating knowledge-based systems (KBS) throughout their entire lifecycle. Method's name is MAVERICK. It stands for Method for Automated Validation Embedded into the Reusable and Incremental CommonKADS. The lack of suitable, rigorous and general validation methods has become a serious obstacle to user acceptance of KBS for critical applications. In spite of recent significant advances in validation of KBS, it still remains an open problem. The ideas presented in this paper are based on the concept that validation should be performed in a structured and guided manner, integrated within a knowledge-based systems' lifecycle development method.. We define an incremental validation method for KBS based on extracting test cases from CommonKADS. Furthermore, we introduce our method for reducing the number of test cases and thus reducing validation's effort and cost.

Index Terms - Validation, CommonKADS, Knowledge-based systems, Test case.

1. INTRODUCTION

This paper describes a method that integrates validation within a life-cycle development method. The most comprehensive definition of validation was recently introduced by Gonzalez et al. [1] in the context of knowledge-based systems: "Validation is the process of ensuring that the output of the intelligent system is equivalent to those of human experts when given the same input." We adopted this definition because it is general and because it states that validation is comparing the system to the real world. Different methods for the validation of knowledge bases have been developed such as BKB [2], VKB [3], KVAT [4], SEEK and SEEK2 [5]. Furthermore, methods for system validation were developed, such as Bi-directional many-sided explanation typed multi-step validation, VESA [3], CORUS [6], CASE VALIDATOR [7], KJ3 [8], VVR [4] and quasi-exhaustive set validation [9]. Additionally, other multi-purpose validation tools were developed such as SHIVA, DIVER, EITHER and EMBODY [10]. None of these methods is fully incorporated into a life-cycle model.

2. BACKGROUND

Validation can and should be performed at any and all levels of the system development stages [1] [11]. O'Keefe et al. [11] and Lee et al. [12] have looked into incorporating validation into a conceptual software development model. However, their success was limited. After working with different general validation approaches, O'Keefe et al. [11] concluded that "We should build validation into the development cycle". However, none of the existing methods perform formal validation across all development phases. Furthermore, none of the mainstream methods presented here is completely based on a life-cycle model for system development. In this paper, we introduce a formal method towards achieving the goal of having a guided and incremental validation. This will be done through CommonKADS. Anderson et al. [13] conducted a study to measure the benefits of incremental validation using many systems in many domains. They came out with the following conclusions:

1. Rates of uncovering errors early in development were better.

2. Validation and verification found 2.3 to 5.5 errors per thousand lines of code.
3. Over 85% of the found errors affect reliability and maintainability.
4. Early error detection saved 20-28% of validation costs if validation begins at coding phase.
5. Incremental validation saved 92-100% of validation costs if validation begins at requirement phase.

Gilb et al. [14] did a similar study and illustrated their results. They concluded that when validation is postponed, costs will grow exponentially. Incremental validation can prevent this increase in costs. Incremental validation helps the user in getting frequent information about the development process of the system, helps the knowledge engineer in finding early comprehensive solutions instead of rushing fixes to meet deadlines and helps the manager in decision-making and instant feedback.

3. COMMONKADS SET OF MODELS

CommonKADS (Knowledge Acquisition and Design Support) is based on KADS. It concentrates on the conceptual structure of the knowledge and the system. The most accepted KBS development method is CommonKADS. It doesn't currently include guidelines for validation, verification or testing in any of its models. The six CommonKADS models are categorized in three groups [15]:

1. Context Models:

Organization model: Supports the description and the analysis of the organization.

Task model: Describes the tasks that might be performed by the system within the organization.

Agent model: Supports the capabilities, constraints and roles of the agents performing the tasks.

2. Concept Models:

Knowledge/Expertise model: Supports the description of the knowledge invoked in the tasks.

Communication model: Describes the relation between the agents, their interaction and their communication.

3. Artifact Models:

Design Model: Supports the design and the structure of the system.

Figure 1 illustrates the CommonKADS set of models. These models are presented in worksheets, UML diagrams, pseudo code and text. All the models are mapped to implementation to form the system. Tools were developed to help in implementing CommonKADS such as Model-K and OMOS [15]. The development of these and other tools reflects the general acceptance of CommonKADS by the KBS development community. Conceptual model languages had been introduced to support CommonKADS representation formally such as ML^2, VITAL and FORKADS [15]. CommonKADS supports reusability, and offers guidelines for the developer to achieve high quality systems. CommonKADS is a knowledge representation dependent model and was not created independently from other software models. Rather, other software models (e.g. object-oriented paradigm) influenced the definition of CommonKADS. CommonKADS has a powerful organizational sub-model that can represent many domains. CommonKADS offers a *de facto* standard for building systems and ensures a modular approach. CommonKADS is the most used knowledge-based systems lifecycle model and is the most accepted [15] [16]. Considering all the advantages of CommonKADS mentioned above, it should be no surprise that we chose it as our knowledge-based system development model for our validation method

In the next three sections, the validation lifecycle, test cases extraction and reduction are introduced.

4. MAVERICK

Incremental validation is based on the idea that "prevention is better than cure". Incremental validation locates the problem in its early stages. For example, if there is an error that is created during knowledge elicitation as a result of miscommunication between the expert and the knowledge engineer, incremental validation helps in identifying the error before it's absorbed into the design and then the implementation. The deeper this error is absorbed the harder it will be to identify it. Therefore, based on the CommonKADS structure, MAVERICK is performed at five levels in the following order:

1. Context Test Cases Extraction: This step defines the test cases that need to be executed after defining the first three models (the Context models: Organization, Task and Agent).
2. Analysis Test Cases Extraction: In this step, the test cases are extracted from the communication and knowledge models. In CommonKADS, the analysis phase is done after building five models: organization, task, agent, communication and knowledge. These five models represent all the requirements of the system. After those five models are defined and before moving into the design model, analysis validation is performed. Inspection validation starts here, first step of inspection validation is analysis validation. This validation checks for conflicting requirements, missing aspects in the analysis and any ambiguities. This validation is performed by the experts and the knowledge engineer manually on all the documents and diagrams defined so far.

3. Design Test Cases Extraction: This is the last step for test case extraction where test cases are extracted from the design model. Inspection validation stops here, second step of inspection validation is design validation. It is performed after this step and before implementation of the knowledge-based system starts. Validation inspects the Class diagrams for DM1 to check the initial design. DM1 represents the whole system.
4. Spiral System Implementation: Implementation of the system is performed iteratively. While iterating, system development proceeds and validation is performed by executing test cases. Test cases are selected in every iteration by the CBV tool described later in this dissertation.
5. Spiral System Validation: Validation is performed spirally, test case selection occurs iteratively and test cases are executed on the system. The validation approach is discussed and introduced in greater detail in section 6. Steps 4 and 5 are indicated to as CBV.
Figure 1 illustrates our general approach towards performing incremental validation within the CommonKADS steps. Different validation steps are performed during the building of the CommonKADS models and the system.

5. COMMONKADS TEST CASE EXTRACTION

The test case extraction starts early, while defining the Organization model. The first worksheet from which to extract cases from is OM3. OM1, OM2 and OM4 are used to introduce the knowledge engineer to the process that needs to be developed into the knowledge-based system and the assets of the organization. Nothing from OM1, OM2 and OM4 is used as a part of the target system. OM3 is the process break down sheet. All the processes in OM3 breakdown into the Task model for more details. In this sheet, each task is defined with who is performing it and what part of knowledge is needed for it. This worksheet doesn't involve the essence of the task. That's the goal of the Task model. Example: Task1 is performed by Paul Hewson and for this task documents 1 and 2 are needed. When the system is built, a test case would be necessary to check the availability of the needed documents when this task is performed by the mentioned employee. The test case would have the following format:
1. Test case ID: *1*.
2. CommonKADS model: *Organizational model (worksheet: OM3 (organization tasks))*.
3. Input variables: *Paul Hewson's user name and password.*
4. Test setup values: *Logout from all accounts and close all documents.*
5. Test execution steps: *Run task 1 by clicking on the "start task" button, log in as Paul Hewson* and *click on "get documents 1 and 2"*
6. Expected solution: *Two PDF files opening on your computer with documents 1 and 2.*
7. System's solution: *Document 1 opened but document 2 didn't.*
8. Local Importance: *2.5.*
9. Number of execution times: *1.*
10. Informal description: *Paul Hewson needs access to documents 1 and 2 with task 1.*

OM2 has a "culture and power" part in the worksheet that deals with social issues, political constraints and rules of thumbs at the organization. This part doesn't apply to many organizations, but in case it's necessary, then for every point in this part of the worksheet there should be test cases to cover it.

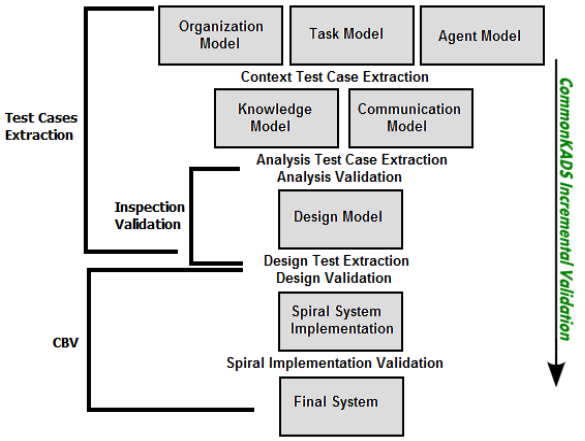


Figure 1 MAVERICK

An important part where test cases are to be extracted is the worksheet TM1. TM2 deals with making the knowledge engineer familiar with assigning tasks to knowledge. It won't be used for test case extraction. In worksheet TM1 however, each task is likely to need a number of test cases, where the inputs of the test case come from the dependency and flow section. In this section, the input objects and the output objects are defined, which are then transformed to the input variables and the test setup values of the test case. In the expected output part of the test case format, the quality and performance part are used. The quality and performance part in the worksheet deals with expected outcome of the task; this would be the criteria for the test case failure or success. Furthermore, in TM1, one part discusses the preconditions and the post conditions of the task. For each condition a set of test cases should be defined. Worksheet AM1 defines the agents' access to the system. Test cases extracted from this worksheet are related to security, roles and accesses. As previously introduced in test case 1 example, Paul Hewson needed access to task 1. Similar test cases are extracted from AM1. The Knowledge model is a critical model in CommonKADS as it is transformed to represent the knowledge base. In CommonKADS, the inference structure and the domain schemas

provide the set of test cases to validate the knowledge. The inferences and the transfer functions are parts of the inference structure, each instance of them is presented in a test case. KM1 is a central worksheet for test case extraction as it defines important parts of the knowledge. The knowledge engineer might need to present some domain requirements in the domain schemas of the Knowledge model, as every object in the domain schema is presented by a test case (refer to test case 2 for an example). In KM1, an important part is the "scenarios" section where any scenario related to a certain part of the knowledge is introduced. Other parts in this worksheet include a glossary of terms, the elicitation material and other sections that will not be transformed into a knowledge-based system. An example of a scenario and a test case: scenario (The employee Dave Evans needs knowledge about credit cards overdraft fees to answer a bank's client). A test case for this scenario would be:

1. Test case ID: *2*.
2. CommonKADS model: *Knowledge model (worksheet: KM1)*.
3. Input variables: *Dave Evans user name and password.*
4. Test setup values: *Run the credit card sub-system.*
5. Test execution steps: *log in as Dave Evans*, *enter a clients name and account number*, *click on "Display credit cards fees rules"*
6. Expected solution: *Correct overdraft fees list of rules should display to employee Dave Evans.*
7. System's solution: *Correct overdraft fees list of rules displayed to employee Dave Evans.*
8. Local Importance: *1.75.*
9. Number of execution times: *1*.
10. Informal description: *Overdraft fees rules display when required by the employee.*

The Communication model defines the interaction between the tasks, the agents and the system. CM1 and CM2 are used for test case extraction as both of these worksheets components are built into the targeted knowledge-based system. In CM1 the constraints section is used to extract test cases and the agents involved in this test case.

CM2 defines the contents of the communication messages and the control over the messages, each transaction needs to be tested using at least one test case. In the Communication model, all the information exchange, message sending and processes between agents are represented in a pseudo code defined specifically for CommonKADS.

For each pseudo construct, a set of test cases should be defined. For example, a message for a new loan is to be sent from the teller Adam Clayton to the management department employee Larry Mullen indicating that a new loan is granted to a client ahs the following construct: *SEND tramsaction1(loan granted) from teller to RECEIVE management*.

The dialogue diagram in the Communication model is used to test the sequence of the tasks performed by the system and the agents. The Design model in CommonKADS represents the initial design of the targeted system. DM2, DM3 and DM4 are worksheets that help the knowledge engineer to select the hardware platform, software platform and all technical issues related with building the system, but the real system design is found in DM1. DM1 defines all the subsystems. Test case extraction from this worksheet targets the issue of the integration of those subsystems. Relation between the subsystems is reflected by communication between the subsystems and the tasks sequencing among subsystems. In all the subsystems, the domain specifications are introduced in the Organizational, Task and Agent models. The functional specifications are presented in the Knowledge and Communication models. Using the test case extraction step defined in this section, all the aspects of the knowledge-base are covered and test cases are generated from all the entities included in the targeted system.

6. TEST CASE REDUCTION (CONTEXT BASED VALIDATION)

In our method, Knowledge-based system development and validation are performed using the spiral model. At any iteration of development, variables' values need to be modified and the system undergoes refinement. This work reduces the number of test cases based on the user's needs and the context of validation. This is where the term context-based validation (CBV) came from. In problem solving, the context would inherently contain much knowledge about the situation's context in which the problem is to be solved or the problem's environment [17]. In the case of test case reduction, testing is intensified for the model that failed the most in the previous testing cycle. To reduce the number of test cases, the knowledge engineer chooses what test cases to remove. This is not performed manually; it is performed spirally by the knowledge engineer and based on the CommonKADS models.

Before the knowledge engineer starts with system implementation, it is necessary to define a number of control variables that are used to select what test cases to be used in every cycle. These variables are:

1. Local Importance (LI): Each test case is assigned a local importance variable that falls between 1 and 5. *Local importance = Average of (dependency + domain importance + criticality + occurrence)*. Local importance is a factor of dependency (Value assigned from 1-5), domain importance (Value assigned from 1-5), criticality (Value assigned from 1-5) and occurrence (Value assigned from 1-5). All the values are defined by the knowledge engineer and the expert. Additionally, the frequency of the task is indicated in TM2, this is the basis for defining the occurrence

factor. Dependency is in the nature of CommonKADS, the Design model depends on the Knowledge and Communication models, which depend on defining the task and the Agent models which are both based on the Organization model which is defined based on the knowledge elicitation. The Organization model has the lowest dependency rate (1) and the Design model has the highest dependency rate (5).

2. Model Weight (MW): Every CommonKADS model is assigned a weight after any iteration of development. Initially all the models have the same importance (MW is set to 5), but when the development starts, model weights will constantly change based on the outcomes of the test cases. The model weight values fall between one and ten. Model weight reflects the assurance level in testing for the CommonKADS models. When the assurance of all models reaches 10 and implementation is done, validation stops.

3. N: Represents the number of test cases to be selected in any iteration.

4. Global Importance (GI): This variable is used to decide what test cases to select in any iteration. *Global Importance = Local Importance * Model Weight.*

Approaches to test case reduction have varied between random, formal and informal. Using a well established model like CommonKADS provides a solid ground and an assurance that all the aspects of the system are covered, and that the test cases extracted using this method make sure that the system is well covered for tests. The steps of CBV presented in figure 2 are:

1. Extract test cases from the worksheets and diagrams. Set all the parameters defined previously. Assign each test case to a CommonKADS model
2. Assign *local importance* for each test case.
3. Set the size of test case subset: *N*, initially all the test cases that have global importance more than 20 (LI*MW = 4*5 = 20). All test cases with local importance of 4 or 5 needs to be selected, cases with 1, 2 and 3 importance are less important.
4. Set all models' *weights/assurance* to 5
5. Calculate *global importance* = local importance * model weight. Sort test cases according to global importance
6. Start implementation using the spiral model
7. At the end of the first iteration, select N number of test cases. From the ordered list pick test cases 1 to n.
8. Execute the test cases on the system, and record the results
9. Based on results for each CommonKADS model test cases, re calculate assurance for each model. Example: if 30% of test cases of a certain model went wrong, that model's assurance will be 7 using the following formula: *100 - (percentage of successful test case)/10*
10. Recalculate global importance of test cases and reorder
11. Refine system; go to next iteration (Manual)
12. *Flag* test cases with a positive outcome (not to be picked again unless a change to their status was made), flag test cases with unexpected outcomes (this is used to make sure that the test case is reselected before end of validation), select different test cases every next iteration
13. Stop when assurance of all models is equal to 10. Assurance of all models = average of all models assurances.

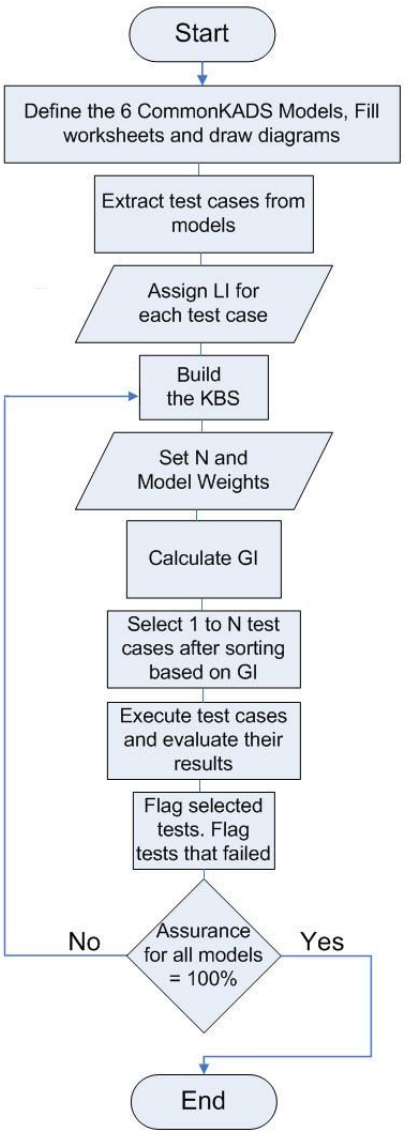


Figure 2 CBV

Test case reduction steps are illustrated in Figure 2.

A Java tool was developed to select, sort and recommend test cases for the knowledge engineer from the universal set of test cases using the method presented in this paper. Figure 3 is a screen shot that represents one panel from the seven panels in the tool. This tool updates the test cases instantly and sorts all the test cases in real time for selection of N test cases.

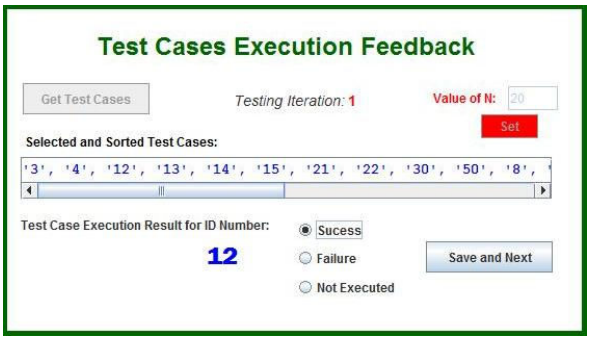


Figure 3 Test execution Java panel

7. CONCLUSIONS

The approach presented in this paper requires some manual work from the knowledge engineer or any other person performing validation but it has many advantages. Advantages of this approach are:

1. Flexibility: the weights and the models could be changed to any other values. This gives the knowledge engineer full control.
2. Usage-oriented: this approach is based on the user needs and a real time testing feedback. It is not a static function, rather a resilient one.
3. It's based on a comprehensive, well defined and well structured model: This function is based on CommonKADS, which as discussed previously, has many advantages.
4. Effort and time reduction: reducing the number of test cases reduces effort and time.

In this paper, we introduced a validation method based on a lifecycle model called CommonKADS; we introduced the validation lifecycle, extracting test cases from the six CommonKADS models and reducing the number of executed test cases and thus reduce time, manpower and expenses.

8. REFERENCES

[1] A.J. Gonzalez, and V. Barr, "Validation and Verification of Intelligent Systems – what are they and how are they different" Proceedings of the Journal of Experimental &Theoretical Artificial Intelligence, pp.407-420, 2000

[2] E. Santos Jr., and H. Dinh, "Consistency of Test Case in Validation of Bayesian Knowledge Bases", Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence – ICTAI 2004.

[3] R. Knauf, S. Tsuruta, and A.J Gonzalez, "Towards Reducing Human Involvement in Validation of Knowledge- Based Systems", Proceedings of the IEEE transaction on Systems, Man and Cybernetics, Volume 37, pp.120-131, January 2007

[4] N. Zlatareva and A. Preece, "State of the Art in Automated Validation of Knowledge-Based Systems", Proceedings of the journal of Expert Systems with Applications, pp.151-168, 1994

[5] A. Ginsberg. S. Weiss, and P. Politakis, "SEEK2: A Generalized Approach to Automatic Knowledge-base Refinement" Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), pp. 367-374, 1985

[6] K. Abdallah, T. Mohammad, and F. Louis, "Validation of Intelligent Systems: a Critical Study and a tool, Corus", Proceedings of the International Journal of Soft Computing, pp.191-198, 2007.

[7] S. Smith and A. Kandel, "Validation of Expert Systems" Proceedings of the Third Florida Artificial Intelligence Research Symposium (FLAIRS), pp.197-201, 1990

[8] Wu, C. and Lee, S. "KJ3- a tool assisting formal validation of knowledge-based systems", Proceedings of the Int. J. Human-Computer Studies, pp. 495-525, 2002.

[9] J. Herrmann, K. Jantke, and R. Knauf, "Using Structural Knowledge for System Validation" Proceedings of the 10th FALIRS Conference, pp. 82-86, 1997.

[10] S. Lockwood, and Z. Chen, "Knowledge Validation of Engineering Expert Systems" Proceedings of the Journal of Advances in Software Engineering, pp. 97-104, 1995.

[11] R. O'Keefe, R. Balci, and E. Smith, "Validating Expert System Performance" IEEE, Proceedings of the IEEE Expert, Volume 2, pp.81-90, 1987

[12] S. Lee, and R. O'Keefe, "Developing a Strategy for Expert System Validation and Verification" , IEEE, Proceedings of the IEEE Transaction on systems, Man and Cybernetics, Volume 24, pp.643-655, 1994.

[13] C. Anderson, T. Thelin, P. Runeson,N. Dzamashvili, "An Experimental Evaluation of Inspection and testing for Detection of Design Faults", Proceedings of the International Symposium on Empirical Software Engineering – ISESE 2003

[14] T. Gilb, and D. Graham "Software Inspection" Published by Addison Wesley 1993

[15] G. Shreiber, H. Akkermans, A. Anjewierden, R. De Hoog, N. Shadbolt, W. Van De Velde, and B. Wielinga,. "Knowledge Engineering and Management-The CommonKADS Methodology" published by The MIT Press 2000

[16] A. Al Korany, K. Shaalan, H. Baraka, and A. Rafea, "An Approach for Automating the Verification of KADS-Based Expert Systems" Proceedings of the 7th International Conference on Applied Informatics and Communications- (WSEAS), pp. 1-22, 2007

[17] A.J Gonzalez, B. Stensrud, and G. Barret, "Formalizing context-based reasoning: A modeling paradigm for representing tactical human behavior", Proceedings of the International Journal of Intelligent Systems, pp. 822-847, 2008

Composing Tactical Agents through Contextual Storyboards

Avelino J. Gonzalez[1], Rainer Knauf[2] and Klaus P. Jantke[3]

(1)Intelligent Systems Laboratory
School of EECS
University of Central Florida
Orlando, FL 32816-2362
USA

(2)Faculty of Artificial Intelligence
Technical University of Ilmenau
PF 10 05 65
98684 Ilmenau
Germany

(3) Fraunhofer IDMT
Children's Media Dept.
Hirschlachufer 7
99084 Erfurt
Germany

ABSTRACT

This paper presents the novel use of storyboards for composing, organizing and visualizing tactical agents designed to serve as computer generated forces. These tactical agents represent enemy forces that act and react to trainee actions and are specifically used here to populate military training scenarios. The tactical agents are based on the Context-based Reasoning human behavior representation paradigm. This application of storyboards facilitates the use and visualization of the contextual elements that make up the composed agents. The use of the approach is described and an informal qualitative evaluation is conducted.

1. INTRODUCTION

Preparing a simulation for a military training session can be a time-consuming process. First of all, training objectives must be expressed by the instructor. Secondly, a mission or task to be executed by the trainee(s) must be specified, and the accompanying environmental conditions must be defined and subsequently reflected in the simulation environment. Thirdly, if the training objectives call for the trainee(s) to be faced with a specific situation, the external entities with which the trainees interact must be designed such that they present that situation to the trainee correctly and at the appropriate time. When this requires the involvement of intelligent software agents, these must be integrated into the simulation in just the right manner to accomplish the desired objective. Planning and organizing the simulation-based training exercise to systematically include these three steps presents a significant problem for simulation-based training.

In recent times, the widespread reuse of standard, reusable scenarios has led to exercises becoming known in advance by the trainees, thereby negating the effect of built-in surprises and diminishing the effectiveness of the training session. This ultimately prematurely requires that new and expensive exercises be created. It would be ideal, therefore, if new training exercises could be easily custom-made for each group of trainees, but that they nevertheless would guarantee an equivalent learning experience for all trainees.

This leads us to the concept of assisted scenario generation for training simulations. While the selection and implementation of certain environmental effects such as weather, time and other such issues is relatively easy, depending on the facilities provided by the simulation infrastructure, others such as the behavior and plans of the external entities typically require much greater care. This is because these intelligent tactical agents could exhibit the wide range of behaviors typically used in these scenarios, thereby resulting in large and complex models. Their large size and high complexity make these agents difficult to build and possibly computationally expensive to run.

However, this is not the entire problem. The external entities are the primary means through which the scenario designer causes the desired situations to be presented to trainees at the right moment. These agents have to be able to react to the trainee actions and still be able to present the desired educational situation. In situations where the roles of the external entity are quick and of a short duration, it may not need to be artificially intelligent. An example of this could be a distracted pedestrian crossing the street in front of the car. In such cases, the model of the pedestrian is simple, as it needs no reaction. Selection and placement of such an external entity would be rather simple. However, for other roles that require extended contact with the trainee such simplicity may not suffice. Examples of this include a driver with road rage, a persistent enemy combatant, or a police officer pursuing a fleeing driver. A more complex process must be developed to assist the training session author in building the appropriate external entities and place them correctly within the simulation.

A tool that helps the session author design the training session – specially the agents used in the training session would be immeasurably helpful. Description of such a tool is our objective here.

2. OVERALL SOLUTION APPROACH

Planning has been a core part of AI research since the beginning. Planning is something that humans do naturally and for the most part, effectively. Many tools have been built to assist planners. We investigated the feasibility of using *storyboards*, as defined by Jantke and Knauf [3], to serve as the infrastructure upon which the agent models could be planned and stored.

The concept of storyboards has been used successfully for many years in many applications such as cinematography, theater, musicals and such time-based works. Storyboarding is a modern approach to planning that actually goes beyond conventional planning. It can be said to be the "… organization of experience" [3]. Jantke [4] asserts that when human activity comes into play (e.g., games, war) predicting the future situations becomes difficult because it is unknown what situation will be faced by the human in a conflict-based context. He maintains that storyboards provide room for such human activity by furnishing means to represent alternative worlds.

Knauf [6] and Knauf et al [7] more recently applied the storyboard concept to course design. They are specifically used to guide the didactic process in traditional learning environments and in e-learning.

The storyboard approach devised by Jantke & Knauf is built upon standard concepts which enjoy (1) *clarity* by providing a high-level modeling approach, (2) *simplicity*, which enables everybody to easily become a storyboard author, and (3) *visual appearance* as graphs. While other means of structuring the contents of the agents exist, such as state diagrams, Petri nets, etc., none meet the above three requirements as easily as does the storyboard tool described here.

Jantke and Knauf define their storyboard as a nested hierarchy of directed graphs with annotated nodes and annotated edges. Nodes can be either *scenes* or *episodes* where scenes denote leaves of the nesting hierarchy and represent a non-decomposable learning activity. A scene can be (1) the presentation of a (media) document, (2) the opening of any other software tool that supports learning (e.g., an URL and/or an e-learning system) or (3) an informal description of the activity. Episodes, on the other hand, denote a sub-graph. Graphs are interpreted by the paths through which they can be traversed. Edges denote transitions between nodes. Figure 1 shows a top-level storyboard that reflects an organization for teaching a college-level course in Artificial Intelligence.

The processes that are commonly represented through storyboarding are characterized by non-determinism, involvement of human players and the attempt to anticipate the behavior of these human players. These characteristics also apply to simulation-based training sessions. Therefore, we propose here to use this storyboard approach to represent the agent being composed for a session in a training simulation.

The agents themselves are defined in the *Context-based Reasoning (CxBR)* modeling paradigm. CxBR specifies that agents built through CxBR be composed of several *major contexts*, some accompanying *minor contexts* and definition of *transition criteria* between the major contexts. While it is *active*, a major context, together with possibly several minor contexts, controls the actions of the agent. When the situation changes so that the context has changed, a transition to a new active context is effected, with its attendant functions and knowledge taking over the control of the agent. Transition criteria determine when the situation calls for a new major context to be made active and the currently active major context to be de-activated. Only one major context can be active at any one time. We expect here that the major contexts will be defined and created a-priori and be available in some repository, providing a baseline behavior for the agent when it finds itself in the correct context. However, the transition criteria are very application-dependent, and must thus be specified carefully for each application. See Gonzalez et al [1] for details about CxBR.

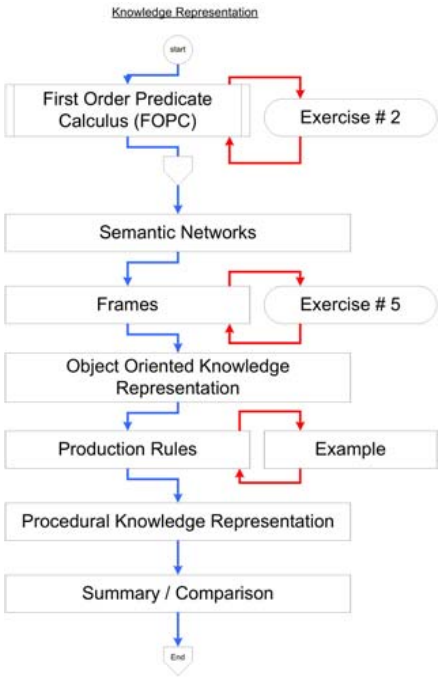


Figure 1 – Application of Story Boarding to Course Definition

We should note that the storyboard is not the agent. It merely helps a human to compose the agents for a specific scenario in a way that is clear, simple and easily visualized. The CxBR-based agents contain the intelligence and the ability to react to events in the simulation exercise.

The objective of the research was not to develop a working model of the tactical agents themselves, but rather to organize their definition in an easily-visualized and manoeuvrable tool. This is what we describe as composing agents from existing components, in our case, major and minor contexts. Our software tool provides a medium for the scenario storyboard to be reflected, provides an infrastructure to store the agent models for all situations, and can assist the session author with customizing the transition criteria for the major contexts vis-à-vis the training session. The storyboard, however, is not an agent representation paradigm. CxBR is the agent representation paradigm used. The storyboard merely helps in composing the agents from previously defined major contexts and easily visualizing the resulting agent. To better describe the concept, we introduce an example military scenario.

3. SPECIFIC SCENARIO USED

The training scenario used for this experiment involves a fictional maritime country (Blue state) with a lightly defended base in an island far off its mainland coast. This island is the subject of a territorial dispute between the Blue state and a neighbouring and also fictitious Red state. In light of current situations that may lead to potential hostilities with the Red state, the Blue state seeks to reinforce the defences on the island by sending a cargo vessel with supplies and armaments needed to enhance the defences of its island base.

This cargo vessel (M1) is escorted by a small task force composed of one anti-aircraft destroyer and flagship of the task force. This vessel is armed with SAM launchers, one torpedo tube and assorted guns. This is the vessel to be directly controlled by the trainees in this training exercise and it is labelled TF1. Three other warships make up this task force. Two anti-submarine frigates respectively labelled BF1 and BF2 come armed with anti-submarine rockets and assorted guns. The fourth warship is a mine layer, armed with mines and a 12.7 mm machine gun. It is labelled BF3. Their mission is to escort and protect the unarmed cargo vessel (M1) containing critical supplies and weapons from the mainland port to the naval base in the island in question. Their orders are to protect the cargo vessel and to confront any force threatening it, whether air, surface or subsurface. The Blue state ships are at the command of the TF1 commander, who can order them to take any action in accordance with the imposed rules of engagement.

Unbeknown to the trainee Blue force, a Red state force intends to land a heavily armed contingent in the island and capture it without a fight, given the light defences of the island base, and its long distance to the mainland. The invading Red force consists of three vessels, and they are labelled RF1, RF2 and RF3. RF2 and RF3 two are AEGIS-type anti-aircraft destroyers. Besides anti-aircraft missiles, they are armed with an assortment of guns. RF1 is a mother ship carrying three landing crafts that can be deployed from her hull. Each landing craft can carry a platoon-size unit with a light armoured vehicle or jeep with machine guns mounted on them. These landing craft are also armed each with one 12mm machine gun.

RF1 will seek to get close enough to the island on its north side so that it can launch the landing craft and land their forces. They are not aware of the Blue state convoy task force, the cargo vessel or its contents. The initial conditions of the developing situation are described in Figure 2 below. Each task force is not initially aware of the other. When the Red task force enters the Blue state's territorial waters, it is detected by an unarmed aerial surveillance aircraft (not shown), that monitors the waters surrounding the island, and continues to monitor the movements of the Red force. Without air or satellite assets, the Red force later discovers the presence of the Blue task force only when the latter gets within range of their ship-based radar. No other aircraft are relevant in this scenario.

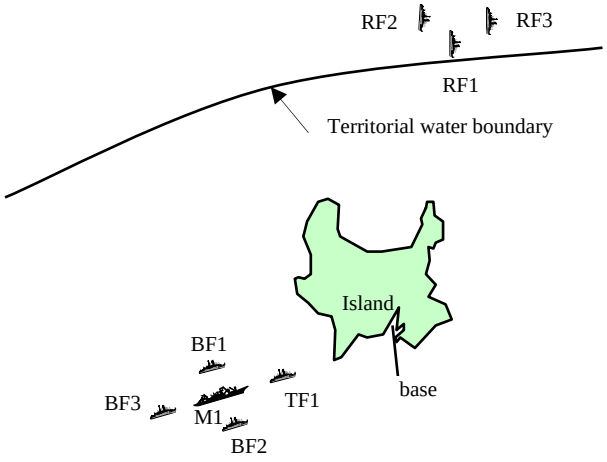


Figure 2 – Initial Conditions of Scenario

In the initial scenario, the Blue force is in a major context that calls for it to escort the cargo vessel. This means that the Blue task force is to sail at full speed toward its destination, maintaining close scrutiny of their sensors for the presence of threats, as the possibility of a Red force attack on the island has been considered a distinct possibility in the recent past. This major context in control is labelled **Escort** and it enforces a diamond shaped formation designed to protect the cargo ship from all directions. This major context looks for the possibility of transitioning to several other contexts, such as **Confront**, **Engage**, **Attack**, **Retreat** and **Dock,** among others.

The Red force, on the other hand, has as its objective to land undetected on the island's north shore which has good beaches for that purpose, deploy its forces and march overland to the base in the south end of the

island and take it through sheer intimidation, preferably without firing any shots. Its initial major context, while in international waters, is simply to navigate to certain coordinates. This major context is called **Transit,** and involves no special care other than to maintain navigational awareness and avoid collision with other objects as well as each other. Upon reaching the target coordinates, it is to transition to a more guarded form of navigation, where they get into a formation that is protective of the mother ship, and proceed in total radio silence, while at the same time in general quarters. This is the **StealthTransit** major context.

Planning in CxBR is carried out rather informally. Unlike other AI planning languages and systems, such planning is reflected merely by a sequence of major contexts with defined transition criteria. These plans are easily visualized via the storyboarding tool described here. The major contexts that compose the agent being built can also be easily described likewise, as can the minor contexts. For example, the plan to be initially followed by the Red force agents as a unit, in terms of a sequence of major contexts is shown below and pictorially in Figure 3.

Red Force: Transit ➔ StealthTransit ➔ Disembark ➔ Retreat ➔ Transit

It is somewhat more complicated for the Blue force. Upon detecting the Red force, the task force splits up and different tasks are assigned by the trainee force flagship (TF1). Thus, the ships do not behave uniformly as a unit as do the Red force ships. In other words, each member of the task force has different tasks to execute. So, we describe each ship individually below:

Blue Force TF1: Escort ➔ Confront ➔ Pursuit ➔ Transit
Blue Force BF1: Escort ➔ Confront ➔ Pursuit ➔ Transit
Blue Force BF2: Escort ➔ StandBy ➔ Confront ➔ Pursuit ➔ Transit
Blue Force BF3: Escort ➔ MineFieldApp ➔ StandBy ➔ MineRetrieval ➔ Rescue ➔Transit
Blue Force M1: Transit ➔ Dock

A full description of the scenario and the composition of the agents involved therein would exceed the page limits of this paper. The reader is referred to [2] for the full details of the scenario and its implementation.

4. MODEL ASSEMBLY WITH TOOL

The storyboard tool presents the availability to create *sheets*, where each of these sheets contains some logic related to the progression of the story. The sheets can contain *episodes*, *scenes* or *to-do* boxes. An episode contains a longer lasting series of actions or sub-actions. It can be composed of other episodes or of scenes. Episodes are depicted by rectangles with small notches at the left and right sides. As the name suggests, scenes contain more temporally short actions. Scenes are depicted by simple rectangles. They intuitively equate to major contexts and minor contexts respectively.

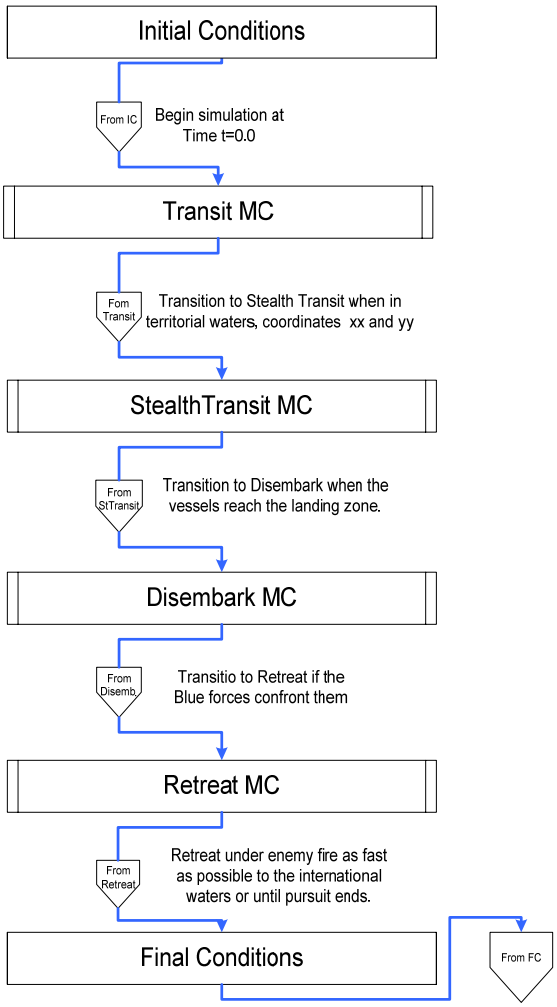


Figure 3 – Red Force Mission Plan

The storyboard tool is based on Microsoft Visio, with some custom-made functions and shapes to allow the free and easy movement among sheets. The main progression of the storyboard is reflected in the *Mission sheet*. This sheet is the plan for the agents that will participate in the scenario. In terms of CxBR, these represent the progression of major contexts to be executed by the agent being composed. These major contexts are represented as episodes in the mission sheet. The all-important transition criteria that triggers transitions between major contexts is found on the mission sheet, placed between the major context episodes.

Figure 3 depicts the Mission sheet for the Red Force in this scenario. The comments shown between each major context represents a textual description of the transition criteria. In the case where the rule language syntax for the system being used is known,

this comment could include the actual code for the transition rule.

Episodes and scenes have the ability to switch to other sheets that may contain an expansion of the elements found in the episode or scene. This provides the ability to quickly inspect a sub-context and its contents.

The storyboard begins with an initial condition and ends with a final condition shape. These shapes are scenes. Clicking twice on the initial conditions scene will take one to the initial condition sheet, which contains the same graph shown above as Figure 2. This is shown in Figure 4 below. The Initial Condition Sheet also refers to a document which describes the initial conditions in a narrative text. This document gives the scenario developer background information on the scenario to be created. Note in Figure 3 the text between the Initial Conditions Scene and the **Transit** major context episode in the mission sheet. This represents the transition to the major context. In this case, the transition is a simple one – commencement of the simulation, at t = 0.0.

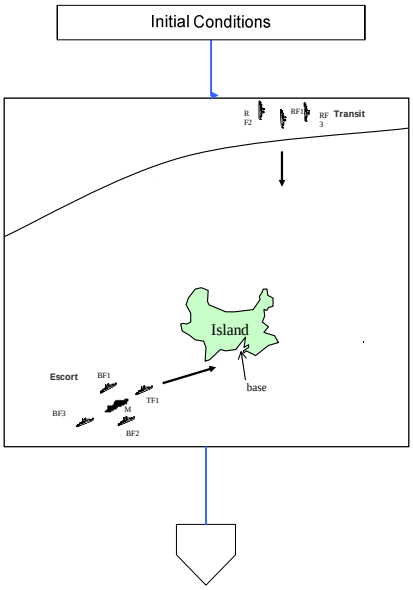


Figure 4 – Initial Conditions page

The funnel-looking pentagon shapes are return "worm holes", so to speak. They represent a way to quickly return the user to the page from which the sub-sheet was called. For example, when double-clicking on the **Transit MC** episode on the mission page, this takes one to the page where the details of the **Transit** major context are described. To return from there back to the mission page, the funnel shape is clicked and the return is executed. Figure 5 shows the Transit major context details. The two worm holes below the sub-contexts depict the return pipe from the respective sub-contexts **Navigate** and **AvoidCollision.** The worm hole below the entire graph is the return pipe to the Mission sheet.

A sub-context sheet is shown in Figure 6. This one in particular is that **Navigate** sub-context. This one is shown for a particular reason. One of the advantages of CxBR is the potential for reusability of lower-level contexts by several major contexts. One of those predictably re-used is the **Navigate** sub-context. It is called by the **Transit MC** and the **Retreat MC**. Conceivably, it is such an important function that it should be called by all major contexts. Once the control passes to the **Navigate** sub-context, a return should be executed to the major context that called it. The ability to remember which major context called it is not intrinsic in Visio, so several return worm holes must be created, one for returning to each of the various major contexts that may call it. While this puts the burden of remembering on the user, it nevertheless works well.

Lastly, an important part of a CxBR is the reactive context set. These major contexts are not included in the mission plan because their use is not expected in the plan. However, the behaviors represented within these reactive contexts could be useful if the mission does not go strictly according to plan (as they rarely ever do). Note that reactive major contexts are structurally similarly to those in the mission plan. It could be that a major context could be reactive in one mission but part of the plan in another. It just depends on the needs of the mission.

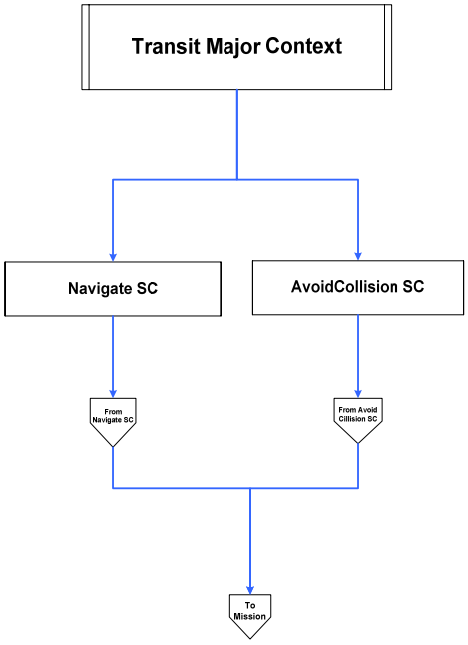


Figure 5 – Transit Major Context Page

The reactive major contexts are contained in a separate sheet called, appropriately enough, "Reactive Major Contexts". This sheet includes an episode for each major context whose activation could be possible in the course of this mission but not explicitly planned. These episodes have a link to its respective major context description page. These include links to the sub-contexts they call, just as was done for those major contexts included in the mission plan.

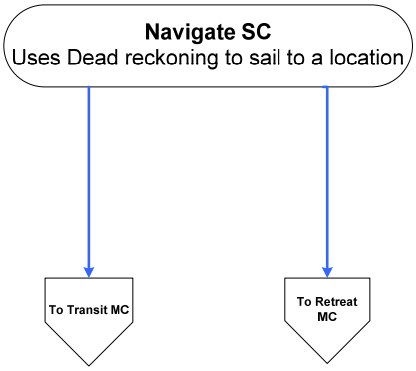


Figure 6 – Navigate Sub-Context Sheet with multiple Returns.

5. EVALUATION AND RESULTS

The tool was used to build the scenario for the intruder interception mission described above. No quantitative evaluation was done, as it is not a performance-oriented tool. Rather, a qualitative and rather informal evaluation was deemed to be the sensible alternative. This was judged by how long it took to learn to use the tool.

As part of this research, the first author used the tool for the first time after only having attended a few paper presentations by the second author, totalling approximately two hours of lecture. These presentations were in the context of the latter's research in didactic design, and not in building tactical agents for a simulation. Learning the use of the tool took approximately another two hours of working with it. This was done without documentation of the tool, other than reading the afore-mentioned papers. [3, 4, 5, 6, 7 and 8] However, it only took the first author a total of approximately 12 working hours to develop and organize the storyboard once he learned how to use the tool. This informal and qualitative evaluation shows that it is indeed an extraordinarily intuitive tool to learn to use, even without formal documentation.

The advantages of this tool go beyond the organization of the agent components. It is quite feasible to have the sheets included in the tool contain the actual source code for each component, such as the major contexts, the minor contexts and all functions that are to be included with the CGF model for the mission in question. The ability to attach files, although not extensively used in this particular work, can serve to attach source code files to each major context and sub-context.

6. SUMMARY

The research preformed here hypothesized that an existing storyboard tool, used previously for academic coursework organization and development, could be used to also define, organize and visualize military missions for the purposes of preparing training scenarios. The research consisted of defining a training scenario that would be typical of a military mission to teach trainees about tactics and doctrinal courses of action. Then, that scenario would be implemented in to the storyboard tool. The objective of the implementation was to gauge its applicability to simulation-based training. The results indicate that, after an informal evaluation, it does indeed satisfy the hypothesis that it would be a highly useful tool for this type of applications. While some improvements can be made to the tool vis-à-vis this type of application, it is useful as is, with only minor modifications made as part of this research.

7. REFERENCES

[1] A.J. Gonzalez, B.S. Stensrud and G. Barrett, "Formalizing Context-Based Reasoning - A Modeling Paradigm for Representing Tactical Human Behavior", International Journal of Intelligent Systems, Vo. 23, No. 7, pp. 822-847, July 2008

[2] A.J Gonzalez, "Composing Tactical Agents through Contextual Storyboards", Final Report, July 16, 2009. Unpublished, but available upon request.

[3] K.P. Jantke and R. Knauf, "Didactic Design though Storyboarding: Standard Concepts for Standard Tools", Proc. 4th Int'l Symp. on Information and Communication Technologies (ISICT) Workshop on Dissemination of e-Learning Technologies and Appl., Cape Town, South Africa, pp. 20-25, Jan. 2005

[4] K.P. Jantke, "Why Storyboarding? Why not Planning?" Computer Methods & Syst., Krakow, Poland, Nov. 2009.

[5] K.P. Jantke, R Knauf and, A.J. Gonzalez, "Storyboarding for Playful Learning", Proc. of World Conf. on E-Learning in Corporate, Government, Healthcare, and Higher Education 2006 *(E-Learn 2006)*, Honolulu, Hawaii.

[6] R. Knauf, "Storyboarding - An AI Technology to Represent, Process, Evaluate, and Refine Didactic Knowledge", Proc. of the Knowledge Media Technologies. First Core-to-Core Workshop, Dagstuhl Castle, Germany, pp. 170-179, 2006.

[7] R. Knauf, Y. Sakurai and S. Tsuruta, "Toward Making Didactics a Subject of Knowledge Engineering", Proc. of the 7th IEEE International Conf. on Advanced Learning Technologies, Niigata (Japan), pp. 788-792, 2007.

[8] S. Dohi, Y. Sakurai, S. Tsuruta and R. Knauf, "Managing Academic Education through Dynamic Storyboarding", Reeves, T.C. & Yamashita, S.F. (Eds.) Proc. of the World Conf. on e-Learning in Corporate, Government, Healthcare, & Higher Education 2006, October 13-17, 2006.

Rule Modularization and Inference Solutions – a Synthetic Overview

Krzysztof Kaczor and Szymon Bobek and Grzegorz J. Nalepa

Institute of Automatics,
AGH University of Science and Technology,
Al. Mickiewicza 30, 30-059 Kraków, Poland

ABSTRACT

Rule–based expert systems proved to be a successful AI technology in a number of areas. Building such systems requires creating a rulebase, as well as providing an effective inference mechanism that fires rules appropriate in a given context. The paper briefly discusses main rule inference algorithms Rete, TREAT and Gator. Since large rulebases often require identifying certain rule clusters, modern inference algorithms support inference rule groups. In the paper the case of the new version of Drools, introducing the RuleFlow module is presented. These solutions are contrasted with a custom rule representation method called XTT2. It introduces explicit structure in the rulebase based on decision tables linked in an inference network. In this case, the classic Rete–based solutions cannot be used. This is why custom inference algorithms are discussed. In the paper possible integration of the XTT2 approach with that of RuleFlow is discussed.

1. INTRODUCTION

Rules constitute a cardinal concept of the rule–based expert systems (RBS for short) [1]. Building such systems requires creating a knowledge base, which in case of RBS can be separated into two parts: factbase containing the set of facts and rulebase containing the set of rules. To make use of this two parts, the inference engine must be provided. The inference engine is responsible for generating findings. This is done according to the current state of the factbase and with the help of the rules. In the first task of the inference mechanism the conditional parts of the rules are checked against the facts from the factbase. This task is performed by *pattern matching algorithm*. The output from the algorithm is the set of rules, which conditional parts are satisfied. This set of rules is called a *conflict set*. The following task of the inference mechanism is the execution of the rules from the *conflict set*. There are many different algorithms for determining an execute order of the rules, but they are not discussed in this paper.

The main problem discussed in this paper concerns inference methods in structured rule-bases. A rule-base can contain thousands or even milions rules. Such large rule-bases cause many problems: 1) Maintenance of the large set of rules. 2) Inference inefficiency – the large number of rules may be unnecessary processed. The modularization of the rule-base that introduces structure to the knowledge base can be considered as the way to avoid these problems. The rules can be grupped in the modules, what can facilitate the maintenance of the large set of rules. What is more, the inference algorithm may be integrated with structured rule-base. The integration can influence the inference performance.

The main focus of this paper is the inference in the structured rule bases. The Section 2 presents the well-known expert system shells such as CLIPS [1], JESS [2] and Drools 5 [3]. It shows how the knowledge base can be structured in these systems and how the inference algorithm can be used over this structure. The next Section 3 describes three main *pattern matching algorithms* such as Rete [4], TREAT and the most recent and general Gator. In the Section 5 the main concepts of the XTT method are introduced. The section presents the structure of the XTT knowledge base. It also introduces the inference methods taking the underlying algorithm into consideration. The conclusions of the paper are included in the Section 6.

2. EXPERT SYSTEMS SHELLS

Expert system shell is a framework that facilitates creation of complete expert systems. Usually, they have most of the important functionalities built-in such as: rule-base, inference algorithm, explanation mechanism, user interface, knowledge base editor.

Such system must be adopted to the domain-specific problem solving. This can be done by creation of the knowledge base. The knowledge engineer must codify the captured knowledge according to the formalism. The knowledge can be captured in a several ways, but this issue is not discussed in this paper.

CLIPS is an expert system tool that is based on Rete algorithm. It provides its own programming language that supports rule-based, procedural and object-oriented programming [1]. Thanks to this variety of programming paradigms implemented in CLIPS, there are three ways to represent knowledge in it:

- rules, which are primarily intended for heuristic knowledge based on experience,
- *deffunctions* and generic functions, which are primarily intended for procedural knowledge,
- object-oriented programming, also primarily intended for procedural knowledge. The generally accepted features of object-oriented programming are supported. Rules may pattern match on objects and facts.

The condition in CLIPS is a test if given fact exists in knowledge database. The right-hand side (RHS) of rule contains actions such like assert or retract that modifies facts database or other operations such like function invocations that does not affect system state.

CLIPS has been written in C language. This makes the tool very efficient and platform independent. However, the integration with other existing systems is not as easy as it is in case JESS.

JESS is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill [2] that derives form CLIPS.

Jess uses a very efficient method known as the Rete algorithm. In the Rete algorithm, inefficiency of the combinatoric explosion of rules analysis is alleviated by remembering the past test results across the iterations of a rule loop. Only new facts are tested against each rule conditional part, but still all rules must be taken into consideration.

Jess supports both *forward-chaining* and *backward chaining*. The default is *forward-chaining*. As the knowledge representation JESS uses rules as well as XML-based language called JessML. JESS uses LISP-like syntax, which is the same as in CLIPS. The JessML is not convenient to read by human. It contains more details, what makes this representation suitable for parsers.

Drools 5 introduces the Business Logic integration Platform which provides a unified and integrated platform for Rules, Workflow and Event Processing. Drools is now split up into 4 main sub projects: 1) Drools Guvnor (BRMS/BPMS) – centralised repository for Drools Knowledge Bases. 2) Drools Expert (rule engine). 3) Drools Flow (process/workflow) provides workflow or (business) process capabilities to the Drools platform. 4) Drools Fusion (event processing/temporal reasoning) – the module responsible for enabling event processing capabilities. Drools Expert is a rule engine dedicated for the Drools 5 rule format.

Drools 5 implements only *forward-chaining* engine, using a Rete-based algorithm – ReteOO. In the future, Drools 5 is promised to support a backward-chaining.

3. RULE INFERENCE ALGHORITHMS

This section discusses three the most important pattern matching algorithms. The descriptions of these algorithms introduce specific nomenclature.

A rule base in the RBS consists of a collection of rules called *productions*. The interpreter operates on the productions in the global memory called *working memory* (WM for short). Each object is related to a number of attribute–value pairs. The set of pairs related to the object and object itself constitute a single *working element*.

By convention, the conditional part (`IF part`) of a rule is called LHS (left–hand side), whereas the conclusion part is known as RHS. The inference algorithm performs the following operations: 1) *Match* – checks LHSs of rules to determine which are satisfied according to the current content of the working memory. 2) *Conflict set resolution* – selects production(s) (*instantiation(s)*) that has satisfied LHS. 3) *Action* – Perform the actions in the RHS of the selected production(s). 4) Goto 1. The first step is a bottleneck of inference process. The algorithms, which are presented in this section, try to alleviate this problem.

The **Rete algorithm** [4] is an efficient pattern matching algorithm for implementing production rule systems. It computes the *conflict set*. The naive implementation of the pattern matching algorithm might check each production against each working element. The main advantage of the Rete algorithm is that it tries to avoid iterating over production and working memory.

Rete can avoid iterating over working memory by storing the information between cycles. Each pattern stores the list of the elements that it matches. Due to this fact, when working memory is changed only the changes are analysed.

Rete also can avoid iterating over production set. This is done by forming a tree-like structure (*network*) that is compiled from the patterns. The network comprise of two types of nodes: intra–elements that involve only one working element and inter–elements that involve more than one working element. At first, the pattern compiler builds a linear sequence of the intra-elements. This part of the network is called *alpha memory* and contains only the *one-input* nodes. After that, the compiler builds the *beta memory* from the inter-elements. The beta memory consists of the *two-input* nodes. Each two-input node (except the first one) joins one two-input node and one one-input node. The first two-input node joins two one-input nodes.

$$\begin{array}{l} R1(a > 17, d(X)), \\ R2(d(X), e(Y), g(Z)), \\ R3(c = on, g(Z)), \\ R4(e(Y), f(W)), \\ R5(b = Friday, f(W)) \end{array} \quad (1)$$

When the working memory is changed, the working elements, that has been changed, are let int to the network. Each node of the network tries to match the given working element. If it matches, then the copy of the element is passed to all the successors of the node. The

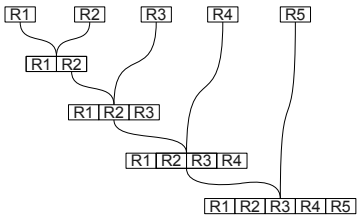


Fig. 1. A general schema of the Rete network.

two-input nodes joins the elements from the two different paths of the network into bigger one. The last two-input element (*terminal element*) is the output from the algorithm and contains the information about changes, which must be applied to the conflict set.
Rete algorithm has been invented by Charles L. Forgy of Carnegie Mellon University. At first, Rete has been assumed as the most efficient algorithm for this problem. The literature did not contain any comparative analysis of the Rete with any other algorithm. Nowadays, other algorithms such as Treat, A-Treat, Gator are known. Some of them are discussed in this paper.

TREAT algorithm. State saving mechanism implemented in Rete is not very efficient. The structure of the Rete network often stores redundant information and number of elements stored in beta-memory nodes may be combinatorially explosive. Moreover cost of join operation in beta-memory are very expensive when many addition and deletion operations are preformed. To address these problems new version of Rete algorithm called TREAT was proposed.

Rete algorithm is based on two concepts: *Memory support* that creates and maintains alpha–memory and *Condition relationship* that join operations in beta–memory. TREAT also uses *Memory support*, but does not use *Condition relationship*. Instead *Conflict set support* and *Condition membership* are used. Absence of *Condition relationship* implies fact that in TREAT network structure there is no beta memory. Hence, the structure of TREAT network is flat.

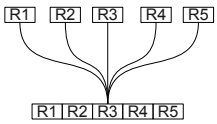


Fig. 2. TREAT network for rule 1

The main idea of the TREAT algorithm is to exploit the *conflict set support* for temporarily redundant systems. The conflict set is explicitly retain across production system cycles which allows for the following advancements comparing to Rete [5]:

- in case of addition of WM element, conflict set remains the same, and constrained search for new instantiation of only those rules that contain newly added WM element is performed.
- deletion from WM triggers direct conflict set examination for rules to remove. No matching is required to process deletion since any instantiation of the rule containing removed element is simply deleted.

Condition membership introduces new property for each rule called *rule-active* that determines weather each of the rule condition elements is partially matched. The match algorithm ignores then rules that are *non-active* during production system cycles.

Gator algorithm. Both Rete and TREAT offer static networks, which structures are defined arbitrary by the design engineer (Rete) and looks mostly the same for all kinds of knowledge databases (Rete and TREAT). This very often leads to the creation of networks that are not optimal for some knowledge bases.

To address this problem a new discrimination network algorithm called Gator was proposed. It is based on Rete, but additionally implements mechanisms for optimizing network structure according to specific knowledge base characteristic. It can be said that Rete and TREAT are special cases of Gator and as reported in [6] it outperforms TREAT and Rete in most cases.

Every rule in production system can be represented by a *condition graph* with nodes for rule condition elements and edges for join conditions.

Gator networks are general tree structures. They consist of alpha–memory elements (leaves), optional beta-memory elements (internal nodes, that can have multiple inputs) and a P–node which is a root of the tree representing a complete RHS of the rule.

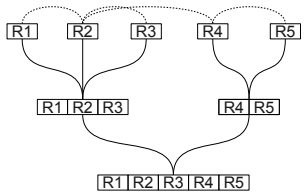


Fig. 3. Gator network for rule 1

The optimizing algorithm is iterative. It starts form networks of size one (which are basically alpha–memory elements) and combine them into larger optimal networks. There is a constraint which states that every newly created network have to be optimal. That ensures that the final network would also be optimal.

The network is built and optimize according to the following rules:

- *Connectivity Heuristic* – do not combine two Gator networks unless there is an explicit connection between them in connectivity graph.
- *Disjointness constraint* – do not combine networks unless their respective sets of rule condition elements do not overlap.
- *Lowest Cost Heuristic* – if there is already a network that covers the same set of condition as the

new network, and the existing network cost (according to the cost function) no more than the new one, discard new network.

More detailed information about cost functions and rules for combining Gator networks can be found in [6].

4. KNOWLEDGE MODULARIZATION

Most of the well–known expert systems have a flat knowledge base. In such case, the inference mechanism have to check each rule against each fact. When the knowledge base contains a large number of rules and facts this process becomes inefficient. This problem can be solved by providing a structure in the knowledge base that allows for checking only a subset of rules. This Section describes the three well–known expert system shells CLIPS, JESS and Drools and knowledge base organisation implementen in them.

CLIPS Modules. CLIPS offers functionality for organising rules into so called *modules*. Modules allows for restriction of access to their elements from other modules, and can be compared to global and local scoping in other programming languages. Modularization of knowledge base helps managing rules, and improves efficiency of rule-based system execution. Modules in CLIPS are defined with *defmodule* command.
In CLIPS each module has its own pattern-matching network for its rules and its own agenda. When a *run* command is given, the agenda of the module which is the current focus is executed. Rule execution continues until another module becomes the current focus, no rules are left on the agenda, or the return function is used from the RHS of a rule. Whenever a module that was focused on runs out of rules on its agenda, the current focus is removed from the focus stack and the next module on the focus stack becomes the current focus. Before a rule executes, the current module is changed to the module in which the executing rule is defined (the current focus). The current focus can be dynamically switched in RHS of the rule with *focus* command.

JESS Modules. Jess provides modules mechanism that helps to manage large numbers of rules. Rules modularisation can be considered as the structure of the rulebase. Modules also provide a control mechanism: the rules in a module will fire only when that module has the focus, and only one module can be in focus at a time. Jess makes the modules defining possible with the help of *defmodule* command. The module name can be considered as a namespace for rules. This means that two different modules can each contain a rule with a the same name without conflicting. Modules can also be used to control execution. In general, although any Jess rule can be activated at any time, only rules in the focus module will fire. It is possible to manually move the focus to another module using the `focus` function. Each rule can decide which module should be focused as the next one. To accomplish that, the operation of the focus changing should be included in the rule conclusion part. This leads to the structured rulebase, but still all rules are checked against the facts. In terms of efficiency the modules mechanism does not influence on the performance of the *conflict set* creation.

Drools RuleFlow. It is a workflow and process engine that allows advanced integration of processes and rules. It provides a graphical interface for processes and rules modelling. Drools have built-in a functionality to define the structure of the rulebase which can determine the order of the rules evaluation and execution. The rules can be gruped in a ruleflow–groups which defines the subset of rules that are evaluated and executed. The ruleflow–groups have a graphical representation as the nodes on the *ruleflow* diagram. The ruleflow–groups are connected with the links what determines the order of its evaluation. A *ruleflow* diagram is a graphical description of a sequence of steps that the rule engine needs to take, where the order is important.

Rules grouping in Drools 5 contributes to the efficiency of the ReteOO algorithm, because only a subset of rules are evaluated and executed. However there is no policy which determines when a rule can be added to the ruleflow-group. Due to this fact, the rules grupping can provide a muddle in the rule base especially in case of large rulebases.

5. XTT–BASED EXPERT SYSTEMS

Knowledge bases in expert system shells described in Section 2 are flat and do not have any internal *structure*. To create a conflict set the entire knowledge base have to be searched, and an intelligent inference control in such unstructuralised system is very difficult. Knowledge representation languages are not formal neither in Drools, Jess, nor in CLIPS and as a consequence there are not formalized methods for verifying and analysing systems designed with those tools. To solve these problems a new knowledge representation method called XTT2 (*Extended Tabular Trees*) was proposed which is part of the HeKatE [7] methodology for designing, implementing and verifying production systems.

5.1. Knowledge representation

Main goals of XTT2 knowledge representation was 1) to provide an expressive formal logical calculus for rules, 2) allow for advanced inference control and formal analysis of the production systems, 3) provide structural and visual knowledge representation. XTT2 incorporates extended attributive table format, where similar rules are grouped within separated tables, and the system is split into such tables linked by arrows representing the control strategy. Each table consist of two parts representing condition and decision part of the rule.

To help creating the XTT2 network, ARD+ diagrams provide the conceptual design. This stage is supported by VARDA tool that generates XML file (called HML in HeKatE methodology) with specification of types, domains, attributes and dependencies between them. Based on this file a XTT2 skeleton is created in HQEd editor, and the tables are filled with rules [8].

Rules representation in XTT2 is based on attributive logic called ALSV(FD) [7]. Each rule in XTT table is of the form:

$$(A_1 \propto_1 V_1) \wedge \ldots \wedge (A_n \propto_n V_n) \longrightarrow RHS \quad (2)$$

where the logical formula on the left describes the rule condition, and RHS is the right-hand side of the rule covering conclusions (see [7] for more details).

The logical rule representation is mapped to the HMR language (*Hekate Meta Representation*) which is an internal rule language for XTT. Following example shows HMR the notation and its pseudocode representation.

```
xrule tab_4/1: [today eq workday,
                hour in [9 to 17]] ==>
  [operation set bizhours].
xrule tab_4/4: [today eq workday,
                hour gt 17] ==>
  [operation set not_bizhours].
```

Pseudocode representation:

```
IF today=workday AND hour>=9 AND hour<=17 THEN
   operation := bizhours
IF today = workday AND hour > 17 THEN
   operation := not_bizhours
```

This formal, logical representation of the rules allows for formal analysis and verification of the system.

5.2. Intelligent inference controll

Described in section 5.1 XTT2 knowledge representation allows for more efficient inference control during rule-based system execution. The inference control is assured thanks to firing only rules necessary for achieving the goal. It is achieved by selecting the desired output tables and identifying the tables necessary to be fired first. The links between tables representing the partial order assure that when passing from a table to another one, the latter can be fired since the former one prepares an appropriate context knowledge. There are four algorithms based on XTT2 notation that control the inference. They were successfully implemented in HeaRT (*HeKatE RunTime*) inference engine [9].

[FOI] The simplest algorithm consists of a hard-coded order of inference, in such way that every table is assigned an integer number; all the numbers are different from one another. The tables are fired in order from the lowest number to the highest one. This inference algorithm is usefull when a reasoning path is well defined and does not change over rule-based system cycles. **[DDI]** A data-driven inference algorithm identifies start tables, and put all tables that are linked to the initial ones in the XTT network into a FIFO queue. When there is no more tables to be added to the queue, algorithm fires selected tables in order they are poped from the queue. This inference mode s especially useful for diagnosis systems, where a lot of symptoms are given as an input that can lead to multiple diagnosis. Choosing apropriate reasoning path by the system saves time and memory. **[GDI]** A goal-driven approach works backwards with respect to selecting the tables necessary for a specific task, and then fires the tables forwards so as to achieve the goal. One or more output tables are identified as the ones that can generate the desired goal values and are put in LIFO queue. As a consequence only those tables that leads to desired solution are fired, and no rules are fired without purpose. This inference algorithm works best in hypotesis-proving systems, where value of attribute from particular table is wanted. **[TDI]** This approach is based on monitoring the partial order of inference defined by the network structure with tokens assigned to tables. A table can be fired only when there is a token at each input. A token at the input is a kind of a flag signalling that the necessary data generated by the preceding table is ready for use. This inference mode was designed to support systems where a lot of dependencies between tables and rules are denoted that would require many redundant conditions XTT tables. Tokens allow to omit those unnecessary conditions, which saves time and memory and makes the system more readable.

The highly modularised knowledge representation that is used in XTT2 was one of the reasons why inference engine – HeaRT – implemented for XTT2 approach does not use matching algorithm based on Rete. Due to the fact that HeaRT was implemented entirely in Prolog, fast and efficient unification algorithm that is implemented in Prolog interpreter was used instead.

5.3. Structure of the Knowledge Base

Considering the differences between the XTT2 approach and the classic Rete-based solutions, at least two meanings of the notion „structure of the rule base" can be given. The first one is related the previously discussed modules in classic expert system shells. There a *physical structure* of the rule base is introduced using modules. The global set of rules is partitioned by the system designer into several parts in an arbitrary way. This is a technical solution, similar to source code partitioning methods such as packages is programming languages. Practically, these partitions are often merged during the inference process. Therefore, the *partitioning process* itself does not support in optimizing the design and inference. The second one is realized in the XTT2 representation. Here rules working in the same context, i.e. having the same conditional attributes are grouped into tables (forming simple rule sets) during the design process. This forms a *logical structure* of the rule base.

This structure is considered during the inference process – only necessary rules are considered, an possibly fired. Therefore, the *modularization process* does support optimization of both the design and inference.

6. CONCLUDING REMARKS

All of the common expert system shells described in this paper use Rete or its variants as a matching algorithm. This is so, because Rete algorithm is very efficient on flat and not structured knowledge base. Once knowledge base becomes modularized, Rete loses its assets. Although idea of modules as sets of not related in any way rules was introduced in CLIPS, the core inference algorithm – Rete – remained the same. Such partial modularisation slightly increases performance of the system, but still did not solve efficient design and verification problems. Most of solutions presented in CLIPS or Jess are just modifications of existing approaches that have their own historical drawbacks.

To address these problems a new knowledge representation called XTT2 was proposed that is a part of newly designed methodology for designing, implementing and verifying expert systems, called HeKatE. It provides visual representation of the knowledge base, formal verification of the rule–based systems and intelligent inference control. XTT2 knowledge base are highly modularized and hence its internal structure allows for more advanced reasoning. Modularisation in XTT is not partial as in CLIPS. XTT tables are not only a mechanism for managing large knowledge bases, but they also allow for context reasoning, due to the fact that each XTT table groups rules that belongs to the same context (have similar LHS and RHS). Moreover, rules in XTT2 are based on attributive logic which allows for formal verification of knowledge base. Table 1 contains the comparison of the expert system shells described in this paper and XTT2 approach.

Table 1. Comparison of expert system shells

Feature	XTT	CLIPS	Jess	Drools
Knowledge modularisation	Yes	Yes	Partial	Yes
Knowledge visualisation	Yes	No	No	Yes
Formal rules representation	Yes	No	No	No
Knowledge base verification	Yes	No	No	No
Inferences strategies	DDI, GDI, TDI, FOI	DDI	DDI, GDI,	DDI
Inference algorithm	HeaRT + Unification	Rete	Rete	Rete
Allows for modelling dynamic processes	No	No	No	Yes

The idea of integrating XTT2 approach with DroolsFlow will allow to combine business processes with formal, modular knowledge representation. Since DroolsFlow diagrams may contain other DroolsFlow diagrams, relations between XTT tables would not be limited to relation table to table, but may also be considered as realtion system to system. Integrating DroolsFlow and XTT can be done by invoking HeaRT from within DroolsFlow blocks directly, using the SWI JPL package for Java integration, or via TCP/IP protocol.

Acknowledgements

Paper is supported by the BIMLOQ Project funded from 2010–12 resources for science as a research project.

7. REFERENCES

[1] Joseph C. Giarratano and Gary D. Riley, *Expert Systems*, Thomson, 2005.

[2] E. Friedman-Hill, *Jess in Action, Rule Based Systems in Java*, Manning, 2003.

[3] Paul Browne, *JBoss Drools Business Rules*, Packt Publishing, 2009.

[4] Charles Forgy, "Rete: A fast algorithm for the many patterns/many objects match problem," *Artif. Intell.*, vol. 19, no. 1, pp. 17–37, 1982.

[5] Daniel P. Miranker, "TREAT: A Better Match Algorithm for AI Production Systems; Long Version," Tech. Rep. 87-58, University of Texas, July 1987.

[6] Eric N. Hanson and Mohammed S. Hasan, "Gator: An Optimized Discrimination Network for Active Database Rule Condition Testing," Tech. Rep. 93-036, CIS Department University of Florida, December 1993.

[7] Grzegorz J. Nalepa and Antoni Ligęza, "HeKatE methodology, hybrid engineering of intelligent systems," *International Journal of Applied Mathematics and Computer Science*, 2010, accepted for publication.

[8] Grzegorz J. Nalepa, Antoni Ligęza, Krzysztof Kaczor, and Weronika T. Furmańska, "HeKatE rule runtime and design framework," in *Proceedings of the 3rd East European Workshop on Rule-Based Applications (RuleApps 2009) Cottbus, Germany, September 21, 2009*, Gerd Wagner Adrian Giurca, Grzegorz J. Nalepa, Ed., Cottbus, Germany, 2009, pp. 21–30.

[9] G. J. Nalepa, S. Bobek, M. Gawędzki, and A. Ligęza, "HeaRT Hybrid XTT2 rule engine design and implementation," Tech. Rep. CSLTR 4/2009, AGH University of Science and Technology, 2009.

AN ADAPTABLE E-LEARNING SYSTEM FOR PUPILS WITH SPECIFIC LEARNING DIFFICULTIES

Petia Kademova-Katzarova, Rumen Andreev, Valentina Terzieva

Institute of Information and Communication Technologies, Bulgarian Academy of Sciences

ABSTRACT

The education of pupils with learning difficulties is very complicated due to great variety of their specific cognitive abilities and psychological factors. It requires the use of personalized learning facilities that can help achievement of their learning goals. For that reason we design an adaptable system for development of tools on the basis of suitable pedagogical methods and learning resources. The system provides facilities for adaptation of learning units to the learning profile of each pupil. The substantial elements of this adaptation technique are carried out by activities of the resource-developer. The paper presents an approach to a description of these activities supported by the adaptable system. The adaptation bases on reusable learning units that can be modified in correspondence with the learner's profile, learning context or scenario.

Index Terms – Learning difficulties, Cognitive abilities, Learning style, Adaptation, Personalisation, Reusable learning units

1. INTRODUCTION

There are many electronic educational systems but for the purposes of school education almost nothing has been done in this regard. Rarely as it may be, e-learning can be found in secondary schools. However teachers don't utilise modern ICT in primary school. Long ago children in kindergartens have been playing on computers, but this interesting "thing" is not set to work in educational process. The reason probably is the difficulty of creating appropriate educational products for young children, because their teaching requires not only a mechanical "dumping" of useful information and knowledge. The learning process is much more complex and includes structured presentation of the learning material in appropriate form and appearance consistent with age and background.

From another point of view, the education in primary school comes across other important problem – certain characteristics of the individuals might hamper them to acquire basic skills such as reading, writing, arithmetic. Many children still lag behind in this early stage of their education not because they are stupid or lazy (common labels), but because they have a special way of perceiving and processing information. These children do not receive teaching adequate to their abilities, the education system rejects them, and society loses specialists with valuable qualities simply because the school failed to discover and develop these qualities on time. Typical examples are children with dyslexia (dyscalculia, dysgraphia), with ADHS and ADS, even with autism.

2. PUPILS WITH LEARNING DIFFICULTIES

Dyslexia, dyscalculia, dysgraphia are disorders in the development of school skills, which are classified in the medical registers, though they are not diseases. The perceiving of environment signals and their processing in the brain shows a specificity that can lead to some distortion of the information and to confusion. For example, in contrast to other people the dyslexics think mostly in pictures [6]. Every thought, every idea and every emotion they "see" as a three-dimensional image in their minds. Consequently, they have problems with two-dimensional symbols and signs which have to be ordered or directed in a certain way to be deciphered correctly. Letters with the same graphical representation but different orientation are confounded (N and Z, b and d). Words without a picture image as prepositions or adverbs hamper them. Therefore, the so-called "cultural techniques" [3] – reading, writing, mathematical expressions are difficult to handle.

3. AN OPPORTUNITY FOR SPECIAL EDUCATION

According to state requirements such children should be integrated together with the others, but they need individual curricula, extra special trained teachers, etc. The aim is to achieve individualization in the teaching process, using pupil's strong skills and personal qualities, and through appropriate exercises to support and develop the weak ones.

That is why these pupils with specific learning (cognitive) difficulties need special education. It could be achieved by development of e-learning system [1] that has to ensure collaboration among all the professionals involved in teaching, generation and adaptation of learning facilities.

Psychological	Pedagogical	Technological
Early screening and identification of children with learning difficulties	Individual curricula, personal teaching assistant	Tool for generation of computerized psychological tests
Detection of cognitive abilities and psychological characteristics	Close collaboration among all professionals concerned with the problem	ICT-based tools allowing collaboration
Defining of psychological profile and learning style	Suggestions for appropriate pedagogical methods and formats: teaching methods arousing interest and catching attention; inducing an emotional connection to the learning matter; illustrative representation of learning units	Authoring tool enabling adaptation of learning resources and building personalized learning paths according to learner's preferences; Incorporation of various instruments for illustration (audio, video, simulation, 3D-modeling, etc.) contributing efficiency to education
Recommendations for learning environment (comfortable, without stress and frustration)	Relaxed and adaptable learning environment enabling to bestow various encouraging bonuses (music, videos, games, etc.)	ICT-based adaptable user-friendly environment (intuitive, language independent, allowing tuning and contextualization)

Table 1 Technological tools meeting psychological and pedagogical requirements for education of pupils with learning difficulties

Table 1 gives an overview of the psychological, pedagogical and technological requirements for the education of pupils with specific learning difficulties.

4. CONCEPTUAL MODEL OF AN ADAPTABLE E-LEARNING SYSTEM

The development of personalized e-learning facilities requires design of adaptable e-learning system that supports production and delivery of learning resources. We suggest a conceptual model of such adaptable e-learning system shown on Figure 1.

The basic elements in this model are the learner's profile, the pedagogical aspects, the resulting pedagogical format and the appropriate learning units.

4.1. The learner's profile

The learner's profile represents cognitive abilities and psychological characteristics. It defines a learning style and appropriate pedagogical methods and tools. The determination of the cognitive abilities depends on the following important characteristics, which are derived during psychological testing:

- Memorizing (short term and long term memory),
- Attention,
- Concentration,
- Absorption capacity,
- Observing ability,
- Working capacity,
- Orientation, Coordination, Balance,
- Motor functions (fine motor skills),
- Communication skills,
- Handling abstract terms and symbols,
- Way of thinking – in terms ("sequential") / in pictures ("quasi parallel").

Some significant psychological features that have influence on the learning process are *self-assessment, imagination, patience, excitability* and *emotionality*. All these characteristics could be easily assessed by computerized psychological tests. They should be in the form of amusing games or entertaining tasks in order to prevent stress and frustration, so that children could do their best. The results and indicators are the basis for the psychological profile of the pupil. This profile determines the teaching style, methods and tools which serve to arrange and to accomplish the education process in the most appropriate way.

4.2. The pedagogical room

The pedagogical room consists of pedagogical methods and pedagogical tools that are in correspondence with the learning style. The most commonly used pedagogical methods are:

- Informational – the teaching is performed using "instructions". Key elements of this method are the messages and the symbols.
- Phenomenological – the knowledge is build up as an event. It is accepted and absorbed through senses and emotions [7].
- Collaborative – this method is connected with the socio-cultural environment. Thus knowledge and skills are formed in a family, in a class, communities, societies, ethnic groups, etc. The knowledge and the skills are "passed over", the experience is shared. Games are typical example of this educational approach.

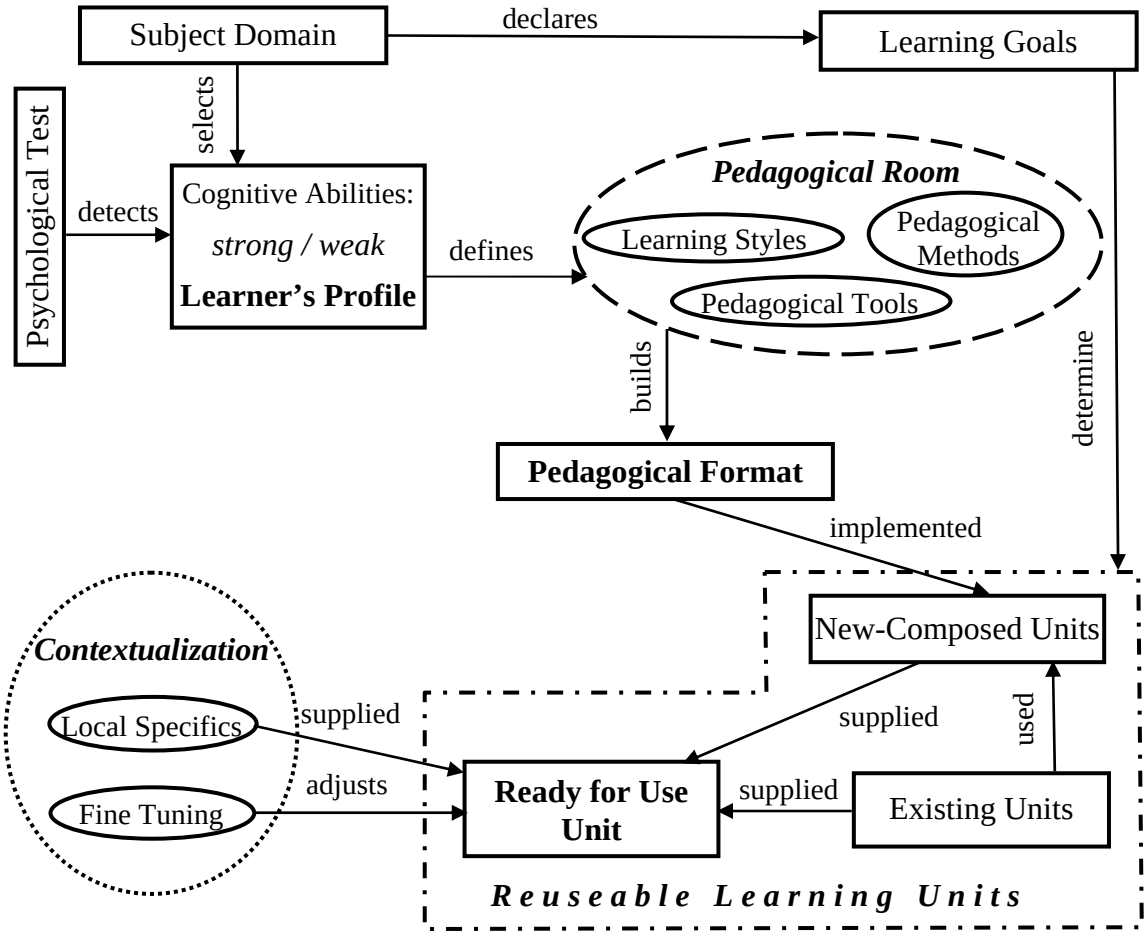


Figure 1 Conceptual model of an adaptable e-learning system

Children with dyslexia are predisposed to learn mostly by the phenomenological method as they can observe the action and get a real idea of the phenomenon. At the same time they can form an emotional connection with the subject matter, which helps focusing attention and supports the memorizing.

4.3. The pedagogical format

The pedagogical format describes the way of knowledge presentation in the learning units. It is built on the basis of the selected pedagogical methods and tools in compliance with pupils' learning style.

4.4. The learning units

The system allows access to learning units stored in databases or repositories. These resources can be modified, adapted and reused in a process of composition of new learning units according to the given pedagogical format [5].

According to the Figure 1, the psychological test detects the strong and week points of cognitive abilities that have to be underlined in the learner's profile. Subject domain contains knowledge about the learning subject(s) (reading, writing, language, mathematics, etc.). It gives the criteria for selecting the appropriate personal features from the learner's profile. On that basis the learning style is determined and the pedagogical methods and tools are chosen. As above mentioned, those are the factors for building the pedagogical format. The latter serves as a frame for composing learning units. The activities regarding constructing of pedagogical formats and learning units are supported by the ICT-based authoring tool. Considering the methodological recommendations and employing the authoring tool, teachers create new learning units, reuse the existing ones or edit, update them and save for future application. Each learning unit done according to the above described procedure is contextualized with regard to the local specifics and learner's preferences. Thus, the composed unit is ready for use.

5. FUNCTIONAL MODEL OF THE E-LEARNING SYSTEM

The functional model of e-learning system can be represented as composed of three parts [4] – the users, the ICT platform and their interactions (Figure 2). Some essential characteristics of the system are:

- To have sufficient technical tools in order to meet the requirements for diverse presentations of the learning matters including sounds, pictures, movies, clips, animations etc.

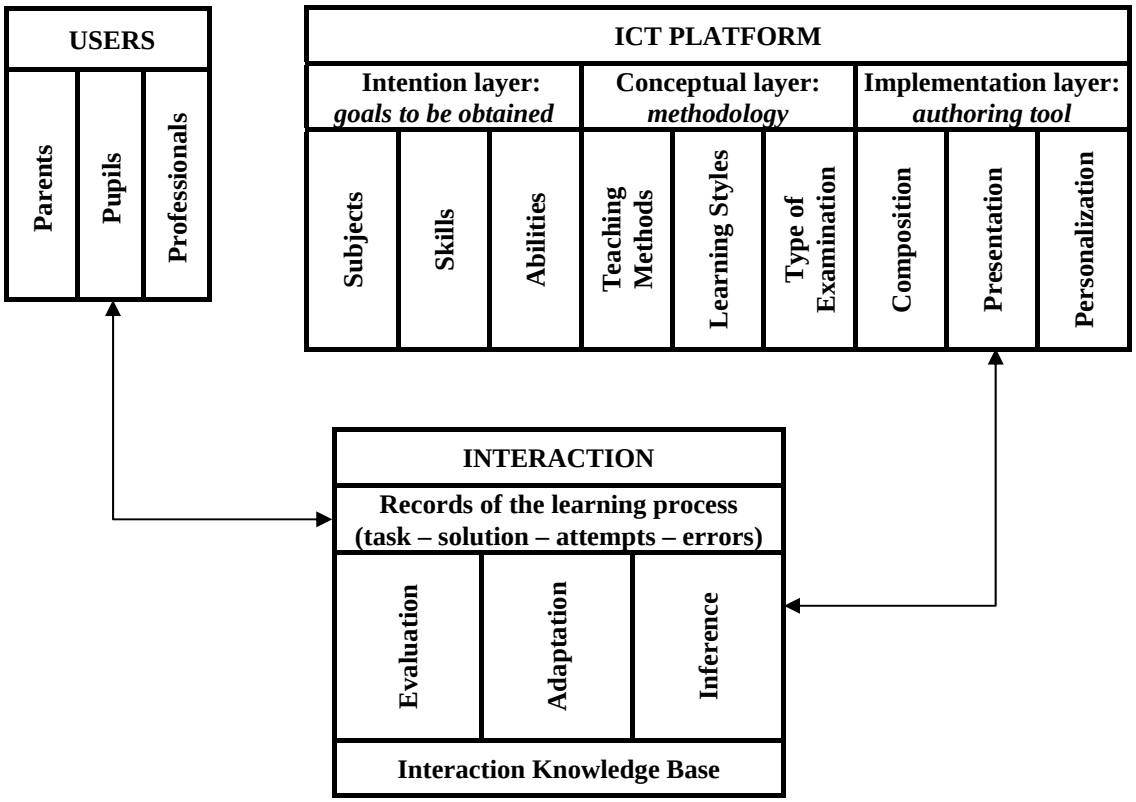


Figure 2 Functional model of the presented e-learning system

- To allow access to appropriate tools for modelling and design of 3D objects.
- To maintain data bases that contain learning resources – learning units, learning facilities.
- To have user friendly design.
- To be flexible and adaptable.

5.1. Users

Among the above mentioned essentials, such a system must have very specific features and characteristics that meet the requirements and perform functions of different types of users:

1. **P**rofessionals (psychologists, pedagogues – methodologists, teachers and speech therapists),
2. **P**upils,
3. **P**arents.

These types of users (**PPP**) exploit the facilities of the system in different ways and in different capacity and therefore they obtain different access rights.

Pupils have access only to learning materials and to some games and entertainment, which they receive as bonuses and rewards in order to be stimulated to deal with the material. It should be noted that the stimulus should only be positive; i.e. there shouldn't be penalties.

Parents have access to the learning outcomes of their children and in case they could support children in their training. Also they can share ideas, impressions and experiences in parental forums. They can seek advices about problems concerning the children's training from the professionals.

Professionals have greater access rights to the system. On the one hand, they must monitor the training process in order to record and analyse the mastering of learning material. Future steps in the learning path are determined by these records and analysis (adaptation). On the other hand, they also participate in the forums – both professional and popular. In the first case they share their problems, ideas, solutions, tips, experiences, arrange conferences. In the latter case, they give advices and suggestions at "common level" in communication with parents, who generally are not experts in the matter.

5.2. ICT platform

The ICT platform consists of three layers – intentional, conceptual and implementation. The *intention layer* presents learning goals that have to be achieved and are built-in parameters of the e-learning system. Those specify the knowledge, skills and abilities (in one or several subjects) that must be acquired, in compliance with the government regulations.

The possible approaches to attain the learning goals are presented in the *conceptual layer*. According to the personal profile of the child the appropriate pedagogical method(s) are selected and

implemented so as to achieve the learning goals efficiently. Furthermore, this profile serves for determination of the learning style and gives guidelines for the choice of relevant modes and forms of the examination. The latter shouldn't be distressful and upsetting, but motivating the pupils to do their best.

Methodologists consider motivation and learner's activity as the most important elements for successful learning strategy. In order to awake the children's activity it is necessary to engage them in the learning process, which could be done best through the emotional connection with the learning matter. Motivation can be provoked by presenting the subject clearly and precisely so as to be understood and absorbed quickly enough. Thus the accomplishment of learning tasks with noticeable results forces the motivation and heightens the self-esteem and self-confidence of the child.

The *implementation layer* includes diverse instruments that serve to gain the learning objectives identified in the intention layer. Modern technologies provide a huge range of capabilities to assist to the full extent the creation of learning units using different pedagogical formats. The latter are implemented by the specially designed authoring tool.

Professionals use this authoring tool to compose learning resources. It supports various functions – development, structuring, reusing and adaptation of learning units, so as to carry out different learning courses and scenarios. In order to meet the necessities of the pupils, determined by their individual cognitive characteristics, the authoring tool must allow adaptation of learning units, regarding the following aspects:

- Volume,
- Presentation (through different types of media – illustrations, simulations),
- Contextualization:
  - Content Localization – language, custom, traditions, etc.,
  - Fine Tuning – font, colour, size; etc.

5.3. Interaction

The interaction part of the system contains a database for every child's reactions (assigned tasks, provided solutions, performed attempts, made errors). On this basis, the professionals can determine the level of the acquired knowledge and infer how to continue the learning path. Besides the above mentioned the system provides opportunities to exchange information – opinions, ideas, plans, experiences, tips – between the different type of users on the one hand and among peers on the other. Therefore the professionals outline and arrange guidelines for both the further development and assembling of learning units and any necessary adaptations to the specific needs of the pupils.

6. AN EXAMPLE

The children with dyslexia (one of the common learning difficulties) need visual representation of every single conception in order to understand its meaning. That is why they have difficulty with prepositions, adverbs and similar words. Therefore a phenomenological approach is applied for solving such problems. The Davis' method [2] is based on this technique. It recommends following steps:

- clear and precise explanation of the selected word;
- helping pupils to use this word in examples;
- motivating them to construct model(s), representing their idea of the word.

The models could be either hand-made of plasticine (clay), or formed using ICT-based tools (e.g. Google SketchUp 6, TopMod3d, etc.). In addition the modelled word has to be written. In this way the child obtains an integral idea of the word: meaning, image and spelling and is able to understand and use it properly. The example on Figure 3 shows the process of building the conception of the adverb "backwards".

Figure 3 "Backwards"

The child's explanation was: "Four balls plus one ball make five balls; '*Backwards*' means the opposite action".

7. CONCLUSIONS

The presented e-learning system exploits effectively ICT for gaining better educational results for all pupils. Obviously, pupils with learning difficulties have characteristics and perceptions that distinguish them from the other pupils. These differences vary in some extent and cause specific knowledge processing. For that reason such pupils demand personalized education. It should be adapted to their individual cognitive abilities and corresponding learning style. This approach is of benefit for all the children as well, but it is crucial for these with learning difficulties as dyslexics.

On the other hand, the composition and adaptation of learning units for learners who need

special education is very complicated and fatigue process, which requires additional teachers' abilities. Hence ICT-based authoring tools are badly needed and of vital importance nowadays. That is the reason for developing a system that integrates ICT tools for:

- Collaboration between professionals,
- Facilities supporting the learning process,
- Creation, reusing and adaptation of learning units.

Furthermore, the system performs a technique for personalisation of learning units in correspondence with the learners' profile of each pupil. All this activities integrated in such a system not only make easier and optimize the teachers' work, but contribute to achieving efficient learning process.

8. REFERENCES

[1] R. Andreev, V. Terzieva, and P. Kademova-Katzarova, "An Approach to Development of Personalized E-learning Environment for Dyslexic Pupils' Acquisition of Reading Competence", International Conference CompSysTech'09. ACM Series, vol. 433, 2009.

[2] Davis, R., The Gift of Learning. The Berkley Publishing Group, USA, 2003.

[3] M. Kalmár, "Theorie und Praxis der metaphonologischen Analyse", mitSPRACHE 36, (2), pp. 5-17, 2004.

[4] R. J. Keeble and R.D. Macredie, "Assistant Agents for the World Wide Web Intelligent Interface Design Challenges", Interacting with Computers, Elsevier, pp. 357-381, 12, 2000.

[5] R. Koper and B. Olliver, "Representing the Learning Design of Units of Learning", Educational Technology & Society, pp. 97-111, 7 (3), 2004.

[6] Temple, R. Dyslexia: Practical and Easy-to-Follow Advice. Element Books Limited, Shaftsbury, Dorset, 1998.

[7] W. Iser, "The Reading Process: a Phenomenological Approach", The Implied Reader, Johns Hopkins UP, pp. 274-294, Baltimore, 1974.

DECISION-MAKER-AWARE DESIGN OF DESCRIPTIVE DATA MINING

Benedikt Kaempgen
Karlsruhe Institute of Technology (AIFB)
Karlsruhe, Germany
benedikt.kaempgen@kit.edu

Florian Lemmerich
University of Würzburg
Department of Computer Science VI
Würzburg, Germany
lemmerich@informatik.uni-wuerzburg.de

Martin Atzmueller
University of Kassel
Knowledge and Data Engineering Group, Kassel, Germany
atzmueller@cs.uni-kassel.de

ABSTRACT

This paper presents two real-world case studies focussing on descriptive data mining for decision-makers. For that, we first propose a process-oriented design of descriptive data mining that helps in describing and performing such projects. Finally, we discuss important lessons learned during the implementation of the respective projects.

1. INTRODUCTION

With the implementation and collection of data in routine fashion, e.g., in industrial, medical, administrative and social-web-based scenarios, the analysis and mining of such accumulated data is of prime importance for intelligent decision support. However, currently up to 60% [1] of data mining projects fail. One problem concerns the integration of the key stakeholders in data mining projects, i.e., the decision-makers. They need to be tightly integrated into the project, similar to the actual data mining engineers. Thus, in order to improve the common understanding on goal, approach and outcome a more transparent data mining process considering both developer team and decision-maker is rather important.

In this paper, we consider two case studies: The first one is concerned with the analysis of the success and failures of (bachelor) student groups in order to help decision support for improving the success rate of individual curricula. The second one is concerned with the evaluation of a web-based training system and aims, e.g., at analyzing the outcomes of different study groups and their learning differences.

We focus on approaches for obtaining descriptive reports and descriptive data mining models, e.g., local patterns and rules as actionable knowledge for decision support. Descriptive data mining focuses on describing the data by the discovered patterns and relations: In contrast to predictive data mining no specialized model is extracted (for later prediction or classification) but a set of patterns and/or relations is mined for characterizing and describing the data and its hidden components.

In this context, the contribution of this work is threefold: First, we propose a process-oriented design for describing and performing projects in the context of decision-maker-aware descriptive data mining. Second, since only few descriptions of successful data mining projects that concentrate on decision-makers as well as the development team are available, we present two such case studies. Third, we discuss specific experiences and lessons learned during the implementation of the case studies. Altogether, it is our motivation to enable more successful descriptive data mining projects.

The rest of the paper is structured as follows: Section 2 discusses related approaches. After that, Section 3 presents the process-oriented design for describing and performing the case studies. Next, the implemented case studies are described in detail. Section 4 reports specific experiences and lessons learned obtained during the implementation of the case studies. Finally, Section 5 concludes the paper with a summary and interesting directions for future work.

2. RELATED WORK

In the following, we describe related work that deals with data mining design and implementations.

Process models provide an high level overview of the input and output of required data mining tasks. According to Kurgan and Musilek [2] CRISP-DM [3] is most prominently used in data mining projects. It consists of six iteratively executed phases: *Business Understanding* and *Data Understanding* make sure that the developer team has necessary background knowledge to deal with the problem of the decision-maker. In *Data Preparation* the available data is transformed for analysis, e.g., by selection, cleaning, construction, transformation and integration. In the *Modeling* step data mining techniques (algorithms) are applied to the prepared data to extract information and knowledge. In the *Evaluation* these results are evaluated, validated and checked against the data mining objectives. Finally, in the *Deployment* phase the results are employed for action, i.e., integrated into the respective processes of the decision-maker.

Marbán et al. [1] discuss the evolvement of data mining to an engineering discipline. They emphasize, that successful projects take more than CRISP-DM's Development Processes: *Organizational Processes* influence the whole organization in which data mining techniques are being used, e.g., continuous improvement and training or establishing of an appropriate data mining infrastructure. *Project Management Processes* assure successful project planing, e.g., by continuous communication with the decision-maker. Furthermore, *Integral Processes* support the development, e.g., documentation or configuration management. Although process models help developer teams and decision-makers to understand what to do in data mining projects, they do not describe *how* it can be done.

In contrast, *methodologies*, e.g., Catalyst [4] feature step-by-step guidance to data mining. However, as methodologies are more dependent on current techniques and systems, they are difficult to keep up to date.

Most *case studies* describe how techniques and systems can be applied in a specific project and concrete application domain. However, while many case studies of data mining projects have been presented (e.g., [5]), they are primarily used for demonstration of specific tools, results or techniques and therefore are seldom more generally applicable.

3. CASE STUDIES

In this section, we present two case studies. After presenting the process-oriented design, we discuss each one in detail.

3.1. Process-Oriented Design

Following Yin's [6] recommendations for well-designed case studies the purpose of the covered case studies is thoroughly describing how descriptive data mining can be successfully applied. As such the case studies are aimed at readers with both some technical background and business interest that consider data mining techniques in a project.

3.1.1. Focused Roles

On the one hand the decision-maker intends to benefit from data mining techniques. More precisely, the decision-maker has access to raw data and expects descriptive data mining techniques to extract information suitable to support his decision(s). The needs of the decision-maker are formalized as *requirements*.

On the other hand, the team of developers intends to fulfill the specified requirements by applying descriptive data mining tasks. The team usually consists of three kinds of experts [7]: *Data mining experts* are familiar with data mining techniques and the respective tools. *Data experts* offer thorough understanding of available and useful data, e.g., the data representation or the data acquisition process, while *domain experts* hold knowledge of the application area.

3.1.2. Focused Processes

We focus on three components (see Figure 1 for an overview): First, decision-maker processes are mainly related to the decision-maker, considering his or her specific needs. They include project definition, engineering of data mining requirements and result presentation. Second, developer team processes deal with techniques and systems that enable the developer team to fulfill the requirements and obtain useful results. Third, organization processes cover functions shared by different projects.

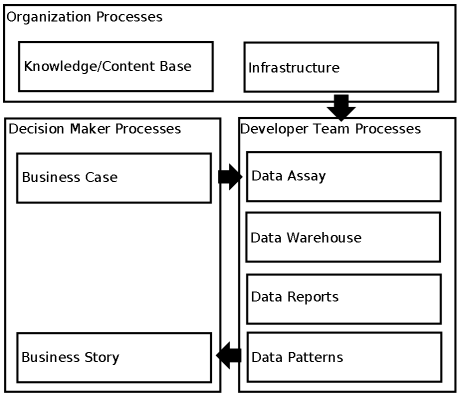


Fig. 1. Case Study Design w/ Information Flow

Decision Maker Processes Based on interviews with the decision-maker and possible feasibility studies, the developer team proposes a data mining approach to the decision-maker's problem in a *Business Case* document written "in management terms" [4, p. 205] and asks for his approval. The Business Case is a central document for any data mining project. It should include the background and motivation of the project, an explicit statement of the problem tackled by the project, a detailed description of the current situation and available data, recommended and alternative solutions, a project plan with time and cost estimations and a glossary.

As decision-maker and developer team mostly have different backgrounds, exact specification of suitable project requirements is a tedious, however, an essential task in descriptive data mining [8].

For that, the problem is restated in single "reporting type questions" [9] asking for attribute-value-pairs in tabular form describing instances of an object. These single Data Reports are then possibly analyzed further by "deeper analytic questions" [9] asking for hidden *Data Patterns* retrieved by techniques ranging from simple visualizations with diagrams or charts up to clustering or classification by machine learning algorithms.

To improve the decision-maker's understanding of the requirements both Data Reports and Patterns may be illustrated by (fictional) examples. Additionally, possibilities for evaluation might be given, e.g., background information and other (secondary) data.

A Business Case is not a static document. In fact, especially requirements will be exposed to constant changes. These are mainly due to results from development processes and have strong influence on the life cycle of a data mining project. In a successful project each requirement is fulfilled and documented in a *Business Story* [4, p. 509].

Developer Team Processes By preparing a *Data Assay* [4, p. 278] Business Understanding, Data Understanding and Data Preparation from CRISP-DM are implemented. It involves a concise description of the raw data, that is made available in a precisely specified tabular form. Additionally, quality issues, for example missing values, should be mentioned explicitly.

Data Preparation is done by making all neccessary data available in a *Data Warehouse*. The team identifies objects, attributes and relationships within the raw data and integrates them in an entity relationship model. Furthermore, data cubes are developed as a more subject-oriented view, if required. Each cell within a data cube can be described by shared attributes (dimensions) and aggregated attributes (measures). From these data cubes, a multidimensional model [10] is developed.

Next, the team creates *Data Reports*, which consist of a query from the data warehouse and additional layout information, e.g., a title or content explaining notes. Additional information can also be included as semantic annotations [11, 12], providing additional presentation possibilites and extended exchangeability. Based on these reports the team applies data mining algorithms to acquire *Data Patterns* specified in the requirements. Both data reports and mined patterns are evaluated and attached to the business story.

Organization Processes To support knowledge management between projects a standardized way of documentation is necessary. Instead of using single documents, we utilize a *Knowledge Base*, cf., [13], that supports references and more efficient searching. Based upon these approaches, we have designed an object-oriented documentation structure, that keeps track of various objects, e.g., goals, tasks, results, tools and documents, and their relationships, and makes these crucial experiences also available across different projects.

Also, a project can only be executed if an appropriate *Infrastructure* of hardware and software is available. For the different steps of our case study design highly specialized software components are available. For the Data Assay, for example, an ETL (Extraction, Transformation, Loading) component can be used, while implementing an entity relationship model or multidimensional model and or effective querying through SQL or MDX [1] is supported by specialized data warehouse components. A data reporting component makes it possible to customize data exports (CSV, ARFF) and to create reports with flexible layout information in various formats (e.g., PDF, XLS). A data mining component is able to read such exports and use data mining techniques (e.g., diagrams, correlation coefficients, subgroup discovery) on their data in order to make data patterns accessible. Finally, a documentation component supports web-based content management of objects, attributes and relationships.

The utilized documentation structure also provided the necessary information for an extensive description of the case studies.

3.2. Case Study I: Student Performance Evaluation

In the following, we describe the decision-maker processes, the developer team processes, and the organizational aspects of the bachelor project.

3.2.1. Decision Maker Processes

In Germany, the introduction of standardized bachelor degrees has been exposed to much criticism lately.

Therefore, for objective assessment on university level an in depth analysis is needed. Basic analytic questions to justify changes in the curriculum are for example: "How do important measures of bachelor degrees evolve?", "How do important measures of exams evolve?" or "What performance do current students achieve?".

The raw data for this proejct was provided by university administration. Since this data includes private student data, it was very carefully selected and precautiously pseudonymized. The legal process for getting permission to access the sensible data took several months in total. The data includes information on:

1. Enrollment information, with the actual semester, number of past semesters and degree of all bachelor students.

2. Exam information, with subject, number of achievable credits, number of lecture hours per week and the type of exam, e.g., module or submodule.

3. Information about student performance in an exam, with pass/fail status, achieved credits and mark.

4. Curricula information, that for each student separately defines categories to exams, e.g., obligatory or compulsory.

[1] http://msdn.microsoft.com/en-us/library/aa216767(SQL.80).aspx

Exemplary requirements, on which the head of the university faculty of (for example) biology, as a relevant decision-maker and the developer team might have agreed, is described as follows: As a Data Report, for each current student of biology the starting semester, number of past semesters, number of university semesters, sum of credits, average credits per semester and overall average grade should be presented. Additionally, the last two measures should be provided for each category of exam separately. As Data Patterns, for a better overview the reports were to be sorted on the number of past semesters and the sum of credits. Also, the histogram of credit points acquired by all students should be provided. This diagram was expected to reveal the number of very unsuccessful (and therefore probable to fail) and very successful (e.g. students already going to university before the end of college) students. Finally, student groups with low/high numbers of semesters and particlarly bad/low marks were to be discovered. This might extract information as "students in their fifth semester have an average mark of 2.0, students in their second semester have an average mark of 3.1, wheras all students have an average mark of 2.6".

During project life cycle these requirements were adapted several times. E.g., the formula for the computation of the overall average grade was not sufficiently specified at the project start. Furthermore, highly detailed requirements on the layout of result representations evolved. Since the utilized open source reporting software could not sufficiently support these requirements, tailored project specific java programs were additionally developed.

As part of the resulting business story the data report was given to the heads of faculities and provided insight into the overall student's performance. The credit distribution indicated a credit threshold for likely-to-fail-students suitable for an automatic warning system, that proposes these students for an additional mentoring program. Influences on student performance indicators will be further enhanced in the future with more information, e.g., survey answers, nationality, gender or age. Such reasons might propose actions towards a more adequate degree program. However, interpretations should be undertaken carefully. Students studying two-subject bachelor degrees need less credits in each subject and may indicate poor performance in comparison to others. Separating these student groups is issued to a follow up project.

3.2.2. Developer Team Processes

The developer team first imported several CSV file exports from the university information system into the data warehouse system. Based on that data, the team developed an entity relationship model made of five entities: Enrollment, person, exam, performance and exam category, each further described by attributes and relationships. Due to the complexity of SQL queries required for the data mining tasks, the ER-model was transformed into a multidimensional model. It contained two data cubes, one of enrollments and one of single performances.

Both an enrollment and a single performance are described by the student, the semester, the number of past semesters, the bachelor degree and an information whether that student is still enrolled in the actual semester. Each single performance is further described by the status, the exam and the type and category of the exam. For a data cell in the enrollment cube the number of individual students and both the minimal and maximal number of past semesters can be calculated. For a data cell of single performances the sum, number and average mark and the sum of credits can be calculated.

Now the team created reports based on data queries in MDX and specified layout informations according to the requirements. Additionally, exports for tools specialized on advanced pattern discovery were created. In this case distribution diagrams were created and subgroup discovery tasks were performed.

3.2.3. Organization Processes

As infrastructure three separate computer systems (each common 32-bit machines, 2 GHz, 2 GB RAM) were used: On one workstation the team mainly used Pentaho Data Integration[2] for the ETL processes and both VIKAMINE[3] and Weka[4] for data mining. On a server, MySQL and Pentaho Mondrian OLAP[5] were used for the data warehouse and Pentaho Business Intelligence Platform[6] was used for creating the data reports. As knowledge base the team used Semantic MediaWiki[7] on another server (for an overview, see Figure 2).

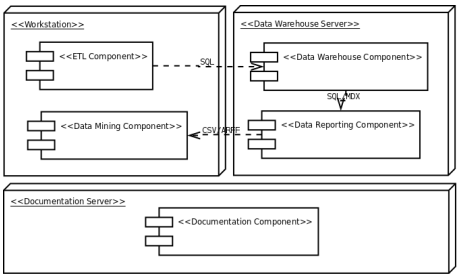


Fig. 2. Bachelor Infrastructure

The results of the project provided valueable insights on the performance of the students, on an automated and on-demand basis.

[2] http://kettle.pentaho.org/
[3] http://www.vikamine.org/
[4] http://www.cs.waikato.ac.nz/ml/weka/
[5] http://mondrian.pentaho.org/
[6] http://community.pentaho.com/projects/bi_platform/
[7] http://www.semantic-mediawiki.org/

3.3. Case Study II: E-Learning system evaluation

Again, the processes centric to the decision-maker, the developer team and the organization are discussed.

3.3.1. Decision Maker Processes

Students at the university of Wuerzburg are offered exam-relevant case-based training courses. The benefits of such a learning system need to be evaluated regularly. Exemplary questions include: "What influence does learning with the system have on exam performances?" or "How satisfied are users of the learning system?". User logs can provide useful data to answer such questions:

1. Log data tracks information about users learning with single cases. Each case execution consists of questions each offering a single score that is accumulated to a total score. The log data also contains information on the usage of help functions, e.g., asking for background information, reading hints or taking a break. Furthermore, at the end of most cases the user is asked for system evaluation: A mark about the case and the system and some textual feedback.

2. Meta information contains additional facts about cases: The form of case evaluation and the time the author expects a user to finish a case.

3. Exam results are available for some courses supported by case-based training.

Exemplary requirements can be described as follows: As a Data Report, for each exam result of a student the number of processed cases, the overall time used for learning with the system, the average overall practice score and the mark and percentage of correct answers in the exam are presented in tabular form. As Data Patterns, correlations between the engagement of the students with the system and their performances at the exam should be discovered, e.g., using a scatter plot and correlation coefficients. This requirement was initially expected to show a high influence of a student's effort with the system and his exam results, showing the effectiveness of the system. While providing promising results, however, no statistically significant correlation was discovered, in contrast to expectations: This is possibly due to not considered influences on student performances, e.g., present knowledge (level) of students, and due to a limited availability of (external) exam results in the considered sample of data.

3.3.2. Developer Team processes

The developer team first imported the provided data into the data warehouse system. This was a non-trivial task, since some data was available in a semi-structured form. Then, the team developed an entity-relationship model made of eight entities: student, case, case execution, evaluation, exam result, score, score action and case action. A multidimensional model consisting of three cubes was added for better querying. Each cube is described by several partially shared dimensions, e.g., student, case and date of execution. A case action is further described by the time of action (beginning and end of case execution) and the kind of action (e.g., pause, case summary, link). A case execution is further described by the exam that execution was relevant to. For a data cell of case execution actions the number and overall time of the actions can be calculated. For a data cell of case executions can be given e.g., the number of case executions, the average overall score, the overall time and the average performance of corresponding exams. For a data cell of scores the number of scores, the average score and the average/overall time taken for viewing the question and answer hints can be calculated. Similar to the bachelor case study, the developer team now designed data reports and exports as stated in the requirements, e.g., correlation mining.

3.3.3. Organization Processes

The Organization processes were executed similar to the bachelor case study. Both projects could not only use the same knowledge base but basically rely on the same infrastructure.

For examining the learning behavior of the students using the CaseTrain system, the performed reports and descriptive data mining results proved promising. Therefore, similar data mining approaches will be implemented as routine mechanisms within the CaseTrain system in the near future.

4. LESSONS LEARNED

From the case studies we could obtain several lessons learned: The proposed methodology appears to be generally applicable: Both projects – though substantially different in domain and requirements – were successfully finished; Data Reports in tabular form are flexible enough to contain most kinds of information; from simple diagrams to sophisticated machine learning algorithms – Data Patterns include the whole range of techniques to retrieve knowledge from this preprocessed raw data. Moreover, for most neccessary components open source software is available.

More than 70% of development time was used for the Data Assay and Data Warehouse. Changes to the data structure, e.g., when adding new features, result in significant additional work. Versionizing and refactoring of raw data description and preprocessing steps that get repeated several times would have been useful and seem essential in bigger projects.

Intensive documentation obviously is crucial for long-running data mining projects, especially if team members change. By documenting not only the project itself, but also sharing experiences and best practices, e.g., on applied tools and techniques, the documentation of one project proved to be extremely helpful for the other. Further cross-project benefits were achieved, since both projects shared a common infrastructure of hardware and software.

Legal aspects of a project should be addressed very early in a project, since the reviewing of data privacy issues and the integration of additional data can require a substantial amount of time. For having several and long running projects a framework of tools as used here seem crucial due to synergistic effects. The projects could be executed exclusively using open source systems. However, some components of current open-source system showed to be insufficient to match project requirements, e.g., highly specialized layouting of the results. Specifically tailored scripts were suitable to fill this gap. This combination of a tool suite for general purpose tasks and additional project specific implementations seems to be well suitable to handle highly specialized requirements.

5. CONCLUSIONS

This paper presented two case studies of successful descriptive data mining projects in two different contexts, i.e., the context of the analysis of university students performance and in usage data evaluation of an e-learning system. We proposed a decision-maker-aware approach for descriptive data mining, and discussed important lessons learned. In the future, in order to fully evaluate the decision-maker-awareness, retrieve general best practices and finally develop a full-scale methodology for descriptive data mining we aim to apply our design to further case studies in various domains.

6. ACKNOWLEDGEMENTS

Part of this work has been funded by the EU IST FP7 project ACTIVE under grant 215040, and by the German Research Council (DFG) under grant Pu 129/8-2. Furthermore, this work has been partially supported by the VENUS research cluster at the interdisciplinary Research Center for Information System Design (ITeG) at Kassel University.

7. REFERENCES

[1] Oscar Marbán, Javier Segovia, Ernestina Menasalvas, and Covadonga Fernández-Baizán, "Toward Data Mining Engineering: A Software Engineering Approach," *Information Systems*, vol. 34, no. 1, pp. 87 – 107, 2009.

[2] Lukasz A. Kurgan and Petr Musilek, "A Survey of Knowledge Discovery and Data Mining Process Models," *Knowl. Eng. Rev.*, vol. 21, no. 1, pp. 1–24, 2006.

[3] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rudiger Wirth, "CRISP-DM 1.0 Step-by-step Data Mining Guide," Tech. Rep., The CRISP-DM consortium, August 2000.

[4] Dorian Pyle, *Business Modeling and Data Mining*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[5] Michael Brydon and Andrew Gemino, "Classification Trees and Decision-Analytic Feedforward Control: A Case Study from the Video Game Industry," *Data Min. Knowl. Discov.*, vol. 17, no. 2, pp. 317–342, 2008.

[6] Robert K. Yin, *Case Study Research*, Number 5 in Applied social research methods series. Sage, Thousand Oaks, Calif. [u.a.], 4. ed. edition, 2009.

[7] Sarabot S. Anand and Alex G. Buchner, *Decision Support Using Data Mining*, Trans-Atlantic Publications, 1998.

[8] Paola Britos, Oscar Dieste, and Ramón García-Martínez, "Requirements Elicitation in Data Mining for Business Intelligence Projects," in *Advances in Information Systems Research, Education and Practice*. 2008, pp. 139–150, Springer Boston.

[9] Ron Kohavi, Llew Mason, Rajesh Parekh, and Zijian Zheng, "Lessons and Challenges from Mining Retail E-Commerce Data," *Mach. Learn.*, vol. 57, no. 1-2, pp. 83–113, 2004.

[10] Sergio Luján-Mora, Juan Trujillo, and Il-Yeol Song, "A UML profile for Multidimensional Modeling in Data Warehouses," *Data Knowl. Eng.*, vol. 59, no. 3, pp. 725–769, 2006.

[11] Martin Atzmueller, Fabian Haupt, Stephanie Beer, and Frank Puppe, "Knowta: Wiki-Enabled Social Tagging for Collaborative Knowledge and Experience Management," in *Proc. Intl. Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS)*, 2009, vol. CEUR-WS.

[12] Martin Atzmueller, Florian Lemmerich, Jochen Reutelshoefer, and Frank Puppe, "Wiki-Enabled Semantic Data Mining - Task Design, Evaluation and Refinement," in *CEUR-WS 545*, 2009.

[13] Karin Becker and Cinara Ghedini, "A Documentation Infrastructure for the Management of Data Mining Projects," *Information & Software Technology*, vol. 47, no. 2, pp. 95–111, 2005.

VALIDATION OF A DATA MINING METHOD FOR OPTIMAL UNIVERSITY CURRICULA

R. Knauf *

Ilmenau University of Technology
Faculty of Computer Science
and Automation
PO Box 100565, 98684 Ilmenau
Germany

Y. Sakurai, K. Takada, S. Tsuruta

Tokyo Denki University
School of Information Environment
2-1200 MuZai Gakuendai
Inzai, Chiba, 270-1383
Japan

ABSTRACT

The paper deals with modeling, processing, evaluating and refining processes with humans involved like learning. A formerly developed concept called storyboarding has been applied at Tokyo Denki University to model the various ways to study at this university. Along with this storyboard, we developed a data mining technology to estimate success chances of curricula. Here, we introduce a validation method for this technology and its results. Further, we discuss chances to improve these results by implementing a formerly introduced learner profiling concept that represents the students' individual properties, talents and preferences for personalized data mining.

Index Terms— modeling learning processes, storyboarding, data mining, validation

1. INTRODUCTION

Learning systems suffer from a lack of an explicit and adaptive didactic design. University education is especially effected by this lack, because university professors are not necessarily educational experts. One way of didactic support is providing a modeling concept for didactic design, which allows the anticipation of the learning processes.

An explicit formal didactic design provides a firm basis to verify and validate the didactics behind a learning process by knowledge engineering techniques such as machine learning and data mining. A modeling concept called storyboarding [1] has been developed formerly as a means of modeling learning processes. Besides providing didactic support, this semi-formal model is setting the stage to apply knowledge engineering technologies to verify and validate the didactics behind a learning process. The verification may include both logical consistency issues and formally to check didactic issues. According to different learning and teaching preferences, it includes alternative paths and possible detours if certain concepts to be learned need reinforcement. Using modern media technology, a storyboard also plays the role of a server that provides the appropriate content material.

By storyboarding, didactics can be refined according to revealed weaknesses and proven excellence. Successful didactic patterns can be explored by applying data mining techniques to the various ways students went through a storyboard and their related success. As a result, future instructors and students may utilize these results by preferring those ways through a storyboard, which turned out to be the most promising ones. In [2], a data mining technology, which allows students to utilize mined "experience" of former students to compose curricula with an optimal success chance, is introduced.

However, so far we did not have a practically proven significance, that this method is appropriate. The basic problem so far was the collection of data, which has to be accumulated during a complete undergraduate study, which needs a period of four years. Meanwhile, we could gain a significant amount of data to validate the technology.

The paper is organized as follows. Section 2 introduces the storyboard concept including the present state of the current development. Section 3 provides an overview on our data mining technique to compose optimal curricula for university studies. In section 4, we describe the available data. Section 5 introduces the validation technology and provides its results. In section 6, we outline a refinement of the technology and section 7 summarizes the paper.

2. STORYBOARDING

Our storyboard concept was introduced in [1] und later refined (see [2] for the latest version). A storyboard is a nested hierarchy of directed graphs with anno-

*This author performed the work while at Tokyo Denki University and was sponsored by the Japan Society for the Promotion of Science (JSPS) with an Award-Fellowship for Rainer Knauf (Fellow's ID S-08742) and the Research Institute for Science and Technology of Tokyo Denki University.

tated nodes and annotated edges. Nodes are *scenes* or *episodes*. *Scenes* are not further structured, *episodes* have a sub-graph as its implementation. Also, there is exactly one *start node* and one *end node* in each graph. Edges specify transitions between nodes and may be single-color or bi-color. Nodes and edges can carry attributes.

A storyboard may be seen as a model of an anticipated reception process that is interpreted as follows.

Scenes denote a non-decomposable learning activity that can be implemented in any way, e.g. by the presentation of a (media) document, opening a tool that supports learning (an URL or an e-learning system) or an informal activity description. *Episodes* are defined by their sub-graph. *Graphs* are interpreted by the paths, on which they can be traversed.

A *start node* of a graph defines the starting point of a legal graph traversing. An *end node* of a graph defines the final target point of a legal graph traversing.

Edges denote transitions between nodes. There are rules to leave a node by an outgoing edge, namely (1) The outgoing edge must have the same color as the incoming edge by which the node was reached and (2) If there is a condition specified as the edge's key attribute, this condition has to be met for leaving the node by this edge. So the colors express the dependence of ways leaving a node from the way of arriving there.

Key attributes of nodes specify application driven information, which is necessary for all nodes of the same type, e.g. actors and locations. *Key attributes of edges* specify conditions, which have to be true for traversing on this edge. Free attributes specify whatever the storyboard author wants the user to know: didactic intentions, useful methods, necessary equipment, e.g. For further information, the reader may see [3] or [4].

3. CURRICULUM VALIDATION BY DATA MINING

A basic objective of storyboarding is to use knowledge engineering technologies on the (semi-) formal process models [3] [4].

In particular, we aim at inductively "learning" successful storyboard patterns and recommendable paths. This is some sort of meta-learning, i.e. the learning of learning knowledge. It is performed by an analysis of the paths where former students went through the storyboard [2].

To show the feasibility and benefit of high level storyboarding for its qualified assistance of students suffering from the "jungle of opportunities and constraints" in university education, we developed a simple prototype storyboard for curricula of a university study.

This prototype is used to validate curricula, which are created or modified by the students in advance of their study [4][2] based on the success of former students, who went a similar path through their study.

For this purpose, we introduced a concept to estimate success chances of curricula, which are composed by students at the School of Information Environment of the Tokyo Denki University in their curriculum planning class in the first semester. Along with the estimation, the students also receive (1) a significance of the provided estimation statement (according to the sufficiency of the available data) and (2) a recommendation for modifications of their plan with respect to an optimal success chance.

For such curricula we developed a data mining technique, which is applied to storyboard paths that (former) students went. Based on these examples, the success chance of intended paths can be estimated [2].

The data mining technique is applied to the paths of students through a storyboard, which anticipates possible ways through a complete study.

In a pre-processing step to determine the paths, the individually visited items (episodes and scenes) in the storyboard graph-hierarchy are "flatten down" to a big graph that contains scenes only. This is performed by systematically replacing episodes by the individually visited items of the episode's related sub-graph.

In the granularity of this storyboard application, a scene is a course that holds over one semester. As a result, we have a linear list of course sets, in which each list item is the set of courses that the student took in the subsequent semesters.

The technique consists of two steps, namely (1) constructing a decision from the examples of former students and (2) applying this decision tree to the planned curricula.

The decision tree is based on the concept of bundling common starting sequences of the various paths to a node of the tree. Different subsequent following (next) nodes of the paths will result in different sub-trees right below the actual root on the last node of the common starting sequence.

This continues for each lower level sub-tree accordingly. If there are different paths with a common starting sequence from the root to the actual root different in the next (subsequent) nodes, related sub-trees will be established.

The utilization or application of this decision tree is performed as follows.

If a submitted path is already represented in the decision tree, the prediction or estimation is very easily done through presenting the average Grade Point Average (average of a numeric performance metric of a student over all subjects, weighted by the number of each subject) that students gained, who went exactly this paths, too.

In the other case, the longest leading (starting and its succeeding) part in common with the path representing the submitted curriculum plan will be identified and

code	subject
1	Advanced Project A
2	Advanced Project B
3	Agent Technology
⋮	⋮
155	Workshop

Table 1. Subject list

the average GPA of all students' paths in the sub-trees that start from that point, will be presented as a success estimation. Additionally, the degree of similarity and a recommended change of the submitted path will be presented. T he data mining technology is described more detailed in [2].

4. DATA PREPROCESSING

We collected 188 individual storyboard paths of students, who studied Information Environment at the School of Information Environment of Tokyo Denki University from 2005 till 2009.

From these samples, we removed two samples of students, who joint the university after taking several semesters elsewhere, because their marks were derived by recognition of marks received in similar subjects at another university. This led to 186 samples.

After collecting and studying all the samples and organizational material rules to compose a curriculum, which was available in Japanese only, we chose a compact data representation by coding the particular subjects and the particular students. Table 1 shows an extract from the subject coding list.

By using subject codes 1-155 and student IDs 1-186, we composed a complete decision tree from the 186 samples.

To make sure that identical starting sequences of semester curricula really end up in the same path, the decision tree is well sorted: (1) the subject sequence within a semester is sorted by ascending subject codes and (2) the students samples are sorted by the code lists, which are, compared element by element, ascending, too. We adopted this technology from a similar technology, which is usually performed in data mining for item lists to efficiently generate association rules.

Figure 1 shows an extract of the decision tree composed by all the samples. For each student (coded by his/her ID),

- each semester (columns s, with yellow-brown background),
- the subjects (courses, columns c with light green background),
- their number of units (columns u with light yellow background) and
- the achieved results (with light blue background), i.e. the mark (columns m: S, A, B, C, D, or E) and the number of grade points (columns GP: 4, 3, 2, or 0)

are listed up.

The last row contains a weighted (by the number of units) grade point average GPA, which quantifies the degree of success in the study. Again, both the subject lists of the students within a semester and the complete students' samples (which are lists of lists), are sorted by subject code. The bars between the paths show, up to which semester the curricula of adjacent students are identical (circles) respectively from which semester they are different from each other (bullets). Thus, the grey bars separate the sub-trees from each other.

The entire table has 42 columns and 1616 rows. Figuratively spoken, the table illustrates the decision tree in a horizontal direction wit the root being on the very left hand side and the leaves being on the very right hand side. The grey bars separate sub-trees from each other.

Before applying the validation technology, we found some "exotic samples" of students, who are not representative. This applies to those students, who never finished their study (as this was the case with students 8, 11, 59, 97, 113, 118, 121 and 153) and removed them because of incomplete data, i.e. 177 samples left. As a "learning curve", in future validations, we will leave at least those "dead end" paths in the set, which are caused by a lack of performance.

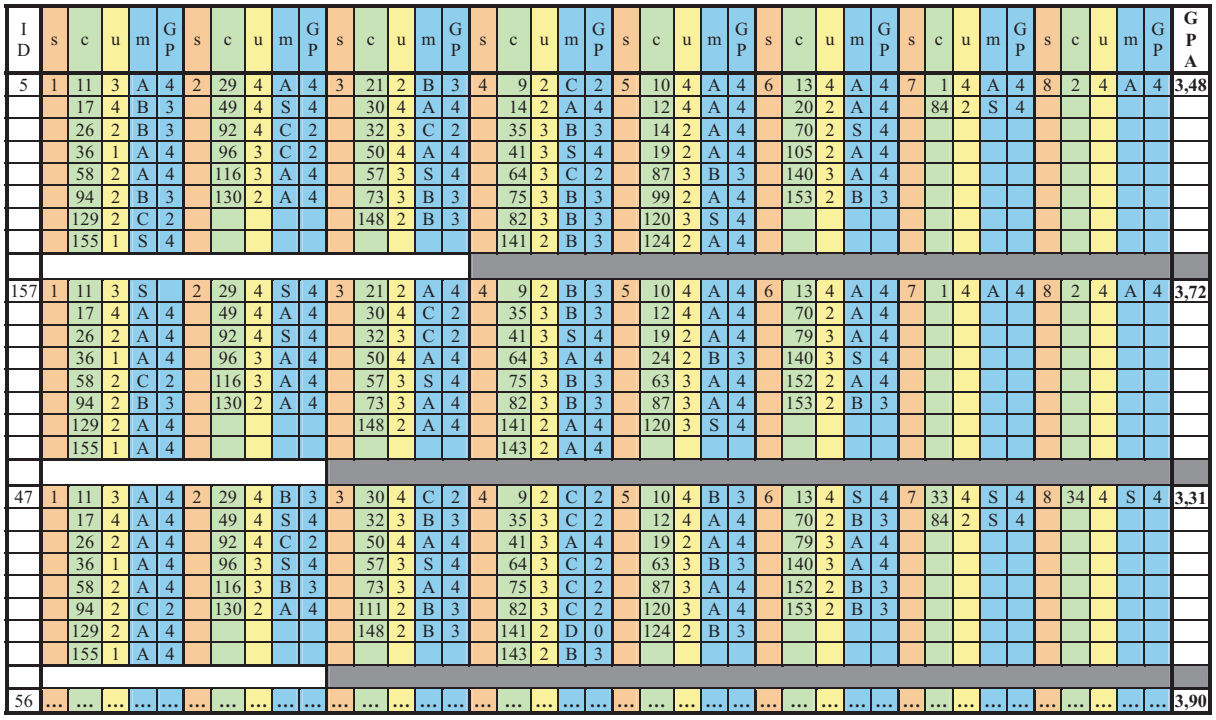
Our validation technology uses an example set to construct a decision tree and a test set to check its performance. Both the example set and the test set are recruited from the given samples.

Those storyboard paths, which are unique and do not have anything in common with any other path, are not appropriate for such a technology, because the test set origins from the same source of data. If the test set contained samples that do not have anything in common with any path of the decision tree, any data mining can not really work because of missing data.

In practice, our data mining technology degenerates to merge all paths of the decision tree and provides the average degree of success of all former students.

Since this is not really a result of data mining, we excluded such paths, which led us to 104 remaining paths, which are used to validate the technology.

For practical use in the success estimation of new paths submitted by students, however, we kept these 73 "lonely" paths, of course, because new paths may be similar to them as well. In fact, any new path is "lonely" when somebody goes it the first time, before it may gain popularity and grow evolutionary towards a sub-tree.



ID	s	c	u	m	GP	s	c	u	m	GP	s	c	u	m	GP	s	c	u	m	GP	s	c	u	m	GP	s	c	u	m	GP	s	c	u	m	GP	s	c	u	m	GP	GPA
5	1	11	3	A	4	2	29	4	A	4	3	21	2	B	3	4	9	2	C	2	5	10	4	A	4	6	13	4	A	4	7	1	4	A	4	8	2	4	A	4	3,48
		17	4	B	3		49	4	S	4		30	4	A	4		14	2	A	4		12	4	A	4		20	2	A	4		84	2	S	4						
		26	2	B	3		92	4	C	2		32	3	C	2		35	3	B	3		14	2	A	4		70	2	S	4											
		36	1	A	4		96	3	C	2		50	4	A	4		41	3	S	4		19	2	A	4		105	2	A	4											
		58	2	A	4		116	3	A	4		57	3	S	4		64	3	C	2		87	3	B	3		140	3	A	4											
		94	2	B	3		130	2	A	4		73	3	B	3		75	3	B	3		99	2	A	4		153	2	B	3											
		129	2	C	2							148	2	B	3		82	3	B	3		120	3	S	4																
		155	1	S	4												141	2	B	3		124	2	A	4																
157	1	11	3	S		2	29	4	S	4	3	21	2	A	4	4	9	2	B	3	5	10	4	A	4	6	13	4	A	4	7	1	4	A	4	8	2	4	A	4	3,72
		17	4	A	4		49	4	A	4		30	4	C	2		35	3	B	3		12	4	A	4		70	2	A	4											
		26	2	A	4		92	4	S	4		32	3	C	2		41	3	S	4		19	2	A	4		79	3	A	4											
		36	1	A	4		96	3	A	4		50	4	A	4		64	3	A	4		24	2	B	3		140	3	S	4											
		58	2	C	2		116	3	A	4		57	3	S	4		75	3	B	3		63	3	A	4		152	2	A	4											
		94	2	B	3		130	2	A	4		73	3	A	4		82	3	B	3		87	3	A	4		153	2	B	3											
		129	2	A	4							148	2	A	4		141	2	A	4		120	3	S	4																
		155	1	A	4												143	2	A	4																					
47	1	11	3	A	4	2	29	4	B	3	3	30	4	C	2	4	9	2	C	2	5	10	4	B	3	6	13	4	S	4	7	33	4	S	4	8	34	4	S	4	3,31
		17	4	A	4		49	4	S	4		32	3	B	3		35	3	C	2		12	4	A	4		70	2	B	3		84	2	S	4						
		26	2	A	4		92	4	C	2		50	4	A	4		41	3	A	4		19	2	A	4		79	3	A	4											
		36	1	A	4		96	3	S	4		57	3	S	4		64	3	C	2		63	3	B	3		140	3	A	4											
		58	2	A	4		116	3	B	3		73	3	A	4		75	3	C	2		87	3	A	4		152	2	B	3											
		94	2	C	2		130	2	A	4		111	2	B	3		82	3	C	2		120	3	A	4		153	2	B	3											
		129	2	A	4							148	2	B	3		141	2	D	0		124	2	B	3																
		155	1	A	4												143	2	B	3																					
56	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	3,90

Fig. 1. Extract from the decision tree data

5. VALIDATION TECHNOLOGY AND RESULTS

There are several approaches to validate data mining technologies.

The *holdout method* splits the data into a training set and a test set, typically in the ratio 2/3 by 1/3. The data mining technology is applied to the training set and validated with the test set. This method suffers from the fact that it does not use the available data exhaustively. A sample, which is in the test set, is not available for building the model (the decision tree, in our case) and thus, decreases the performance of the model. Thus, some performance features of the data mining technology may not be revealed by such a testing method. The splitting ratio is a trade off between the quality of the model and a trustable statement about the performance of the data mining technology.

Random sub-sampling is a refinement of this method, which is a repeated holdout with various splits of the available data and thus, uses the data a little more exhaustively. However, there is no control on the issue, how often a data object is used for building the model and how often it is used for test.

A more exhaustive utilization of the available data is done by *cross validation*. Here, each data object is used for training with the same frequency and for test exactly once. The data set is split into k equally sized subsets. In k cycles, each subset is used for test, whereas the the other $k-1$ sets is used for training.

stud. ID	GPA	GPA estimation	difference
89	3.40	3.23	0.17
148	3,04	3,26	0,22
179	3,30	3,24	0,06
92	3,55	3,63	0,08
178	3,91	3,40	0,51
164	3,29	3,71	0,42
177	3,52	3,60	0,08
⋮	⋮	⋮	⋮

Table 2. Validation results

The *leave one out* approach is a special case of cross validation with k being the number of data objects and makes the most exhaustive use of the data.

Finally, we used this approach to validate our data mining technology. In 104 cycles, we removed one path from the complete decision tree and used this sample to check the remaining decision tree.

As a result, we received a list of all the 104 samples along with their original GPA and the GPA as estimated by the data mining technology as shown in Table 2. The mean of the difference between both was 0.43 with a standard deviation of 0.30.

Having in mind that this result is just based on a statistical analysis of former students' curricula and their related success, an average error of 0.43 grade points is

not too bad and promises remarkable results, when the learner' individual characteristics are also included in the data mining technology.

6. PERSONALIZED DATA MINING AND ITS REALIZATION

Individual learning plans should not only be based on the success of former students who went similar ways. Additionally, individual properties, talents and preferences should be considered.

For example, some students are more talented for analytical challenges, some are more successful in creative or composing tasks, and others may have an extraordinary talent to memorize a lot of factual knowledge. Consequently, we need to include individual learner profiles to avoid lavishing the students with suggestions that don't match their individual preferences and talents.

In [5], we introduced an approach of personalized data mining. This approach adopts the GARDNER'S theory of multiple intelligences [6] and the learning style model of FELDER and SILVERMAN [7]. The assumption behind this approach is that there is a link between

- typical "competence traits" (according to GARDNER) and subjects that typically challenge the one or other "kind of intelligence" more than others and
- typical teaching methods (according to FELDER and SILVERMAN) and subjects that are typically taught with these methods.

According to [5], the next steps of collecting and processing data to integrate this technology, are (1) the appraisal of the learner profile introduced in [5] for the very best students in each subject, (2) the derivation a typical "success profile" for each subject, (3) the estimation of learner profiles for all students as a (by success degree) weighted average success profile of the subjects they took, and (4) the application of the same technology to the data of "personalized" decision trees for each learner, which are composed by samples of learners, which have a similar learner profile.

The appraisal of the GARDNER - like items in the learner profile can be performed by a questionnaire, which derives an estimation of a human's intelligence distribution by his/her answers on 70 questions. This questionnaire is available to the public in the Internet as a downloadable Microsoft Excel file.[1]

The FELDER-SILVERMAN - like items of the learner profile can be estimated by a questionnaire as well. This questionnaire is also available to the public in the Internet.[2]

[1] see http://www.businessballs.com/howardgardnermultiple...
...intelligences.htm

[2] see http://www.engr.ncsu.edu/learningstyles/ilsweb.html

attribute	attribute description	value range
d_1	Linguistic intelligence	$0 \leq v_1 \leq 1$
d_2	Logical-mathematical intelligence	$0 \leq v_2 \leq 1$
d_3	Musical intelligence	$0 \leq v_3 \leq 1$
d_4	Bodily-kinesthetic intelligence	$0 \leq v_4 \leq 1$
d_5	Spatial intelligence	$0 \leq v_5 \leq 1$
d_6	Interpersonal intelligence	$0 \leq v_6 \leq 1$
d_7	Intrapersonal intelligence	$0 \leq v_7 \leq 1$
d_8	Active vs. Reflective style	$0 \leq v_8 \leq 1$
d_9	Sensing vs. Intuitive style	$0 \leq v_9 \leq 1$
d_{10}	Visual vs. Verbal style	$0 \leq v_{10} \leq 1$
d_{11}	Sequential vs. Global style	$0 \leq v_{11} \leq 1$

Table 3. Derived Learner Profile

We consider both in our model, which is defined as an array of 11 attribute-value pairs that contains 7 intelligence attributes and 4 learning style attributes. Both can be appraised by questionnaires that are available to the public in the web.

To make the dimensions of both sources comparable to each other and see the quantitative relations, we normalized them in a way that they all have the same range of values. The intelligence dimensions rage from 10 to 40. The learning style dimensions range from -11 to +11 (opposite algebraic sign for opposite styles). The normalization can be done by

- $v = result/40$ for the intelligence dimensions according to GARDNER and
- $v = (result + 11)/22$ for the learning style dimensions accodrding to FELDER and SILVERMAN.

Finally, our learner model looks as shown in Table 3.

However, it turned out to be very hard to find former students, who are still accessible and, moreover, willing to fill in such questionnaires to obtain their learner profiles. Our students are very sensible in respecting privacy and, vice versa, in expecting the same respect from others. Since answers to the questions in the questionnaire may reveal some private issues, it is hard to ask them to answer these questions.

However, there are some students, who we dare to ask for filling in the questionnaires because they had a quite confidential relation to the one or other professor, but these students are not necessarily the best ones.

Therefore, steps one and two of this plan need to be changed. To infer a typical "success profile" of a subject, we can collect the questionnaire answers be some student, which are not necessarily the best ones.

Thus, we modified the approach of computing an "average profile" of the best students towards a

"weighted average profile" of all available students, who took part in a particular subject.

Let $L(s)$ be the set of learners, who took part in the subject s and for who a learner profile can be composed from the questionnaires' answers. So for each learner $l^i \in L(s)$, $i = 1...|L(s)|$, a learner profile $p(l^i) = [d_1^i, d_2^i, \cdots, d_{11}^i$ is available. Let

$$succ_s^i = \begin{cases} 1.00 & \text{, if } l^i \text{ received in subject } s \text{ mark } S \\ 0.80 & \text{, if } l^i \text{ received in subject } s \text{ mark } A \\ 0.60 & \text{, if } l^i \text{ received in subject } s \text{ mark } B \\ 0.40 & \text{, if } l^i \text{ received in subject } s \text{ mark } C \\ 0.20 & \text{, if } l^i \text{ received in subject } s \text{ mark } D \\ 0.00 & \text{, if } l^i \text{ received in subject } s \text{ mark } E \end{cases}$$

be the success degree of the learner $l1i$ in subject s.

By using this success degree as a weight factor, the "typical success profile" of a subject s can be computed as

$$p(s) = \frac{1}{\sum_{i=1}^{|L(S)|} succ_s^i} \begin{pmatrix} \sum_{i=1}^{|L(s)|}(succ_s^i * d_1^i) \\ \sum_{i=1}^{|L(s)|}(succ_s^i * d_2^i) \\ \vdots \\ \sum_{i=1}^{|L(s)|}(succ_s^i * d_{11}^i) \end{pmatrix}$$

This calculation has to be done for each subject separately and the set of "most successful students" differs from subject to subject, of course. The idea behind is to mine a "typical success profile" for each subject separately.

After performing these computations, steps three and four can be conducted as planned originally and described in [5]. As a result of processing this additional data in the way sketched above, we expect a remarkable improvement the performance compared to the results presented in section 5.

7. SUMMARY AND OUTLOOK

The research reported here is focused on modeling, processing, evaluating and refining processes with humans involved like learning. A formerly developed concept called storyboarding is briefly introduced.

Along with a storyboard application, we developed a data mining technology to estimate success chances of curricula, which are composed by students. So far, there was no practical significance for the performance of this technology.

The basic problem so far was the collection of data, which has to be accumulated during a complete undergraduate study of, which needs a period of four years. Meanwhile, we could gain a significant amount of data to validate the technology.

By cross validation with the available data, we could empirically show performance of our data mining technology.

However, the currently implemented way of statistically analyzing all former students' curricula ignores the fact that the success chance heavily depends on individual properties.

A formerly developed approach to validate curricula personalized by building the decision tree based on former students with a similar learner profile only, was refined here. This was necessary, because the required personal data is not available.

As a result of practically implementing this refined approach, we expect a remarkable improvement of these results.

8. REFERENCES

[1] K.P. Jantke and R. Knauf, "Didactic design though storyboarding: Standard concepts for standard tools," in *Proc. of 4th Int. Symposium on Information and Communication Technologies, Workshop on Dissemination of e-Learning Technologies and Applications, Cape Town, South Africa*. 2005, ISBN 0-9544145-6-X, pp. 20–25, New York: ACM Press.

[2] R. Knauf, R. Böck, Y. Sakurai, S. Dohi, and S. Tsuruta, "Knowledge mining for supporting learning processes," in *Proc. of the 2008 IEEE Int. Conference on Systems, Man, and Cybernetics (SMC 2008), Singapore*. IEEE, Piscataway, NJ, USA, 2008, IEEE Catalog number CFP08SMC-USB, ISBN 978-1-4244-2384-2, Library of Congress: 2008903109, pp. 2615–2621.

[3] R. Knauf, Y. Sakurai, S. Tsuruta, and K.P. Jantke, "Modeling didactic knowledge by storyboarding," *Journal of Educational Computing and Research*, vol. 42, no. 4, pp. in press, 2010.

[4] Y. Sakurai, S. Dohi, S. Tsuruta, and R. Knauf, "Modeling academic education processes by dynamic storyboarding," *Journal of Educational Technology & Society*, vol. 12, no. 2, ISSN 1436-4522 (online) and 1176-3647 (print), pp. 307–333, April 2009.

[5] R. Knauf, Y. Sakurai, S. Tsuruta, K. Takada, and S. Dohi, "Personalized curriculum composition by learner profile driven data mining," in *Proc. of the 2009 IEEE Int. Conference on Systems, Man, and Cybernetics (SMC 2009), San Antonio, TX, USA*, 2009, ISBN 978-1-4244- 2794-9, pp. 2137–2142.

[6] H. Gardner, *Frames of Mind: The Theory of Multiple Intelligences*, New York: Basic Books, 1993.

[7] R.M. Felder and L.K. Silverman, "Learning and teaching styles in en-gineering education," *Engineering Education*, vol. 78, no. 7, pp. 647–681, 1988.

Part II

First International Workshop on Evolution Support for Model-Based Development and Testing (EMDT2010)

Preface

Welcome to the First International Workshop on Evolution Support for Model-Based Development and Testing (EMDT2010), September 16, 2010, Ilmenau, Germany.

History The workshop continues a series of successful workshops on software evolution on the national level: Object-orientation, Reengineering and Architecture ORA2006 and ORA2007, Model-driven Software architecture – Evolution, Integration and Migration MSEIM 2008 and MSEIM 2009. With EMDT2010 we wanted to extend the series on an international level to integrate it with a wider community.

Motivation The growing size and complexity of modern systems is one of the major reasons for the adaptation of model-based development and testing techniques. There is an increase in academic and industrial interest in model-based and model-driven development in recent years. However, the rapid evolution of systems due to changing requirements and technological advancements is still a challenge for practitioners and researchers. The goal of this workshop is to identify the key challenges, research questions and ideas for the support of evolution in software development and testing. With EMDT2010 we intended to bring together the industrial practitioners and academic researchers to exchange their experiences and ideas.

Topics of Interest Topics of interest for the workshop include but are not limited to:

- Architectural design methods supporting evolution and evolvability
- Model-driven software evolution and maintenance
- Traceability from requirements models to design and test models
- Model-based reengineering and refactoring for evolution support
- Model comparison and impact analysis
- Model transformation for test generation
- Model-based testing, validation and verification
- Model-based test specification
- Test model evolution and regression testing
- Model-based test processes
- Evolution support for system management
- Tool support for model-based development and testing
- Case studies and application of model-based development and testing
- Experiences of using models and relating models with their applications in the real-world development process.

Workshop Format & Facts To enable interaction and discussion between the participants, the workshop was held in two parts. First, the full papers and invited talks were presented including a short discussion after each presentation. In the second part, short position statements were given, to initiate a focussed discussion with all attendees on key challenges, research questions and ideas for the support of evolution in software development and testing.

There were six submitted full papers. The reviewing process was performed anonymously with three peer reviews per submission. We could accept three full papers. This results in an acceptance rate of 50% for full papers for EMDT2010. Furthermore, we had one invited talk with an additional paper. Moreover, it was a great pleasure to have Prof. Bernd-Holger Schlingloff from Humboldt University, Berlin, Germany as a keynote speaker. During our workshop we had a number of 12 participants, which contributed to our discussion

Acknowledgement We would like to thank the contributors to the workshop for making this workshop possible – the authors for their submissions, the speakers for their presentations. We would like to express our special thanks to the program committee members for their extensive feedback in the reviews, which contributed to high quality level of the discussions. We also thank all participants for their comments in the discussions and the organizers of the umbrella conference IWK for their support in organizing the workshop.

Ilmenau, September 2010

Stephan Bode
Qurat-Ul-Ann Farooq
Matthias Riebisch

Organizing Committee

Stephan Bode, Ilmenau University of Technology
Qurat-Ul-Ann Farooq, Ilmenau University of Technology
Matthias Riebsich, Ilmenau University of Technology

Program Committee

Stephan Bode, Ilmenau University of Technology, Germany
Steve Counsell, Brunel University, UK
Zhen Ru Dai, Hamburg University of Applied Science, Germany
Qurat-ul-Ann Farooq, Ilmenau University of Technology, Germany
Peter Hänsgen, Intershop AG, Jena, Germany
Wilhelm Hasselbring, University Kiel, Germany
Zohaib Zafar Iqbal, Simula Research Labortries, Norway
Norbert Klein, Capgemini sd&m Research, Germany
Michael Mlynarski, University of Paderborn, Germany
Naouel Moha, INRIA, University of Rennes, France
Ilka Philippow, Ilmenau University of Technology, Germany
Matthias Riebisch, Ilmenau University of Technology, Germany
Bernd-Holger Schlingloff, Humboldt University, Germany
Detlef Streitferdt, ABB Corporate Research, Ladenburg, Germany
Mario Winter, University of Applied Science (FH) Koeln, Germany
Tao Yue, Carleton University, Canada
Justyna Zander, Fraunhofer Fokus, Germany & Harvard University, USA
Steffen Zschaler, Lancaster University, UK

Cooperative Organizations

The workshop was organized in cooperation with the SIG OOSE of the German Computer Society GI and with ACM/SIGSOFT.

It was held as part of the umbrella conference 55th International Scientific Colloquium (IWK2010), September 13-17, 2010, Ilmenau, Germany.

KEYNOTE:

MODEL-BASED SOFTWARE DEVELOPMENT – PERSPECTIVES AND CHALLENGES

Bernd-Holger Schlingloff

Humboldt University
Kekuléstr. 7, 12489 Berlin, Germany

ABSTRACT

Model-based software development and testing has turned out to be the method of choice for safety-critical embedded systems. An abstract model reflects requirements and environmental conditions for the system. Such a model can be used in two ways—as a development model in a stepwise refinement process to derive the actual implementation, or as a testing model in order to derive test cases for some system under test. In this talk we discuss commonalities and differences between development models and testing models, discuss the formalization of requirements in models, and show how to automatically evaluate observations about a system with respect to a model. We illustrate our ideas with examples from aerospace, automation and medical devices. Finally, we discuss some recent trends and challenges in the area of model-based development and testing.

(INVITED TALK)

AGILITY VS. MODEL-BASED TESTING: A FAIR PLAY?

Baris Güldali, Michael Mlynarski

Software Quality Lab (s-lab), University of Paderborn
Warburger Str. 100, 33098 Paderborn/Germany
{bguldali,mmlynarski}@s-lab.upb.de

ABSTRACT

Agile manifesto defines principles for a light-weight software development process aiming at an improved customer satisfaction. *Automated testing* plays an important role in fulfilling these principles, because it enables efficient execution of *test scripts* for checking the quality of delivered software. However, the implementation and the maintenance of the test scripts can be very tedious and error-prone. In order to deal with that, *model-based testing* extends the automated test execution by test design and test implementation. Thus, model-based testing can speed up the test automation and improve the maintenance of test scripts. Nevertheless, introducing model-based testing requires some initial and some continual efforts, like creating test models, buying or developing tools, etc. In this talk, we will discuss how model-based testing can support agile development without conflicting with the principles of agile manifesto.

Index Terms - Agile manifesto, Automated testing, Model-based Testing

1. INTRODUCTION

As the complexity of software rises, novel software development techniques are required in order to cope with the technical and the organizational challenges in the development process. Model-based software development (MBSD) proposes using *abstract models* for better communication, for maintainable software specification and for efficient code generation. In this context, model-based testing (MBT) proposes using models for automating some of the testing activities, e.g. test case generation, evaluation of test results, which are tedious and error-prone tasks if they are manually done. In order to profit from model-based techniques in development process, however, some efforts must be expended, e.g. for introducing tools, for training developers and testers, for creating and maintaining models, etc. That is why MBSD is said to be a "heavyweight" technique for creating better software.

In contrast, agile manifesto [1] proposes a "light-weight" development process where (1) individuals and interactions are favored over processes and tools, (2) working software is favored over comprehensive documentation, (3) customer collaboration is favored over contract negotiation and (4) responding to change is favored over following a plan [1]. However, in the practice, these principles are likely to be misinterpreted such that developers often neglect documenting customer requirements properly. Frequently, this leads to chaos in the development process and to conflicts during the delivery and acceptance. Thus, it is a challenge to follow the principles of agile manifesto and thereby not to lose sight of the proper documentation and communication of customer needs and of the efficient and effective development.

We believe that, model-based techniques can help in dealing with these challenges. In the rest of paper, we will discuss how agility and model-based paradigm fits together. Thereby, we will mainly focus on the integration of model-based testing in agile development process as an enabling technology for the principles of the agile manifesto.

2. AGILE MANIFESTO

In 2001 seventeen software experts, who have introduced well-known agile methods (e.g. Scrum, Test driven Development (TDD), Extreme Programming (XP) etc.) have defined common principles for a lightweight development process. The new development paradigm should be an alternative to documentation-driven, heavyweight software development processes. They called these principles "agile manifesto". Agile manifesto includes the following principles (based on [1]):

1. *Customer satisfaction*: The highest priority in agile development has the customer satisfaction, which can be achieved by early and continuous delivery of valuable software. This principle has the highest priority in agile manifesto. All other principles serve to achieve this goal.
2. *Fast adaptation*: In agile development, requirements changes of the customer are

welcome, even in the late phases of the development. The flexibility in agile processes enables changes in software for assuring the customer's competitive advantage.

3. *Frequent delivery*: For customer satisfaction, it is important to show that the development process makes progress. For showing this to the customer, deliver new versions of software frequently. Define together with the customer what "frequent" means. The time slots can range from a couple of weeks to a couple of months. Try to keep the time slots as short as possible, because frequent delivery leads to frequent feedback.
4. *Close collaboration*: For achieving fast adaptation and frequent delivery, it is important to understand customer's business needs and consider them during the development continuously. For that, business people and developers must work together every day throughout the project.
5. *Motivated members*: Identify motivated team members who can push on the project. Provide them with the resources they need and support them while getting the job done.
6. *Conversation*: For achieving fast adaptation and frequent delivery, besides close collaboration with the customer, also the efficient communication between team members is important. The most efficient and effective method of exchanging information is face-to-face conversation.
7. *Working software*: Supply the customer with working software which is the main measure of progress. Delivering working software is indispensible for customer satisfaction.
8. *Sustainable development*: Agile processes promote sustainable development.
9. *Constant pace*: The customers and developers should be able to keep a constant pace for the whole time of project.
10. *Good design*: Continuous awareness for technical quality and good design improves agility.
11. *Simplicity*: Simplicity is crucial, which means that the amount of work to be done should be kept minimal.
12. *Self-organization*: Motivate team members to organize themselves.
13. *Reflection*: Motivate team members to reflect their experiences at regular intervals. Team members should discuss on how to improve the effectiveness and the efficiency in team and should suggest improvements accordingly.

Existing agile methods aim at enabling these principles. For example, Scrum promotes the close collaboration of customer or product owner at identifying software functionalities to be implemented in the next development cycles [4]. TDD advocates continuing programming until all predefined test cases are passed [1]. Since test cases are seen as specification, the resulting software is assumed to be correct with respect to the specification. Test automation plays in important role in agile methods supporting an efficient and effective development process. Having different focus, agile methods mostly should be combined in order to fulfill all principles of agile manifesto.

3. MODEL-BASED TESTING VS. AGILITY

We believe that model-based techniques can help in combining the different tasks in agile development by using abstract models as primary development artifacts. Models can support *communication* between team members and customers, *documentation* of customer requirements and design decisions and *automation* of code generation and testing. Thus, model-based techniques can enable an integrated development throughout the whole project. As next, we want to focus on how the documentation of customer requirements and their validation can be supported by model-based testing while following principles of agile manifesto.

3.1. Model-based Testing

With the emerging popularity of model-based software development, the usage of models in software testing is also desired. There are several definitions of model-based testing (MBT) in the literature, but the common understanding is that MBT is "the automation of test design of black-box tests" [2]. Therefore, MBT uses abstract models (test models) of the system under test (SUT) or its environment as the source for test generation. In addition to models of SUT and the environment, also the testware itself can be modeled [3].

There are three main advantages of MBT, which make this technique interesting: a) enabling high coverage, b) need for lower effort and c) enabling early testing. Because MBT uses sophisticated algorithms and tools for automatic test generation, far more test cases than while manual testing can be generated. This way a very high coverage of the system specification and/or requirements can be reached. While test cases are not designed and implemented manually anymore, the effort for this task is significantly low. This works under the assumption that the modeling effort is lower than the manual test design activity. Last but not least the early creation of test models supports the validation of requirements even before the system is implemented.

3.2. MBT as a technical enabler for Agility

Using MBT, the requirements can be captured and communicated in form of models. Unified Modeling Language (UML) provides many types of visual diagrams for describing the desired structure and behavior of software. Most of the diagrams have a quite simple syntax and fairly clear semantics such that customer and developer can easily learn how to

express their requirements more precisely, thus enabling the principle *close collaboration*. The changes in requirements can easily be made on the already created models, thus improving *fast adaptation*. Models can also support the *conversation* between team members, where the results of a discussion can be edited into the models immediately. Also the *simplicity* principle can be supported by models by using the abstraction, modularization and decomposition features of modeling.

There are different scenarios for creating and using models in MBT [9]. While some scenarios propose sharing models (one model for test team and development team), some scenarios require separated models (one models for each test and development team respectively). Using shared models can support *close collaboration,* face-to-face *conversation* and *simplicity*. However, if same models are used for development and testing, specification errors cannot be found [9]. Using separate models makes the teams for development and test more independent and enables finding specification errors, thus assuring *working software*.

Models having a well-defined syntax and semantics can be handled by computers, which obviously bring efficiency into the test process. The state-of-the-art modeling techniques support creating *good design*. Depending on the context of development, formal or semi-formal notations can be used. The more formal the models are, the better automatable are the test activities. Especially the automation of the test design task, which is the most costly and time consuming part in testing [5], leads to more efficiency. Test automation is the key for assuring *working software, frequent delivery*, *sustainable development* and *constant pace*.

Within MBT several coverage criteria for selecting test cases can be used. One possibility is to cover the customer requirements, which directly correlates with several agile principles. The *customer satisfaction* and *close collaboration* principles are supported by refining and understanding customer requirements while modeling them and showing that those requirements were successfully tested. The usage of different selection criteria and possibly combining them leads to higher defect detection rate and therefore facilitates *working software*. Due to changeable coverage criteria and automated test case generation, the test team can conduct different testing scenarios and gain experience for further development cycles and projects. This flexibility and configurability of MBT enables *reflection* in agile development.

4. A FAIR PLAY?

As discussed in the last section, MBT can definitely enable many principles of the agile manifesto. The main advantage of MBT for the agile world is the usage of models as primary artifacts and the automation of several test activities. This way MBT fits very well with agility!

However, MBT is not for free. Introducing MBT into the agile development process requires some initial and continual efforts as discussed in [6] and [7]. These include:

- Training team members for modeling
- Buying or developing modeling tools
- Buying or developing test drivers and test adapters
- Defining modeling notations and test selection criteria
- Creating and maintaining models
- Eventually extending generated test cases by test data
- Eventually evaluation of test results

At first sight, these efforts seem to be not proportional to the lightweight development purposes of agile manifesto. However, test automation is an indispensible part of agility enabling the efficient and effective process. Fewster and Graham said in 1999 that "automating chaos just gives faster chaos". MBT is an attempt to make test automation more systematic, more maintainable.

In this paper, we have discussed how agility and MBT conceptually fits together. A concrete approach for combining agility and MBT can be read in [8]. There, we have described a concrete approach including tool support for integrating MBT into Scrum.

5. REFERENCES

[1] Beck, K. et al.: Manifesto for Agile Software Development. Online resource at agilemanifesto.org (Last visited: 29.07.2010)

[2] Utting, M. and Legeard, B.: Practical Model-Based Testing: A Tools Approach, Morgan Kaufmann, 2007

[3] Baker, P. et al.: Model-Driven Testing: Using the UML Testing Profile, Springer Verlag, 2008

[4] Schwaber, K., and Beedle, M.: Agile Software Development with Scrum, Prentice Hall, 2002.

[5] Pol, M. and Koomen, T. and Spillner, A.: Management und Optimierung des Testprozesses, dpunkt.verlag, 2002

[6] Güldali, B. and Jungmayr, S. and Mlynarski, M. and Neumann, S. and Winter, M.: Starthilfe für modellbasiertes Testen. OBJEKTspektrum, 2010, 3, 63-69

[7] Güldali, B. and Mlynarski, M. and Sancar, Y.: Effort Comparison of Model-based Testing Scenarios. Proc. of Quombat Workshop at ICST, 2010

[8] Löffler, R., Güldali, B., Geisen, S.: Towards Model-based Acceptance Testing for Scrum. Softwaretechnik-Trends, GI, 2010 (to be published)

[9] Pretschner, A., Philips, J.: Methodological Issues in Model-Based Testing. In M. Broy, et.al. (Eds.), Model-Based Testing of Reactive Systems, LNCS no. 3472, Springer-Verlag, 2005, pp. 281-291.

[10] Beck, K. Test-Driven Development: By Example. Addison-Wesley Longman, 2003

A TEST CASE GENERATION TECHNIQUE AND PROCESS

Nicha Kosindrdecha and Jirapun Daengdej

Autonomous System Research Laboratory
Faculty of Science and Technology
Assumption University, Thailand
p4919741@au.edu, jirapun@scitech.au.edu

ABSTRACT

It has been proven that the software testing phase is one of the most critical and important phases in the software development life cycle. In general, the software testing phase takes around 40-70% of the effort, time, and cost. This area is well researched over a long period of time. Unfortunately, while many researchers have found methods of reducing time and cost during the testing process, there are still a number of important related issues that need to be researched. This paper introduces a new high level test case generation process with a requirement prioritization method to resolve the following research problems: unable to identify suitable test cases with limited resources, lack of an ability to identify critical domain requirements in the test case generation process and ignore a number of generated test cases. Also, this paper proposes a practical test case generation technique derived from use case diagram.

Index Terms - test generation, testing and quality, test case generation, test generation technique and generate tests

1. INTRODUCTION

Software testing is known as a key critical phase in the software development life cycle, which account for a large part of the development effort. A way of reducing testing effort, while ensuring its effectiveness, is to generate test cases automatically from artifacts used in the early phases of software development. Many test case generation techniques have been proposed [2], [4], [10], [11], [12], [15], [21], [22], [42], [47], [50], mainly random, path-oriented, goal-oriented and model-based approaches. Random techniques determine a set of test cases based on assumptions concerning fault distribution. Path-oriented techniques generally use control flow graph to identify paths to be covered and generate the appropriate test cases for those paths. Goal-oriented techniques identify test cases covering a selected goal such as a statement or branch, irrespective of the path taken. There are many researchers and practitioners who have been working in generating a set of test cases based on the specifications. Modeling languages are used to get the specification and generate test cases. Since Unified Modeling Language (UML) is the most widely used language, many researchers are using UML diagrams such as state diagrams, use-case diagrams and sequence diagrams to generate test cases and this has led to model-based test case generation techniques. In this paper, an approach with additional requirement prioritization step is proposed toward test cases generation from requirements captured as use cases [23], [24], [33]. A use case is the specification of interconnected sequences of actions that a system can perform, interacting with actors of the system. Use cases have become one of the favorite approaches for requirements capture. Test cases derived from use cases can ensure compliance of an application with its functional requirements. However, one difficulty is that there are a large number of functional requirements and use cases. A second research challenge is to ensure that test cases are able to preserve and identify critical domain requirements [5]. Finally, a third problem is to minimize a number of test cases while preserving an ability to reveal faults. For example, there are a lot of functional requirements in the large software development. Software test engineers may not be able to design test cases to cover important requirements and generate a minimum set of test cases. Therefore, test cases derived from large requirements or use cases are not effective in the practical large system. This paper presents an approach with additional requirement prioritization process for automated generation of abstract presentation of test purposes called test scenarios. This paper also introduces a new test case generation process to support and resolve the above research challenges. We overcome the problem of large numbers of requirements and use cases. This allows software testing engineer to prioritize critical requirements and reasonably design test cases for them. Also, this allows us to be able to identify a high percentage of each test case's critical domain coverage.

The rest of the paper is organized as follow. Section 2 discusses the comprehensive set of test case generation techniques. Section 3 proposes the outstanding research challenges that motivated this study. Section 4 introduces a new test generation process and technique. Section 5 describes an experiment, measurement metrics and results. Section 6 provides the conclusion and research directions in

the test case generation field. The last section represents all source references used in this paper.

2. LITERATURE REVIEW

Model-based techniques are popular and most researchers have proposed several techniques. One of the reasons why those model-based techniques are popular is that wrong interpretations of complex software from non-formal specification can result in incorrect implementations leading to testing them for conformance to its specification standard [43]. A major advantage of model-based V&V is that it can be easily automated, saving time and resources. Other advantages are shifting the testing activities to an earlier part of the software development process and generating test cases that are independent of any particular implementation of the design [7]. The model-based techniques are method to generate test cases from model diagrams like UML Use Case diagram [23], [24], [33], UML Sequence diagram [7] and UML State diagram [5], [43], [22], [2], [21], [15], [32], [4]. There are many researchers who investigated in generating test cases from those diagrams. The following paragraphs show examples of model-based test generation techniques that have been proposed for a long time.

Heumann [23] presented how using use cases to generate test cases can help launch the testing process early in the development lifecycle and also help with testing methodology. In a software development project, use cases define system software requirements. Use case development begins early on, so real use cases for key product functionality are available in early iterations. According to the Rational Unified Process (RUP), a use case is used to describe fully a sequence of actions performed by a system to provide an observable result of value to a person or another system using the product under development." Use cases tell the customer what to expect, the developer what to code, the technical writer what to document, and the tester what to test. He proposed three-step process to generate test cases from a fully detailed use case: (a) for each use case, generate a full set of use-case scenarios (b) for each scenario, identify at least one test case and the conditions that will make it execute and (c) for each test case, identify the data values with which to test. Ryser [24] raised the practical problems in software testing as follows: (1) Lack in planning/time and cost pressure, (2) Lacking test documentation, (3) Lacking tool support, (4) Formal language/specific testing languages required, (5) Lacking measures, measurements and data to quantify testing and evaluate test quality and (6) Insufficient test quality. They proposed their approach to resolve the above problems. Their approach is to derive test case from scenario / UML use case and state diagram. In their work, the generation of test cases is done in three processes: (a) preliminary test case definition and test preparation during scenario creation (b) test case generation from Statechart and from dependency charts and (c) test set refinement by application dependent strategies.

3. RESEARCH CHALLENGES

This section discusses the details of research issues related to test case generation techniques and research problems, which are motivated this study. Every test case generation technique has weak and strong points, as addressed in the literature survey. In general, referring to the literature review, the following lists major outstanding research challenges. The first research problem is that existing test case generation methods are lack of ability to identify domain specific requirements**.** The study [5] shows that domain specific requirements are some of the most critical requirements required to be captured for implementation and testing, such as constraints requirements and database specific requirements. Existing approaches ignore an ability to address domain specific requirements. Consequently, software testing engineers may ignore the critical functionality related to the critical domain specific requirements. Thus, this paper introduces an approach to priority those specific requirements and generates an effective test case. The second problem is that existing test case generation techniques aim to generate test cases which maximize cover for each scenario. Sometimes, they generate a huge number of test cases which are impossible to execute given limited time and resources. As a result, those unexecuted test cases are useless. The last problem is to unable to identify suitable test cases in case that there are limited resources (e.g. time, effort and cost). The study reveals that existing techniques aim to maximum and generate all possible test cases. This can lead to unable to select necessary test cases to be executed during software testing activities, in case that there are limited resources.

4. PROPOSED METHOD

This section presents a new high-level process to generate a set of test cases introduced by using the above comprehensive literature review and previous works [43].

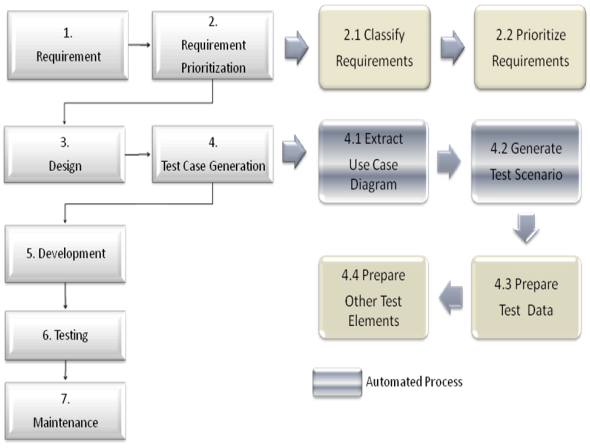


Figure 1 A Proposed Process to Generate Test Cases

From the above figure, the left-hand side process is a general waterfall process. We propose to add two additional processes: (a) requirement prioritization and (b) test case generation.

The requirement prioritization process aims to be able to effectively handle with a large number of requirements. The objective of this process is to prioritize and organize requirements in an appropriate way in order to effectively design and prepare test cases [16], [25], [37]. There are two sub-processes: (a) classify requirements and (b) prioritize requirements.

The classify requirement process primarily divides and classifies requirements into four groups [30]: (a) "Must-Have" (b) "Should-Have" (c) "Could-Have" and (d) "Wish". The "Must-Have" requirements are mandatory requirements that need to be implemented in the system. The "Should-Have" requirements are requirements that should be implemented if there are available resources. The "Could-Have" requirements are additional requirements that are able to be implemented if there are adequate resources. The "Wish" requirements are "would like to have in the future" requirements that may be ignored if there are inadequate resources. This paper introduces five factors to classify the above requirements, as follows:

Table 1 Requirement Classification

Group	Time	Cost	People	Scope	Success
Must have	Y	Y	Y	N	Y
Should have	Y	Y	Y	N	N
Could have	N	N	Y	Y	N
Wish	N	N	N	Y	N

From the above table, the following shortly describes a meaning of the above factors:

- *Time* – The requirement must be implemented in the current version or release of software.
- *Cost* – There is an available of budget or fund to implement the requirement.
- *People* – There is an available of human resources to develop and test the requirement.
- *Scope* – The requirement can be removed out of the current version or release of software.
- *Success* – The success of system development rely on the requirement.

In addition, this paper secondary divides those requirements into two groups: (a) functional and (b) non-functional. The functional requirements can be categorized into two groups: (a) domain specific requirements and (b) non- domain specific requirements. The domain specific requirements are able to identify as database specific and constraints requirements. For example, database connection specific requirements and requirements for an interface with other systems. The non-functional requirements can be vary, such as performance, security, operability and maintainability requirements. The following displays the classify requirement tree:

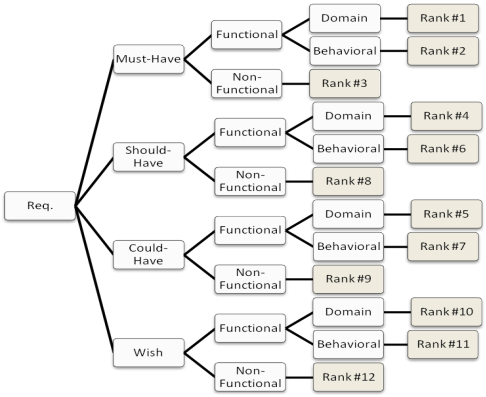


Figure 2 A Classify Requirement Tree

From the above figure, we propose a ranking number for each requirement. This paper prioritizes "Must-Have" requirements as top three ranking and "Wish" requirements as last three ranking. The study [5] reveals that domain specific requirements should have higher priority than both of behavioral and non-functional requirements.

However, when the requirement is already classified, the next process is to prioritize those requirements. In the requirement prioritization process, this paper proposes to use a cost-value approach to weight and prioritize requirements. This paper also proposes to use the following formula:

$$P(Req) = (Cost * CP) \quad (1)$$

Where:

- *P* is a prioritization value.
- *Req* is a requirement required to be prioritized.
- *Cost* is a total estimated cost of coding and testing for each requirement.
- *CP* is an user-defined customer priority value. This value is in the range between 1 and 10. 10 is the highest priority and 1 is the lowest priority. This value aims to allow customers to identify how important of each requirement is from their perspective.

To compute the above cost for coding and testing, this paper proposes to apply the following formula:

$$Cost= (ECode*CostCode)+(ETest*CostTest) \quad (2)$$

Where:

- *Cost is a total estimated cost.*
- *ECode is an estimated effort of coding for each requirement. The unit is man-hours.*
- *CostCode is a cost of coding that is charged to customers. This paper applies the cost-value approach to identify the cost of coding for each requirement group (e.g. "Must-Have", "Should-Have", "Could-Have" and "Wish"). The unit is US dollar.*
- *ETest is an estimated effort of testing for each requirement. The unit is man-hours.*
- *CostTest is a cost of testing that is charged to customers. The approach to identify this value is*

similar to CostCode's approach. The unit is US dollar.

In this paper, we assumed the following in order to calculate *CostCode* and *CostTest*. Also, this paper assumes that a standard cost for both activities is $100 per man-hours.

- *A value is 1.5 of ("Must-Have", "Should-Have") – this means that "Must-Have" requirements have one and half times cost value than "Should-Have" requirements.*
- *A value is 3 of ("Must-Have", "Could-Have") – this means that "Must-Have" requirements have three times cost value than "Could-Have" requirements.*
- *A value is 2 of ("Should-Have", "Could-Have") – this means that "Should-Have" requirements have two times cost value than "Could-Have" requirements.*
- *A value is approximately 3 of ("Could-Have", "Wish") – this means that "Could-Have" requirements have three times cost value than "Wish" requirements.*

Therefore, the procedure of requirement prioritization process can be shortly described below:

1. Provide estimated efforts of coding and testing for each requirement.
2. Assign cost value for each requirement group based on the previous requirement classification (e.g. "Must-Have", "Should-Have", "Could-Have" and "Wish").
3. Calculate a total estimated cost for coding and testing, by using the formula (2).
4. Define a customer priority for each requirement.
5. Compute a priority value for each requirement by using the formula (1).
6. Prioritize requirements based on the higher priority value.

Once the requirements are prioritized, the next proposed step is to generate test scenario and prepare test case.

This section presents an automated test scenario generation derived from UML Use Case diagram. Our approach is built based on Heumann's algorithm [23]. The limitation of our approach is to ensure that all use cases are fully dressed. The fully dressed use case is a use case with the comprehensive of information, as follows: use case name, use case number, purpose, summary, pre-condition, post-condition, actors, stakeholders, basic events, alternative events, business rules, notes, version, author and date.

The proposed method contains four steps, as follows: (a) extract use case diagram (b) generate test scenario (c) prepare test data and prepare other test elements. These steps can be shortly described as follows:

1. The first step is to extract the following information from fully dressed use cases: (a) use case number (b) purpose (c) summary (d) pre-condition (e) post-condition (f) basic event and (g) alternative events. This information is called use case scenario in this paper. The example fully dressed use cases of ATM withdraw functionality can be found as follows:

Table 2 Example Fully Dressed Use Case

Use Case Id	Use Case Name	Summary	Basic Event	Alternative Events	Business Rules
UC-001	Withdraw	To allow bank's customers to withdraw money from ATM machines anywhere in Thailand.	1. Insert Card 2. Input PIN 3. Select Withdraw 4. Select A/C Type 5. Input Balance 6. Get Money 7. Get Card	1. Select Inquiry 2. Select A/C Type 3. Check Balance	(a) Input amount <= Outstanding Balance (b) Fee charge if using different ATM machines
UC-002	Transfer	To allow users to transfer money to other banks in Thailand from all ATM machines	1. Insert Card 2. Input PIN 3. Select Transfer 4. Select bank 5. Select "To" account 6. Select A/C Type 7. Input Amount 8. Get Receipt 9. Get Card	1. Select Inquiry 2. Select A/C Type 3. Check Balance	Amount <= 50,000 baht

The above use cases can be extracted into the following use case scenarios:

Table 3 Extracted Use Case Scenarios

Scenario Id	Summary	Basic Scenario
Scenario-001	To allow bank's customers to withdraw money from ATM machines anywhere in Thailand.	1. Insert Card 2. Input PIN 3. Select Withdraw 4. Select A/C Type 5. Input Balance 6. Get Money 7. Get Card
Scenario-002	To allow bank's customers to withdraw money from ATM machines anywhere in Thailand.	1. Insert Card 2. Input PIN 3. Select Inquiry 4. Select A/C Type 5. Check Balance 6. Select Withdraw 7. Select A/C Type 8. Input Balance 9. Get Money 10. Get Card

Scenario-003	To allow users to transfer money to other banks in Thailand from all ATM machines	1. Insert Card 2. Input PIN 3. Select Transfer 4. Select bank 5. Select "To" account 6. Select A/C Type 7. Input Amount 8. Get Receipt 9. Get Card
Scenario-004	To allow users to transfer money to other banks in Thailand from all ATM machines	1. Insert Card 2. Input PIN 3. Select Inquiry 4. Select A/C Type 5. Check Balance 6. Select Transfer 7. Select bank 8. Select "To" account 9. Select A/C Type 10. Input Amount 11. Get Receipt 12. Get Card

2. The second step is to automatically generate test scenarios from the previous use case scenarios [23]. From the above table, we automatically generate the following test scenarios:

Table 4 Generated Test Scenarios

Test Scenario Id	Summary	Basic Scenario
TS-001	To allow bank's customers to withdraw money from ATM machines anywhere in Thailand.	1. Insert Card 2. Input PIN 3. Select Withdraw 4. Select A/C Type 5. Input Balance 6. Get Money 7. Get Card
TS-002	To allow bank's customers to withdraw money from ATM machines anywhere in Thailand.	1. Insert Card 2. Input PIN 3. Select Inquiry 4. Select A/C Type 5. Check Balance 6. Select Withdraw 7. Select A/C Type 8. Input Balance 9. Get Money 10. Get Card
TS-003	To allow users to transfer money to other banks in Thailand from all ATM machines	1. Insert Card 2. Input PIN 3. Select Transfer 4. Select bank 5. Select "To" account 6. Select A/C Type 7. Input Amount 8. Get Receipt 9. Get Card
TS-004	To allow users to transfer money to other banks in Thailand from all ATM machines	1. Insert Card 2. Input PIN 3. Select Inquiry 4. Select A/C Type 5. Check Balance 6. Select Transfer 7. Select bank 8. Select "To" account 9. Select A/C Type 10. Input Amount 11. Get Receipt 12. Get Card

3. The next step is to prepare test data. This step allows to manually prepare an input data for each scenarios.

The last step is to prepare other test elements, such as expected output, actual output and pass / fail status.

5. EVALUATION

The section describes the experiments design, measurement metrics and results.

5.1. Experiments Design

A comparative evaluation method has proposed in this experiment design. The high-level overview of this experiment design can be found as follows:

1. **Prepare Experiment Data**. Before evaluating the proposed methods and other methods, preparing experiment data is required. In this step, 50 requirements and 50 use case scenarios are randomly generated.
2. **Generate Test Scenario and Test Case**. A comparative evaluation method has been made among the proposed test generation algorithm, Heumann's technique Jim [23], Ryser's method [24], Nilawar's algorithm [33] and the proposed method presented in the previous section.
3. **Evaluate Results**. In this step, the comparative generation methods are executed by using 50 requirements and 50 use case scenarios. These methods are also executed for 10 times in order to find out the average percentage of critical domain requirement coverage, a size of test cases and total generation time. In total, there are 500 requirements and 500 use case scenarios executed in this experiment.

The following tables present how to randomly generate data for requirements and use case scenarios respectively.

Table 5 Generate Random Requirements

Attribute	Approach
Requirement ID	Randomly generated from the following combination: Req + *Sequence Number*. For example, Req1, Req2, Req3, …, ReqN.
Type of Requirement	Randomly selected from the following values: Functional AND Non-

	Functional.
MoSCoW Criteria	Randomly selected from the following values: Must Have (M), Should Have (S), Could Have (C) and Won't Have (W)
Is it a critical requirement (Y/N)?	Randomly selected from the following values: True (Y) and False (N)

Table 6 Generate Random Use Case Scenario

Attribute	Approach
Use case ID	Randomly generated from the following combination: uCase + *Sequence Number*. For example, $uCase_1, uCase_2, \ldots, uCase_n$.
Purpose	Randomly generated from the following combination: Pur + *Sequence Number same as Use case ID*. For example, $Pur_1, Pur_2, \ldots, Pur_n$.
Basic Scenario	Randomly generated from the following combination: uCase + *Sequence Number*. For example, $basic_1, basic_2, \ldots, basic_n$.

5.2. Measurement Metrics

The section lists the measurement metrics used in the experiment. This paper proposes to use three metrics, which are: (a) size of test cases (b) total time and (c) percentage of critical domain requirement coverage. The following describe the measurement in details.

1. **A Number of Test Cases**: This is the total number of generated test cases, expressed as a percentage, as follows:

$$\%\ Size = (\#\ Size\ /\ \#\ \text{of}\ Total\ Size)*100 \quad (3)$$

Where:

- *% Size is a percentage of the number of test cases.*
- *# of Size is a number of test cases.*
- *# of Total Size is the maximum number of test cases in the experiment, which is assigned 1,000.*

2. **A Domain Specific Requirement Coverage:** This is an indicator to identify the number of requirements covered in the system, particularly critical requirements, and critical domain requirements [5]. Due to the fact that one of the goals of software testing is to verify and validate requirements covered by the system, this metric is a must. Therefore, a high percentage of critical requirement coverage is desirable.

It can be calculated using the following formula:

$$\%\ CRC = (\#\ of\ Critical\ /\ \#\ of\ Total)*100 \quad (4)$$

Where:

- *% CRC is the percentage of critical requirement coverage.*
- *# of Critical is the number of critical requirements covered.*
- *# of Total is the total number of requirements.*

3. **Total Time**: This is the total number of times the generation methods are run in the experiment. This metric is related to the time used during the testing development phase (e.g. design test scenario and produce test case). Therefore, less time is desirable.

It can be calculated using the following formula:

$$Total = PTime + CTime + RTime \quad (5)$$

Where:

- *Total* is the total amount of times consumed by running generation methods.
- *PTime* is the total amount of time consumed by preparation before generating test cases.
- *CTime* is the time to compile source code / binary code in order to execute the program.
- *RTime* is the total time to run the program under this experiment.

5.3. Results and Discussion

This section discusses an evaluation result of the above experiment. This section presents a graph that compares the above proposed method to other three existing test case generation techniques, based on the following measurements: (a) size of test cases (b) critical domain coverage and (c) total time. Those three techniques are: (a) Heumman's method (b) Ryser's work and (c) Nilawar's approach. There are two dimensions in the following graph: (a) horizontal and (b) vertical axis. The horizontal represents three measurements whereas the vertical axis represents the percentage value.

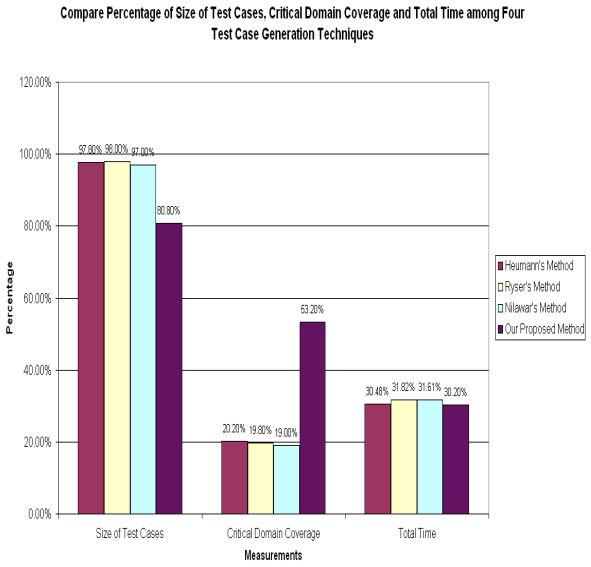


Figure 3 An Evaluation Result

The above graph shows that the above proposed method generates the smallest set of test cases. It is calculated as 80.80% where as the other techniques is computed over 97%. Those techniques generated a bigger set of test cases, than a set generated by the proposed method. The literature review reveals that the smaller set of test cases is desirable. Also, the graph shows that the proposed method consumes the least total time during a generation process, comparing to other techniques. It used only 30.20%, which is slightly less than others. Finally, the graph

presents that the proposed method is the best techniques to coverage critical domains. Its percentage is much greater than other techniques' percentage, over 30%.

6. CONCLUSION

This paper concentrates on resolving the following research problems: (a) an inefficient test case generation method with limited resources (b) a lack of ability to identify and coverage the critical domain requirements and (c) an ignorance of a size of test cases. Furthermore, this paper proposes an effective test case generation process by adding additional prioritization process. The new process aims to improve the ability to: (a) generate test cases with limited resources (b) include more critical domain specific requirements and (c) minimize a number of test cases. Also, this paper introduces an automated test scenario generation technique to address critical domain specific requirements. This paper proposes to compare to other three test case generation techniques, which are: Heummann's work, Ryser's method and Nilawar's technique. As a result, this study found that the proposed method is the most recommended method to generate the smallest size of test cases with the maximum of critical domain specific requirement coverage and the least time consumed in the test case generation process.

7. REFERENCES

[1] Ahl, V., "An Experimental Comparison of Five Prioritization Methods", Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden, 2005.

[2] Alessandra Cavarra, Charles Crichton, Jim Davies, Alan Hartman, Thierry Jeron and Laurent Mounier, "Using UML for Automatic Test Generation", Oxford University Computing Laboratory, Tools and Algorithms for the Construction and Analysis of Systems, TACAS'2000, 2000.

[3] Amaral, "A.S.M.S. Test case generation of systems specified in Statecharts", M.S. thesis – Laboratory of Computing and Applied Mathematics, INPE, Brazil, 2006.

[4] Annelises A. Andrews, Jeff Offutt and Roger T. Alexander, "Testing Web Applications", *Software and Systems Modeling*, 2004.

[5] Avik Sinha, Ph.D and Dr. Carol S. Smidts, "Domain Specific Test Case Generation Using Higher Ordered Typed Languages fro Specification" Ph. D. Dissertation, 2005.

[6] A. Bertolino, "Software Testing Research and Practice", *10th International Workshop on Abstract State Machines (ASM'2003)*, Taormina, Italy, 2003.

[7] A.Z. Javed, P.A. Strooper and G.N. Watson. "Automated Generation of Test Cases Using Model-Driven Architecture", Second International Workshop on Automation of Software Test (AST'07), 2007.

[8] Beck, K. & Andres, C., "Extreme Programming Explained: Embrace Change", 2nd ed. Boston, MA: Addison-Wesley, 2004.

[9] Boehm, B. & Ross, R.. "Theory-W Software Project Management: Principles and Examples", IEEE Transactions on Software Engineering 15, 4: 902-916, 1989.

[10] B.M. Subraya, S.V. Subrahmanya, "Object driven performance testing in Web applications", in: *Proceedings of the First Asia-Pacific Conference on Quality Software (APAQS'00)*, pp. 17-26, Hong Kong, China, 2000.

[11] Chien-Hung Liu, David C. Kung, Pei Hsia and Chih-Tung Hsu, "Object-Based Data Flow Testing of Web Applications", *Proceedings of the First Asia-Pacific Conference on Quality Software* (APAQS'00), pp. 7-16, Hong Kong, China, 2000.

[12] C.H. Liu, D.C. Kung, P. Hsia, C.T. Hsu, "Structural testing of Web applications", in: *Proceedings of 11th International Symposium on Software Reliability Engineering* (ISSRE 2000), pp. 84-96, 2000.

[13] Davis, A., "The Art of Requirements Triage", IEEE Computer 36, 3 p: 42-49, 2003.

[14] Davis, A., "Just Enough Requirements Management: Where Software Development Meets Marketing", New York: Dorset House (ISBN 0-932633-64-1), 2005.

[15] David C. Kung, Chien-Hung Liu and Pei Hsia, "An Object-Oriented Web Test Model for Testing Web Applications", *In Proceedings of the First Asia-Pacific Conference on Quality Software (APAQS'00)*, page 111, Los Alamitos, CA, 2000.

[16] Donald Firesmith, "Prioritizing Requirements", *Journal of Object Technology*, Vol.3, No8, 2004.

[17] D. Harel, "On visual formalisms", Communications of the ACM, vol. 31, no. 5, pp. 514-530, 1988.

[18] D. Harel, "Statecharts: A Visual Formulation for Complex System", Sci.Comput. Program. 8(3):232-274, 1987.

[19] Flippo Ricca and Paolo Tonella, "Analysis and Testing of Web Applications", *Proc. of the 23rd International Conference on Software Engineering*, Toronto, Ontario, Canada. pp.25-34, 2001.

[20] Harel, D., "Statecharts: a visual formalism for complex system", *Science of Computer Programming*, v. 8, p. 231-274, 1987.

[21] Hassan Reza, Kirk Ogaard and Amarnath Malge, "A Model Based Testing Technique to Test Web Applications Using Statecharts", *Fifth International Conference on Information Technology*, 2008.

[22] Ibrahim K. El-Far and James A. Whittaker, "Model-based Software Testing", 2001.

[23] Jim Heumann., "Generating Test Cases From Use Cases", Rational Software, 2001.

[24] Johannes Ryser and Martin Glinz, "SCENT: A Method Employing Scenarios to Systematically Derive Test Cases for System Test", 2000.

[25] Karl E. Wiegers, "First Things First: Prioritizing Requirements", Published in Software Development, 1999.

[26] Karlsson, J., "Software Requirements Prioritizing", *Proceedings of the Second International Conference on Requirements Engineering (ICRE'96)*. Colorado Springs, CO, April 15-18, 1996. Los Alamitos, CA: IEEE Computer Society, p 110-116, 1996.

[27] Karlsson, J., "Towards a Strategy for Software Requirements Selection. Licentiate", Thesis 513, Linköping University, 1995.

[28] Karlsson, J. & Ryan, K., "A Cost-Value Approach for Prioritizing Requirements", *IEEE Software* September/October, p67-75, 1997.

[29] Leffingwell, D. & Widrig, D., "Managing Software Requirements: A Use Case Approach", 2nd ed. Boston, MA: Addison-Wesley, 2003.

[30] Leslie M. Tierstein, "Managing a Designer / 2000 Project", *NYOUG Fall'97 Conference*, 1997.

[31] L. Brim, I. Cerna, P. Varekova, and B. Zimmerova, "Component-interaction automata as a verification oriented component-based system specification", *In: Proceedings (SAVCBS'05)*, pp. 31-38, Lisbon, Portugal, 2005.
[32] Mahnaz Shams, Diwakar Krishnamurthy and Behrouz Far, "A Model-Based Approach for Testing the Performance of Web Applications", *Proceedings of the Third International Workshop on Software Quality Assurance (SOQUA'06)*, 2006.
[33] Manish Nilawar and Dr. Sergiu Dascalu, "A UML-Based Approach for Testing Web Applications", Master of Science with major in Computer Science, University of Nevada, Reno, 2003.
[34] Moisiadis, F., "Prioritising Scenario Evolution", *International Conference on Requirements Engineering (ICRE 2000)*, 2000.
[35] Moisiadis, F., "A Requirements Prioritisation Tool", *6th Australian Workshop on Requirements Engineering (AWRE 2001)*. Sydney, Australia, 2001.
[36] M. Prasanna S.N. Sivanandam R.Venkatesan R.Sundarrajan, "A Survey on Automatic Test Case Generation", Academic Open Internet Journal, 2005.
[37] Nancy R. Mead, "Requirements Prioritization Introduction", Software Engineering Institute, Carnegie Mellon University, 2008.
[38] Park, J.; Port, D.; & Boehm B., "Supporting Distributed Collaborative Prioritization for Win-Win Requirements Capture and Negotiation 578-584", *Proceedings of the International Third World Multi-conference on Systemics, Cybernetics and Informatics (SCI'99)* Vol. 2. Orlando, FL, July 31-August 4, 1999. Orlando, FL: International Institute of Informatics and Systemic (IIIS), 1999.
[39] Rajib, "Software Test Metric", *QCON*, 2006.
[40] Robert Nilsson, Jeff Offutt and Jonas Mellin, "Test Case Generation for Mutation-based Testing of Timeliness", 2006.
[41] Saaty, T. L., "The Analytic Hierarchy Process", New York, NY: McGraw-Hill, 1980.
[42] Shengbo Chen, Huaikou Miao, Zhongsheng Qian, "Automatic Generating Test Cases for Testing Web Applications", International Conference on Computational Intelligence and Security Workshops, 2007.
[43] Valdivino Santiago, Ana Silvia Martins do Amaral, N.L. Vijaykumar, Maria de Fatima, Mattiello-Francisco, Eliane Martins and Odnei Cuesta Lopes, "A Practical Approach for Automated Test Case Generation using Statecharts", 2006.
[44] Vijaykumar, N. L.; Carvalho, S. V.; Abdurahiman, V., "On proposing Statecharts to specify performance models", *International Transactions in Operational Research*, 9, 321-336, 2002.
[45] Wiegers, K., "E. Software Requirements", 2nd ed. Redmond, WA: Microsoft Press, 2003.
[46] Xiaoping Jia, Hongming Liu and Lizhang Qin, "Formal Structured Specification for Web Application Testing". *Proc. of the 2003 Midwest Software Engineering Conference (MSEC'03)*. Chicago, IL, USA. pp.88-97, 2003.
[47] Yang, J.T., Huang, J.L., Wang, F.J. and Chu, W.C., "Constructing an object-oriented architecture for Web application testing", *Journal of Information Science and Engineering* 18, 59-84, 2002.
[48] Ye Wu and Jeff Offutt, "Modeling and Testing Web-based Applications", 2002.
[49] Ye Wu, Jeff Offutt and Xiaochen, "Modeling and Testing of Dynamic Aspects of Web Applications, *Submitted for publication*. Technical Report ISE-TR-04-01, www.ise.gmu.edu/techreps/, 2004.
[50] Zhu, H., Hall, P., May, J., "Software Unit Test Coverage and Adequacy", ACM Comp. Survey 29(4), pp 366~427, 1997.

FROM NATURAL LANGUAGE REQUIREMENTS TO A CONCEPTUAL MODEL

Christian Kop, Günther Fliedl, Heinrich C. Mayr

Alpen-Adria Universität Klagenfurt
Applied Informatics/Application Engineering
Universitätsstasse 65 – 67, 9020 Klagenfurt
(chris | guenther | heinrich)@ifit.uni-klu.ac.at

ABSTRACT

In literature it is described in great detail how class diagrams and ER diagrams or UML class diagrams are derived from natural language sentences. It is normally assumed, that there is a direct correspondence between natural language elements (e.g., words) and conceptual model elements. We do not strictly follow this assumption because of the complexity of natural language with its ambiguities and ellipsis. Hence in this paper a stepwise generation of a conceptual model out of natural language requirements sentences is proposed. According to the ideas of MDA we assume that automatic transformation steps from the source model (in our case natural language) to the target conceptual model (e.g., UML class diagram) make sense. In addition to that we suggest that the designer should play an important part during transformation. It is furthermore proposed to introduce an interlingua which helps to detect defects and provides traceability between sentences and the model elements.

Index Terms – natural language processing, interlingua, conceptual modeling, defect detection

1. INTRODUCTION

In most cases the requirements are presented on two levels: the level of end user needs and the level of developers or requirements engineers models. End user requirements usually are expressed via natural language; requirements handled by engineers are usually expressed through formal, conceptual models. In many cases this diverging way of representing knowledge is the main reason for misunderstandings between users and engineers concerning initial requirements. The discrepancy disables the possibility of validating requirements, which is an important step in the process of requirements engineering.

To handle such problems we proposed an intermediate level for requirements representation, an interlingua connecting the natural language level of the end user and conceptual model level produced by engineers. The approach provides instruments for the representation of intermediate results and the traceability between intermediate results and the original sentences. It supports automated mapping from natural language requirements to interlingua specifications and automated mapping from the interlingua representation to the conceptual models.

The linguistic processing step focuses on the transfer of written textual requirements to an interlingua, the so called Pre-design Model. The "Klagenfurt Conceptual Pre-design Model (KCPM)" [6] provides a glossary and a graphical representation and it is used as a basis for the mapping to the conceptual model (e.g., UML). We propose that the basic notions introduced in this interlingua should correspond to hypothetical basic linguistic categories like nouns, verbs, etc. Thus, the goal of the whole process which is called NIBA ("Natürlichsprachliche Informationsbedarfsanalyse") is to automate the process of producing pre-design models by extracting their entries from the end-user's natural language requirements statements.

To enhance the mapping process a specific framework for annotating natural language descriptions on different layers was developed.

The paper is structured as follows. In the next section the related work is described. The linguistic processing step is introduced in Section 3. Section 4 explains the interpretation step. Section 5 focuses on the interlingua and their possibilities. Section 6 gives an overview of the mapping to the conceptual model. The paper is summarized in Section 7.

2. RELATED WORK

The interpretation of natural language has a long tradition. In earlier approaches heuristics were proposed. Some of these approaches were described in [3] [1] [8] [7]. Chen presented 11 rules to generate conceptual model elements (entity types and relationship types) from structured sentence. Excerpts of these rules can be found in the next listing [3].

- (Rule 1) A common noun in English corresponds to an entity type.
- (Rule 2) A transitive verb in English corresponds to a relationship type in an ER diagram.
- (Rule 3) An adjective in English corresponds to an attribute of an entity in an ER diagram.
- (Rule 4) An adverb in English corresponds to an attribute of a relationship in an ER diagram.
- (Rule 5) If the sentence has the form: „There are … X in Y“ then we can convert it into the equivalent form „Y has … X “.
- (Rule 7) If the sentence has the form „The X of Y is Z“ and if Z is not a proper noun, we may treat X as an attribute of Y.

Abbot [1] used heuristics for the generation of program specifications. Parsing techniques were introduced in [2] and [11]. NL-OOPS [14] uses the LOLITA [15] natural language processing toolkit with an internal knowledge base to generate first cut conceptual models. Meanwhile tagging and chunking is the state of the art for the linguistic step. In [13] an approach is described which uses part of speech tagging and morphological analysis for the generation of conceptual model element candidates. Additionally an ontology (world model) was used to refine the candidates for the project specific conceptual model (discourse model).

3. LINGUISTIC PROCESSING

The system solves the task of Natural Language Processing of English requirements texts by producing chunked and semantically annotated text, which is made ready for the KCPM modeling notions extraction in the interpretation stage of the project. In a first stage it accepts the tagged sentences which are produced by QTag [16]. This output is refined and certain structures are chunked together. Figure 1 in the appendix shows such a chunk tree representing the syntactic structure including phrasal, feature inheriting nodes.

This chunking output was processed by a modular system of linguistic subsystems including the following functions:

- The identification of compound nouns. We suppose that unclear compound boundaries are very often motivated through ambiguity of complex terms, e.g., the implicit structure of compounds or other groups of words.
- The extraction and generation of inflectional word forms.
- Extraction of derivational morphological information.
- The identification of multi-words units and idiomatic expression identification. This is made possible by dynamically extending linguistic knowledge inside the lexicon component.
- Verb subclass identification. The filtered verb classes are based on the NTMS-system (“Natürlichkeitstheoretische Morphosyntax”) [4] included in the NIBA framework.

4. INTERPRETATION

4.1 General guidelines for interpretation

Following the different approaches mentioned in the related work section, the following can be learned for the interpretation of natural language sentences:

- Common (individual) nouns are candidates for classes and attributes.
- An adjective and a noun together are candidates for specialized classes.
- Proper nouns are candidates for instance labels.
- A transitive verb is a candidate for a relationship type.
- The nouns related to the verbs are the involved classes of the relationship type.
- Also prepositions can be candidates for relationship types.

In other words, given a source language (e.g., natural language) and a “meta model” (i.e., the grammar description of the sentence) as well as a target language (e.g., a conceptual model and its meta model), certain instances of the source language can be mapped to instances of the target language. This is achieved by defining equivalences between syntactic structures of the source model and syntactic structures of the target model.

These general rules must be adopted for the certain situation (i.e., the annotated natural language). In our case the NTMS was used for annotating the natural language sentences with syntactic grammar information. Since the NTMS defines N0 as a noun and N3 as a noun phrase, a class can be derived from a noun (N0) or noun phrase (N3) respectively. If we find a verb (V0) together with two noun phrases then a relationship can be derived from such a pattern. Figure 1 in the appendix shows such an example.

Although these and other heuristics are commonly used they cannot really support the interpretation. The next section will explain some difficulties of interpretation.

4.2 Problems of Interpretation

The problems of interpretation arise since the same syntactic structure of a phrase can be interpreted differently. A typical example of this problem is that the combination of an adjective and a noun can be seen as a specialization of that noun. It is also possible that the adjective together with the noun is the needed concept. Another problem: It is not always possible to distinguish between a class and an attribute just by analyzing one single sentence. In literature [11] the subject-predicate-object structure with the predicate "has" (e.g., X has Y) is interpreted as follows. The subject X is a class and the object Y is an attribute. However in [9] it was shown that the verb "has" is very ambiguous.

Since mainly syntactic structures are analyzed and mapped to elements of the conceptual model there is no guarantee that all the extracted elements are relevant for the target model. There is no guarantee that the model assembled only with the extracted elements will be complete or consistent. Even worse if an arbitrary text is taken for analyzing and interpretation there is no guarantee that the intention of the customer fits with the intentions of the designer.

3.4 Solution

As one possible solution it is necessary to give the designer the freedom to select those extracted model elements which seem to be necessary for the target model. Furthermore it is necessary to introduce an interlingua. This interlingua presents the designer the result of the extraction process and the designer can maintain and refine the results. Hence the model presented in the interlingua does not represent the final result or final conceptual model. It represents a intermediate result that must be discussed, refined and improved. A tool was implemented with which the designer can select necessary model elements and manage the elements in the model of the interlingua. This also includes a tool feature for the mapping from the interlingua to the conceptual model.

5. INTERLINGUA

5.1 Overview

According to the underlying paradigm of how a stakeholder perceives the "world", two types of conceptual modeling approaches can be distinguished:

- Entity type and object oriented approaches.
- Fact oriented approaches.

In the first paradigm the "world" is seen as a world of objects which have properties. Therefore a clear distinction is made between object and object types respectively and their properties. Representatives of this paradigm are the classical ER approach and UML. Fact oriented approaches on the other hand see the "world" as a world of facts. Facts describe objects and their roles within a relationship. No distinction is made between objects and their properties. Every concept is treated equally in a first step. Representatives of this kind of paradigm are NIAM [7] and its successor ORM [5]. Both approaches have pros and cons. Object oriented approaches look very compact. In a typical object oriented class diagram attributes are embedded in the class representation. No additional connections between classes and attributes are necessary which would expand the diagram. On the other hand, many revisions must be made if such a diagram is used too early in the design phase. Due to information that is collected, classes might become attributes and attributes might become classes. According to [5] this is a reason why fact oriented approaches are better suited to be used as an interlingua.

Since the interlingua is placed before the conceptual model during an early phase of design the fact oriented paradigm was preferred. Nevertheless there must also be the necessity to provide an easy transformation from the interlingua to a conceptual model like UML since it is actually the standard for conceptual modeling. Hence the interlingua for conceptual modeling of structural aspects of an information system consists of the following basic notions:

- Thing type: Any notion which is important in a certain universe of discourse is treated as a thing type. Since attributes are not defined also notions like person name, course id etc. are seen as thing types.
- Connection type: Connection types relate thing types to each other. Special connection types like generalization or aggregation can be defined.

The aim of the interlingua is also to be a support for all kinds of stakeholders (designers and end users). Therefore a graphical and glossary based representation was used for the collection of requirements (see Figure 3 in the appendix for the graphical representation – the glossary representation is hidden).

5.2 Defect detection support

Beside the purpose to provide a communication platform between stakeholders, the interlingua can also support the detection of structural inconsistencies and incompleteness. The simplest one can be detected if the designer takes a look at the cardinality definitions of the connection types. As it can be easily seen, all of these cardinality descriptions have a "?..?". This means that cardinalities could not be extracted from the textual description.
Another possibility is to count the number of connection types of a thing type. This is described in detail in [12]. With this strategy, centered thing types

can be detected (see Figure 4 in the appendix). The more connection types a thing type has, the more centered or important it is. Such centered thing types appear with a bigger rectangular and in another color (e.g., green) than other thing types which seem to be less important. However, this must not necessarily reflect the end users intention. Therefore this strategy is used to confront the end user with the result and to discuss the result with him. For instance if the end user wonders why certain thing types like course and professor are not so important (they appear in white color and the rectangular is not so big as the rectangular for assistant or employee) then this can be the hint for a defect in the original specification.

If a mapping preview is made, then orphan classes [10] can be detected. The Figure 5 shows such a case for the university example. In this case thing types like university, faculty, department, assistant, employee, professor, budget, ut8 and ut3 were detected to be class candidates. All the thing types which appear in white color are currently candidates for attributes. Once again this is not the final result but a starting point for communication, discussion and refinement. As can be seen in Figure 5, professor, budget, faculty and university do not have any related attributes. Hence the mapping preview gives also hints for defects.

5.3 Traceability

Sentences from which thing types and connection types can be extracted are also stored as "Sources" in the interlingua model. If a thing type was extracted from the sentence, then a relation between the thing type and the sentence exists. The same holds for connection types.

6. MAPPING TO THE CONCEPTUAL MODEL

In order to guarantee the mapping to a conceptual model rules are applied. These rules can be classified into

- Laws vs. proposals.
- Direct vs. indirect rules.

Laws are much stricter than proposals. If a mapping rule is a law than a mapping to a certain target concept (e.g., class) cannot be ignored otherwise the syntax of the conceptual target model will be incorrect. Proposals on the other hand only give hints. The syntax of the target model will not be wrong if these hints are ignored.

An indirect rule not only uses the semantic relationship to decide about the mapping but also information about previous mappings. For example, if a concept X is already mapped to an attribute and a concept Y is related to that attribute X then an indirect rule for Y detects a mapping possibility (Y will become a class).

This mapping approach also applies meta-rules to resolve conflicting situations between the rules. An example of a meta rule is: "Laws overrule proposals".

7. CONCLUSION AND FUTURE WORK

In this paper an overview of a mapping process from natural language descriptions to a conceptual model was given. It was also described that such a process is not straight forward. Instead the designer must handle problems. As one possible solution the interlingua (KCPM) was introduced. This model gives the designer an overview of the output of natural language processing and provides him with some help to improve it. Without generating the UML target model, he is able to revise it. Different presentation techniques (e.g., graphical view and glossary view) make it possible to communicate with the end user.

In future, it is planned to find more possibilities to detect defects. These defect detection strategies should then be applied on the notions which were extracted from English or from German requirements sentences.

8. REFERENCES

[1] R.J. Abbot, "Program Design by Informal English Descriptions," Communication of the ACM, Vol. 26 No. 11, pp. 882 – 894, 1983.

[2] E. Buchholz, H. Cyriaks, A. Düsterhöft, H. Mehlan, B. Thalheim, B.. "Applying a Natural Language Dialogue Tool for Designing Databases,". International Workshop on Applications of Natural Language to Databases (NLDB'95), pp. 119 – 133, 1995.

[3] P. Chen "English Sentence Structure and Entity Relationship Diagrams," International Journal of Information Siences, Vol. 29., pp. 127-149, 1983

[4] G. Fliedl, Natürlichkeitstheoretische Morpho-syntax – Aspekte der Theorie und Implementierung, Gunter Narr Verlag Tübingen, 1999.

[5] T. Halpin, "UML Data Models from an ORM Perspective Part 1," Journal of Conceptual Modeling 1998.

[6] H.C. Mayr, Ch. Kop, "A User Centered Approach to Requirements Modeling, Proceedings Modelierung," Lecture Notes in Informatics LNI, p-12, GI-Edition, pp. 75-86, 2002.

[7] G.M. Nijssen, T.A Halpin, Conceptual Schema and Relational Database Design – A fact oriented approach. Prentice Hall Publishing Company, 1989.

[8] M. Saeki, H. Horai, H. Enomoto, "Software Development from Natural Language Specification," Proceedings of the 11th International Conference on Software Engineering, pp. 64 – 73, 1989.

[9] V.C. Storey, "Understanding Semantic Relationships," VLDB Journal, Vol. 2, pp. 455 – 488., 1993.

[10] B. Tauzovich, "An Expert System for Conceptual Data Modeling," Proceedings of the 8th International Conference on Entity Relationship Approach, North Holland Publ. Company, pp. 205 – 220, 1989.

[11] A.M. Tjoa, A.M.; L. Berger, "Transformation of Requirement Specification Expressed in Natural Language into an EER Model, " Proceedings of the 12th International Conference on Entity Realtionship Approach, Springer Verlag, New York, pp. 127-149, 1993.

[12] Ch. Kop, "Visualizing Concetual Schemas with their Sources and Progress," International Journal on Advances in Software, Vol. 2. u. 3., pp. 245 – 258, 2009.

[13] H. M. Harmain, R. Gaizauskas, "CM-Builder: An Automated NL-based Case Tool," 15th IEEE International Confernce on Automated Software Engineering (ASE'00), pp. 45 – 54, 2000.

[14] L. Mich, J. Mylopoulos, N. Zeni, "Improving the Quality of Conceptual Models with NLP Tools: An Experiment," Technical Report DIT-02-0047, Dept. of Information and Communication Technology, Univ. of Trento, 2002.

[15] R. Garigliano, R. Morgan, M. Smith, "The LOLITA System as a Contents Scanning Tool," Proceedings of the 13th International Conference Artificial Intelligence, Expert Systems, and Natural Language Processing, 1993.

[16] D. Tufis, O. Mason, "Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger," Proceedings of the First International Conference on Language Resources & Evaluation (LREC), Granada (Spain), p.589-596, 1998.

APPENDIX

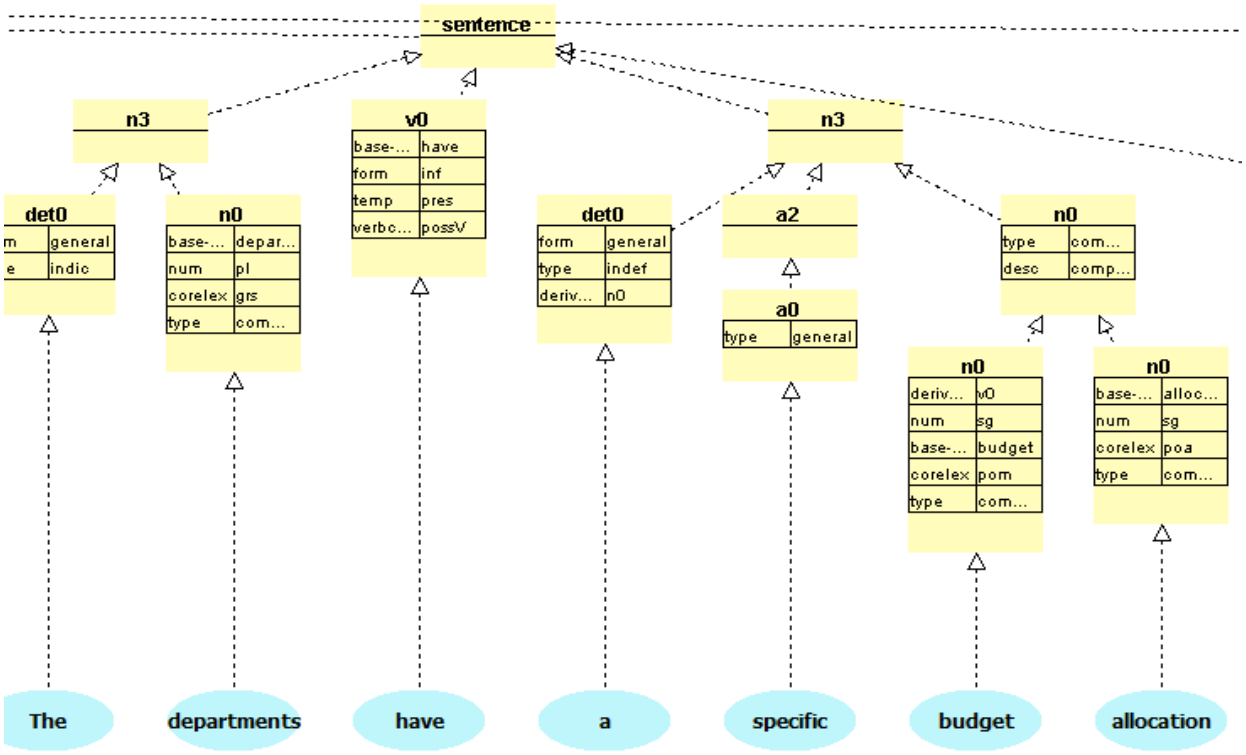


Fig. 1. Tagged sentence with chunk tree

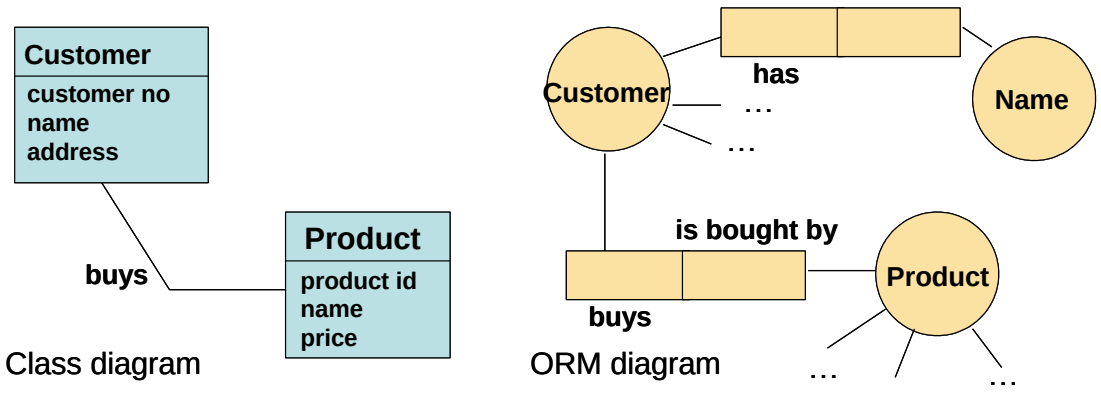


Fig. 2. Class diagram versus ORM diagram

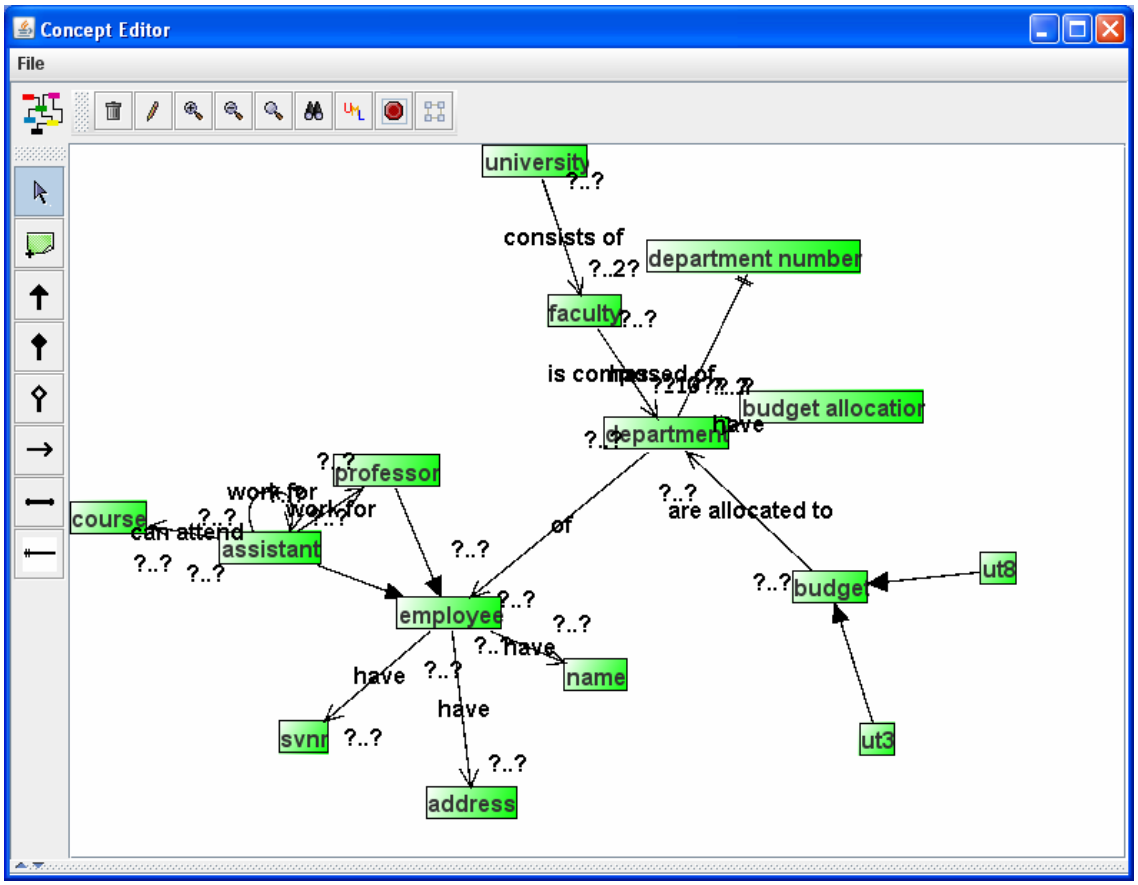


Fig. 3. Graphical representation of the interlingua (university example)

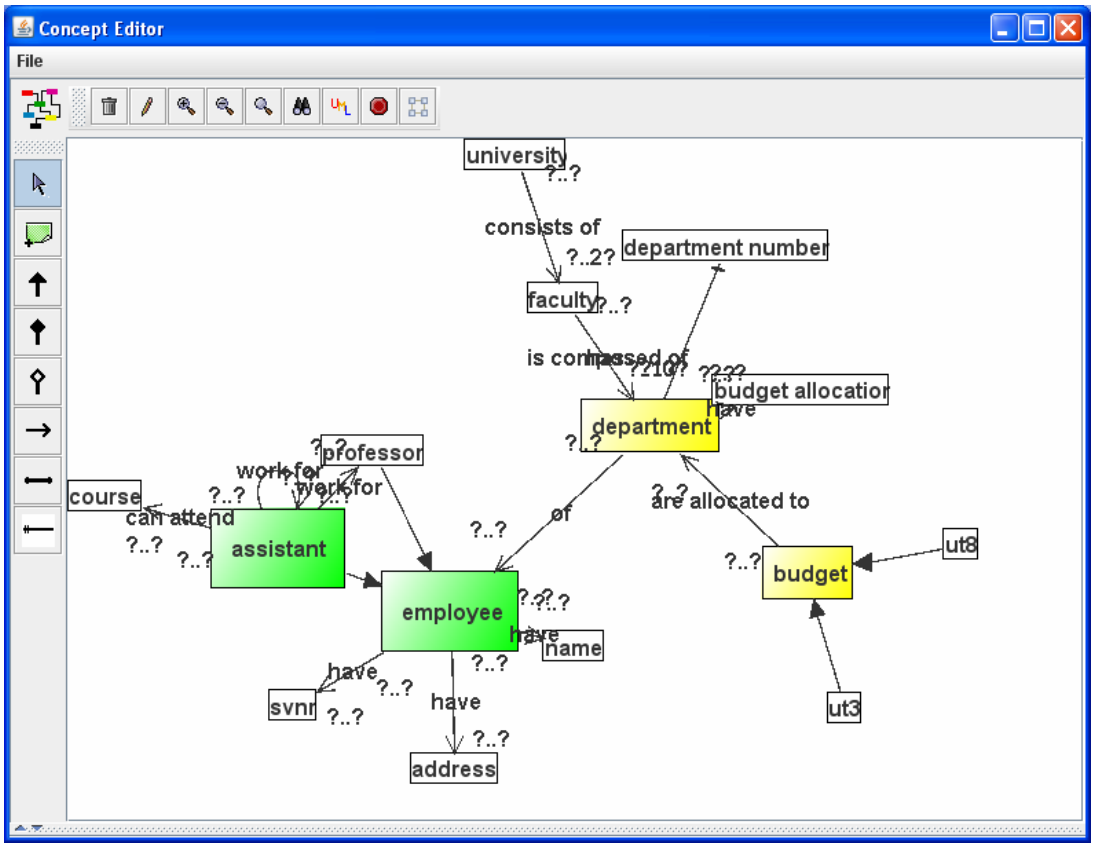


Fig. 4. Visualization of centered thing types

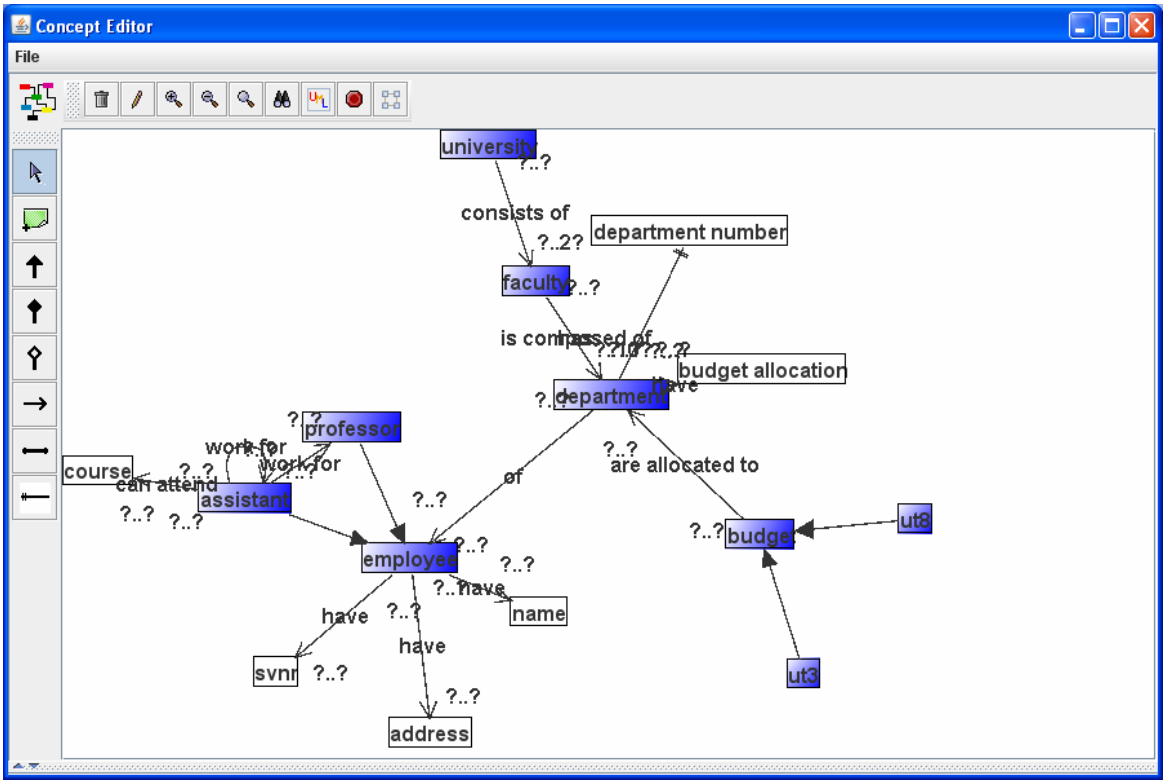


Fig. 5. Mapping preview

TEST CASE REDUCTION METHODS BY USING CBR

Siripong Roongruangsuwan and Jirapun Daengdej

Autonomous System Research Laboratory
Faculty of Science and Technology
Assumption University, Thailand
p4919742@au.edu, jirapun@scitech.au.edu

ABSTRACT

It has been proven that software testing usually consumes over 50% of the costs associated with the development of commercial software systems. Particularly, regression testing activities has been shown to be a critically important phase of software testing. Many reduction techniques have been proposed to reduce costs. Unfortunately, the cost is usually over budget and those methods are failed to reasonably control costs. The primarily outstanding issue is non-effective methods to remove redundancy tests while a bigger size of tests and a significant amount of time are still remaining. To resolve the issue, this paper proposes an artificial intelligent concept of case-based reasoning (CBR). CBR has an uncontrollable costs issue as same as testing. There are many effective algorithms researched over a long period of time. This study introduces three methods combined between CBR's deletion algorithm and testing activities. Those methods aim to minimize size of tests and time, while preserving fault detection.

Index Terms - test case reduction, test reduction, test reduction CBR, CBR for testing and test reduction techniques

1. INTRODUCTION

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test [7], with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of implementation of the software. Test techniques include the process of executing a program or application with the intent of finding software bugs. It can also be stated as the process of validating and verifying that software meets the business and technical requirements that guided its design and development, so that it works as expected. Software Testing can be implemented at any time in the development process; however, the most test effort is employed after the requirements have been defined and coding process has been completed.

Many researchers [10], [11], [12], [13], [14], [15], [23], [24], [25], [26], [29], [36], [37], [39] have proven that these test case reduction methods can reserve the fault detection capability. There are many outstanding research issues in this area. In this paper, the research issues are: redundancy test cases are still remained, an uncontrollable growth of test cases and existing reduction methods consume a great deal of time and cost during a reduction process. The literature review [16] shows that there are many techniques to resolve those three issues. One of effective approaches is to apply the concept of artificial intelligent. There are many artificial intelligent concepts, such as neutral network, fuzzy logic, learning algorithms and case-based reasoning (CBR). CBR is one of the most popular and actively researched areas in the past. The researches [4], [8], [16], [26] show that CBR has identical problems as same as software testing topic. In software testing field, particularly during regression testing activities, the key research issues are: (a) too many redundancy test cases after reduction process (b) a decrease of test cases' ability to reveal faults and (c) uncontrollable grow of test cases. Meanwhile, the key research issues in CBR field are: (a) there are too many redundancy cases in the CBR system (b) a size of CBR system is continuously growing all the time and (c) existing CBR deletion algorithms take longer time to remove all redundancy cases in the CBR system. Those issues in CBR field can be elaborated as follows: Fundamentally, there are four steps in the CBR system, which are: retrieve, reuse, revise and retain. These steps can lead to a serious problem of uncontrollably growing cases in the system. However, the study shows that there are many proposed techniques in order to control a number of cases in the CBR system, such as add algorithms, deletion algorithms and maintenance approaches. CBR have been investigated by CBR researchers in order to ensure that only small amounts of efficient cases are stored in the case base. The previous work [28] shows that deletion algorithms are the most popular and effective approaches to maintain a size of the CBR system. There are many researchers have proposed several deletion algorithms [4], [8], [31], such as random method, utility approach and footprint algorithm. These algorithms aim to: (a) remove all redundancy or unnecessary cases (b) minimize size of system and reduction time and (c) preserve the ability of solving problems. Nevertheless, each technique has strength and weakness. Some methods are suitable for

removing cases. Some methods are perfectly suitable for reducing time. Some may be used for reserving the problem solving capability. Eventually, the previous work [28] discovered several effective methods (e.g. confidential case filtering method, coverage value algorithm and confidential coverage approach) to remove those cases, minimize size of CBR and reduce amount of time, while preserving the ability of CBR system's problem solving skill. Therefore, this paper applies those effective deletion techniques to resolve the problems of software testing. In the light of software testing, the proposed techniques focus on how to maintain the test case or test data while the ability to reveal faults is still preserved. It is assumed that *test cases* or *test data* in this paper are treated as *cases* in the CBR system. Also, there is an assumption that a given set of test cases are generated by a path-oriented test case generation technique. The path-oriented technique is widely used for a white-box testing, which this paper does not address how to generate test cases with path-oriented methods.

Section 2 discusses an overview of test case reduction techniques and processes. Also, section 2 discusses a concept of CBR. Section 3 provides a definition of terminologies used in this paper. Section 4 lists the outstanding research issues motivated this study. Section 5 proposes three new test case reduction methods. Section 6 describes an evaluation method and discusses a result. The last section represents all source references used in this paper.

2. LITERATURE REVIEW

This section describes an overview of test case reduction techniques and the concept of CBR. The following describes those two areas in details.

2.1. Test Case Reduction Techniques

This section discusses and organizes test case reduction (or TCR) techniques researched in 1995-2006. This study shows that there are many researchers who proposed a method to reduce unnecessary test cases (also known as redundancy test cases), like Offutt [5], Rothermel [12], McMaster [24] and Sampth [27]. These techniques aim to remove and minimize a size of test cases while maintaining the ability to detect faults. The literature review [1], [10], [11], [12], [13], [14], [15], [24], [25], [36], [37], [39] shows that there are two types of reduction techniques, which are: (a) pre-process and (b) post-process. First, the pre-process is a process that immediately reduces a size of test cases after generating. Typically, it is occurred before regression testing phase. Second, the post-process is a process that maintains and removes unnecessary test cases, after running the first regression testing activities. Although these techniques can reduce the size of test cases, but the ability to reveal faults seems slightly to be dropped. However, Jefferson Offutt [5] and Rothermel [10], [11], [12], [13], [14], [15], [30], [31], [33] has proven that these test case reduction techniques have many benefits, particularly during the regression testing phase, and most of reduction techniques can maintain an acceptable rate of fault detection. The advantages of these techniques are: (a) to spend less time in executing test cases, particularly during the regression testing phase (b) to significantly reduce time and cost of manually comparing test results and (c) to effectively manage the test data associated with test cases. This study proposes a new "2C" classification of test case reduction techniques, classified based on their characteristics, as follows: (a) coverage-based techniques and (b) concept analysis-based techniques.

2.2. Case-Based Reasoning (CBR)

Over the time, CBR is growing. When the uncontrollable case-based growth is occurred, the performance of CBR is decreasing. Therefore, the maintenance process is required in order to preserve or improve the performance of the system. The process of maintaining CBR is called CBM. David C. Wilson [8] presented the overall concepts of CBR and case based maintenance. This paper focused on the case based maintenance (CBM) approach in term of the framework. In other words, this paper described the type of data collection and how the case based maintenance works. There were so many policies for CBM, for example, addition, deletion, and retain.

"CBM was defined as the process of refining a CBR system's case-base to improve the system's performance. It implements policies for revising the organization or contents (representation, domain content, accounting information, or implementation) of the case-base in order to facilitate future reasoning for a particular set of performance objectives."

These studies [4], [5], [6], [8], [19], [20], [28] reveal that several deletion algorithms have been proposed. For example, a random approach (RD), utility deletion algorithm (UD), footprint deletion algorithm (FD), footprint utility deletion algorithm (FUD) and iterative case filtering algorithm (ICF).

RD is the simplest approach, which removes the case randomly. UD deletes the case that has minimum utility value. Footprint algorithm uses the competence model and removes the auxiliary case from the system. FUD is a hybrid approach between Utility algorithm and Footprint algorithm, and is concerned with the competence model and the utility value. Finally, ICF focuses on the case, which the reachability set is greater than the coverage set [19], [28].

3. DEFINITION

This section describes a definition of terminologies.

Definition 1: Barry [4] defined the *CBR, case base, auxiliary case and pivotal case* as follows:

"Case-Based Reasoning is one of the Artificial Intelligence-based algorithms, which solve the problems by searching through the case storage for

the most similar cases. CBR has to store their solved cases back to their memory or storage in order to learn from their experience."

"Case Base is a collection of cases in CBR, which can be defined as the following: Given a case - base C = {c1… cn}, for c ε C whereas C = CBR, c = case"

Definition 2: *"Auxiliary Case is a case that does not have a direct effect on the competence of a system when it is deleted. The definition of auxiliary case can be described as follows:*

Auxiliary cases do not affect competence at all. Their deletion only reduces the efficiency of the system. A case is an auxiliary case if the coverage it provides is subsumed by the coverage of one of its reachable cases."

Definition 3: *"Pivotal Case is the case that does have a direct effect on the competence of a system if it is deleted.*

A case is a pivotal case if its deletion directly reduces the competence of a system (irrespective of the other cases in the case-base) [2], [3]. Using the above estimates of coverage and reachability a case is pivotal if it is reachable by no other case but itself."

4. RESEARCH CHALLENGES

This section discusses the details of research issues motivated this study. The literature review reveals that [7], [22], [24], [25], [27], [38] those research issues are: (a) too many redundancy test cases after reduction process (b) a decrease of test cases' ability to reveal faults and (c) uncontrollable grow of test cases. These research issues can be elaborated in details as follows: First, the literature review shows that redundancy test cases are test cases tested by multiple test cases. Many test cases that are designed to test the same things (e.g. same functions, same line of code or same requirements) are duplicated. Those duplicated tests are typically occurred during testing activities, particularly during regression testing activities [7], [22], [24], [25], [27], [38]. Those duplicated tests can be eventually removed in order to minimize time and cost to execute tests. The study shows that there are many proposed methods to delete those duplicated test cases such as McMaster's work [24] [25], Jeff's method [7] and Khan's approach [22]. Also, the study shows that one of the most interesting research issues is to minimize those duplicated tests and reduce cost of executing tests. Although there are many proposed methods to resolve that issue, that issue is still remaining. Thus, it is a challenge for researchers to continuously improve the ability to remove duplicated tests. Second, test cases are designed to reveal faults during software testing phase. The empirical studies [10], [11], [12], [23], [30], [31], [33], [39] describe that reducing test cases may impact to the ability of detect faults. Many reduction methods decrease a capability of testing and reveal those faults. Therefore, one of outstanding research challenges for researchers is to remove tests while preserving the ability to defect faults. Last, this paper shows that uncontrollable grow of test cases can be typically occurred during software testing process and evolution. Even if there are many reduction methods proposed to control and limit growth of tests, unfortunately it appears that a number of test cases is still large. Obviously, the greater size of test cases takes longer time and cost to execute.

5. PROPOSED METHODS

For evolving software, test cases are growing dramatically. The more test cases software test engineers have, the more time and cost software test engineers consume. The literature review shows that regression testing activities consume a significant amount of time and cost. Although, a comprehensive set of regression selection techniques [10], [11], [12], [13] has been proposed to minimize time and cost, there is an available room to minimize size of tests and clean up all unnecessary test cases. Thus, removing all redundancy test cases is desirable. There are many approaches to reduce redundancy test cases and applying an artificial intelligent concept in the test case reduction process is an innovated approach. The literature review [16], [28] shows that there are many areas of artificial intelligent concept, such as artificial neutral network, fuzzy logic, learning algorithms and CBR concept. Also, it reveals that CBR has a same research issue as software testing has. The issue is that cases in the CBR system will be consistency growing bigger and larger all the time. There are four steps in CBR that can uncontrollably grow a size of the system: retrieve, reuse, revise and retain. Therefore, many CBR papers aim to reduce all redundancy cases, known as "deletion algorithms". The smaller size of CBR system is better and desirable. Due to the fact that CBR has the same problem as software testing and this paper focuses on reduction methods, therefore, this paper proposes to apply CBR deletion algorithms to the test case reduction techniques. This paper introduces three reduction methods that apply CBR deletion algorithms: TCCF, TCIF and PCF methods. Those techniques aim to reduce a number of test cases generated by path-oriented test case generation technique. This technique is used for white-box testing only. However, the generation methods are out of the scope of this paper.

5.1. Example of Test Cases

Given a set of test cases generated, this study discusses the use of a number of case maintenance techniques, which have been investigated by CBR researchers in ensuring that only small amount of cases are stored in the case base, thereby reducing number of test cases should be used in software testing. Similar to what happen to software testing, a number of CBR researchers have focused on finding approaches especially for reducing cases in the CBR

systems' storages. This paper proposes to use the path coverage criteria in order to reduce redundancy test cases. This is because path coverage has a huge benefit of required very thorough testing activities. The following describes in details of the above path coverage using in the software testing field. Let S = {s1, s2, s3, s4, s5} to be a set of stage in the control flow graph. The control flow graph can be derived from the source-code or program. It is a white-box testing. Thus, each state represents a block of code. The techniques that aim to generate and derive test cases from the control flow graph are well-known as path-oriented test case generation techniques. These techniques are widely used to generate test cases. There are many research papers on this area. However, the test case generation techniques are out of scope in this paper.

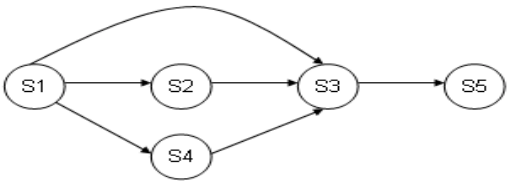


Figure 1 An Example of Control Flow Graph

From the above figure, this paper assumes that each state can reveal a fault. Thus, an ability to reveal faults of five states is equal to 5. Also, it is assumed that every single transaction must be tested. This example is used in the rest of paper.

Let $TC_n = \{s_1, s_2, \ldots, s_n\}$ where TC is a test case and s_n is a stage or node in the path-oriented graph that is used to be tested. From the above figure, a set of test cases can be derived as follows:

$TC_1 = \{s_1, s_2\}$
$TC_2 = \{s_1, s_3\}$
$TC_3 = \{s_1, s_4\}$
$TC_4 = \{s_1, s_2, s_3\}$
$TC_5 = \{s_1, s_3, s_5\}$
$TC_6 = \{s_1, s_4, s_3\}$
$TC_7 = \{s_1, s_2, s_3, s_5\}$
$TC_8 = \{s_1, s_4, s_3, s_5\}$
$TC_9 = \{s_2, s_3\}$
$TC_{10} = \{s_2, s_3, s_5\}$
$TC_{11} = \{s_3, s_5\}$
$TC_{12} = \{s_4, s_3\}$
$TC_{13} = \{s_4, s_3, s_5\}$

The following describes the proposed methods that apply the concept of CBR in details:

5.2. Test Case Complexity for Filtering (TCCF)

A complexity of test case is the significant criteria in this proposed method [2], [19]. In this paper, the complexity of test case measures a number of states included in each test case.

Let Cplx(TC) = {High, Medium, Low} where Cplx is a complexity of test case, TC is a test case and the complexity value can be measured as:

- *High* when a number of states are greater than an average number of states in the test suite.
- *Medium* when a number of states are equal to an average number of states in test suites.
- *Low* when a number of states are less than an average number of states in the test suites.

The procedures of this method can be described briefly in the following steps.

The first step is to determine a coverage set. From figure 1, each coverage set can be identified as follows:

Coverage (1) = $\{TC_1\}$
Coverage (2) = $\{TC_2\}$
Coverage (3) = $\{TC_3\}$
Coverage (4) = $\{TC_1, TC_4, TC_9\}$
Coverage (5) = $\{TC_2, TC_5, TC_{11}\}$
Coverage (6) = $\{TC_3, TC_6, TC_{12}\}$
Coverage (7) = $\{TC_1, TC_4, TC_7, TC_9, TC_{10}, TC_{11}\}$
Coverage (8) = $\{TC_3, TC_6, TC_8, TC_{11}, TC_{12}, TC_{13}\}$
Coverage (9) = $\{TC_9\}$
Coverage (10) = $\{TC_9, TC_{10}, TC_{11}\}$
Coverage (11) = $\{TC_{11}\}$
Coverage (12) = $\{TC_{12}\}$
Coverage (13) = $\{TC_{11}, TC_{12}, TC_{13}\}$

The second step is also to determine a reachability set. The reachability set can be figured out from the above coverage set, based on the given definition in this paper. Therefore, the reachability set can be identified as follows:

Reachability (TC_1) = {1, 4, 7}
Reachability (TC_2) = {2, 5}
Reachability (TC_3) = {3, 6, 8}
Reachability (TC_4) = {4, 7}
Reachability (TC_5) = {5}
Reachability (TC_6) = {6, 8}
Reachability (TC_7) = {7}
Reachability (TC_8) = {8}
Reachability (TC_9) = {4, 7, 9, 10}
Reachability (TC_{10}) = {7, 10}
Reachability (TC_{11}) = {5, 7, 8, 10, 11, 13}
Reachability (TC_{12}) = {6, 8, 12, 13}
Reachability (TC_{13}) = {8, 13}

Next, the step is to define an auxiliary set. The given definition of auxiliary set is to find a test case that does not have a direct effect on the ability to reveal faults when it is removed. From figure 1, therefore, the auxiliary set can be identified as follows:

Auxiliary set = $\{TC_1, TC_2, TC_3, TC_4, TC_5, TC_6, TC_9, TC_{10}, TC_{11}, TC_{12}, TC_{13}\}$

Afterward, the method computes a complexity value for all test cases in the above auxiliary set. From figure 1 and test suites that contain 13 test cases, the average number of states is equal to 3. Therefore, the complexity value for each test case can be computed as follows:

$Cplx(TC_1) = Low$, $Cplx(TC_2) = Low$, $Cplx(TC_3) = Low$, $Cplx(TC_4) = Medium$, $Cplx(TC_5) = Medium$, $Cplx(TC_6) = Medium$, $Cplx(TC_9) = Low$, $Cplx(TC_{10}) = Medium$, $Cplx(TC_{11}) = Low$, $Cplx(TC_{12}) = Low$ and $Cplx(TC_{13}) = Medium$

Finally, the last step removes test cases with minimum of complexity value from the auxiliary set. Thus, TC_1, TC_2, TC_3, TC_9, TC_{11} and TC_12 are removed.

5.3. Test Case Impact for Filtering (TCIF)

The study [21] shows that software is error-ridden in part because of its growing complexity. Software is growing more complex every day. The size of

software products is no longer measured in thousands of lines of code, but it measures in millions. Software developers already spend approximately 80 percent of development costs [21] on identifying and correcting defects, and yet few products of any type other than software are shipped with such high levels of errors. Other factors contributing to quality problems include marketing strategies, limited liability by software vendors, and decreasing returns on testing and debugging, according to the study. At the core of these issues is difficulty in defining and measuring software quality. Due to the fact that defining and measuring a quality of software is important and difficult, the impact of inadequate testing must not be ignorance. The impact of inadequate testing could be lead to the problem of poor quality, expensive costs and huge time-to-market. In conclusion, software testing engineers require identifying the impact of each test case in order to acknowledge and understand clearly the impact of ignoring some test cases. In this paper, an impact value is an impact of test cases in term of the ability to detect faults if those test cases are removed and not be tested.

Let Imp(TC) = {High, Medium, Low} where Imp is an impact if a test case is removed, TC is a test case and the impact value can be measured as:

- *High* when the test case has revealed at least one fault for many times.
- *Medium* when the test case has revealed faults for only one time.
- *Low* when the test case has never revealed faults.

The procedure of this method is similar to the previous method. The only different is that this method aims to use an impact value instead of complexity value. Therefore, the fire three steps are to: identify coverage set, define reachability set and determine an auxiliary set. Afterward, the next step is to compute and assign an impact value. The method computes the impact value for all test cases in the above auxiliary set. From figure 1, the impact value for each test case can be computed as follows:

$$Imp(TC_1) = Low, Imp(TC_2) = High, Imp(TC_3) = Medium, Imp(TC_4) = Low, Imp(TC_5) = High, Imp(TC_6) = Medium, Imp(TC_9) = Low, Imp(TC_{10}) = Low, Imp(TC_{11}) = Low, Imp(TC_{12}) = Low \text{ and } Imp(TC_{13}) = Low$$

Finally, the last step removes test cases with minimum of impact value from the auxiliary set. Thus, TC_1, TC_4, TC_7, TC_9, TC_{10}, TC_{11}, TC_{12} and TC_{13} are removed.

5.4. Path Coverage for Filtering (PCF) Method

Code coverage analysis is a structural testing technique (also known as white box testing). Structural testing compares test program behaviour against the apparent intention of the source code. This contrasts with functional testing (also referred to black-box testing), which compares test program behaviour against a requirements specification. Structural testing examines how the program works, taking into account possible pitfalls in the structure and logic. Functional testing examines what the program accomplishes, without regard to how it works internally. Structural testing is also called path testing since you choose test cases that cause paths to be taken through the structure of the program. The advantage of path cover is that it takes responsible for all statements as well as branches across a method. It requires very thorough testing. This is an effective substitute of other coverage criteria. The path coverage is used as coverage value in this technique. The Coverage value is combined into the addition policy for adding significant case [17]. Within the adding algorithm along with the coverage weight value stated in the review, the concept of deletion algorithm and the coverage have been proposed. The coverage value can specify how many nodes that the test case can cover. In other words, the coverage value is an indicator to measure that each test case covers nodes. It means that the higher coverage value is, the more nodes can be contained and covered in the test case. Let *Cov*(n) = value where *Cov* is a coverage value, value is a number of test cases in each coverage group and n is a coverage relationship.

The procedure of this method can be elaborated briefly as the following steps. From figure 1, the first step is to identify a coverage set, which has been already identified in the previous method. The next step is to calculate a coverage value. This paper proposes to calculate a coverage value based on a number of test cases in each coverage group. Therefore, the coverage value can be computed as follows:

Cov(1) = 1, *Cov*(2) = 1, *Cov*(3) = 3, *Cov*(4) = 3, *Cov*(5) = 3, *Cov*(6) = 4, *Cov*(7) = 6, *Cov*(8) = 6, *Cov*(9) = 1, *Cov*(10) = 3, *Cov*(11) = 1, *Cov*(12) = 1 and *Cov*(13) = 3.

The last step removes all test cases with minimum coverage value, in the potential removal set. Therefore, TC_1, TC_2, TC_9, TC_{11} and TC_{12} are removed.

6. EVALUATION

This section describes an experiments design, measurement metrics and results. This paragraph designs an experiment used to evaluate and determine the best reduction methods. This paper proposes the following three steps. First, the experiment proposes to randomly generate 2,000 test data used in the telecommunication industry. In this experiment, the test data is represented as test case. Second, the experiment executes reduction methods with the generated test cases and compares among the following reduction methods: RD, UD, FD, FUD, ICF and three proposed methods (e.g. TCCF, TCIF and PCF). This step randomly simulates defects for each test case in order to determine an ability to reveal faults. Third, the experiment aims to run the above methods for 10 times in order to calculate the average value for each metric. The metrics used in this

experiment are described in details in next section. Afterward, the experiment compares the values and evaluates a result by generating a comparison graph in order to determine the most recommended reduction approach.

The following table lists the description of each test data that need to be generated randomly.

Table 1 An Example Form of Test Cases

Attribute	**Description**	**Data Type**
Test Id	A unique index to reference test data. The value is a sequence number, starting at 1.	Numeric
Full Name	A first and last name who own the mobile phone.	String
Name	A mobile brand name. The value is a range of iPhone, BlackBerry, Nokia, LG, Sony Ericsson and Samsung.	String
Coverage Value	A value of Coverage set, which is defined by the user.	Numeric
Impact Value	An impact value of each case, in this work. This can be matched to the impact value.	Numeric
wCoverageValue	The weight value for coverage set	Numeric
A set of states	A set of states that required to be tested. State is directly derived from control flow graph. The control flow graph is a result of path-oriented test case generation techniques.	Array
Complexity	An indicator to represent a complexity of test case. The complexity of test cases represents how difficult to execute each test case.	Numeric
Impact	An indicator to represent an impact value in case that test case is ignored.	Numeric
Coverage	An indicator to represent how many states each test case cover.	Numeric
Status	An indicator to represent that test case can reveal faults or not. The status can be only either pass or fail. If the status is fail, it mean that fault is detected.	Boolean

The following table describes an approach to generate random data using the above attributes respectively.

Table 2 Approach to Generate Random Test Case

Attribute	**Approach**
Test Id	Generate randomly from the following combination: t + *Sequence Number*. For example, t1, t2, t3, …, tn.
Name	Random from the following values: iPhone, BlackBerry, Nokia, LG, Sony and Samsung.
ImpValue	Set as a zero (0) at the beginning
wCoverageValue	Set as a one (1) at the beginning
A set of states	There are two elements needed to be randomly generated: (a) a number of states that needed to be tested by each test case and be generated between 1 and 100. (b) states themselves that described as follows: Generate randomly from the following combination: s + *Sequence Number*. For example, s1, s2, s3, …, sn.
Cplx	Random from the following values: 1-100
Impact	Random from the following values: 1-100
Coverage	Compute a number of states from “a set of states” field

The paragraph lists the measurement metrics used in the experiment. The first measurement is a number of test cases. The large number of test cases consumes time, effort and cost more than the smaller size of test cases. Many reduction or minimization approaches [1], [10], [11], [12], [13], [14], [15], [24], [25], [36], [37], [39] have been proposed to minimize size of test cases. This has proven that size is one of important metrics in software testing area. The second is an ability to reveal faults. It aims to measure the percentage of faults detection. One of the goals of test case with a set of data is to find defects. Thus, this metric is important criteria to measure and determine which reduction methods can preserve the high ability to reveal faults. The last measurement is a total of reduction time: It is the total number of times running the reduction methods in the experiment. This metric is related to time used during execution time and maintenance time of test case reduction methods. Therefore, less time is desirable. This paragraph discusses an evaluation result of the above experiment. This section presents the reduction methods results in term of: (a) a number of test cases (b) ability to reveal faults and (c) total reduction time. The comparative methods are: RD, UD, FD, FUD, ICF, TCCF, TCIF and PCF. Additionally, this section shows a graph format. There are two dimensions in the following graph: (a) horizontal and (b) vertical axis. The horizontal represents three measurements whereas the vertical axis represents the percentage value.

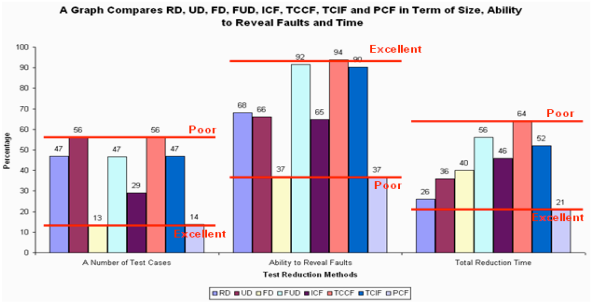


Figure 2 A Graph Comparison of Deletion Methods

The above graph presents that both of FD and PCF minimize a number of test cases by far better than other reductions methods, approximately over 15%. Meanwhile, both of them are the worst methods for preserving an ability to reveal faults. FUD, TCCF and TCIF are best top three methods to reserve a capability to detect faults. They are greater than other methods over 22%. Unfortunately, they are also the worst three methods that require a lot of time during a reduction process. In the mean time, both of RD and PCF take the least total reduction time among other methods. The evaluation result suggests that FD and PCF is perfectly suitable for a scenario that does not directly concern about an ability to reveal faults and total reduction time. Both of FD and PCF are two of the most excellent methods to minimize a number of test cases. Meanwhile, FUD, TCCF and TCIF are the most recommended methods to delete tests while preserving the ability to detect faults. In addition, both of RD and PCF are excellent in case that total time is matter.

7. CONCLUSION

This paper reveals that there are many research challenges and gaps in the test case reduction area. Those challenges and gaps can give the research direction in this field. However, the research issues that motivated this study are: (a) too many redundancy test cases after reduction process (b) a decrease of test cases' ability to reveal faults and (c) uncontrollable grow of test cases. This paper combines the concept of software testing and CBR. Those two concepts could be used together on practical software development scenarios. The proposed maintenance algorithms are significant approaches for removing unnecessary test cases and are used for controlling the growth of test cases. Those approaches are aimed at maintaining the large test cases by minimizing the time consumed by execution & maintenance and reducing the size of the test cases along with preserving the ability to reveal faults as much as possible. Also, the evaluation reveals that they have been achieved by removing a number of test cases, minimizing time for executing & maintenance and preserving the fault-detection ability with sample of 2,000 test cases. However, the primarily limitation of those approaches is about the path coverage. The path coverage may be not an effective coverage factor for a huge system that contains million lines of code. This is because it requires an exhaustive time and cost of identify coverage from a huge amount of codes. Thus, one of the future works is to apply other coverage factors for those approaches. Finally, this paper recommends researchers to improve the ability to reduce duplicated or unnecessary test cases from multiple test suites, enhance the capability to reduce test cases in the large commercial system and develop a systematic approach to identify an impact and complexity of tests.

8. REFERENCES

[1] A. Jefferson Offutt, Jie Pan and Jeffery M. Voas, "Procedures for Reducing the Size of Coverage-based Test Sets", 1995.

[2] Barry Smyth & Keane. "Remembering To Forget: A Competence Preserving Deletion Policy for Case-Based Reasoning Systems" In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 377-382. Morgan-Kaufman, 1995.

[3] Barry Smyth Ph.D. Thesis. "Case Based Design" Department of Computer Science, Trinity College, Dublin Ireland, 1996.

[4] Barry W. Boehm, "A Spiral Model of Software Development and Enhancement", TRW Defense Systems Group, 1998.

[5] Boris Beizer, "Software Testing Techniques, Van Nostrand Reinhold", Inc, New York NY, 2nd edition. ISBN 0-442-20672-0, 1990.

[6] Bo Qu, Changhai Nie, Baowen Xu and Xiaofang Zhang, "Test Case Prioritization for Black Box Testing", 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), 2007.

[7] Cem Kaner, "Exploratory Testing", Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, FL, 2006.

[8] David C. Wilson. Ph.D. Thesis "A Case-Based Maintenance: The husbandry of experiences." Department of Computer Science, Indiana University, 2001.

[9] E. Lehmann and J. Wegener, "Test case design by means of the CTE XL", In Proc. of the 8th European International Conf. on Software Testing, Analysis & Review (EuroSTAR 2000), 2000.

[10] Gregg Rothermel, Roland H. Untch, Chengyun Chu and Mary Jean Harrold, "Prioritizing Test Cases for Regression Testing", IEEE Transactions on Software Engineering, 2001.

[11] Gregg Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test case prioritization: An empirical study", In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 179-188, Oxford, England, UK, 1999.

[12] Gregg Rothermel, Mary Jean Harrold, Jeffery Ostrin and Christie Hong, "An Empirical Study of the Effects of Minimization on the Fault Detection Capabilities of Test Suites", In *Proceedings of IEEE International Test Conference on Software Maintenance (ITCSM'98)*, Washington D.C., pp. 34-43, 1998.

[13] Gregg Rothermel, Mary Jean Harrold, Jeffery von Ronne and Christie Hong, "Empirical Studies of Test-Suite

Reduction", In Journal of Software Testing, Verification, and Reliability, Vol. 12, No. 4, 2002.
[14] Gregg Rothermel and Mary Jean Harrold, "A Safe, Efficient Regression Test Selection Technique", ACM Transactions on Softw. Eng. And Methodology, 6(2): 173-210, 1997.
[15] Gregg Rothermel and Mary Jean Harrold, "Analyzing Regression Test Selection Techniques", IEEE Transactions on Software Engineering, 22(8):529-551, 1996.
[16] Jirapun Daengdej, Ph.D. Thesis, "Adaptable Case Base Reasoning Techniques for Dealing with Highly Noise Cases" The University of New England, Australia, 1998.
[17] Jun Zhu and Quiang Yang. "Remembering To Add Competence-preserving Case Addition Policies for Case Base Maintenance." In *Proceedings of the 16th International Joint Conference in Artificial Intelligence*, 234-241. Morgan-Kaufmann, 1999
[18] Mary Jean Harrold, Rajiv Gupta and Mary Lou Soffa, "A Methodology for Controlling the Size of A Test Suite", ACM Transactions on Software Engineering and Methodology, 2(3):270-285, 1993.
[19] Nicha Kosindrdecha and Jirapun Daengdej, "A Deletion Algorithm for Case-Based Maintenance Based on Accuracy and Competence", Assumption University, Thailand, 2003
[20] Nicha Kosindrdecha and Siripong Roongruangsuwan, "Reducing Test Case Created by Path Oriented Test Case Generation", AIAA 2007 Conference and Exhibition, Rohnert Park, California, USA, 2007.
[21] NIST, "The economic impacts of inadequate infrastructure for software testing", 2002.
[22] Saif-ur-Rebman Khan and Aamer Nadeem, "TestFilter: A Statement-Coverage Based Test Case Reduction Technique", 2006.
[23] Sara Sprenkle, Sreedevi Sampath and Amie Souter, "An Empirical Comparison of Test Suite Reduction Techniques for User-session-based Testing of Web Applications", Journal of Software. Testing, Verification, and Reliability, 4(2), 2002.
[24] Scott McMaster and Atif Memon, "Call Stack Coverage for Test Suite Reduction", *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05)*, pages 539-548, Budapest, Hungary, 2005.
[25] Scott McMaster and Atif Memon, "Call Stack Coverage for GUI Test-Suite Reduction", Proceedings of the 17th IEEE International Symposium on Software Reliability Engineering (ISSRE 2006), NC, USA, 2006.
[26] Scott McMaster and Atif Memon, "Fault Detection Probability Analysis for Coverage-Based Test Suite Reduction", IEEE, 2007.
[27] Sreedevi Sampath, Sara Sprenkle, Emily Gibson and Lori Pollock, "Web Application Testing with Customized Test Requirements – An Experimental Comparison Study", 17th International Symposium on Software Reliability Engineering (ISSRE'06), 2006.
[28] Siripong Roongruangsuwan and Jirapun Daengdej, "Techniques for improving case-based maintenance", Assumption University, Thailand, 2003
[29] Siripong Roongruangsuwan and Jirapun Daengdej, "Test Case Reduction", Technical Report 25521. Assumption University, Thailand, 2009.
[30] S. Elbaum, A. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies", IEEE Trans. on Software Engineering, vol. 28, 2002.
[31] S. Elbaum, A. G. Malishevsky and G. Rothermel, "Prioritizing Test Cases for Regression Testing", In *Proceedings of the International Symposium on Software Testing and Analysis*, pages 102-112, 2000.
[32] S. Elbaum, P. Kallakuri, A. G. Malishevsky, G. Rothermel, and S. Kanduri, "Understanding the effects of changes on the cost-effectiveness of regression testing techniques", Journal of Software Testing, Verification, and Reliability, 13(2):65-83, 2003.
[33] Todd L. Graves, Mary Jean Harrold, Jung-Min Kim, Adam Porter and Gregg Rothermel, "An Empirical Study of Regression Test Selection Techniques", 2000.
[34] W. Eric Wong, J. R. Horgan, Saul London and Hira Agrawal, "A Study of Effective Regression Testing in Practice", 8th IEEE International Symposium on Software Reliability Engineering (ISSRE'97), 1997.
[35] W. Eric Wong, Joseph R. Horgan, Saul London and Aditya P. Mathur, "Effect of Test Set Minimization on the Fault Detection Effectiveness of the All-Uses Criterion", In *Proceedings of the 17th International Conference on Software Engineering*, pages 41-50, 1995.
[36] Xiaofang Zhang, Baowen Xu, Changhai Nie and Liang Shi, "An Approach for Optimizing Test Suite Based on Testing Requirement Reduction", Journal of Software (in Chinese), 18(4): 821-831, 2007.
[37] Xiaofang Zhang, Baowen Xu, Changhai Nie and Liang Shi, "Test Suite Optimization Based on Testing Requirements Reduction", International Journal of Electronics & Computer Science, 7(1): 9-15, 2005.
[38] Xue-ying MA, Bin-kui Sheng, Zhen-feng HE and Cheng-qing YE, "A Genetic Algorithm for Test-Suite Reduction", IEEE, China, 2006.
[39] Yanbing Yu, James A. Jones and Mary Jean Harrold, "An Empirical Study of the Effects of Test-Suite Reduction on Fault Localization", *Proceedings of ICSE'08*, Germany, 2008.

EVOLUTION SUPPORT FOR MODEL-BASED DEVELOPMENT AND TESTING SUMMARY

Stephan Bode, Qurat-Ul-Ann Farooq, Matthias Riebisch
{stephan.bode | qurat-ul-ann.farooq | matthias.riebisch}@tu-ilmenau.de

Ilmenau University of Technology, Ilmenau, Germany

1. INTRODUCTION

The First International Workshop on Evolution Support for Model-Based Development and Testing (EMDT2010) was held on September 16, 2010 in Ilmenau, Germany. After a keynote and several paper presentations a workshop discussion was held.

2. GOALS OF THE DISCUSSION

The goal of the workshop discussion was to identify the key challenges, research questions and ideas for the support of evolution in software development and testing. Initiated by keynote and presentations, the participants from industry and academia should exchange their experiences and ideas.

3. DISCUSSION OF THE TERM SOFTWARE EVOLUTION

3.1. Key aspects of the term Evolution

Unfortunately, a clear definition of the term evolution is missing. According to Lehman and Ramil (chapter 1 of [1]), the term evolution reflects "a process of progressive, for example beneficial, change in the attributes of the evolving entity or that of one or more of its constituent elements. What is accepted as progressive must be determined in each context. It is also appropriate to apply the term evolution when long-term change trends are beneficial even though isolated or short sequences of changes may appear degenerative. For example, an entity or collection of entities may be said to be evolving if their value or fitness is increasing over time. Individually or collectively they are becoming more meaningful, more complete or more adapted to a changing environment. "

Our understanding of the term related to the workshop theme covers the following key aspects:

- Modification, change, progress, extension over time
- State of an artefact at different points of time
- Models change: dynamic versus static
- Evolution versus revolution while revolution means the replacement of an existing system by a new one

Two examples for evolution shall illustrate the change of the states:

- A change of natural language requirements leads to a change of the conceptual model, which in turn leads to a change of the class diagram as vertical evolution. This chain has to be traceable backwards.
- A change of the initial requirements (e.g. use cases) leads to a change of the functional specification, e.g. expressed by a visual contract: with pre and post conditions.

3.2. Levels and dimensions of evolution

Evolution of models in a stepwise incremental development in two dimensions:

Horizontal: to add one part after the other, leading to an increased functionality:

- V1 views PDF files,
- V2 views PDF and JPG files

Vertical: to develop parts to detailed level, leading to further refinement:

- From abstract specification to components and to code
- To achieve horizontal evolution, some vertical evolution steps may be necessary.

Evolution results in a traceable sequence of parts.

4. ASPECTS OF SEMANTICS TO BE EXPRESSED IN MODELS

A formal definition of semantics is important for transformability. The following aspects have to be expressed in such a way:

- Structure
- Class diagrams, component diagrams
- Behaviour
- State Charts, Petri Nets
- Conceptual models
- Ontology
- Functional specification
- Visual contracts [2], Java Modeling Language JML: pre and post conditions

5. IDENTIFIED RESEARCH CHALLENGES

The participants identified a set of research challenges

- Mastering complexity: modularization vs. comprehension
- Appropriate level of detail
- Appropriate models (views) for different types of tests
- Appropriate models (notations) for different domains
- Models to cover the relevant aspects of real world
- Decision on separate models for specification, for testing and development as an overhead or necessity
- Expression of semantics of data transformation / functionality
- Dependency relations between models
- Means to bridge the gaps / to overcome the walls between the stages of development
- Usage of ontologies to bridge the gap between informal requirements and design models
- Identification of generalized change types according to their consequences for different development activities
- Tool integration: establishing appropriate meta models and interfaces for:
    - Model creation
    - Code generation
    - Test case generation
- Impact analysis for evolution support: how to identify artefacts affected by changes
- Analysis of the impact of evolution on generated artefacts
- Reuse of the development artefacts during evolution, including test cases
- Software product lines – planned reuse vs. evolution
- Definition of formal criteria for evolution: legal issues for example regarding copyright

From the discussion we can conclude that all mentioned issues are related to the questions:

- Which models are necessary
- How to express the relevant aspects in models
- How to evaluate and to utilize models

6. ACKNOWLEDGEMENTS

We want to thank the contributors of the discussion for their input and their statements: Sven Biegler, Ilmenau University of Technology, Germany; Jirapun Daengdej, Assumption University, Thailand; Stefan Groß, Ilmenau University of Technology Germany; Baris Güldali, University Paderborn, Germany; Christian Kop, University Klagenfurt, Austria; Bernd-Holger Schlingloff, Humboldt University, Germany.

7. REFERENCES

[1] Meir Lehman and Juan C. Fernández-Ramil: Software Evolution. In: Nazim H. Madhavji, Juan C. Fernández-Ramil and Dewayne E. Perry: Software Evolution and Feedback: Theory and Practice, Wiley, 2006, pp 7-40.

[2] Mark Lohmann, Stefan Sauer, Gregor Engels: Executable visual contracts. In: 2005 IEEE Symposium on Visual Languages and Human-Centric Computing, pp. 63-70, IEEE CS Press, 2005.