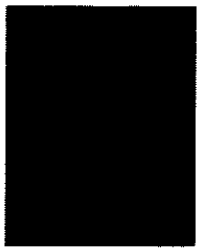
Workshop proceedings

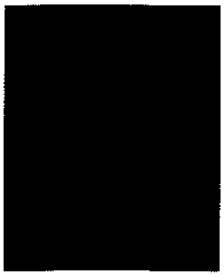
11th International Conference on Model Driven Engineering Languages and Systems
Toulouse, France
September 28 – October 3, 2008

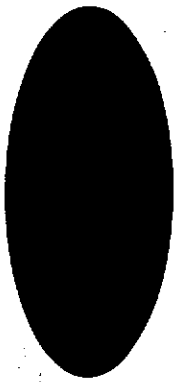
Empirical Studies of Model-Driven Engineering (ESMDE'08)

September 29

ISSN 1613-0073 Vol 392
http://ceur-ws.org/Vol-392
CEUR-WS.org







simula . research laboratory]

Preface

It is often difficult to rigorously evaluate Model-Driven Engineering (MDE) technologies. Performing empirical studies require skills, experience and tacit knowledge that are in many ways very different from the "core" MDE research. Furthermore, empirical studies often entail large investments in terms of human resources, time and money. Nevertheless, evaluations of MDE technologies are needed in order to demonstrate the soundness, applicability, and cost effectiveness of proposed technologies in various development contexts.

The aim of this workshop is to exemplify and discuss ways in which proposed model-driven engineering (MDE) technologies should be evaluated, with a specific emphasis on how to plan, conduct, analyze and report the results of empirical studies. The workshop will have focus on the challenges of empirical studies involving human users, since MDE technologies are typically expected to be used by software engineers to improve various quality aspects of software systems and the productivity of software development. More detailed topics include: What are the main obstacles and potential remedies when performing empirical studies of MDE? What are the main threats to validity of empirical studies of MDE, and how should they be dealt with? For example, since MDE often represent new and complex technology, the selection and training of human subjects who participate in empirical studies often become critical factors. What are the most important outcome variables of the costs and benefits of MDE? How can quality be measured in the context of MDE? And can we define an unambiguous set of (benchmark) outcome measures to facilitate meta-analyses across subjects, systems, tasks and technologies?

The goal of the workshop is to pave the way for the development of a MDE-specific framework for empirical evaluation of MDE technologies, or at least provide a minimum standard for evaluation that published work in the MDE community should abide by.

Erik Arisholm
Lionel Briand
Bente Anda

Program committee

Colin Atkinson, University of Mannheim, Germany

Christian Bunse, International University, Germany

Michel Chaudron, Eindhoven University of Technology, Netherlands

Massimiliano Di Penta, University of Sannio, Italy

Robert B. France, Colorado State University, USA

Marcela Genero, University of Castilla-La Mancha, Spain

Marianne Huchard, University of Montpellier II, France

Ferhat Khendek, Concordia University, Canada

Yves Le Traon, IT/Telecom Bretagne, France

Tim Menzies, West Virginia University, USA

Alexander Pretschner, ETH Zurich, Switzerland

Per Runeson, Lund University, Sweden

Houari Sahraoui, University of Montreal, Canada

Miroslaw Staron, IT University of Göteborg, Sweden

Content

Towards Quality-Driven Model Transformations: A Replication Study

Emilio Insfran[1], José Ángel Carsí[1], Silvia Abrahão[1], Marcela Genero[2], Isidro Ramos[1], Mario Piattini[2]

[1] ISSI Group, Department of Information Systems and Computation
Universidad Politécnica de Valencia
Camino de Vera, s/n, 46022, Valencia, Spain
{einsfran, pcarsi, sabrahao, iramos}@dsic.upv.es

[2] ALARCOS Group, Department of Information Systems and Technologies,
University of Castilla-La-Mancha
Paseo de la Universidad Nº 4, 13071, Ciudad Real, Spain
{Marcela.Genero, Mario.Piattini}@uclm.es

Abstract. Commonly, there are several ways to transform a source model into a target model. These alternative target models may have the same functionality but can differ in their quality attributes. One of the key challenges of an automated transformation process is to identify the transformations that will produce a target model with the desired quality attributes. In this paper, we present a replica of a controlled experiment to investigate the selection of alternative transformations to obtain UML class models from a Requirements Model. This is a concrete instantiation of a wider domain-independent approach for quality-driven model transformation. Specifically, we focus on a set of transformations related to structural relationships between classes (association, aggregation and association class) and the understandability quality attribute. Although, some results could be foreseen even by a superficial analysis of the alternatives, the goal of this work is to use experimentation to gather empirical evidence about which alternative transformation produces the UML class model that is the easiest to understand. The empirical results support the original results showing that there is a tendency to favor the use of association relationships to drive these transformations when understandability is chosen.

Keywords: Model transformations, MDA, Software Quality, Requirements, UML class model, Empirical Software Engineering.

1 Introduction

Model-Driven Architecture (MDA) is an emerging approach to software development. It promotes the use of models and model transformations as the primary artefacts to be built and maintained. In essence, an MDA development process transforms a Platform-Independent Model (PIM) into one or more Platform-Specific Models (PSM), which are then transformed into code (Code Model – CM). Therefore,

in this context, a model is no longer simply a means for describing software, but rather an essential piece of the software development process. Consequently, the quality of the models built throughout this process is of great significance since these models will determine the quality of the software product that is finally deployed.

Usually, in an MDA development process there are several ways to transform a source model into a target model. Alternative target models may have the same functionality but differ in their quality attributes. One of the key challenges for an *automated transformation process* is to identify which transformations will produce a target model with the desired quality attributes (e.g., understandability, modifiability).

In the last few years, some approaches that deal with quality in Model-Driven Engineering (MDE) have been proposed [15] [14] [12] [13] [8] [9]. One disadvantage of these approaches is that the practical applicability of model transformations is often reported based on the intuition of the researcher. As pointed out by Czarnecki and Helsen [4], there is a lack of controlled experiments to fully validate the observations made by the researchers in the field of MDE. Therefore, more systematic approaches to ensure quality in MDE processes are needed.

In this paper, we present a replica of a controlled experiment to investigate the selection of alternative transformations to obtain UML class models from a Requirements Model [5] [6]. This work is part of a project on *quality-driven model transformations* whose overall goal is the definition of a quality metamodel to drive the selection of alternative model transformations according to different quality attributes.

Specifically, we focus on a sub-set of transformations that are related to structural relationships between classes (association, aggregation, and association class) and the 'understandability' quality attribute. These transformations have been selected because the determination of structural relationships has a great impact on the UML class model. Understandability has been selected since it is recognized as the main quality attribute that influences maintainability. A UML class model must be understood before any desired change to it can be identified, designed or implemented.

The goal of the experimentation is to gather empirical evidence about which alternative transformation produces the UML class model that is the easiest to understand. The empirical evaluation of the best transformation is particularly important when the transformations are automatically applied, which is the main reason for adopting MDA [11].

This paper is organized as follows. Section 2 gives an overview of our approach to transform a Requirements Model into UML class models. This section also shows the analysis of alternative model transformations. Section 3 describes the design of the original experiment and its replica to empirically validate the selection of alternative transformations with regard to the understandability. Section 4 presents the data analysis and the interpretation of the results. Finally, section 5 presents our conclusions and future work.

2 Transforming Requirements into UML Class Models

Following an MDA development approach, it is important to ensure quality in every step of the development process. In this context, automated model transformation plays a key role for success. We propose to empirically validate model transformations with regard to quality attributes and use this information to drive the selection of alternative transformations. A quality attribute is a measurable physical or abstract property of an entity (e.g., conceptual model) [7].

A model transformation is executed taking a transformation definition as input. A transformation definition contains transformation rules that relate constructs in the source model to constructs in the target model. We use another input for the transformation process that is the definition of the quality attributes together with the corresponding empirical evidence gathered from controlled experiments. This information will feed the transformation process with the criteria to choose the alternative transformation that maximizes the selected quality attribute. The rationale of this approach is to be able to automatically select the alternative transformation that an experienced software developer would select if the transformation process were manually applied. In this section, we present a concrete application of the quality-driven model transformation approach to transform software requirements into UML class models (see Fig. 1).

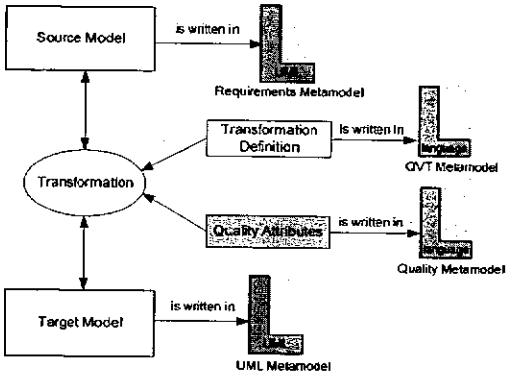


Fig. 1. The quality-driven model transformation approach

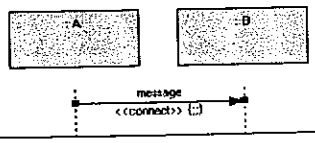
The Requirements Model [5] [6] defines the structures and the process followed to capture the software requirements following an MDA approach. It is composed of a *Functions Refinement Tree (FRT)* to specify the hierarchical decomposition of the system functionality, a *Use Case Model* to specify the system communication and functionality, and *Sequence Diagrams* to specify the required object-interactions necessary to realize each Use Case. This Requirements Model is supported by a Requirements Engineering TOol – RETO (http://reto.dsic.upv.es). Once the Requirements Model has been specified, a conceptual model including a UML class model can be obtained by applying a set of transformations from the Transformation Rules Catalog (TRC) [5]. These transformation rules establish traceability relationships between the Requirements Model and the UML class models.

The application of a Transformation Rule (TR) implies that a certain structural pattern match in the Requirements Model and that a resultant structure in the UML class model can be generated while establishing a traceability relationship between them. Some transformations are easy to apply once the transformation pattern has been matched (a *one-to-one* relation). For example, the generation of classes for the UML class model is based on the analysis of participating classes in all the Sequence Diagrams. However, other transformations are not easy to identify or apply for three main reasons: the complexity of the transformation pattern, the non-disjoint condition pattern of the transformation, and the multiple valid representation of a conceptual model for a given requirement pattern.

2.1 Analyzing Alternative Transformations

Following, we explain with examples some transformations where multiple alternatives arise because of non-disjoint condition patterns and multiple possible representations of the same pattern. Fig. 2 shows the Sequence Diagram used to specify the necessary object interactions to realize the Use Case *Create Insurance* of a Car Rental system. This Use Case represents the creation of a car *Insurance* policy that must be bought from an *Insurance Company* and assigned to the *Car* before using the car for rentals. The actor *Administrator* initiates the interaction (message 1). After introducing the necessary data and checking the existence of the corresponding car (messages 2 to 4), a new *Insurance* object is created (message 5). In addition, an *Insurance Company* object and a *Car* object must be connected to the newly created *Insurance* policy object (messages 6 and 7).

After analyzing the requirement specification provided by the previous Sequence Diagram, the analyst of the system could possibly determine that the partial Class Model that best represents this requirement is defined by two association relationships that relate the newly created class *Insurance* (message 5) to the *InsuranceCompany* and *Car* classes as shown in Fig. 3(a). This partial Class Model can be obtained by applying twice the following transformation rule TR15 (Association) from the Transformation Rule Catalog to the messages 6 and 7:

TR15 (Association): *For every message between two classes labeled with the stereotype «connect», THEN an association relationship between these classes will be generated.*	
Pattern condition:	**Result:**
	

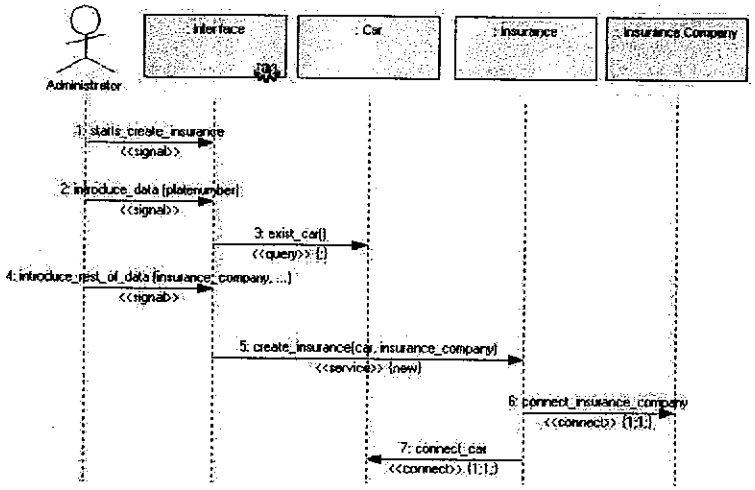


Fig. 2. Sequence Diagram showing the required interactions for the Use Case *Create Insurance*

As an alternative solution to the same requirement specification, the analyst could prefer a solution that uses an association class (named *Insurance*) to relate the *Car* rented with the *Insurance Company* as shown in Fig. 3 (b). This partial Class Model can be obtained by applying the transformation rule TR39 (AssociationClass) to the messages 5, 6 and 7. Finally, another possibility is the definition of two aggregation relationships, one between *Insurance Company* and *Insurance* classes and another between *Car* and *Insurance* classes as shown in Fig. 3 (c). This partial Class Model can be obtained by applying twice the transformation rule TR28b (Aggregation) to the messages 6 and 7.

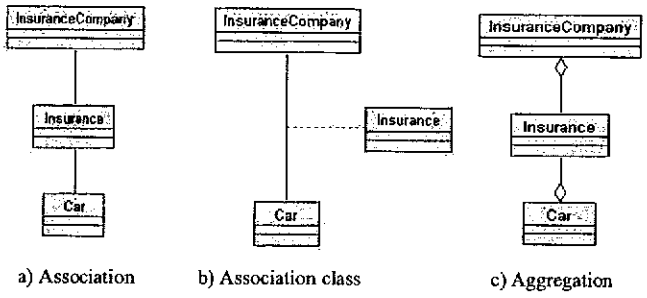


Fig. 3. Two partial Class Models for the Car Rental system

An important issue derived from these examples is that different alternative structural relationships can be derived from the analysis of the requirements specifications described using Sequence Diagrams. And what is more important is the fact that the part of the information used in the Sequence Diagrams for deriving these

structural relationships follows the same pattern: a message with the stereotype *new*, followed by two messages with a stereotype *connect*.

In the examples, we have shown that TR15, TR39, or TR28b transformation rules could be applied, depending on the interpretation of the analyst if the transformation process is performed manually. In an automated transformation process, the information provided by the stereotyped Sequence Diagrams is not sufficient to automatically determine which structural relationship of a Class Model best realizes the specified requirement. Because these alternatives exist, it is the human analyst who should decide which alternative better represent a solution in the corresponding problem domain. In this work, considering that alternatives exist, we use controlled experiments to discover which alternative transformation maximizes a given quality attribute of the resulting target model (e.g., understandability).

3 A Controlled Experiment and its Replica

In a previous work [1], we presented a controlled experiment to determine which of the transformation rules for structural relationships: association (A1), aggregation (A2), or association class (A3), obtained the easiest to understand UML class model. The results show a slight tendency to favor the transformations related to associations (A1). However, as Basili et al. [2] suggested relevant and credible results can only be obtained by replicating the experiments. In other words, single studies rarely provide definitive answers. Therefore, we carried out an internal strict replication (changing only the subjects) to corroborate the findings. To provide an overall view of the experimentation, we explain both the original experiment and the replica.

3.1 Planning

The participants in the original experiment were 39 fourth-year students in Computer Science at the Universidad Politécnica de Valencia, who were taking part in the second Software Engineering course. We took a "convenience sample" (i.e., all the students in the class). The subjects had six month of experience in modeling with UML and three years of experience in the OO paradigm. The subjects that participated in the replica were a different group of 37 students from the same course.

The independent variables for the experiments were the transformation rules for structural relationships between classes (A1, A2 and A3). The dependent variable was understandability. The experimental material and tasks consisted of:

- 9 Sequence Diagrams from three different case studies (a car rental system, a hotel management system, and a singer contest system), with 3 UML class models each. These were obtained by applying the alternative transformation rules. The material used is available at: www.dsic.upv.es/~einsfran/experiment.
- Each Sequence Diagram had a questionnaire attached consisting of 6 Yes/No questions to test the subjects' understanding of the Sequence Diagrams. The effectiveness of the subjects in answering the questionnaires (number of correct

answers by number of answers) was used to exclude those observations that did not fulfill a minimum level of quality.

- Each of the three UML class models had a questionnaire attached (with 6 questions) for assessing which UML class model was best understood by the subjects. In addition, the subjects had to write down the starting and ending times for completing the questionnaires. For this purpose we used a wall clock. From this understanding task, we obtained three measures for understandability:
  Understandability Time, which reflects the time, in seconds, that the subjects spent answering each questionnaire (calculated by the difference between the ending time and the starting time). Each subject completed 4 questionnaires detailing 3 alternatives (A1, A2 and A3). Three understandability time measures (A1Time, A2Time and A3Time) were obtained.
  Effectiveness, which reflects the correctness of the answers (number of correct answers by number of answers). Three understandability effectiveness measures (A2Effec, A2Effec and A3Effec) were obtained.
  Efficiency, which reflects the correctness of the answers by time (number of correct answers by understandability time). Three measures for understandability efficiency (A2Effic, A2Effic and A3Effic) were obtained.
- The final task of each test asked the subjects which of the three alternative UML class models best reflected the problem modeled in the Sequence Diagram. It is a subjective measure (*Alternative Selected*) based on the subjects' perception.

The following hypotheses were formulated:

- **$H1_0$**: The use of different alternative transformations does not affect the Understandability Time (A1Time, A2Time and A3Time). $H1_1 = \neg H1_0$
- **$H2_0$**: The use of different alternative transformations does not affect the Understandability Effectiveness (A1Effec, A2Effec and A3Effec). $H2_1 = \neg H2_0$
- **$H3_0$**: The use of different alternative transformations does not affect the Understandability Efficiency (A1Effic, A2Effic and A3Effic). $H3_1 = \neg H3_0$
- **$H4_0$**: There is no correlation between the Alternative Selected and the means of objective Understandability variables (Understandability Time, Effectiveness, and Efficiency). $H4_1 = \neg H4_0$

We selected a balanced within-subject design, i.e., each subject received the same experimental material.

3.2 Execution

Both the original experiment and the replica started with an introductory session in which we reviewed the main concepts of the Requirements Model (e.g., the notation of Sequence Diagrams). The goal of the experiment was not disclosed to the subjects. Then, we showed an example of the experimental material, which was similar to the material they would use during the execution of the experiment.

Each subject was assigned all the material, with the nine tests (balanced within-subject design). The models were assigned in different order to limit learning effects. We showed them how to develop the experimental tasks, and they had a maximum of two hours to complete all the tasks.

After the experiment took place, we collected the experiment data, which consisted of a table of 351 rows (9 models x 39 subjects) and 9 columns (A1Time, A2Time, A3Time, A1Effec, A2Effec, A3Effec, A1Effic, A2Effec, A3Effec). The replica had the same structure, but with 331 rows (9 models x 37 subjects). In both samples, we performed a "data cleaning", excluding the observations that were not complete because the subjects had not written down the time or because the subjects did not select the best alternative. All the questions were answered in each questionnaire, thereby assuring the completeness of the performed tasks. We also excluded the observations that had a value of effectiveness of 50% or less for each Sequence Diagram. We considered that if the level of correct answers was low in relation to the Sequence Diagram, the subjects had not really understood the model, and they would probably not perform well in the following tasks, so we discarded them. Therefore, the final data for testing the hypotheses were 325 observations for the original experiment and 293 for the replica.

4 Data Analysis and Interpretation

The following statistical analyses were performed to analyze the data: (1) a descriptive study was done to characterize the variables Alternative Selected, Understandability Time, Effectiveness, and Efficiency; (2) Hypotheses H1, H2, and H3 were tested using an ANOVA test with repeated measures; (3) Hypothesis H4 was tested using the Spearman correlation coefficient.

We used SPSS to carry out the data analyses presented in this study. The transformation most selected by the subjects was A1, i.e., the subjects believed that the use of associations obtained the easiest to understand UML class model. Association class (i.e., alternative A3) was the transformation that was least selected, which reveals that it could be the least appropriate transformation.

The descriptive statistics we carried out for Understandability Time, Effectiveness and Efficiency suggest the following:

- **Original Experiment.** On average, the subjects spent less time performing the tasks related to alternative A2; however, the difference with the others is not very significant (≈ 8 seconds for A1 and A3). The subjects were more effective and efficient performing the tasks related to alternative A1; however, the difference in effectiveness with the other alternatives is not very significant.
- **Replica.** The measures related to A1 have the best values, which means that, on average, the subjects spent less time and were more effective and more efficient performing the tasks related to the class model that was obtained via associations.

In summary, the descriptive statistics show a slight tendency in favor of A1, the transformation based on associations. Surprisingly, the difference between the minimum and maximum time values is significant. This may be due to the fact that the subjects were novice modelers.

To test the hypotheses presented in section 3.1, we carried out an ANOVA for repeated measures, which is the appropriate statistical test for analyzing the collected data [10]. Due to space constraints, we will briefly present the main findings obtained through the ANOVA for each data sample:

- **Original Experiment.** We can reject hypotheses $H1_0$ ($p = 0.0002$), $H2_0$ ($p = 0.0001$), and $H3_0$ ($p = 0.0005$), with a significance level = 0.05. This means that the use of different alternative transformations really affects understandability time, effectiveness, and efficiency.
- **Replica.** We can reject $H1_0$ ($p = 0.0028$) and $H2_0$ ($p = 0.0003$), which means that the use of one alternative or another does not affect efficiency but does affect time and effectiveness when the subjects understand the class models.

When planning the experiment, we designed it in such a way as to alleviate the threats to the internal validity.

One limitation to the external validity (i.e., the generalization of the findings) of this study is the fact that the three alternative transformation rules cannot be applied simultaneously to all modeling situations. For instance, to establish an association class relationship (A3), at least one «service/new» message and two «connect» messages are needed in the source model. The goal of this experimentation was to gather empirical evidence for the specific case when the three alternative transformations could be applied to obtain a relationship between classes. We are aware that, more alternatives may be possible to represent structural relationships between classes. More experimentation is needed to validate these other combinations. Another limitation to the external validity might be the use of students as experimental subjects. However, the students who participated in the experiment can be considered to be representative of novice users of conceptual modeling approaches. To increase external validity, the current study needs to be replicated using experienced practitioners.

5 Conclusions and Future Work

This paper has presented an analysis of alternative model transformations and how controlled experiments can be used to provide useful information to guide the selection of transformations in an automated transformation process. In particular, we presented the results of an experiment to gather evidence about which alternative transformation produces the UML class model that is easiest to understand.

The main findings obtained from the experimentation are the following: (a) the transformation that was most selected in the original experiment and the replica was the association transformation (A1); (b) the results of the replica confirm the results of the original experiment for effectiveness. The subjects were more effective when they understood the class models with association relationships; (c) the fact that the hypothesis related to efficiency could not be confirmed in the replication has no great impact on our approach since the transformations are automatically executed in a Model Management framework (MOMENT) [3].

These results show that there is a slight tendency to favor the use of association relationships as part of an automated transformation process. A possible reason for this could be that this relationship has less semantic strength than the other kinds of relationships. When an aggregation relationship is chosen instead of an association relationship, analysts know that they are defining a part-of relationship. In the case of an association class, it is possible to represent almost the same relationship using two

association relationships. Although the results obtained can be quite obvious the important point is the systematic approach presented to validate this 'obvious' results. That the association relationship is more understandable than the aggregation relationship or association classes is something that almost all the people can say but until this moment no one has the data to confirm that result but merely by intuition.

These preliminary results provided empirical evidence that can be further used to define a domain-independent quality metamodel to drive the execution of model transformations. Nevertheless, more replication is needed for building a body of knowledge. We plan to replicate this experiment with practitioners. Future work also includes the evaluation of the remaining transformations of the Transformation Rules Catalog taking into account other quality attributes (e.g., usability, modifiability).

Acknowledgments. This research is part of the META project TIN2006-15175-C05-05, the MECENAS project PBI06-0024, and the IDONEO project PAC08-0160-6141.

References

1. Abrahão, S., Genero, M., Insfran, E., Carsí, J.A., Ramos, I., Piattini, M.: Quality-Driven Model Transformations: From Requirements to UML Class Diagrams. In: Model-Driven Software Development: Integrating Quality Assurance, IGI Publishing, 2008.
2. Basili, V., Shull, F., Lanubile F.: Building Knowledge through Families of Experiments. In: IEEE Transactions on Software Engineering, 25(4): 435–437, 1999.
3. Boronat, A., Carsí, J.A., Ramos, I.: Algebraic Specification of a Model Transformation Engine. In: Fundamental Approaches to Software Engineering (FASE'06). ETAPS'06. Vienna, Austria, 2006, pp. 262–277.
4. Czarnecki, K., Helsen, S.: Feature-Based Survey of Model Transformation Approaches. In: IBM Systems Journal 45(3): 621–645, 2006.
5. Insfran, E.: A Requirements Engineering Approach for Object-Oriented Conceptual Modeling, PhD Thesis, DSIC, Valencia University of Technology, 2003.
6. Insfran, E., Pastor, O., Wieringa, R.: Requirements Engineering-Based Conceptual Modelling. In: Journal of Requirements Engineering 7(2): 61–72, 2002.
7. ISO, ISO/IEC 9126-1, (2001). Software Engineering – Product quality P1:Quality model.
8. Ivkovic, I., Kontogiannis K.: A Framework for Software Architecture Refactoring using Model Transformations and Semantic Annotations. In: Conf. on Software Maintenance and Reengineering, 2006, pp. 135–144.
9. Kerhervé, B., Nguyen, K.K., Gerbé, O., Jaumard, B.: A Framework for Quality-Driven Delivery in Distributed Multimedia Systems. In: AICT/ICIW 2006, 2006, pp. 195–205.
10. Kirk, R.E.: Experimental design. Procedures for the behavioural sciences. Brooks/Cole Publishing Company, 1995.
11. Kontio, M.: Architectural Manifesto: The MDA adoption manual, (2005), Accessible at http://www-128.ibm.com/ developerworks/wireless/library/wi-arch17.html
12. Kurtev, I.: Adaptability of Model Transformations. PhD Thesis, Univ. of Twente, 2005.
13. Markovic, S., Baar, T.: Refactoring OCL Annotated UML Class Diagrams. In: 8th Int. Conf. on Model Driven Engineering Languages and Systems, 2005, pp. 280–294.
14. Merilinna, J.: A Tool for Quality-Driven Architecture Model Transformation. In: Espoo, VTT Electronics, VTT Publications, 2005.
15. Rottger, S., Zschaler, S.: Model-Driven Development for Non-functional Properties: Refinement Through Model Transformation. In: The Unified Modelling Language (UML) Conference, LNCS Volume 3273, 2004, pp. 275–289.