



**ACM/IEEE 14<sup>th</sup> International  
Conference on Model Driven  
Engineering Languages and  
Systems, Wellington, New Zealand,  
October 16-21, 2011**

**Experiences and Empirical Studies  
in Software Modelling (EESSMod 2011)**

**October 17**

Michel Chaudron, Marcela Genero,  
Silvia Abrahão, Parastoo Mohagheghi,  
Lars Pareto (Eds)



# **EESSMod 2011**

## **First International Workshop on Experiences and Empirical Studies in Software Modelling**

Michel Chaudron<sup>1</sup>, Marcela Genero<sup>2</sup>, Silvia Abrahão<sup>3</sup>, Parastoo Mohagheghi<sup>4</sup>, Lars Pareto<sup>5</sup>

<sup>1</sup>LIACS – Leiden University  
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands  
chaudron@liacs.nl

<sup>2</sup>ALARCOS Research Group, University of Castilla-La Mancha  
Paseo de la Universidad 4, 13071, Ciudad Real, Spain  
Marcela.Genero@uclm.es

<sup>3</sup>ISSI Research Group, Department of Information Systems and Computation – Universitat Politècnica de València  
Camino de Vera, s/n, 46022, Valencia, Spain  
sabrahao@dsic.upv.es

<sup>4</sup>SINTEF and Norwegian University of Science and Technology  
Forskingsveien 1, 0373 Oslo, Norway  
parastoo.mohagheghi@sintef.no

<sup>5</sup>Chalmers – University of Gothenburg  
Gothenburg, Sweden  
pareto@chalmers.se

### **Preface**

Most software development projects apply modelling in some stages of development and to various degrees in order to take advantage of the many and varied benefits of it. Modelling is, for example, applied for facilitating communication by hiding technical details, analysing a system from different perspectives, specifying its structure and behaviour in an understandable way, or even for enabling simulations and generating test cases in a mode-driven engineering approach. Thus, the evaluation of modelling techniques, languages and tools is needed in order to assess their advantages and disadvantages, to ensure their applicability to different contexts, their ease of use, and other issues such as required skills and costs; either isolated or in comparison with other methods.

The need to reflect and advance on empirical methods and techniques that help improving the adoption of software modelling in industry led us to organize the first edition of the International Workshop on Experiences and Empirical Studies in Software Modelling (EESSMod 2011) that was held in conjunction with the ACM/IEEE 14th International Conference on Model Driven Engineering Languages and Systems (MoDELS 2011). The main purpose of the workshop was to bring together professionals

and researchers interested in software modelling to discuss in which way software modelling techniques may be evaluated, share experiences of performing such evaluations and discuss ideas for further research in this area. The workshop accepted both experience reports of applying software modelling in industry and research papers that describe more rigorous empirical studies performed in industry or academia.

These proceedings collect the papers presented at the Workshop. All the submitted papers were peer-reviewed by three independent reviewers. The accepted papers (5 regular papers) discuss theoretical and practical issues related to experimentation in software modelling or the use of modelling techniques in industry.

In particular, the paper by *Fernández-Sáez et al.* presents a controlled experiment for analysing the influence of the level of detail of UML models on the maintenance of the corresponding source code. The paper by *Zugal et al.* proposes a framework for assessing the impact of hierarchy on model understandability and discusses the implications for experiments investigating the impact of modularization on conceptual models. The paper by *Carver et al.* analyses the frequency with which empirical evaluation has been reported in the software modelling community. The results of an analysis of papers published in the MoDELS conference (from 2006-2010) showed that, of 266 papers, 195 of them (73%) performed no empirical evaluation. The paper by *Leotta et al.* presents an experience report on the use of a model-driven method for developing VECM-based systems in the context of two Italian companies. Finally, the paper by *Cadavid et al.* proposes a process for analysing meta-models expressed using MOF and OCL and reports on the pre-processing of 52 meta-models in order to get them ready for automatic empirical analysis.

We would like to thank the authors for submitting their papers to the Workshop. We are also grateful to the members of the Program Committee for their efforts in the reviewing process, and to the MoDELS2011 organizers for their support and assistance during the workshop organization. More details on the Workshop are available at <http://www.eesmod.org>.

Leiden, Ciudad Real, Valencia, Oslo,  
Gothenburg  
28 September 2011

*Michel Chaudron*  
*Marcela Genero*  
*Silvia Abrahão*  
*Parastoo Mohagheghi*  
*Lars Pareto*

## **Program Committee**

Bente Anda, University of Oslo, Norway  
Teresa Baldassarre, Universita' Degli Studi di Bari, Italy  
Narasimha Bolloju, University of Hong Kong, China  
Lionel Briand, Simula Research Laboratory, Norway  
Danilo Caivano, Universita' Degli Studi di Bari, Italy  
Karl Cox, University of Brighton, UK  
Jose Antonio Cruz-Lemus, University of Castilla-La Mancha, Spain  
H. Eichelberger, Universität Hildesheim, Germany  
Felix Garcia, University of Castilla-La Mancha, Spain  
Carmine Gravino, University of Salerno, Italy  
Torchiano Marco, Politecnico di Torino, Italy  
Jan Mendling, Humboldt-University Berlin, Germany  
James Nelson, Southern Illinois University, USA  
Ariadi Nugroho, LIACS, Leiden University, The Netherlands  
Jeffrey Parson, Memorial University of Newfoundland, Canada  
Keith Phalp, Bournemouth University, UK  
Geert Poels, University of Ghent, Belgium  
Jan Recker, Queensland University of Technology, Australia  
Giuseppe Scaniello, Universita' Degli Studi della Basilicata, Italy  
Samira Si-Said Cherfi, CEDRIC-CENAM  
Keng Siau, University of Nebraska-Lincoln, USA  
Dag Sjøberg, University of Oslo, Norway  
Sara Sprenkle, Washington & Lee University, USA  
Miroslaw Staron, University of Gothenburg, Sweden



## Content

Preface	i
Program committee	iii
What do 449 MDE Practitioners Think About MDE? (Keynote Speech) ..... Jon Whittle	1
Does the Level of Detail of UML Models Affect the Maintainability of Source Code? ..... A. M. Fernández-Sález, M. Genero and M. R.V. Chaudron	3
Assessing the Impact of Hierarchy on Model Understandability – A Cognitive Perspective..... S. Zugal, J. Pinggera, B. Weber, J. Mendling and H. A. Reijers	18
Assessing the Frequency of Empirical Evaluation in Software Modeling Research..... Jeffrey C. Carver, Eugene Syriani and Jeff Gray	28
Building VECM-based Systems with a Model Driven Approach: an Experience Report..... M. Leotta, G. Reggio, F. Ricca and E. Astesiano	38
Empirical evaluation of the conjunct use of MOF and OCL ..... J. Cadavid, B. Baudry and B. Combemale	48





## **What do 449 MDE Practitioners Think About MDE? (Keynote Speech)**

Jon Whittle

Computing Department  
Lancaster University, UK  
whittle@comp.lancs.ac.uk

This talk will present the results of an in-depth survey of model-driven engineering (MDE) industrial practice. The survey, disseminated electronically, consisted of 35 questions on MDE use and received 449 responses. The study focused on six key criteria related to productivity and maintainability for evaluating MDE success. Each of these can be impacted positively or negatively depending on how MDE is applied. The study aimed to understand whether, in current practice, the positive impacts outweigh the negative ones. Findings indicate that productivity gains from code generation tend to outweigh losses from integration with existing code. Successful MDE practitioners follow best practice guidelines by making changes at the model level. MDE allows for faster turn-arounds on new requirements, but there is a risk that it may prevent organizations from responding to new business opportunities. Findings also indicate that MDE increases overall training costs. Finally, UML is not yet universally accepted as the modeling language of choice and, in fact, domain-specific modeling languages are much more prevalent than anticipated. This is joint work with John Hutchinson and Mark Rouncefield.

### **Biography**

Jon Whittle is a full Professor, Chair of Software Engineering and Royal Society Wolfson Merit Scholar at Lancaster University. He has parallel research interests in model-driven engineering and social computing. In particular, his recent interests are in how new social media and social networking can influence or contribute to MDE. Jon has been intimately involved with the MDE community for over ten years, having served as Chair of the Steering Committee of MODELS from 2006-2008 and serving as PC Chair in 2011. He also sits on the editorial board of the Journal of Software and System Modeling. Jon is currently principal or co-investigator on a number of interdisciplinary research projects, with a total net worth of around £5M.



## Does the Level of Detail of UML Models Affect the Maintainability of Source Code?

Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

<sup>1</sup>Alarcos Quality Center, S.L., Department of Technologies and Information Systems,  
University of Castilla-La Mancha  
Paseo de la Universidad 4, 13071, Ciudad Real, Spain  
+34 926295300 ext.6648  
[ana.fernandez@alarcosqualitycenter.com](mailto:ana.fernandez@alarcosqualitycenter.com)

<sup>2</sup>ALARCOS Research Group, Department of Technologies and Information Systems,  
University of Castilla-La Mancha  
Paseo de la Universidad 4, 13071, Ciudad Real, Spain  
+34 926295300 Ext. 3740  
[Marcela.Genero@uclm.es](mailto:Marcela.Genero@uclm.es)

<sup>3</sup>LIACS - Leiden University  
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands  
+31 715277065 (secr 7061)  
[chaudron@liacs.nl](mailto:chaudron@liacs.nl)

**Abstract.** This paper presents an experiment carried out as a pilot study to obtain a first insight into the influence of the quality of UML models on the maintenance of the corresponding source code. The quality of the UML models is assessed by studying the amount of information they contain as measured through a level of detail metric. The experiment was carried out with 11 Computer Science students from the University of Leiden. The results obtained indicate a slight tendency towards obtaining better results when using low level of detail UML models, which contradicts our expectations based on previous research found in literature. Nevertheless, we are conscious that the results should be considered as preliminary results given the low number of subjects that participated in the experiment. Further replications of this experiment are planned with students and professionals in order to obtain more conclusive results.

**Keywords:** UML, maintenance, empirical studies, controlled experiment

### 1 Introduction

The current increasing complexity of software projects [1] has led to the emergence of UML [2] as a tool with which to increase the understanding between customer and developer and to improve communication among team members [3]. Despite this, not all UML diagrams have the same complexity, layout, level of abstraction, etc. Previous studies have shown that the style and rigor used in the diagrams may vary

2 Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

considerably throughout software projects [4], in addition to affecting the source code of the system in a different way.

On the one hand, the different purposes for which a model may be intended (for example: architecting solutions, communicating design decisions, detailed specification for implementation, or automatically generating implementation code) signifies that the same system can be represented with different styles. On the other hand, the development diagrams are sometimes available for maintainers, but this is not always the case, and the diagrams must be generated with a reverse engineering process. The difference in the origin of the models and the different techniques that can be used to generate a reverse engineering model result in different styles of models. Some of the most notable differences between these models may be the level of detail shown. In this work we therefore analyze whether the different levels of detail (LoD) affect the work that must be carried out by a maintainer.

This document is organized as follows. Section 2 presents the related work. Section 3 presents the description of the experiment. The results obtained in the experiment are presented in Section 4, whilst the threats to validity are summarized in Section 5. Finally, Section 6 outlines our main conclusions and future work.

## 2 Related work

We performed an SLR [5] to discover all the empirical studies performed as regards the use of UML in maintenance, and found only the following two works related to the maintenance of source code:

- In [6] an experiment was performed to investigate whether the use of UML influences maintenance in comparison to the use of only source code. The results of this work show a positive influence of the presence of UML for maintainers.
- In the work presented in [7], the experiment performed is focused on the comprehension and the difficulties involved in maintaining object-oriented systems. UML models were also presented to the subjects of the experiment, but they were only focused on exploring the participant's strategies and problems while they were conducting maintenance tasks on an object-oriented application.

We therefore decided to perform an experiment related to the influence of different levels of detail on UML diagrams when assisting in maintenance tasks. We found a paper [8] focused on the understandability of models with different LoD in the development phase. The results show a better understanding of models when they have a high LoD. We would like to discover whether high LoD diagrams help workers to perform the changes that need to be made to the source code during the maintenance phase.

## 3 Experiment description

The experiment was carried out at the University of Leiden (The Netherlands) in March 2011. In order to run and report this experiment, we followed the

recommendations provided in several works [9-11]. The experiment was presented by following the guidelines for reporting empirical research in software engineering [11] as closely as possible. The experimental material is available for downloading at:

<http://alarcos.esi.uclm.es/experimentUMLmaintenance/>

In the following subsections we shall describe the main characteristics of the experiment, including goal, context, variables, subjects, design, hypotheses, material, tasks, experiment procedure and analysis procedure.

### 3.1 Goal

The principal goal of this experiment was to investigate whether the LoD in UML models influences the maintenance of source code. The GQM template for goal definition [12, 13] was used to define the goal of our experiment as follows: “*Analyze the level of detail in UML models with the purpose of evaluating it with respect to the maintainability of source code from the point of view of researchers, in the context of Computer Science students at the University of Leiden.*”

As in [3], we considered that the LoD in UML models should be defined as the amount of information that is used to represent a modeling element. LoD is a 'continuous' metric, but for the experiment we have taken two “extremes” - high and low LoD.

We decided to use 3 different types of diagrams (use case, sequence and class diagrams) since they are those most frequently used. When the LoD used in a UML model is low, it typically employs only a few syntactical features, such as class-name and associations, without specifying any further facts about the class. When it is high, the model also includes class attributes and operations, association names, association directionality, and multiplicity. In sequence diagrams, in which there is a low LoD, the messages among objects have an informal label, and when the LoD is high the label is a method name plus the parameter list. We consider that it is not possible to distinguish between low and high LoD in use case diagrams because they are very simple diagrams. The elements that fit each level of detail are shown in Table 1.

**Table 1.** Levels of detail in UML models

Diagram	Element	Low LoD	High LoD
Class diagram	Classes (box and name)	✓	✓
	Attributes	✗	✓
	Types in attributes	✗	✓
	Operations	✗	✓
	Parameters in operations	✗	✓
	Associations	✓	✓
	Association directionalities	✗	✓
	Association multiplicities	✗	✓
	Aggregations	✓	✓
	Compositions	✓	✓
Sequence diagram	Actors	✓	✓
	Objects	✓	✓

4 Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

	Messages in informal language	✓	✗
	Messages with formal language (name of a method)	✗	✓
	Parameters in messages	✗	✓
	Labels in return messages	✗	✓

### 3.2 Context Selection

The experimental objects consisted of use case, class and sequence diagrams and the JAVA code of two software systems, which are summarized below:

- A-H: high LoD diagrams and JAVA code of system A.
- A-L: low LoD diagrams and JAVA code of system A.
- B-H: high LoD diagrams and JAVA code of system B.
- B-L: low LoD diagrams and JAVA code of system B.

Diagrams A-x described a library domain from which a user can borrow books. Diagrams B-x described a sport centre domain from which users can rent services (tennis courts, etc.). System A is a Library extracted from [14]. We decided to use it because it was a representative system, it was complete (source code and models were available) and it gave us a starting point from which to compare our results (it was only possible to compare the results obtained from the subjects who received System A with high LoD with [7]). System B is a Sport centre application created as part of the Master's degree Thesis of a student from the University of Castilla-La Mancha, and we therefore consider it to be a real system. Both systems are desktop applications and have more or less the same complexity. These experimental objects were presented in English.

The subjects students on a Software Engineering course from which they had acquired training in UML diagrams. Their knowledge was sufficient for them to understand the given systems, and they had roughly the same background. They had knowledge about the use of UML diagrams in general, but they were taught about UML diagrams and JAVA in a training session organized to take place the day before the experiment was carried out.

The experiment was carried out by 11 Computer Science students from the University of Leiden (The Netherlands) who were taking the Software Engineering course in the second-year of their B.Sc.

Working with students also implies various advantages, such as the fact that their prior knowledge is fairly homogeneous, there is the possible availability of a large number of subjects [15], and there is the chance to test experimental design and initial hypotheses [16]. An additional advantage of using novices as subjects in experiments on comprehensibility and modifiability is that the cognitive complexity of the objects under study is not hidden by the subjects' experience. Nonetheless, we also wish to test the findings with practitioners in order to strengthen the external validity of the results obtained.

The students who participated in the experiment were volunteers selected for convenience (the students available in the corresponding course). Social threats

caused by evaluation apprehension were avoided by not grading the students on their performance.

### 3.3 Variables selection

The independent variable (also called “main factor”) is the LoD, which is a nominal variable with two values (low LoD and high LoD). We combined each level of the independent variable with the two different systems used to obtain four treatments (see Table 2).

The dependent variables are modifiability and understandability. These two variables were considered because understandability and modifiability directly influence maintainability [17]. In order to measure these dependant variables, we defined the following measures:

- **Understandability Effectiveness ( $U_{Effec}$ ):** This measure reflects the ability to correctly understand the system presented. It is calculated with the following formula: number of correct answers / number of questions. A higher value of this measure reflects a better understandability.
- **Modifiability Effectiveness ( $U_{Effic}$ ):** This measure reflects the ability to correctly modify the system presented. It is calculated with the following formula: number of correctly performed modification tasks / number of modification tasks. A higher value of this measure reflects a better modifiability.
- **Understandability Efficiency ( $M_{Effec}$ ):** This measure also reflects the ability to correctly understand the system presented. It is calculated with the following formula: time spent / number of correctly answered questions. A lower value of this measure reflects a better understandability.
- **Modifiability Efficiency ( $M_{Effic}$ ):** This measure also reflects the ability to correctly modify the system presented. It is calculated with the following formula: time spent / number of correctly performed tasks. A lower value of this measure reflects a better modifiability.

Additional independent variables (called “co-factors”) were considered according to the experimental design of the replication, and their effect has been controlled and analyzed:

- **Order.** The selected design (see Table 2), i.e., the variation in the order of application of each method (low LoD, high LoD), was intended to alleviate learning effects. Nonetheless, we analyzed whether the order in which the LoD were used by the subjects biased the results.
- **System.** This factor indicates the systems (i.e., A and B) used as experimental objects. The design selected for the experiment (see Table 2) forced us to choose two application domains in order to avoid learning effects. Our intention was that the system factor would not be a confounding factor that might also influence the subjects’ performances. We therefore selected well-known domains and experimental objects of a similar complexity.

6 Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

### 3.4 Hypotheses formulation

Based on the assumption that the more information a model contains, the more is known about the concepts/knowledge described in the model, the hypothesis are:

1.  $H_{1,0}$ : There is no significant difference in the subjects' understandability effectiveness when working with UML diagrams modeled using high or low levels of detail.

$$H_{1,1}: \neg H_{1,0}$$

2.  $H_{2,0}$ : There is no significant difference in the subjects' understandability efficiency when working with UML diagrams modeled using high or low levels of detail.

$$H_{2,1}: \neg H_{2,0}$$

3.  $H_{3,0}$ : There is no significant difference in the subjects' modifiability effectiveness when working with UML diagrams modeled using high or low levels of detail.

$$H_{3,1}: \neg H_{3,0}$$

4.  $H_{4,0}$ : There is no significant difference in the subjects' modifiability efficiency when working with UML diagrams modeled using high or low levels of detail.

$$H_{4,1}: \neg H_{4,0}$$

The goal of the statistical analysis will be to reject these null hypotheses and possibly to accept the alternative ones (e.g.,  $H_{n1} = \neg H_{n0}$ ).

### 3.5 Experimental design

We selected a balanced factorial design in which the group-interaction acted as a confounding factor [18] which permits the lessening of the effects of learning and fatigue. The experiment's execution consisted of two runs. In each round, each of the groups was given a different treatment. The corresponding system (source code + UML models) was assigned to each group at random, but was given out in a different order in each case. Table 2 presents the outline of the experimental design.

**Table 2.** Experimental design

RUN 1		LoD		RUN 2		LoD	
		Low	High			Low	High
System	A	Group 1	Group 2	System	A	Group 3	Group 4
	B	Group 3	Group 4		B	Group 2	Group 1

Before carrying out the experiment, we provided the subjects with a background questionnaire and assigned them to the 4 groups randomly, based on the marks obtained in the aforementioned questionnaire (blocked design by experience) in an attempt to alleviate experience effects. To avoid a possible learning effect, the diagrams came from different application domains (A-a Library and B-a Sport centre).

When designing the experiment we attempted to alleviate several issues that might threaten the validity of the research done by considering the suggestions provided in [19].



### 3.6 Experimental tasks

The tasks to be performed did not require high levels of industrial experience, so we believed that the use of students could be considered appropriate, as suggested in literature [20, 21]. The material used was written in English.

There were three kinds of tasks:

- **Understandability task:** This contained 3 questions concerning the semantics of the system, i.e. the semantics of diagrams and the semantics of code. These questions were multiple choice questions and were used to obtain  $U_{Effec}$  and  $U_{Effic}$ .
- **Modifiability task:** The subjects received a list of requirements in order to modify the code of the system in order to add/change certain functionalities. This part of the experiment contained 3 modifiability tasks and allowed us to calculate  $M_{Effec}$  and  $M_{Effic}$ . The subjects were provided with answer sheets to allow them to structure their responses related to maintenance tasks. They had to fill in a different form depending on the element that they wished to maintain. The answer sheets can be found at:  
<http://alarcos.esi.uclm.es/experimentUMLmaintenance/>
- **Post-questionnaire task:** At the end of the execution of each run, the subjects were asked to fill in a post-experiment questionnaire, whose goal was to obtain feedback about the subjects' perception of the experiment execution, which could be used to explain the results obtained. The answers to the questions were based on a five-point Likert scale [22].

### 3.7 Experimental procedure

The experiment took place in two sessions of two hours each. The subjects first attended a training session in which detailed instructions on the experiment were presented and the main concepts of UML and JAVA were revised. In this session, the subjects carried out an exercise similar to those in the experimental tasks in collaboration with the instructor. During the training session, the subjects were required to fill in a background questionnaire. Based on the marks obtained in this questionnaire, the subjects were randomly assigned to the 4 groups shown in Table 2, thus obtaining balanced groups in accordance with the marks obtained in the background questionnaire.

The experiment then took place in a second session, consisting of two runs. In each run, each of the groups was given a different treatment, as is shown in Table 2.

The experiment was conducted in a classroom, where the students were supervised by the instructor and no communication among them was allowed.

After the experiment execution, the data collected from the experiment were placed on an excel sheet.

### 3.8 Analysis procedure

The data analysis was carried out by considering the following steps:

8 Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

1. We first carried out a descriptive study of the measures of the dependent variables, i.e., understandability and modifiability.
2. We then tested the formulated hypotheses using the non-parametric Kruskal-Wallis test [23] for the data collected in the experiment. The use of this test was possible because, according the design of the controlled experiment, we obtained paired samples. In addition, Kruskal-Wallis is the most appropriate test with which to explore the results of a factorial design with confounded interaction [18, 24], i.e., the design used in our experiment, when there is non-normal distribution of the data.
3. We next used the Kruskal-Wallis test to analyze the influence of the co-factors (i.e., System and Order).
4. The data collected from the post-experiment questionnaire was finally analyzed using bar graphs.

## 4 Results

The following subsections show the results of the data analysis of the experiment performed using SPSS [25].

### 4.1 Descriptive statistics and exploratory analysis

Table 3 and Table 4 show the descriptive statistics of the Understandability and Modifiability measures, respectively (i.e., mean ( $\bar{X}$ ), standard error (SE), and standard deviation (SD)), grouped by LoD.

**Table 3.** Descriptive statistics for  $U_{Effec}$  and  $U_{Effic}$ .

LoD	Subjects	$U_{effec}$			$U_{Effic}$		
		$\bar{X}$	SE	SD	$\bar{X}$	SE	SD
Low	N = 10 (1 outlier)	<b>0.767</b>	0.051	0.161	<b>334.500</b>	36.308	114.816
High	N = 11	0.758	0.650	0.215	363.924	82.602	273.960

**Table 4.** Descriptive statistics for  $M_{effec}$  and  $M_{Effic}$ .

LoD	Subjects	$M_{effec}$			$M_{Effic}$		
		$\bar{X}$	SE	SD	$\bar{X}$	SE	SD
Low	N = 11	<b>0.437</b>	0.066	<b>0.221</b>	<b>240.121</b>	41.008	136.007
High	N = 11	0.402	0.050	0.169	294.637	47.198	156.539

At a glance, we can observe that when the subjects used low LoD diagrams they obtained better values in all variables. This indicates that low LoD diagrams may, to some extent, improve the comprehension and modification of the source code.

## 4.2 Influence of LoD

In order to test the formulated hypotheses we analyzed the effect of the main factor (i.e. LoD) on the dependent variables considered (i.e.,  $U_{\text{Effec}}$ ,  $U_{\text{Effic}}$ ,  $M_{\text{Effec}}$  and  $M_{\text{Effic}}$ ) using the Kruskal-Wallis test (see Table 5).

**Table 5.** Kruskal-Wallis test results for  $U_{\text{Effec}}$ ,  $U_{\text{Effic}}$ ,  $M_{\text{Effec}}$  and  $M_{\text{Effic}}$

	$U_{\text{Effec}}$	$U_{\text{Effic}}$	$M_{\text{Effec}}$	$M_{\text{Effic}}$
LoD	1	0.439	0.792	0.491

### Testing $H_{1,0}$ ( $U_{\text{Effec}}$ )

The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in  $U_{\text{Effec}}$  in either group.

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is higher than a high LoD value):

**Table 6.** Comparison of subjects' results for each measure

	low LoD = high LoD	low LoD < high LoD	low LoD > high LoD
$U_{\text{Effec}}$	6	3	2
$U_{\text{Effic}}$	0	7	4
$M_{\text{Effec}}$	0	7	4
$M_{\text{Effic}}$	0	5	6

As Table 6 shows, the number of subjects who obtained the same results for both treatments (high and low LoD) is relatively high. There were more subjects who performed better with a high LoD than with a low LoD, but the differences in comparison to the opposite group is very small (only one subject).

### Testing $H_{2,0}$ ( $U_{\text{Effic}}$ )

The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in  $U_{\text{Effic}}$  in either group.

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is smaller than a high LoD value):

As Table 6 shows, no subjects obtained the same  $U_{\text{Effic}}$  for both treatments (high and low LoD). More subjects performed better with a low LoD than with a high LoD.

### Testing $H_{3,0}$ ( $M_{\text{Effec}}$ )

The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in  $M_{\text{Effec}}$  in either group.

10 Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is higher than a high LoD value):

As Table 6 shows, no subjects obtained the same  $M_{\text{Effic}}$  for both treatments (high and low LoD). More subjects performed better with a high LoD than with a low LoD.

#### Testing $H_{4,0}(M_{\text{Effic}})$

The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in  $M_{\text{Effic}}$  in either group.

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is smaller than a high LoD value):

As Table 6 shows, no subjects obtained the same  $M_{\text{Effic}}$  for both treatments (high and low LoD). More subjects performed better with a high LoD than with a low LoD, but the differences in comparison to the opposite group are also small.

### 4.3 Influence of system

In order to test the effect of the co-factor System, we performed a Kruskal-Wallis test whose results are shown in Table 7. As all the p-values were higher than 0.05, except in one case ( $U_{\text{Effic}}$ ), we did not have sufficient evidence to reject the hypothesis, i.e. it seems that the system did not influence the subjects' performance (and this was therefore a controlled co-factor).

**Table 7.** Kruskal-Wallis test results for the influence of the System.

	$U_{\text{effec}}$	$U_{\text{Effic}}$	$M_{\text{effec}}$	$M_{\text{Effic}}$
System	0.804	<b>0.035</b>	0.575	0.061

### 4.4 Influence of order

In order to test the effect of Order, we performed a Kruskal-Wallis test (see Table 8). As all p-values were higher than 0.05, we did not have sufficient evidence to reject the hypothesis, i.e. the order did not influence the subjects' performance (and this was therefore a controlled co-factor).

**Table 8.** Kruskal-Wallis tests results.

	$U_{\text{effec}}$	$U_{\text{Effic}}$	$M_{\text{effec}}$	$M_{\text{Effic}}$
Order	1	0.105	0.223	0.341

### 4.5 Post- experiment survey questionnaire results

The analysis of the answers to the post-experiment survey questionnaire revealed that the time needed to carry out the comprehension and modification tasks was

considered to be inappropriate (more time was needed), and that the subjects considered the tasks to be quite difficult (Fig. 1).

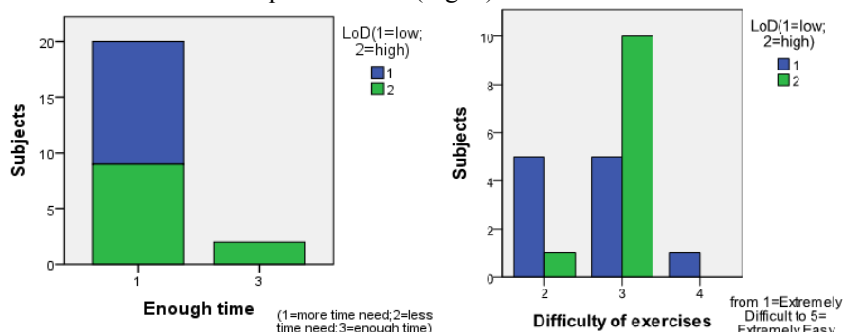


Fig. 1. Subjects' perception of the experiment.

We also asked about the subjects' perception of some of the items that appeared in the high LoD diagrams but did not appear in the low LoD diagrams. Fig. 2 shows that high LoD elements seem to be appreciated by the subjects. With regard to the histograms in Fig. 2, if a subject responds 1 or 2, this indicates that s/he thinks that the element in the question was helpful, while a response of 4 or 5 indicates that the elements in the question are not helpful (3 is a neutral response). If we focus on the elements related to class diagrams (upper histograms) we can see that attributes are helpful for 9 subjects (versus 1 subject who does not believe them to be helpful). The same is true of operations (10 subjects vs. 1 subject). If we focus on the elements related to sequence diagrams (lower histograms) we can see that formal messages are more helpful (16 subjects) than natural language messages (0 subjects), and the same can also be said of the appearance of parameters in messages (13 subjects vs. 2 subjects).

#### 4.6 Summary and discussion of the data analysis

The null hypothesis cannot be rejected for any of the dependent variables. Although we cannot draw conclusive results on the main factor (LoD), we have found that co-factors (system, order) have not influenced the results.

Nevertheless, the descriptive statistics in general showed a slight tendency in favor of using low LoD diagrams in contrary to what we believed, as the diagrams with a high LoD helped developers in the software development stage [8]. This may result from the fact that the subjects did not have the expected amount of knowledge about UML (a mean of 8.8 correct answers out of 16 questions) and JAVA (a mean of 4.9 correct answers out of 9 questions) tested in the background questionnaire. The results of the experiment must be considered as preliminary results owing to the small size of the group of subjects who participated in the experiment.

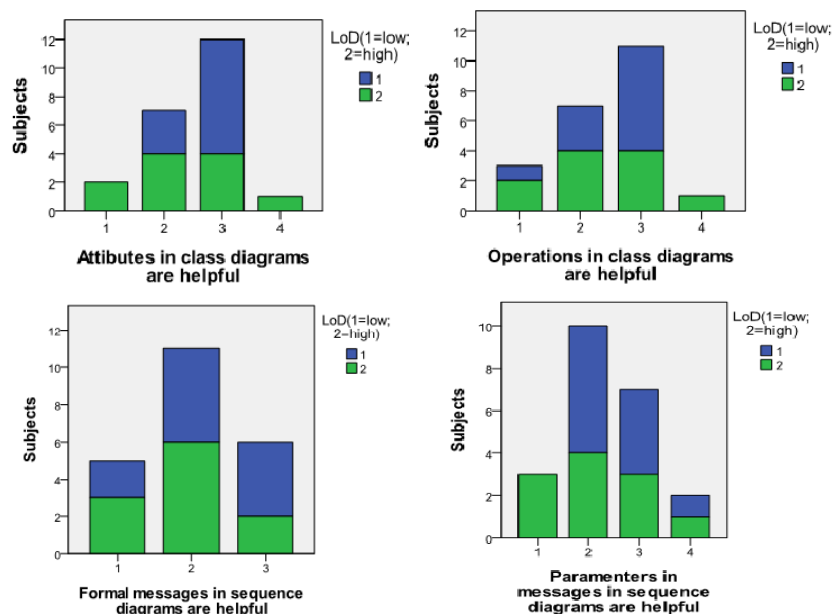
12 Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

Fig. 2. Subjects' opinion of LoD (1=Complete Agreement 2=Partial Agreement 3=Neither agree/ nor disagree 4=Partial Disagreement 5=Total disagreement)

## 5 Threats to Validity

We must consider certain issues which may have threatened the validity of the experiment:

- **External validity:** External validity may be threatened when experiments are performed with students, and the representativeness of the subjects in comparison to software professionals may be doubtful. In spite of this, the tasks to be performed did not require high levels of industrial experience, so we believed that this experiment could be considered appropriate, as suggested in literature [13]. There are no threats related to the material used since the systems used were real ones.
- **Internal validity:** Internal validity threats are mitigated by the design of the experiment. Each group of subjects worked on the same system in different orders. Nevertheless, there is still the risk that the subjects might have learned how to improve their performances from one performance to the other. Moreover, the instrumentation was tested in a pilot study in order to check its validity. In addition, mortality threats were mitigated by offering the subjects extra points in their final marks.
- **Conclusion validity:** Conclusion validity concerns the data collection, the reliability of the measurement, and the validity of the statistical tests. Statistical tests were used to reject the null hypotheses. We have explicitly mentioned and discussed when non-significant differences were present. What is more,

conclusion validity might also be affected by the number of observations. Further replications on larger datasets are thus required to confirm or contradict the results.

- **Construct validity:** This may be influenced by the measures used to obtain a quantitative evaluation of the subjects' performance, the comprehension questionnaires, the maintenance tasks, and the post-experiment questionnaire. The metrics used were selected to achieve a balance between the correctness and completeness of the answers. The questionnaires were defined to obtain sufficiently complex questions without them being too obvious. The post-experiment questionnaire was designed using standard forms and scales. Social threats (e.g., evaluation apprehension) have been avoided, since the students were not graded on the results obtained.

## 6 Conclusions and future work

The main concern of the research presented in this paper is the use of a controlled experiment to investigate whether the use of low or high level of detail in UML diagrams influences the maintainer's performance when understanding and modifying source code. The experiment was carried out by 11 academic students from the University of Leiden in the Netherlands.

The results obtained are not significant owing to various factors such as the fact that the subjects selected had a low level of experience in using UML and JAVA code, and the small size of the group of subjects who participated in the experiment. It is only possible to observe a slight tendency towards obtaining better results with low LoD diagrams, contrary to the results obtained in [8].

Despite these drawbacks, we have ensured that the experimental results were not influenced by other co-factors such as the system used or the order in which the subjects received the experimental material.

We are planning to perform two replications with students from the University of Castilla-La Mancha (Spain) and students from the University of Bari (Italy). A third possible replication with professionals is also being planned. All the drawbacks found in the execution of this experiment will be taken into account in the replications.

**Acknowledges.** This research has been funded by the following projects: MEDUSAS (CDTI-MICINN and FEDER IDI- 20090557), ORIGIN (CDTI-MICINN and FEDER IDI-2010043(1-5), PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01), EECOO (MICINN TRA2009-0074), MECCA (JCMM PII2I09-0075-8394) and IMPACTUM (PEII 11-0330-4414).

### References

1. Van Vliet, H., *Software Engineering: Principles and Practices* 3rd ed. 2008: Wiley.
2. OMG. *The Unified Modeling Language. Documents associated with UML Version 2.3* 2010; Available from: <http://www.omg.org/spec/UML/2.3>.
3. Nugroho, A. and M.R.V. Chaudron. *Evaluating the impact of UML modeling on software quality: An industrial case study*. in *Proceeding of 12th International*

14 Ana M. Fernández-Sáez<sup>1</sup>, Marcela Genero<sup>2</sup>, and Michel R.V. Chaudron<sup>3</sup>

- Conference on Model Driven Engineering Languages and Systems (MODELS'09)*. 2009.
4. Lange, C.F.J. and M.R.V. Chaudron, *In practice: UML software architecture and design description*. IEEE Software, 2006. **23**(2): p. 40-46.
  5. Fernández-Sáez, A.M., M. Genero, and M.R.V. Chaudron, *Empirical studies on the influence of UML in software maintenance tasks: A systematic literature review*. Submitted to Science of Computer Programming - Special issue on Software Evolution, Adaptability and Maintenance, Elsevier.
  6. Dzidek, W.J., E. Arisholm, and L.C. Briand, *A realistic empirical evaluation of the costs and benefits of UML in software maintenance*. IEEE Transactions on Software Engineering, 2008. **34**(3): p. 407-432.
  7. Karahasanovic, A. and R. Thomas. *Difficulties Experienced by Students in Maintaining Object-Oriented Systems: an Empirical Study*. in *Proceedings of the Australasian Computing Education Conference (ACE'2007)* 2007.
  8. Nugroho, A., *Level of detail in UML models and its impact on model comprehension: A controlled experiment*. Information and Software Technology, 2009. **51**(12): p. 1670-1685.
  9. Juristo, N. and A. Moreno, *Basics of Software Engineering Experimentation*. 2001: Kluwer Academic Publishers.
  10. Wohlin, C., et al., *Experimentation in Software Engineering: an Introduction*. 2000: Kluwer Academic Publisher.
  11. Jedlitschka, A., M. Ciolkowski, and D. Pfahl, *Reporting Experiments in Software Engineering*, in *Guide to Advanced Empirical Software Engineering* F. Shull, J. Singer, and D.I.K. Sjøberg, Editors. 2008, Springer Verlag.
  12. Basili, V. and D. Weiss, *A Methodology for Collecting Valid Software Engineering Data*. IEEE Transactions on Software Engineering, 1984. **10**(6): p. 728-738.
  13. Basili, V., F. Shull, and F. Lanubile, *Building Knowledge through Families of Experiments*. IEEE Transactions on Software Engineering, 1999. **25**: p. 456-473.
  14. Eriksson, H.E., et al., *UML 2 Toolkit*. 2004: Wiley.
  15. Verelst, J. *The Influence of Abstraction on the Evolvability of Conceptual Models of Information Systems*. in *International Symposium on Empirical Software Engineering (ISESE'04)*. 2004.
  16. Sjøberg, D.I.K., et al., *A Survey of Controlled Experiments in Software Engineering*. IEEE Transaction on Software Engineering, 2005. **31**(9): p. 733-753.
  17. ISO/IEC, *ISO/IEC 25000: Software Engineering, in Software product quality requirements and evaluation (SQuaRe)*. 2008, International Organization for Standardization.
  18. Kirk, R.E., *Experimental Design. Procedures for the Behavioural Sciences*. 1995: Brooks/Cole Publishing Company.
  19. Wohlin, C., et al., *Experimentation in Software Engineering: An Introduction*. 2000, Norwell, MA, USA: Kluwer Academic Publishers.
  20. Basili, V., F. Shull, and F. Lanubile, *Building Knowledge through Families of Experiments*. IEEE Transactions on Software Engineering, 1999. **25**(4): p. 456-473.
  21. Höst, M., B. Regnell, and C. Wholin. *Using students as subjects - a comparative study of students and professionals in lead-time impact assessment*. in *4th*



Does the Level of Detail of UML Models Affect the Maintainability of Source Code? 15

*Conference on Empirical Assessment and Evaluation in Software Engineering.*  
2000.

22. Oppenheim, A.N., *Questionnaire Design, Interviewing and Attitude Measurement.*  
1992: Pinter Publishers.

23. Conover, W.J., *Practical Nonparametric Statistics.* 3rd ed. 1998: Wiley.

24. Winer, B.J., D.R. Brown, and K.M. Michels, *Statistical Principles in  
Experimental Design.* 3rd ed. 1991: Mc Graw Hill Series in Psychology.

25. SPSS, *SPSS 12.0, Syntax Reference Guide.* 2003, Chicago, USA: SPSS Inc.

# Assessing the Impact of Hierarchy on Model Understandability—A Cognitive Perspective

Stefan Zugal<sup>1</sup>, Jakob Pinggera<sup>1</sup>, Barbara Weber<sup>1</sup>, Jan Mendling<sup>2</sup>, and Hajo A. Reijers<sup>3</sup>

<sup>1</sup> University of Innsbruck, Austria

{stefan.zugal|jakob.pinggera|barbara.weber}@uibk.ac.at

<sup>2</sup> Humboldt-Universität zu Berlin, Germany

jan.mendling@wiwi.hu-berlin.de

<sup>3</sup> Eindhoven University of Technology, The Netherlands

h.a.reijers@tue.nl

**Abstract.** Modularity is a widely advocated strategy for handling complexity in conceptual models. Nevertheless, a systematic literature review revealed that it is not yet entirely clear under which circumstances modularity is most beneficial. Quite the contrary, empirical findings are contradictory, some authors even show that modularity can lead to decreased model understandability. In this work, we draw on insights from cognitive psychology to develop a framework for assessing the impact of hierarchy on model understandability. In particular, we identify abstraction and the split-attention effect as two opposing forces that presumably mediate the influence of modularity. Based on our framework, we describe an approach to estimate the impact of modularization on understandability and discuss implications for experiments investigating the impact of modularization on conceptual models.

## 1 Introduction

The use of modularization to hierarchically structure information has for decades been identified as a viable approach to deal with complexity [1]. Not surprisingly, many conceptual modeling languages provide support for hierarchical structures, such as sub-processes in business process modeling languages like BPMN and YAWL [2] or composite states in UML statecharts. While hierarchical structures have been recognized as an important factor influencing model understandability [3, 4], there are no definitive guidelines on their use yet. For instance, for business process models, recommendations for the size of a sub-process, i.e., sub-model, range from 5–7 model elements [5] over 5–15 model elements [6] to up to 50 model elements [7]. Also in empirical research into conceptual models (e.g., ER diagrams or UML statecharts) the question of *whether and when* hierarchical structures are beneficial for model understandability seems not to be entirely clear. While it is common belief that hierarchy has a positive influence on the understandability of a model, reported data seems often inconclusive or even contradictory, cf. [8, 9].

As suggested by existing empirical evidence, hierarchy is not beneficial by default [10] and can even lead to performance decrease [8]. The goal of this paper is to have a detailed look at which factors cause such discrepancies between the common belief in positive effects of hierarchy and reported data. In particular, we draw on concepts from cognitive psychology to develop a framework that describes how the impact of hierarchy on model understandability can be assessed. The contribution of this theoretical discussion is a perspective to disentangle the diverse findings from prior experiments.

The remainder of this paper is structured as follows. In Sect. 2 a systematic literature review about empirical investigations into hierarchical structuring is described. Afterwards, concepts from cognitive psychology are introduced and put in the context of conceptual models. Then, in Sect. 3 the introduced concepts are used as basis for our framework for assessing the impact of hierarchy on understandability, before Sect. 4 concludes with a summary and an outlook.

## 2 The Impact of Hierarchy on Model Understandability

In this section we revisit results from prior experiments on the influence of hierarchy on model understandability, and analyze them from a cognitive perspective. Sect. 2.1 summarizes literature reporting experimental results. Sect. 2.2 describes cognitive foundations of working with hierarchical models.

### 2.1 Existing Empirical Research into Hierarchical Models

The concept of hierarchical structuring is not only applied to various domains, but also known under several synonyms. In particular, we identified synonyms *hierarchy*, *hierarchical*, *modularity*, *decomposition*, *refinement*, *sub-model*, *sub-process*, *fragment* and *module*. Similarly, model understandability is referred to as *understandability* or *comprehensibility*. To systematically identify existing *empirical investigations* into the *impact of hierarchy on understandability* within the domain of conceptual modeling, we conducted a systematic literature review [11]. More specifically, we derived the following key-word pattern for our search: (synonym modularity) X (synonym understandability) X experiment X model. Subsequently, we utilized the cross-product of all key-words for a full-text search in the online portals of Springer<sup>1</sup>, Elsevier<sup>2</sup>, ACM<sup>3</sup> and IEEE<sup>4</sup> to cover the most important publishers in computer science, leading to 9,778 hits. We did not use any restriction with respect to publication date, still we are aware that online portals might provide only publications of a certain time period. In the next step, we removed all publications that were not related, i.e., did not consider the impact of hierarchy on model understandability or did not report

<sup>1</sup> <http://www.springerlink.com>

<sup>2</sup> <http://www.sciencedirect.com>

<sup>3</sup> <http://portal.acm.org>

<sup>4</sup> <http://ieeexplore.ieee.org>

empirical data. All in all, 10 relevant publications passed the manual check, resulting in the list summarized in Table 1. Having collected the data, all papers were systematically checked for the influence of hierarchy. As Table 1 shows, reported data ranges from negative influence [12] over no influence [12–14] to mostly positive influence [15]. These experiments have been conducted with a wide spectrum of modeling languages. It is interesting to note though that diverse effects have been observed for a specific notation such as statecharts or ER-models. In general, most experiments are able to show an effect of hierarchy either in a positive *or* a negative direction. However, it remains unclear *under which circumstances* positive or negative influences can be expected. To approach this issue, in the following, we will employ concepts from cognitive psychology to provide a systematic view on which factors influence understandability.

Work	Findings
Moody [15] Domain: ER-Models	Positive influence on accuracy, no influence / negative influence on time
Reijers et al. [16, 17] Domain: Business Process Models	Positive influence on understandability for one out of two models
Cruz-Lemus et al. [9, 18] Domain: UML Statecharts	Series of experiments, positive influence on understandability in last experiment
Cruz-Lemus et al. [13] Domain: UML Statecharts	Hierarchy depth of statecharts has no influence
Shoval et al. [14] Domain: ER-Models	Hierarchy has no influence
Cruz-Lemus et al. [8] Domain: UML Statecharts	Positive influence on understandability for first experiment, negative influence in replication
Cruz-Lemus et al. [12, 19] Domain: UML Statecharts	Hierarchy depth has a negative influence

**Table 1.** Empirical studies into hierarchical structuring

## 2.2 Inference: A General-Purpose Problem Solving Process

As discussed in Sect. 2.1, the impact of hierarchy on understandability can range from negative over neutral to positive. To provide explanations for these diverse findings, we turn to insights from cognitive psychology. In experiments, the understandability of a conceptual model is usually estimated by the difficulty of answering questions about the model. From the viewpoint of cognitive psychology, answering a question refers to a *problem solving task*. Thereby, three different problem-solving “*programs*” or “*processes*” are known: search, recognition and inference [20]. Search and recognition allow for the identification of information of low complexity, i.e., locating an object or the recognition of patterns. Most conceptual models, however, go well beyond complexity that can be handled by search and recognition. Here, the human brain as a “*truly generic problem solver*” [21] comes into play. Any task that can not be solved by search or recognition, has to be solved by deliberate thinking, i.e., inference, making *inference* the *most important cognitive process* for understanding conceptual models. Thereby, it is widely acknowledged that the human mind is limited by the capacity of its

working memory, usually quantified to as  $7\pm 2$  slots [22]. As soon as a mental task, e.g., answering a question about a model, overstrains this capacity, errors are likely to occur [23]. Consequently, mental tasks should always be designed such that they can be processed within this limit; the amount of working memory a certain task thereby utilizes is referred to as *mental effort* [24].

In the context of this work and similar to [25], we take the view that the impact of modularization on understandability, i.e., the influence on inference, ranges from negative over neutral to positive. Seen from the viewpoint of cognitive psychology, we can identify two *opposing forces* influencing the understandability of a hierarchically structured model. Positively, hierarchical structuring can help to reduce the mental effort through *abstraction* by reducing the number of model elements to be considered at the same time [15]. Negatively, the introduction of sub-models may force the reader to *switch her attention* between the sub-models, leading to the so-called *split-attention effect* [26]. Subsequently, we will discuss how these two forces presumably influence understandability.

*Abstraction.* Through the introduction of hierarchy it is possible to group a part of a model into a sub-model. When referring to such a sub-model, its content is hidden by providing an abstract description, such as a complex activity in a business process model or a composite state in an UML statechart. The concept of abstraction is far from new and known since the 1970s as “*information hiding*” [1]. In the context of our work, it is of interest in how far abstraction influences model understandability. From a theoretical point of view, abstraction should show a positive influence, as abstraction reduces the amount of elements that have to be considered simultaneously, i.e., abstraction can hide irrelevant information, cf. [15]. However, if positive effects depend on whether information can be hidden, the way how hierarchy is displayed apparently plays an important role. Here, we assume, similar to [15, 17], that each sub-model is presented separately. In other words, each sub-model is displayed in a separate window if viewed on a computer, or printed on a single sheet of paper. The reader may arrange the sub-models according to her preferences and may close a window or put away a paper to hide information. To illustrate the impact of abstraction, consider the BPMN model shown in Fig. 1. Assume the reader wants to determine whether the model allows for the execution of sequence A, B, C. Through the abstraction introduced by sub-processes A and C, the reader can answer this question by looking at the top-level process only (i.e., activities A, B and C); the model allows to hide the content of sub-processes A and C for answering this specific question, hence reducing the number of elements to be considered.

*Split-Attention Effect.* So far we have illustrated that abstraction through hierarchical structuring can help to reduce mental effort. However, the introduction of sub-models also has its downsides. When extracting information from the model, the reader has to take into account several sub-models, thereby switching attention between sub-models. The resulting *split-attention effect* [26] then leads to increased mental effort, nullifying beneficial effects from abstraction. In fact, too many sub-models impede understandability, as pointed out in [4].

Again, as for abstraction, we assume that sub-models are viewed separately. To illustrate this, consider the BPMN model shown in Fig. 1. To assess whether activity J can be executed after activity E, the reader has to switch between the top-process as well as sub-processes A and C, causing her attention to split between these models, thus increasing mental effort.

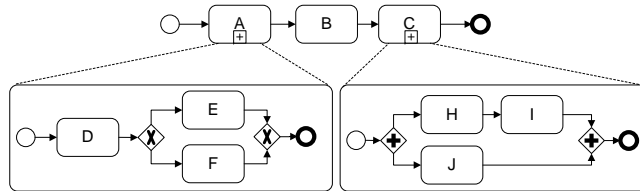


Fig. 1. Example of hierarchical structuring

While the example is certainly artificial and small, it illustrates that it is not always obvious in how far hierarchical structuring impacts a model’s understandability.<sup>5</sup>

### 3 Assessing the Impact of Hierarchy

Up to now we discussed how the cognitive process of inferencing is influenced by different degrees of hierarchical structuring. In Sect. 3.1, we define a theoretical framework that draws on cognitive psychology to explain and integrate these observations. We also discuss the measurement of the impact of hierarchy on understanding in Sect. 3.2 along with its sensitivity to model size in Sect. 3.3 and experience in Sect. 3.4. Furthermore, we discuss the implications of this framework in Sect. 3.5 and potential limitations in Sect. 3.6.

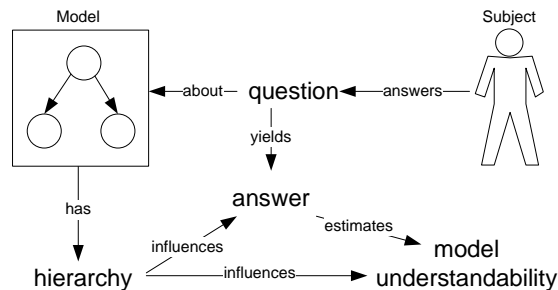


Fig. 2. Research model

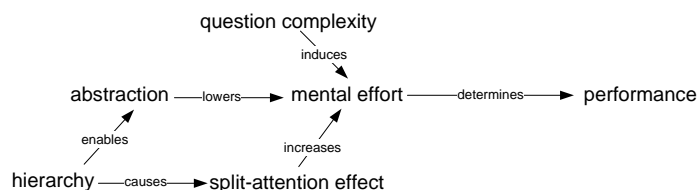
#### 3.1 Towards a Cognitive Framework

The typical research setup of experiments investigating the impact of hierarchy, e.g., as used in [8, 9, 15, 17, 18], is shown in Fig. 2. The posed research question

<sup>5</sup> At this point we would like to remark that we do not take into account class diagrams hierarchy metrics, e.g. [27], since such hierarchies *do not* provide abstraction in the sense we define it. Hence, they fall outside our framework.

thereby is how the *hierarchy* of a model influences *understandability*. In order to operationalize and measure model understandability, a common approach is to use the performance of answering questions about a model, e.g., accuracy or time, to *estimate* model understandability [9, 17, 18]. In this sense, a *subject* is asked to *answer questions* about a model; whether the model is hierarchically structured or not serves as treatment.

When taking into account the interplay of abstraction and split-attention effect, as discussed in Sect. 2.2, it becomes apparent that the impact of hierarchy on the performance of answering a question might not be uniform. Rather, each individual question may benefit from or be impaired by hierarchy. As the estimate of understandability is the average answering performance, it is essential to understand how *a single question* is influenced by hierarchy. To approach this influence, we propose a framework that is centered around the concept of mental effort, i.e., the load imposed on the working memory [24], as shown in Fig. 3. In contrast to most existing works, where hierarchy is considered as a dichotomous variable, i.e., hierarchy is present or not, we propose to view the impact of hierarchy as the result of two opposing forces. In particular, every *question* induces a certain *mental effort* on the reader caused by the question's complexity, also referred to as *intrinsic cognitive load* [23]. This value *depends* on model-specific factors, e.g., model size, question type or layout, and person-specific factors, e.g., experience, but is *independent* of the model's hierarchical structure. If hierarchy is present, the resulting mental effort is decreased by *abstraction*, but increased by the *split-attention effect*. Based on the resulting mental effort, a certain answering performance, e.g., accuracy or time, can be expected. In the following, we discuss the implications of this framework. In particular, we discuss how to measure the impact of hierarchy, then we use our framework to explain why model size is important and why experience affects reliable measurements.

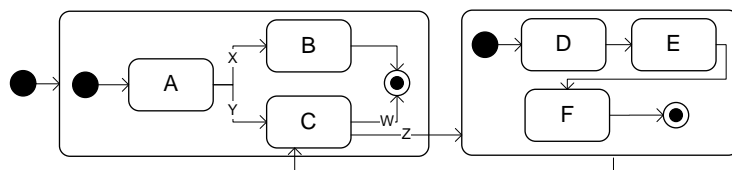


**Fig. 3.** Theoretical framework for assessing understandability

### 3.2 Measuring the Impact on Model Understandability.

As indicated [9, 8, 15, 17, 18] it is unclear whether and under which circumstances hierarchy is beneficial. As argued in Sect. 2.2, hierarchical structuring can affect answering performance positively by abstraction and negatively by the split-attention effect. To make this trade-off measurable *for a single question*, we provide an operationalization in the following. We propose to estimate the gains of abstraction by counting the number of model elements that can be “hidden” for answering a specific question. Contrariwise, the loss through the split-attention effect can be estimated by the number of context switches, i.e., switches between sub-models, that are required to answer a specific question. To illustrate

the suggested operationalization, consider the UML statechart in Fig. 4. When answering the question whether sequence A, B is possible, the reader presumably benefits from the abstraction of state C, i.e., states D, E and F are hidden—leading to a gain of three (hidden model elements). On the contrary, when answering the question, whether the sequence A, D, E, F is possible, the reader does not benefit from abstraction, but has to switch between the top-level state and composite state C. In terms of our operationalisation, no gains are to be expected, since no model element is hidden. However, two context switches when following sequence A, D, E, F, namely from the top-level state to C and back, are required. Overall, it can be expected hierarchy *compromises* this question.



**Fig. 4.** Abstraction versus split-attention effect

Regarding the use of this operationalization we have two primary purposes in mind. First, it shall help experimenters to design experiments that are not biased toward/against hierarchy by selecting appropriate questions. Second, on the long run, the operationalization could help to estimate the impact of hierarchy on a conceptual model. Please note that these applications are to be viewed under some limitations as discussed in Sect. 3.6.

### 3.3 Model Size

Our framework defines two major forces that influence the impact of hierarchy on understandability: abstraction (positively) and the split-attention effect (negatively). In order that hierarchy is able to provide benefits, the model must be large enough to benefit from abstraction. Empirical evidence for this theory can be found in [9]. The authors conducted a series of experiments to assess the understandability of UML statecharts with composite states. For the first four experiments no significant differences between flattened models and hierarchical ones could be found. Finally, the last experiment showed significantly better results for the hierarchical model—the authors identified increased complexity, i.e., model size, as one of the main factors for this result. While it seems very likely that there is a certain complexity threshold that must be exceeded, so that desired effects can be observed, it is not yet clear where exactly this threshold lies. To illustrate how difficult it is to define this threshold, we would like to provide an example from the domain of business process modeling, where estimations range from 5–7 model elements [5] over 5–15 elements [6] to 50 elements [7]. In order to investigate whether such a threshold indeed exists and how it can be computed, we envision a series of controlled experiments. Therein, we will systematically combine different model sizes with degrees of abstraction and measure the impact on the subject’s answering performance.



### 3.4 Experience

Besides the size of the model, the reader's experience is an important subject-related factor that should be taken into account [28]. To systematically answer why this is the case, we would like to refer to Cognitive Load Theory [23]. As introduced, it is known that the human working memory has a certain capacity, if it is overstrained by some mental task, errors are likely. As learning causes additional load on the working memory, novices are more likely to make mistakes, as their working memory is more likely to be overloaded by the complexity of the problem solving task in combination with learning. Similarly, less capacity is free for carrying out the problem solving task, i.e., answering the question, hence lower performance with respect to time is to be expected. Hence, experimental settings should ensure that most mental effort is used for problem solving instead of learning. In other words, subjects are not required to be experts, but *must be* familiar with hierarchical structures. Otherwise, it is very likely that results are influenced by the effort needed for learning. To strengthen this case, we would like to refer to [8], where the authors investigated composite states in UML statecharts. The first experiment showed *significant benefits* for composite states, i.e., hierarchy, whereas the *replication* showed *significant disadvantages* for composite states. The authors state that the "*skill of the subjects using UML for modeling, especially UML statechart diagrams, was much lower in this replication*", indicating that experience plays an important role.

### 3.5 Discussion

The implications of our work are threefold. First, hierarchy presumably does not impact answering performance uniformly. Hence, when estimating model understandability, results depend on *which* questions are asked. For instance, when only questions are asked that do not benefit from abstraction, but suffer from the split-attention effect, a bias adversely affecting hierarchy can be expected. None of the experiments presented in Sect. 2.1 describes a procedure for defining questions, hence inconclusive results may be attributed to unbalanced questions. Second, for positive effects of hierarchy to appear, presumably a certain model size is required [9]. Third, a certain level of expertise is required that the impact of hierarchy instead of learning is measured, as to be observed in [8].

### 3.6 Limitations

While the proposed framework is based on established concepts from cognitive psychology and our findings coincide with existing empirical research, there are some limitations. First, our proposed framework is currently based on theory only, an empirical evaluation is yet missing. To counteract this problem, we are currently planning a thorough empirical validation, cf. Sect. 4. In this vein, also the operationalization of abstraction and split-attention effect needs to be investigated. For instance, we do not know yet whether a linear increase in context switches also results in a linearly decreased understandability, or the correlation

can be described by, e.g., a quadratic or logarithmic behavior. Second, our proposal focuses on the effects on a single question, i.e., we can not yet assess the impact on the understandability of the entire model. Still, we think that the proposed framework is a first step towards assessing the impact on model understandability, as it is assumed that the overall understandability can be computed by averaging the understandability of all possible individual questions [29].

## 4 Summary and Outlook

We first had a look at studies on the understandability of hierarchically structured conceptual models. Hierarchy is widely recognized as viable approach to handle complexity—still, reported empirical data seems contradictory. We draw from cognitive psychology to define a framework for assessing the impact of hierarchy on model understandability. In particular, we identify abstraction and the split-attention effect as opposing forces that can be used to estimate the impact of hierarchy with respect to the performance of answering a question about a model. In addition, we use our framework to explain why model size is a prerequisite for a positive influence of modularization and why insufficient experience can bias measurement in experiments. We acknowledge that this work is just the first step towards assessing the impact of hierarchy on model understandability. Hence, future work clearly focuses on empirical investigation. First, the proposed framework is based on well-established theory, still, a thorough empirical validation is needed. We are currently preparing an experiment for verifying that the interplay of abstraction and split-attention effect can actually be observed in hierarchies. In this vein, we also pursue the validation and further refinement of the operationalization for abstraction and split-attention effect.

## References

1. Parnas, D.L.: On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM* **15** (1972) 1053–1058
2. van der Aalst, W., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. *Information Systems* **30** (2005) 245–275
3. Davies, R.: *Business Process Modelling With Aris: A Practical Guide*. Springer (2001)
4. Damij, N.: Business process modelling using diagrammatic and tabular techniques. *Business Process Management Journal* **13** (2007) 70–90
5. Sharp, A., McDermott, P.: *Workow Modeling: Tools for Process Improvement and Application Development*. Artech House (2011)
6. Kock, N.F.: Product flow, breadth and complexity of business processes: An empirical study of 15 business processes in three organizations. *Business Process Re-engineering & Management Journal* **2** (1996) 8–22
7. Mendling, J., Reijers, H.A., van der Aalst, W.M.P.: Seven process modeling guidelines (7pmg). *Information & Software Technology* **52** (2010) 127–136
8. Cruz-Lemus, J.A., Genero, M., Manso, M.E., Piattini, M.: Evaluating the Effect of Composite States on the Understandability of UML Statechart Diagrams. In: *Proc. MODELS '05*. (2005) 113–125

10 S. Zugal et al.

9. Cruz-Lemus, J.A., Genero, M., Manso, M.E., Morasca, S., Piattini, M.: Assessing the understandability of UML statechart diagrams with composite states—A family of empirical studies. *Empir Software Eng* **25** (2009) 685–719
10. Burton-Jones, A., Meso, P.N.: Conceptualizing systems for understanding: An empirical test of decomposition principles in object-oriented analysis. *ISR* **17** (2006) 38–60
11. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. *JSS* **80** (2007) 571–583
12. Cruz-Lemus, J., Genero, M., Piattini, M.: Using controlled experiments for validating uml statechart diagrams measures. In: *Software Process and Product Measurement*. Volume 4895 of LNCS. Springer Berlin / Heidelberg (2008) 129–138
13. Cruz-Lemus, J., Genero, M., Piattini, M., Toval, A.: Investigating the nesting level of composite states in uml statechart diagrams. In: *Proc. QAOOSE '05*. (2005) 97–108
14. Shoval, P., Danoch, R., Balabam, M.: Hierarchical entity-relationship diagrams: the model, method of creation and experimental evaluation. *Requirements Engineering* **9** (2004) 217–228
15. Moody, D.L.: Cognitive Load Effects on End User Understanding of Conceptual Models: An Experimental Analysis. In: *Proc. ADBIS '04*. (2004) 129–143
16. Reijers, H., Mendling, J., Dijkman, R.: Human and automatic modularizations of process models to enhance their comprehension. *Inf. Systems* **36** (2011) 881–897
17. Reijers, H., Mendling, J.: Modularity in Process Models: Review and Effects. In: *Proc. BPM '08*. (2008) 20–35
18. Cruz-Lemus, J.A., Genero, M., Morasca, S., Piattini, M.: Using Practitioners for Assessing the Understandability of UML Statechart Diagrams with Composite States. In: *Proc. ER Workshops '07*. (2007) 213–222
19. Cruz-Lemus, J.A., Genero, M., Piattini, M., Toval, A.: An empirical study of the nesting level of composite states within uml statechart diagrams. In: *Proc. ER Workshops*. (2005) 12–22
20. Larkin, J.H., Simon, H.A.: Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* **11** (1987) 65–100
21. Tracz, W.J.: Computer programming and the human thought process. *Software: Practice and Experience* **9** (1979) 127–137
22. Miller, G.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review* **63** (1956) 81–97
23. Sweller, J.: Cognitive load during problem solving: Effects on learning. *Cognitive Science* **12** (1988) 257–285
24. Paas, F., Tuovinen, J.E., Tabbers, H., Gerven, P.W.M.V.: Cognitive Load Measurement as a Means to Advance Cognitive Load Theory. *Educational Psychologist* **38** (2003) 63–71
25. Wand, Y., Weber, R.: An ontological model of an information system. *IEEE TSE* **16** (1990) 1282–1292
26. Sweller, J., Chandler, P.: Why Some Material Is Difficult to Learn. *Cognition and Instruction* **12** (1994) 185–233
27. Chidamber, S.R., Kemerer, C.F.: A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.* **20** (1994) 476–493
28. Reijers, H.A., Mendling, J.: A Study into the Factors that Influence the Understandability of Business Process Models. *SMCA* **41** (2011) 449–462
29. Melcher, J., Mendling, J., Reijers, H.A., Seese, D.: On Measuring the Understandability of Process Models. In: *Proc. BPM Workshops '09*. (2009) 465–476

## Assessing the Frequency of Empirical Evaluation in Software Modeling Research

Jeffrey C. Carver, Eugene Syriani, Jeff Gray

University of Alabama, Department of Computer Science  
Tuscaloosa, Alabama USA  
{carver, esyriani, gray}@cs.ua.edu

**Abstract.** Researchers in software modeling often publish new tools or methodologies that claim to offer some advantage to the modeling community. There are different methods by which those claims can be evaluated. In this paper, we examine the degree to which such claims are supported by various types of empirical evaluation. We surveyed five editions of the MoDELS conference from 2006-2010, as well as the primary conference that focuses on empirical software engineering (the *International Symposium on Empirical Software Engineering and Metrics*), to understand the frequency with which empirical evaluation has been reported in the software modeling community. Our summary of 266 MoDELS papers found that 195 (73%) of the publications performed no empirical evaluation. This paper summarizes our findings from that survey and offers recommendations for improving the awareness and need for empirical evaluation in software modeling research.

**Keywords:** Empirical software engineering, Model-driven Engineering

### 1. Introduction

Research into software modeling has attracted many creative and transformative ideas over the past decade, ranging from new methods for defining languages and transforming their model instances, to higher level performance analysis and verification tools that abstract the essence of some system property. Although the novelty of software modeling research has led to numerous advances, the collective body of work in this area has not always followed the typical tenets of a scientific discipline. One of the key precepts of scientific investigation is the ability to repeat an experiment to verify that some new scientific discovery can be confirmed under numerous scenarios. For most contributions in model-driven engineering, some new tool or technique is often proposed and discussed through an illustrative case study, but generally is not evaluated at the level of rigor assumed for a traditional empirical evaluation.

Our suspicion about the level of empirical studies in modeling research led to this summary paper that analyzes the degree of empirical evaluation in software modeling research. To approach this topic, we analyzed the most recent five editions (from 2006 through 2010) of the most influential conference in software modeling – the

conference on *Model-Driven Engineering, Languages and Systems* (MoDELS). Two of the authors of this paper (Gray and Syriani) have themselves published papers at this conference that did not contain an empirical evaluation. We were curious about the extent to which this practice is common in the software modeling community. In addition to observing contributions at MoDELS, we also considered the prevalence of modeling papers at a venue focused on empirical software engineering. The remainder of this paper summarizes our findings from an analysis of 266 MoDELS papers. Our suspicions were confirmed by our analysis, which suggests that a large majority of research papers in the modeling community fail to provide any level of empirical evidence to support the claims of benefit made in those papers.

The next section of this paper provides an overview of empirical studies and the methodology that we used in conducting our analysis of MoDELS papers. Section 3 presents the results of our analysis of the MoDELS conference and our analysis of software modeling papers that have appeared in the flagship empirical software engineering conference, the *International Symposium on Empirical Software Engineering and Measurement* (ESEM). Section 4 discusses the results of the analysis in more detail. Finally, we conclude the paper in Section 5.

## 2. Overview of Empirical Studies and Methodology

For a new modeling tool or technique to become used, the developer of the tool or technique must demonstrate its value. Although a proof-of-concept or illustrative example are important first steps in establishing the usefulness of a technique or tool, claims about the usefulness of modeling techniques and tools cannot be fully validated without the use of various types of empirical studies. An empirical study is a validation method that draws conclusions based on observations (as opposed to proof, argumentation, or expert opinion).

In the larger software engineering community, empirical studies have commonly been used to understand developer behavior in a number of important areas. There is an entire sub-community focused on validating software engineering claims via empirical study. This sub-community has a conference (ESEM), a Springer journal (*Empirical Software Engineering*) and a number of handbooks [3], [9], [14]. The first author of this paper comes from this community.

The goal of this investigation was to determine how many papers had some type of empirical evaluation of their claims. We realize that there are evaluation methods other than empirical studies (e.g., demonstration/proof-of-concept or theoretical proof). But, in this paper, we focus only on empirical evaluation. Among the three authors, two are experts in the modeling domain and one is an expert in the empirical software engineering domain. Working together, we were able to complement each other's expertise to perform this analysis.

We used a three-step process for identifying which papers contain an empirical component. The first step was to develop an initial characterization scheme. Next, the two modeling experts individually analyzed the proceedings of various years of the MoDELS proceedings to identify and classify the papers. Third, the empirical studies expert reviewed the papers identified in step two and validated the classification of

those papers. Step 3 resulted in some modifications to the characterization scheme. The remainder of this section describes each step in more detail.

### **2.1. Step 1 – Develop an initial characterization scheme**

We began with the assumption that there are two types of empirical studies: those that are more analytical (i.e., perform some type of analysis of a tool and its properties without using humans) and those that are human-based (i.e., they involved studying one or more people using a modeling technique). For each type of study, we created two categories: “non-rigorous” and “rigorous.” The difference between rigorous and non-rigorous was subjective and ill-defined at this first stage of analysis.

Because we had no preconceived notions of the results of the literature search, this initial characterization scheme was necessarily vague. We realized that after examining the actual papers, we would have to refine the characterization scheme to accurately describe the identified papers.

### **2.2. Step 2 – Identification of candidate papers**

The two modeling experts divided the five years of MoDELS proceedings between them and individually analyzed all of the papers. For each paper, they first determined whether there was any type of empirical study and whether it was human-based. At this stage, they also made a subjective determination as to whether a paper was rigorous. After this step, we developed a spreadsheet that characterized each paper into one of five categories: *no empirical study*, *non-rigorous non-human*, *rigorous non-human*, *non-rigorous human* and *rigorous human*.

### **2.3. Step 3 – Review of candidate papers and finalization of characterization**

The empirical software engineering expert then reviewed each paper that the modeling experts identified during Step 2 as having an empirical study. The goal of this process was to provide a second observation to validate the characterization from Step 2. During the review, it quickly became apparent that our initial characterization scheme was inadequate. We refined the initial characterization as follows.

First, we more clearly defined the term “empirical study.” Some of the candidate papers identified during Step 2 really contained just a demonstration or implementation of the new tool or technique rather than an empirical study. In fact, several MoDELS papers had an “Evaluation” section that was merely a discussion of lessons learned, rather than what those in the empirical software engineering community would call an empirical study. We clarified the definition of what we considered as an empirical study to exclude papers that clearly did not gather any type of data to evaluate the proposed tool or technique.

In reviewing the papers, we identified two types of empirical papers:

1. Papers that propose a new tool or technique and then perform some type of evaluation of it.
2. Papers that gather information about the use of modeling techniques in practice. These papers do not propose new approaches; rather, they study existing approaches or survey users to develop requirements for tools or techniques that may be needed. We call these papers Formative Case Studies, as opposed to the Illustrative Case Studies that just illustrate the use of a new tool or technique.

Second, we refined the original characterization scheme to define more concretely the categories into which the papers could be classified. The revised characterization scheme is as follows:

1. *No empirical evaluation* – the paper did not provide any type of empirical evaluation of the proposed tool or technique (this, unfortunately, represented the overwhelming majority of the papers we analyzed).
2. *Non-human evaluation of the proposed tool/technique only* – the paper offered some type of empirical evaluation (e.g., performance or correctness) of the proposed tool, but did not compare the new tool against other tools or benchmarks.
3. *Non-human evaluation of proposed tool/technique by comparison with other tools* – the paper provided an empirical evaluation by comparing the proposed tool/technique against one or more existing tools or benchmarks to evaluate some aspect of the new tool/technique.
4. *Observation of humans using new tool/technique* – the paper discussed and analyzed the results from the use of the new tool/technique by one or more people other than the authors of the paper.
5. *Human-based controlled experiment* – the paper described a controlled experiment where the new tool/technique was compared against one or more existing approaches through a human-based controlled experiment where each participant used one or more approaches and provided data that could be analyzed to evaluate the new tool/technique.
6. *Formative case study* – as defined above.

### **3. Results of Literature Survey**

This section summarizes the results of our survey of the MoDELS papers and of the modeling papers that appeared in the ESEM conference.

**Table 1.** Results of the survey of the papers published at MoDELS 2006-2010

Year	Total	No Evaluation	Non-Human			Human	
			No comparison	Comparison	Observation	Controlled Experiment	Formative Case Study
2006	51	42 (82%)	6 (12%)	0 (0%)	1 (2%)	1 (2%)	1 (2%)
2007	45	36 (80%)	2 (4%)	5 (11%)	0 (0%)	2 (4%)	0 (0%)
2008	58	39 (67%)	8 (14%)	2 (3%)	2 (3%)	4 (7%)	3 (5%)
2009	58	45 (78%)	5 (9%)	2 (3%)	2 (3%)	1 (2%)	3 (5%)
2010	54	33 (61%)	8 (15%)	4 (7%)	2 (4%)	4 (7%)	3 (6%)
<b>Total</b>	<b>266</b>	<b>195 (73%)</b>	<b>29 (10%)</b>	<b>13 (4%)</b>	<b>7 (2%)</b>	<b>12 (4%)</b>	<b>10 (3%)</b>

### 3.1. Results of MoDELS Survey

In the empirical evaluations conducted, we analyzed a total of 266 papers published at MoDELS from 2006-2010. The complete analysis of the papers took approximately 18 hours of observation and recording. Table 1 summarizes the results of this assessment.

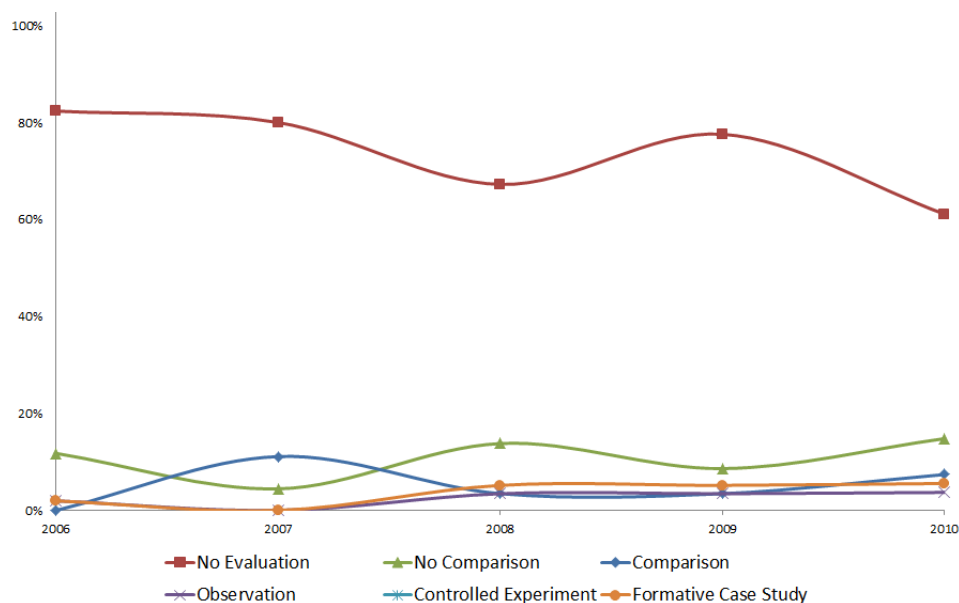
It is very clear that, for each year, the number of papers without any evaluation was predominant: ranging from 61% in 2010 up to 82% in 2006. However, the tendency seems to suggest a rising awareness and influence of the need for empirical studies, as we note an average decrease of about 4% each year in the number of papers with no evaluation (there is a 21% drop in the “No Evaluation” category from the beginning of our study period to the end of the period over the five years observed). We have no direct evidence for the cause of this improvement, but feedback sent to authors on reviews over the period of the study may suggest the emerging demand among the Program Committee for more rigorous evaluation.

Those papers that did have some form of empirical study were often restricted to simple evaluations of performance or correctness of the proposed tool/technique without comparing it to other results (41% of the those papers describing an empirical study were in the “No comparison” category). The papers in 2007 seem to be the only exception, where 11% of all papers addressed comparisons with other tools or benchmarks.

On average, about 11% of the papers were supported by empirical studies involving humans. In this category, 42% of the papers contained controlled experiments, representing not more than 7% of all papers (years 2008 and 2010). The number of papers where the evaluation was observed by at least one external participant has been quite steady at about 3% of all papers. Formative case studies are gaining popularity with up to 6% of all the papers in 2010.

The “Total” row (at the bottom of Table 1) shows the portions occupied by each of the categorizations defined in Section 2 across all years. Although 73% of the papers published at MoDELS do not contain an evaluation, 10% of the papers only evaluate their own tool without any comparison to other approaches. Thus, only the remaining 17% of the papers involve an empirical evaluation of the proposed tool or technique. However, according to Fig. 1, this number is increasing every year: up to 24% in 2010. This trend may suggest that authors are aware of the lack of empirical evidence in the modeling community and are now working on filling this gap.





**Fig. 1.** Evolution of the empirical studies involved in MoDELS 2006-2010 papers

### 3.2 Results of ESEM Survey

As evidenced by the discussion in the previous section, the MoDELS conference appears to be focused mainly on proposing new tools and techniques without rigorous evaluation. To the authors' credit, the paper length restrictions of the LNCS format used in MoDELS leave little space for discussion of formal evaluation. The *ESEM* conference is a general software engineering conference that focuses on the empirical evaluation of newly proposed techniques across all software engineering topics. We analyzed the same five years of the ESEM conference to determine whether more formal evaluations of modeling research were being published there. To identify the set of papers, we queried the proceedings using the following keywords: "UML," "DSL," "metamodel," "model," and "model-driven." The modeling experts then vetted the results of the search to ensure that the papers were within the scope of software modeling.

Based on this analysis, we can make a few interesting observations. The ESEM conference has three types of papers: Regular Papers, Short Papers, and Posters. In total, we only found 17 modeling papers across the five years that we analyzed ESEM. Of those 17 papers, only 4 were Regular Papers (10 pages IEEE or ACM format) out of a total of 178 Regular Papers and 10 were Short Papers (4 pages) out of a total of 118 Short Papers. Thus, even when software modeling papers are published in an empirical venue, they tend to be shorter and do not provide a high-level of detail. In analyzing the five years of ESEM, we were not able to identify any trends

that would suggest the prominence of modeling papers is increasing in the empirical software engineering community. One final observation, in comparing the author lists and titles of the ESEM papers against the empirical MoDELS papers, we found very little overlap; only one paper seemed to be about the same tool or technique. Thus, the cross-pollination of results across the two communities seems to be very low.

## **4. Observations from Our Survey**

This section provides a summary of our observations about the papers that focused on controlled experiments and formative case studies.

### **4.1. Controlled Experiments**

Across the five years of the MoDELS conference, we found twelve controlled experiments [1], [2], [5], [6], [7], [8], [10], [11], [12], [13], [15], [16]. This category of papers serves as an example of the types of papers that we feel should be more prevalent within the MoDELS community. In this section, we provide a brief discussion of some of the trends observed in these controlled experiment papers. Overall, the level of detail reported by the authors of these papers is quite low. We realize that this level of reporting is likely affected by paper length restrictions and the need to fully describe the newly proposed tool or technique as the core contribution of the paper. Although we do not have the space to evaluate the quality of each study in detail, there are two important factors that are relatively easy to evaluate: 1) the number of participants, and 2) whether the participants were students or professionals.

In terms of the number of participants in the studies, one half of the identified papers had less than 25 participants, and only two studies had more than 50 participants. Furthermore, one study did not even report the number of participants. In terms of the type of participant, only one study had professionals as a portion of the participants. The overwhelming majority of the studies relied on undergraduates with only a few using graduate students. Over 33% of the studies did not specify whether the participants were students or professionals. The use of student participants is not necessarily bad, but researchers need to make a clear case as to why student participants are a valid population for the question under investigation [4].

There does not appear to be a significant trend in the number of controlled experiments reported. From 2006 through 2008, the number was increasing. Then, there was a large drop of such experiments in 2009. The percentage of controlled experiments in 2010 was equivalent to the percentage reported in 2008. Even in the best years, only 7% of the papers reported controlled experiments. In general, we would like to see an increase in both the frequency and diversity of controlled experiments within the modeling community.

## 4.2. Formative Case Studies

Across the five years of the MoDELS conference, we found ten Formative Case Study papers. There were two types of Formative Case Studies. First, there were four studies that did not involve humans. These studies tended to analyze some existing source code to understand how various modeling tools would or would not work effectively. Second, there were six studies that focused on humans. These studies mostly used a survey method to understand how existing tools were not meeting the needs of developers. The output of many of these studies was a set of requirements for new tools that were needed. Contrary to the Controlled Experiments, which focused heavily on student participants, the Formative Case Studies were focused more on industrial settings. Similar to the Controlled Experiments, we would also like to see additional Formative Case Studies that provide input to tool and method developers to help ensure that their work is relevant to the needs of practitioners.

## 5. Conclusion

This paper provides evidence that the rigor of empirically validated research in software modeling is rather weak and should be a focus of future authors of MoDELS papers. The high-level of incidence of papers with no evaluation is somewhat alarming when compared to other software engineering venues (e.g., ICSE) where empirical evaluation is more expected as a scientific contribution. Overall, the level of empirical evaluation as seen in the software modeling community is quite low for a scientific and engineering discipline. A goal of this paper is to raise the awareness of this issue to assist in progressing the area of software modeling with a more scientific underpinning. Our own future work will include a similar analysis of papers in the software modeling community's flagship journal – *Software and Systems Modeling*.

As part of this work, we posit that there is a need for more controlled experiments within the modeling community. We realize that there are at least three factors that are hindering these types of studies being conducted. First, many researchers in the modeling community may lack the background or training to carry out empirical studies. This situation is evidenced by the fact that authors frequently mention “validation experiments” which are nothing more than the application of the findings or a toy example. Second, many researchers in the modeling community are more interested in creating new tools and techniques than they are in performing the rigorous evaluation of those techniques. Third, given the length restrictions of the formatting style in the MoDELS conference, there is often not adequate space to discuss both the new tool or technique and its validation, so most researchers seem to opt for devoting space to the definition of the tool or technique as representing the core contribution of their paper.

Our goal in this paper is to stress the importance of building a culture that values and expects empirical validation of newly proposed tools and methods. To help facilitate this goal, we propose the following solutions to the problem. First, researchers in the modeling domain who are interested in conducting appropriate empirical evaluations themselves need to collaborate more often with researchers who

have expertise in empirical evaluation of software engineering methods (as the authors of this paper are doing). Such a collaboration allows both types of researchers to do what they are interested in and what they do best. Second, we suggest that more rigorous empirical evaluations of modeling research be published in the ESEM conference, where the focus is on the empirical evaluation, to cross-pollinate the contributions of the modeling community with those explicitly working in empirical techniques. In that venue, authors can devote more space to describing the evaluation and interpreting the results. A somewhat radical suggestion is to afford MoDELS authors an additional two to three pages of space for any paper that includes a more rigorous evaluation based on an empirical study.

A spreadsheet representing the results of our analysis of MoDELS conferences, and a summary of the papers analyzed for the ESEM conferences, is available at: <http://www.cs.ua.edu/~carver/Data/2011/EESSMOD/>

**Acknowledgments.** This research was supported in part by NSF CAREER award CCF-1052616.

#### References

1. Almeida da Silva, M., Bendraou, R., Blanc, X. et al.: Early Deviation Detection in Modeling Activities of MDE Processes. In: Petriu, D., Rouquette, N. and Haugen, Ø. (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 6395, pp. 303-317. Oslo, Norway (2010)
2. Almeida da Silva, M., Mougenot, A., Bendraou, R. et al.: Artifact or Process Guidance, an Empirical Study. In: Petriu, D., Rouquette, N. and Haugen, Ø. (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 6395, pp. 318-330. Oslo, Norway (2010)
3. Boehm, B., Rombach, H. D., Zelkowitz, M. V.: Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili. Springer (2005)
4. Carver, J., Jaccheri, L., Morasca, S. et al.: A Checklist for Integrating Student Empirical Studies with Research and Teaching Goals. *Empirical Software Engineering*, **15** (2010) 35-59
5. Correa, A., Werner, C., Barros, M.: An Empirical Study of the Impact of OCL Smells and Refactorings on the Understandability of OCL Specifications. In: Engels, G., Opdyke, B., Schmidt, D., et al (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 4735, pp. 76-90. Nashville, TN (2007)
6. Fuhrmann, H., & von Hanxleden, R.: Taming Graphical Modeling. In: Petriu, D., Rouquette, N. and Haugen, Ø. (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 6394, pp. 196-210. Oslo, Norway (2010)
7. Genero, M., Cruz-Lemus, J., Caivano, D. et al.: Assessing the Influence of Stereotypes on the Comprehension of UML Sequence Diagrams: A Controlled Experiment. In: Czarnecki, K., Ober, I., Bruel, J., et al (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 5301, pp. 280-294. Toulouse, France (2008)

8. Gravino, C., Scanniello, G., Tortora, G.: An Empirical Investigation on Dynamic Modeling in Requirements Engineering. In: Czarnecki, K., Ober, I., Bruel, J., et al (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 5301, pp. 615-629. Toulouse, France (2008)
9. Juristo, N., & Moreno, A.: Lecture notes on empirical software engineering. World Scientific, Singapore (2003)
10. Lange, C., DuBois, B., Chaudron, M. et al.: An Experimental Investigation of UML Modeling Conventions. In: Nierstrasz, O., Whittle, J., Harel, D., et al (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 4199, pp. 27-41. Genova, Italy (2006)
11. Lucrédio, D., de M. Fortes, R., Whittle, J.: MOOGLE: A Model Search Engine. In: Czarnecki, K., Ober, I., Bruel, J., et al (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 5301, pp. 296-310. Toulouse, France (2008)
12. Mäder, P., & Cleland-Huang, J.: A Visual Traceability Modeling Language. In: Petriu, D., Rouquette, N. and Haugen, Ø. (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 6394, pp. 226-240. Oslo, Norway (2010)
13. Prochnow, S., & von Hanxleden, R.: Statechart Development Beyond WYSIWYG. In: Engels, G., Opdyke, B., Schmidt, D., et al (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 4735, pp. 635-649. Nashville, TN (2007)
14. Shull, F., Singer, J., Sjøberg, D. I. K.: Guide to Advanced Empirical Software Engineering. Springer (2008)
15. Stålhane, T., & Sindre, G.: Safety Hazard Identification by Misuse Cases: Experimental Comparison of Text and Diagrams. In: Czarnecki, K., Ober, I., Bruel, J., et al (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 5301, pp. 721-735. Toulouse, France (2008)
16. Yue, T., Briand, L., Labiche, Y.: A Use Case Modeling Approach to Facilitate the Transition towards Analysis Models: Concepts and Empirical Evaluation. In: Schürr, A. and Selic, B. (eds.) Model Driven Engineering Languages and Systems, LNCS vol. 5795, pp. 484-498. Denver, CO (2009)

# Building VECM-based Systems with a Model Driven Approach: an Experience Report

Maurizio Leotta, Gianna Reggio, Filippo Ricca, and Egidio Astesiano

Dipartimento di Informatica e Scienze dell'Informazione - DISI  
Università di Genova  
16146 Genova, Italy

{maurizio.leotta|gianna.reggio|filippo.ricca|astes}@disi.unige.it

**Abstract.** Recently, we took part in a project with two local companies about the creation of a UML-based Model Driven rigorous method to develop VECM-based systems. VECM is a way to abstract from the details of different Enterprise Content Management (ECM) systems used within the same organization. This report details the experience made using our method to develop V-Protocol: a system able to protocol, sign and archive public competition announcements received by a company.

## 1 Introduction

The authors have been recently involved in the VirtualECM project<sup>1</sup> aimed at developing the VECM technology and a UML-based Model Driven (MD) rigorous method for building systems based on VECM. A VECM-based system (shortly V-System) is a system that uses the VECM software interface (Sect. 2). In a nutshell, a VECM abstracts the basic operations offered by an ECM. An ECM is a system used to capture, manage, store, and deliver enterprise content. It provides operations on documents such as: *createDocument()* and *deleteDocument()*. There are a lot of ECM systems available on the market, e.g., Alfresco (open source), SharePoint (Microsoft), Oracle Content Management (Oracle), and Documentum (EMC).

On the top of this heterogeneous ECM environment, usually, the companies build their systems using several ECM systems characterized by different interfaces; for example, a bank that uses an ECM system to manage credit transfer and another one for loans. Often, the consequence of this practice is the development of a system highly coupled with the underlying ECM systems. The VECM software interface solves this problem. In practice, the VECM allows to develop systems not tied to the specific characteristics of a particular ECM.

In this paper we propose a UML rigorous method useful to develop VECM-based systems (Sect. 3). It follows the MD approach [2] for the development of software systems. In particular, several UML models are created starting from the business process model to obtain a detailed design model that can

---

<sup>1</sup> VirtualECM project was supported by “Programma Operativo Regionale POR-FESR (2007-2013)”, Liguria, Italy

2 Maurizio Leotta, Gianna Reggio, Filippo Ricca, and Egidio Astesiano

be transformed/refined in a running system. We have applied this method to build V-Protocol, the selected case study for the project, which is a system used to protocol, sign and archive public competition announcements received by a company.

## 2 VECM and ECM

In this section we explain what a VECM is and report on the factors that have motivated the development of the VECM software interface, which is the aim of the VirtualECM project.

There are a lot of ECM systems available on the market. Even if each of them has some distinguishing features, they provide substantially the same operations often called in different ways and with different parameters.

Usually, in a big company it is possible to find different ECM systems chosen by and used in different branches of the same organization. This can happen for several reasons: for example because different branches of the same company have chosen different ECM systems for money matters, licence or specific characteristics. Thus, a company often has to build a system in an heterogeneous ECM environment using different underlying ECM systems. The result of this practice is a system that interacts with different ECM systems with their own interfaces, languages, and characteristics. Systems built in this way are tightly coupled to the set of used ECM systems, and thus tend to exhibit well-known problems (e.g., difficulties in changing, reusing, and testing software).

That problem can be solved with an additional layer of abstraction placed between the system and the different ECM systems. Such software layer is called VECM, a sort of *virtualization* of ECM systems. In practice, without the VECM a system has to interact with a set of different ECM systems and it has to know their different interfaces; instead, with the VECM a system has to know only the VECM interface. The management of the interaction with the different underlying ECM systems is totally delegated to the VECM. When the system uses a functionality offered by the VECM, it does not to know what type of ECM is actually used. In this way it is possible to replace an ECM with another without problems.

Fig. 1 shows a simplified definition of the functionalities offered by the VECM, which are an abstraction of a well defined subset of operations typically offered by an ECM. As we can see, the VECM operations cover different areas and work on or produce content.

## 3 The Method

We propose a method for the development of systems based on the VECM that follows the MD approach. The starting point is a UML model representing the target business process including at least an activity diagram and the final result is a detailed design model that can be transformed into a running system. In

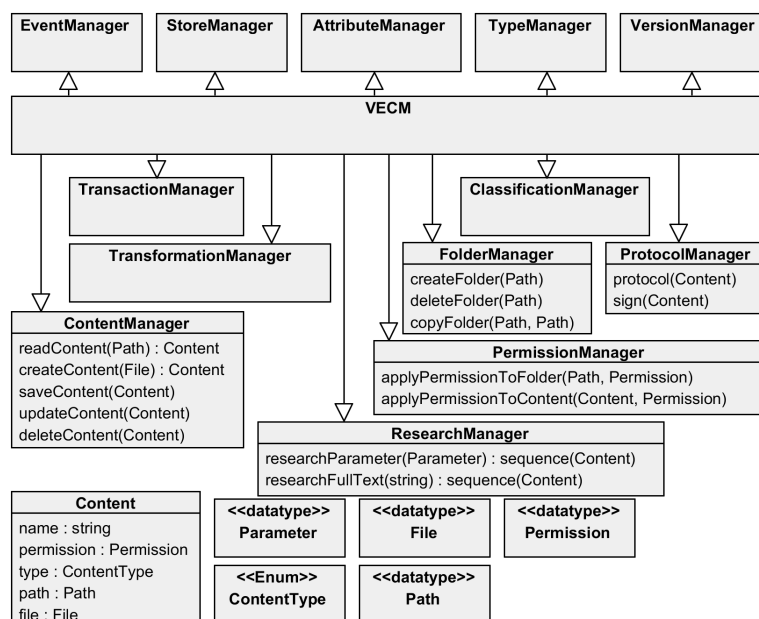


Fig. 1. UML presentation of the VECM functionalities

our method, the activity diagrams used to represent the business processes are created following the “precise style” introduced in [6].

In a nutshell, the “precise style” prescribes that the participants of a business process are explicitly listed and precisely modelled with UML by means of classes. Moreover, the behavioural view of the business process is given by an activity diagram where the basic activities and the conditions are written by respectively using the language for the actions of UML and OCL (the textual language for Boolean expressions part of UML). Thus, our UML precise model of a business process consists of: a class diagram, introducing the classes needed to type its participants, the list of its participants, and an activity diagram representing its behaviour, where all nodes (arcs) are decorated by operations calls (OCL expressions). From a previous study, we have seen that this style improves the quality of the business process models expressed as activity diagrams [5], and that it is better than a “light stile” [1].

The method for developing a V-System consists of four phases:

- Business Process Modelling
- V-System Placement
- V-System High Level Design
- V-System Detailed Design

We present our method using the Protocol case study selected in the VirtualECM project for the realization of a demonstrator named V-Protocol. The system V-Protocol has been developed by the two companies following the above



four phases. The Protocol case study is about the management of the announcements of public competitions received by a company. It can shortly be described in this way: “*First an announcement is received by the company and managed by a clerk. Subsequently, the announcement is checked by a clerk to verify its validity, and if it is valid the announcement is protocolled, digitally signed and saved in a repository*”.

### 3.1 Phase 1: Business Process Modelling

The first phase of our method consists in describing the business process/domain “*as it is*”, i.e., by means of the UML model called *BusinessProcessModel* that represents the business before the introduction of the system based on the VECM. At this level the UML model has to be close as much as possible to the given description of the business. Participants of the business will have a name and will be typed by a class with stereotype either `<<businessWorker>>` (human being) or `<<system>>` (hardware/software systems). Also the business objects will have a name and will be typed by a class stereotyped by `<<businessObject>>`.

Since it is not possible to use the V-System in the *BusinessProcessModel* and moreover the textual description could be incomplete or quite abstract, in some cases it could be difficult assigning the activities involving a business object to a business worker or to a system (e.g., in the business underlying the Protocol case study, it is not specified who will protocol, sign and save an announcement). In those cases, it is advisable assigning such activities directly to the business objects over which they will be executed. We have decided to describe those activities in the model using the passive form (e.g., `DOC.saved()` in Fig. 2).

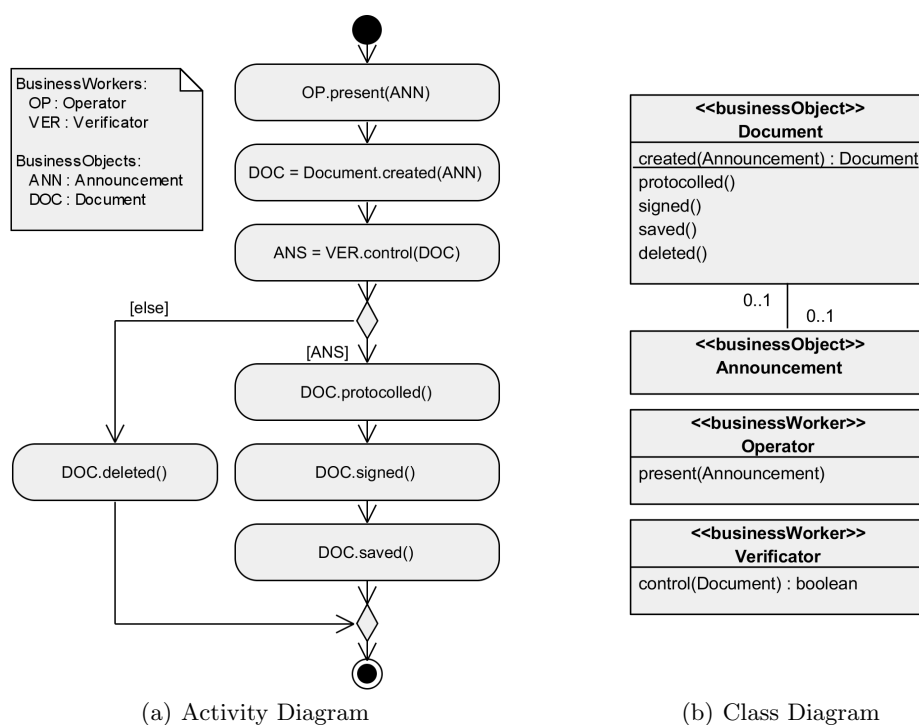
The result of this activity in the Protocol case study is a UML model reported in Fig. 2. That model is composed by two diagrams: an activity diagram and a class diagram. In the activity diagram, as reported in the side note, there are two business workers (the operator and the verifier) and two business objects (an announcement and a document). Note that in the class diagram the operations of `Document` are given in passive form, and that the `created()` operation is declared as static because in this phase it is not known who creates the document.

### 3.2 Phase 2: V-System Placement

The aim of the second phase is deciding which of the activities described in the previous phase will be performed by the V-System. The placement of the V-System is done using a swimlane labelled by the name of the system (in our case V-Protocol). During this phase a set of models are created and finally a UML model called *PlacementModel* is obtained. The activities that have to be performed by the V-System will be placed inside the swimlane, the others will remain outside. It could happen that an activity is not totally of competence of the V-System: in this case the activity has to be subdivided in two or more sub-activities assigned to different participants.

The placement is correct only if the following constraints are observed:

- at least one activity is placed inside the V-System swimlane (no activities in the swimlane mean that the system will do nothing);



**Fig. 2.** Protocol Case Study: *BusinessProcessModel*

- no activities performed by business workers are inside the swimlane (the system cannot replace the behaviour of a human being that it is unpredictable and not computable, e.g., an examiner of future employees or of the artistic value of a novel);
- at least an activity should be placed on the border of the swimlane (such an activity will result in communication between the system and some external entity);
- no activity flow (control and object) can cross the swimlane boundary (a crossing flow means a hidden communication between the system and some external entities).

Therefore only the following types of activities can be placed inside the V-System swimlane:

- passive activity of a business object: this means that the V-System will execute the activity on the object (e.g., `DOC.saved()`);
- activity performed by a pre-existing system: this means that the system under development will replace the existent system in the execution of the activity.

Since that is not allowed to place an activity assigned to a business worker inside the V-System swimlane, if the activity has to be performed by the V-System then it can be placed inside the swimlane only after a model refactoring

step in which the activity has to be assigned to a participant stereotyped by `<<system>>` (or `<<businessObject>>` using the passive form). This means that the system will perform an activity that previously was done by a human.

During the placement of the V-System in presence of constraints violations the following cases can occur:



- the placement becomes corrected after performing one or more business process model refactoring;
- it is not possible to refactor the model to satisfy the constraints; in such case the developer has to either change the placement or to conclude that the intended system is not doable.

The result of this phase is the *PlacementModel*, having the form of a *BusinessProcessModel* where part of the activity diagram is included in one swimlane named as the system to develop.

The creation of the *PlacementModel* (that for space limitations we do not report here) for V-Protocol has required a model refactoring step in order to create the activities for the messages exchange between the participants. For example, the activity `control(DOC)` has been broken in several activities because it involve both the operator and the system.

### 3.3 Phase 3: V-System High Level Design

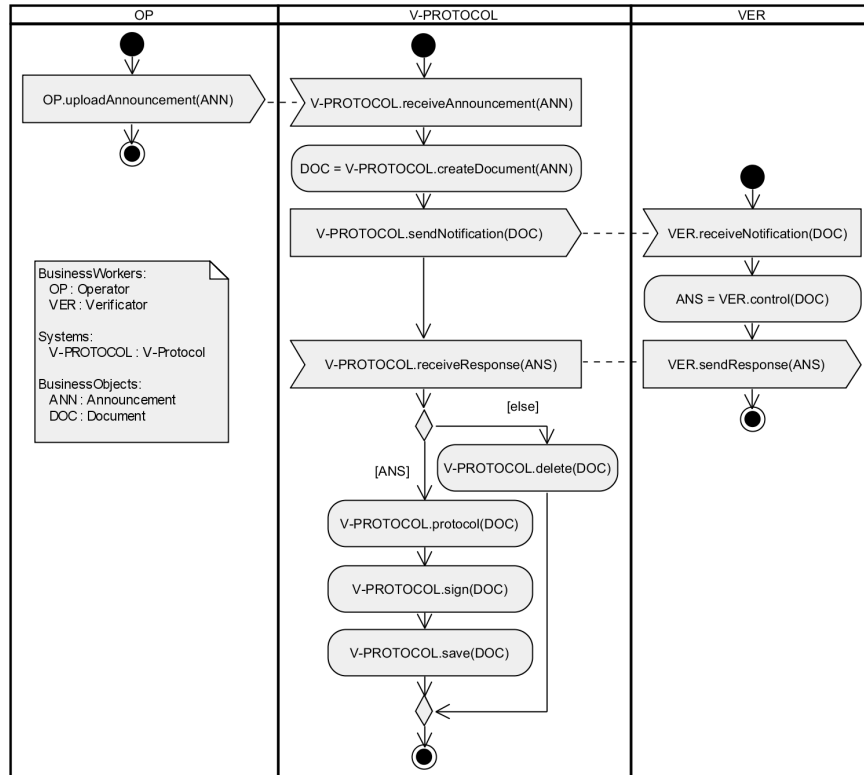
The goal of this phase is providing a high level design of the V-System with a detailed description of the activities carried out by the system. During this phase a UML model called *DesignModel* is produced.

We start from the *PlacementModel* and perform a refinement step. For each participant of type `<<businessWorker>>` and `<<system>>` in the activity diagram we introduce a swimlane labelled by its name. In this model, the involved participants communicate using two type of UML actions: – the call action<sup>2</sup> (  ) used when a participant sends a message to another one and – the accept action (  ) used when a participant receives a message from another one.

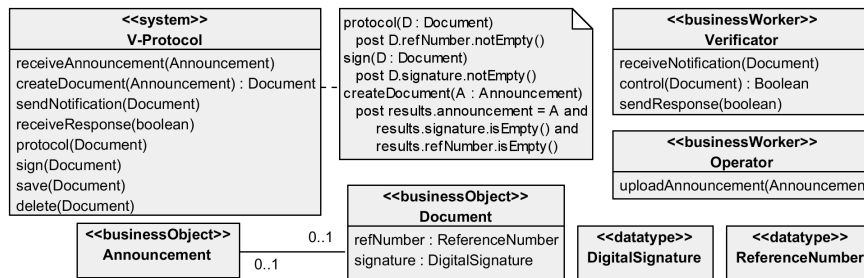
At this level of description, all the activities in the V-System swimlane are reported as executed by the system as a whole. There are no details about the inner structure of the V-System (e.g., information about the number of ECM used).

In Fig. 3 we report the *DesignModel* for the Protocol case study. It contains three swimlanes: one is for V-Protocol and two are dedicated to the business workers that interact with it. The operator, the verifier and V-Protocol perform some actions on the business objects announcement and document. The business process implemented by V-Protocol is triggered by the arrival of a message with attached an announcement (each message exchange is depicted in the activity diagram by means of a dashed line). The starting message is sent by OP (see Fig. 3). The system interacts also with the verifier that checks the correctness of the document. In the end, V-Protocol executes some operations on the document DOC (e.g., `VProtocol.save(DOC)`) and next the process stops.

<sup>2</sup> We are aware that only send signal actions can be represented in this way, but here we use this icon also for call actions.



(a) Activity Diagram



(b) Class Diagram

Fig. 3. Protocol Case Study: DesignModel

In Fig. 3(b), we show the class diagram associated with the activity diagram of Fig. 3(a). The class diagram is used to type and specify the stereotype of each participant and to specify attributes and/or operations of each participant. Moreover, the class diagram can contain datatypes (e.g., DigitalSignature) used to type attributes or operation parameters. The activity diagram created at this level ought to be structured [7] because unstructured diagrams make difficult

the translation of business process models into executable models (e.g., written in BPEL [3]) that often offer structured-programming constructs only.

The models created following our method can be enriched at each phase with other UML diagrams or details that increase the level of information provided; for example in the class diagram in Fig. 3(b) we added some constraints.

### 3.4 Phase 4: V-System Detailed Design

The fourth and last phase produces a UML model called *ArchitectureModel* that depicts the system architecture.

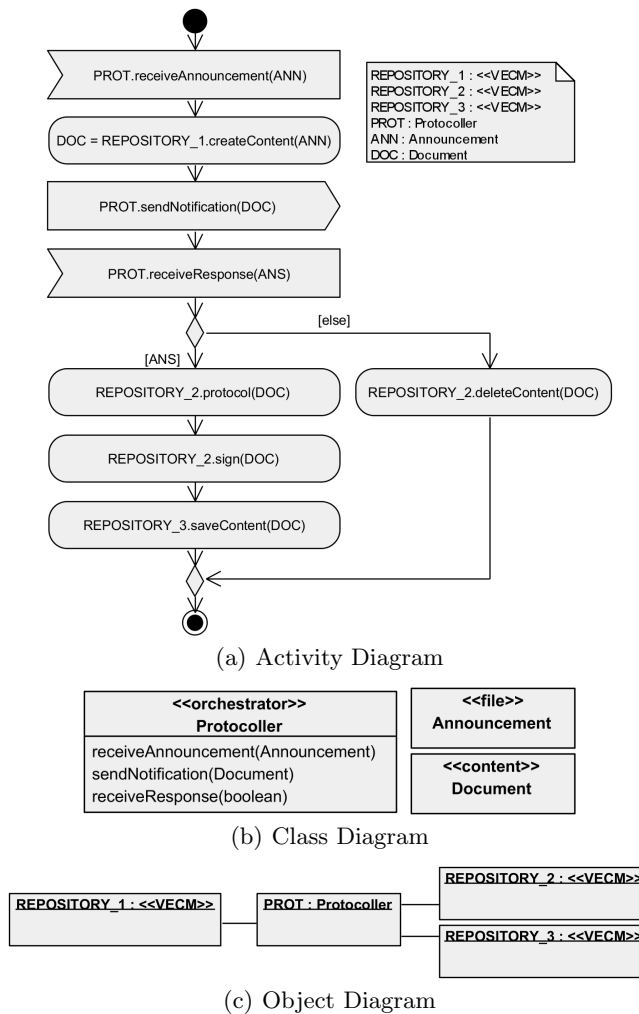


Fig. 4. Protocol Case Study: *ArchitectureModel*

The detailed design is given in terms of the subsystems that constitute the system. The subsystems can be: (1) one or more VECMs that abstract the underlying ECMs; (2) an *orchestrator* that coordinates the execution of the different VECMs and, if necessary, performs some data elaborations. Note that all the computations not assignable to the VECMs are done by the orchestrator. Moreover, the orchestrator manages the interaction with the outside participants (e.g., sending and receiving messages). At this level, for simplicity, the participants not included in the V-System swimlane are removed from the model. The attention is uniquely focused on the V-System architecture.

The subsystems that are typed by VECM can perform only a predefined set of operations (see Fig. 1). All the operations not supported by VECM have to be executed by the orchestrator and if it is not possible they have to be substituted by calls to external services and this require a modification of the V-System design.

In Fig. 4 we show the V-Protocol *ArchitectureModel*. The system is composed by four subsystems: one orchestrator (Protocoller) and three repositories of type VECM. All the operations performed by the repositories are included in the UML presentation of the VECM functionalities in Fig. 1. Note that the repositories do not communicate each other.

## 4 Lessons Learnt

Based on the experience that we have acquired during the development of our method, we summarize the lessons learnt and discuss opportunities for future research.

- It is possible to apply MD to build VECM-based applications without investing in complex tools and expensive training. It is sufficient a UML design tool (e.g., Visual Paradigm<sup>3</sup>) and a basic knowledge of UML. In the case of the VirtualECM project the involved companies had an adequate expertise for the execution of the method.
- Usually business process descriptions and models used in practice and given as starting point to develop a system are ambiguous, inconsistent, over-specific or too generic. Often, models given in “light style” format [6] seem very simple and easy to understand but often they contain subtle flaws that could bring to different interpretations and meanings. Using the “precise style” helps to reduce more common errors and flaws [5].
- The use of VECM is complementary to the use of SOA and not an alternative, given that they differ in the level of application. Indeed, VECM can be placed above a set of SOA-based ECM that though they have a SOA-based interface, the signatures of their operations are not standardized. For this reasons also in a SOA-based ECM environment it is useful to adopt VECM. Moreover, VECM exposes a SOA interface so it can be invoked as a web service.

<sup>3</sup> a UML modeller covering all kinds of UML diagram types. See <http://www.visual-paradigm.com/>

10 Maurizio Leotta, Gianna Reggio, Filippo Ricca, and Egidio Astesiano

- Methods for developing ECM based systems are difficult to find. Some tools that permit to integrate different ECMs exist (e.g., ECM integration layer of SAP NetWeaver or FusionEnterprise) but they are part of complex technology platforms. Instead VECM is a light interface that does not require complex and expensive solutions.
- The adoption of CMIS<sup>4</sup> [4] does not replace the use of VECM. Indeed, even if CMIS will reach in the future a great diffusion among ECM users, a lot of companies will still use obsolete ECMs.
- Currently, the two companies involved in the VirtualECM project implemented by hand the system as a web application starting from the V-Protocol Architecture Model. Future work will be devoted to automate this task.

## 5 Conclusion and Future Work

In this experience report we have presented: (i) an MD method useful to develop VECM-based systems and (ii) its application to the development of the V-Protocol case study. Preliminary applications of our method (as the one reported here) show its effectiveness.

Future work will be devoted to refine our method and test it with more complex real case studies. We also intend developing a tool able to assist the designer in the construction of VECM-based systems. The tool should automatically transform the orchestrator produced during V-System Architecture Design phase in executable code.

## References

1. F. Di Cerbo, G. Doderio, G. Reggio, F. Ricca, and G. Scanniello. Precise vs. Ultra-Light Activity Diagrams - An Experimental Assessment in the Context of Business Process Modelling. In D. Caivano, M. Oivo, M. Baldassarre, and G. Visaggio, editors, *Product-Focused Software Process Improvement*, volume 6759 of *Lecture Notes in Computer Science*, pages 291–305. Springer Berlin / Heidelberg, 2011.
2. S. Kent. Model driven engineering. In M. J. Butler, L. Petre, and K. Sere, editors, *IFM*, volume 2335 of *Lecture Notes in Computer Science*, pages 286–298. Springer, 2002.
3. OASIS. Web Services Business Process Execution Language, v. 2.0. Standard, 2007.
4. OASIS. Content Management Interoperability Services, v. 1.0. Standard, 2010.
5. G. Reggio, M. Leotta, and F. Ricca. Precise is better than Light - A Document Analysis Study about Quality of Business Process Models. In *Proceedings of 1st International Workshop on Empirical Requirements Engineering (EmpiRE 2011 co-located with RE 2011)*. IEEE Digital Library (to Appear), 2011.
6. G. Reggio, F. Ricca, E. Astesiano, and M. Leotta. On Business Process Modelling with the UML: a Discipline and Three Styles. Technical Report DISI-TR-11-03, Dipartimento di Informatica e Scienze dell'Informazione (DISI), Università di Genova, Italy, April 2011. [Online]: <http://softeng.disi.unige.it/tech-rep/TECDOC.pdf>.
7. M. H. Williams and H. L. Ossher. Conversion of Unstructured Flow Diagrams to Structured Form. *The Computer Journal*, 21(2):161–167, 1978.

<sup>4</sup> Content Management Interoperability Services is an OASIS specification for improving interoperability between Enterprise Content Management systems

# Empirical evaluation of the conjunct use of MOF and OCL

Juan Cadavid<sup>1</sup>, Benoit Baudry<sup>1</sup>, Benoît Combemale<sup>2</sup>

<sup>1</sup>INRIA, Centre Rennes – Bretagne Atlantique  
Campus de Beaulieu, 35042 Rennes Cedex, France  
{[juan.cadavid](mailto:juan.cadavid@inria.fr), [benoit.baudry](mailto:benoit.baudry@inria.fr)}@inria.fr

<sup>2</sup>IRISA, Université de Rennes 1 Triskell Team, Rennes, France  
{[benoit.combemale](mailto:benoit.combemale@irisa.fr)}@irisa.fr

**Abstract.** MOF and OCL are commonly used for metamodeling: MOF to model the domain structure, and OCL for the well-formedness rules. Thus, metamodelers have to master both formalisms and understand how to articulate them in order to build metamodels that accurately capture domain knowledge. A systematic empirical analysis of the conjunct use of MOF and OCL in existing metamodels could help metamodelers understand how to use these formalisms. However, existing metamodels usually present anomalies that prevent automatic analysis without prior fixing. In particular, it often happens that both parts of the metamodel (MOF and OCL) are inconsistent. In this paper, we propose a process for analyzing metamodels and we report on the pre-processing phase we went through on 52 metamodels in order to get them ready for automatic empirical analysis.

## 1 Introduction

The Meta-Object Facility (MOF) [2] and the Object Constraint Language (OCL) [4] are commonly used for metamodeling: MOF to define a domain model and OCL to define well-formedness rules in this domain. At first glance, the roles of both standards seem well-delimited, yet many conceptual decisions can be implemented in either one, as we will demonstrate. For this reason, their combined usage has revealed several styles as observed in the panorama of developed mature DSMLs (Domain Specific Modeling Languages). Systematic empirical analysis of how these standards are combined in publicly available metamodels would help understand their usage and propose new methodologies and techniques to assist domain experts when building a new metamodel. Empiric analysis can be established through the systematic collection of metrics over existing metamodels. However, there currently exists no metrics that relate MOF and OCL and there exists no tool to automatically compute metrics on metamodels. Another issue is related to the lack of homogeneous formats to support the automatic analysis of MOF/OCL based metamodels. For example, OCL well-formedness rules are provided in various formats (txt, annotations in Ecore, OCL model). Also, because MOF and OCL parts are not always stored in the same format, both parts



of the metamodel tend to be inconsistent. When gathering data for empirical analysis, it is thus necessary to fix it prior to metrics computation. This paper aims at illustrating the challenges of the conjunct usage of MOF and OCL for metamodeling. We propose initial metric definitions and analysis methodology to empirically understand how both formalisms are related and conjunctly used in existing metamodels. We have collected 52 metamodels for which we have learned a few initial lessons by manually analyzing and fixing them in order to get them ready for automatic measurement. In particular we have found and fixed a number of inconsistencies in OCL invariants defined in OMG (Object Management Group) standard metamodels. Section 2 presents the motivation for this study. Section 3 introduces the problem statement, introduces definitions as well as the presentation of the two standards. Section 4 presents our research process. Section 5 and 6 present the first data findings and conclusions relevant to the first phase of our research process.

## 2 Motivation

This section illustrates the issues for the definition of a correct and precise metamodel through an example. The model in figure 1, expressed in the basic version of MOF called EMOF (Essential MOF), specifies the concepts and relationships of the Petri net domain. A *PetriNet* is composed of several *Arcs* and several *Nodes*. *Arcs* have a source and a target *Node*, while *Nodes* can have several incoming and outgoing *Arcs*. The model distinguishes between two different types of *Nodes*: *Places* or *Transitions*.

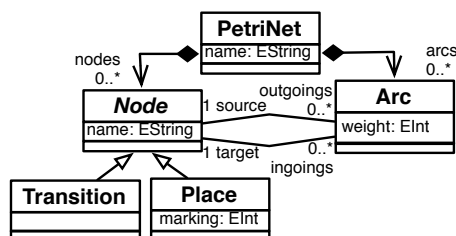


Fig. 1. MOF-based domain structure for Petri nets

This model captures every necessary concept to build Petri nets. However, there can also exist valid instances of this model that are not valid Petri nets. For example, the model does not prevent the construction of a Petri net in which an arc's source and target are only places, instead of linking a place and a transition. Thus, the sole model is not sufficient to precisely model the specific domain of Petri nets, since it still allows the construction of models that are not valid in this domain. The model needs to be enhanced with additional properties to capture the domain more precisely. The following well-formedness rules, expressed in OCL, show some mandatory properties of Petri nets.

1. *noEqualNamesInv*: Two nodes cannot have the same name.

```
context PetriNet inv :
  self.nodes->forAll(n1, n2 | n1 <> n2
implies n1.name <> n2.name)
```

2. *noSameEndTypesInv*: No arc may connect two places or two transitions.

```
context Arc inv: self.source.oclType()
<> self.target.oclType()
```

3. *placeMarkingPositiveInv*: A place's marking must be positive.

```
context Place inv: self.marking >= 0
```

4. *arcWeightPositiveInv*: An arc's weight must be strictly positive.

```
context Arc inv: self.weight > 0
```

In our study we consider that the metamodel for Petri nets is the composition of the model and the associated well-formedness rules. We learn from this example that the construction of a precise metamodel, that consistently captures a domain, requires: (i) mastering two formalisms<sup>1</sup>: EMOF for concepts and relationships; OCL for properties; (ii) building two complimentary views on the domain model; (iii) finding a balance between what is expressed in one or the other formalism, (iv) keeping the views, expressed in different formalisms, consistent. This last point is particularly challenging in case of evolution of one or the other view. One notable case from the OMG and the evolution of the UML standard is that the `AssociationEnd` class disappeared after version 1.4 in 2003, but as late as version 2.2, released in 2009, there were still OCL expressions referring to this metaclass [11]. In the same manner, the OCL 2.2 specification depends on MOF 2.0, however a particular section of the specification defining the binding between MOF and OCL [4, p.169] makes use of the class `ModelElement` which only existed until MOF 1.4.

### 3 Research problem

#### 3.1 Definitions

This section defines the terms we use to designate the focus of our analysis. The relationship between a model and a metamodel can be described as shown in figure 2 [3]. Here the `conformsTo` relation is a predicate function that returns true if all objects in the model are instances of the concepts defined in the metamodel, all relations between objects are valid with respect to relationships defined in the metamodel and if all properties are satisfied.

<sup>1</sup> One can notice that some properties could have been modeled with EMOF by choosing another structure for concepts and relationships. However, the number of concepts and relationships would have increased, hampering the understandability of the metamodel and increasing the distance between the metamodel and a straightforward representation of domain concepts.

4 Juan Cadavid, Benoit Baudry, Benoît Combemale

**Definition 1. Metamodel.** A metamodel is defined as the composition of:

- **Concepts.** The core concepts and attributes that define the domain.
- **Relationships.** Relationships that specify how the concepts can be bound together in a model.
- **Well-formedness rules.** Additional constraints that restrict the way concepts can be assembled to form a valid model.

In this study, we consider metamodels defined with OMG standards. We distinguish two parts as defined below.

**Definition 2. Metamodel under study.** For this work, a metamodel is defined as the composition of:

- **Domain structure.** An EMOF-compliant model portraying the domain concepts as metaclasses and relationships between them.
- **Invariants.** Well-formedness rules that impose invariants over the domain structure and that are expressed in OCL.



**Fig. 2.** Model & MetaModel Definition with Class Diagram Notation

### 3.2 Summary of EMOF and OCL

Figure 3 displays the structure of EMOF [2]. EMOF allows to specify the concepts of a metamodel in a `Package`. This `Package` contains `Classes` and `Propertyts` to model the concepts and relationships. The `Propertyts` of a `Class` can be either: attributes of type `Boolean`, `String` or `Natural`; or references to other `Classes`, in this case the `Property` is of type another `Class`. Figure 4 displays the structure of OCL expressions [4] that can be used to constrain the structure defined with EMOF. The most noticeable constructs for OCL expressions are: the ability to declare `Variables`, whose type is a concept modeled with EMOF; the ability to use control structures such as `IfExp` and `LoopExp`; the ability to have composite OCL expressions, through `CallExps`. Figure 5 displays the connection between EMOF and OCL [4, p.169]. This figure specifies that it is possible to define `Constraints` on `Elements` (everything in EMOF is an `Element`, cf. figure 3). They can be defined as `Expressions`, and one particular type of expression is `ExpressionInOCL`, an expression defined with OCL. The most important concept is the notion of `ExpressionInOCL` that binds an `Element` coming from an EMOF model on one hand to an `OclExpression` on the other hand. The existence of this binding between formalisms is essential for metamodeling: this is how two different formalisms can be smoothly integrated in the construction of a metamodel. This binding is also what allows us to automatically analyze metamodels built with EMOF and OCL.

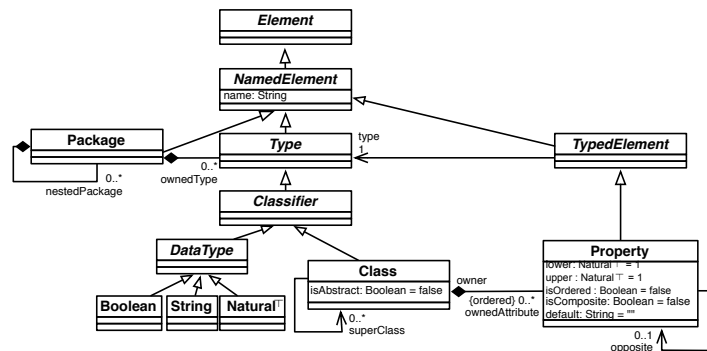


Fig. 3. The EMOF Core with Class Diagram Notation

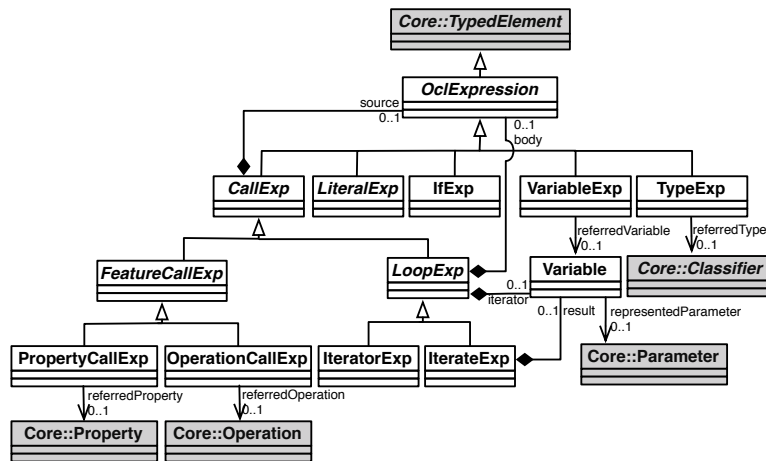


Fig. 4. OCL Expression metamodel

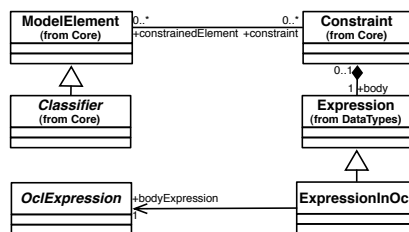


Fig. 5. OCL and MOF binding

6 Juan Cadavid, Benoit Baudry, Benoît Combemale

### 3.3 Metrics

In order to understand the conjunct usage of these two standards, we aim at defining the following metrics.

- Size of the metamodel: The number of constructs that a metamodel provides can change dramatically from one language to the other. Such measure has to be compared when evaluating several metamodels from diverse domains.
- Size of the invariants set: Metamodels can portray different levels of complexity; highly complex domains require a large number of OCL formulae to express their logic, whereas lesser complex domains will express their knowledge with a lower number of constraints. With this metric we aim at understanding the different levels of such complexity.
- Invariant complexity with respect to the underlying domain structure: Some metamodels contain lengthy and complex well-formedness rules, while others seem to define them using simple expressions. We measure how many EMOF elements are used in each OCL invariant and thus the quantity of model elements involved in a constraint.
- Invariant complexity with respect to the OCL syntax metamodel: In order to extract the effective subset of the OCL language that is used in DSML development, we intend to query the invariants for the specific OCL expression types they use.

## 4 Analysis of MOF and OCL in metamodels

The data sets and metrics specified in the previous section are used to build a tool to perform the computations which will provide the data to perform analysis on a metamodel.

### 4.1 The Global Process for Analysis Automation

Figure 6 shows the overall process to analyze a metamodel. The process is composed of three activities with their own tools:

1. If the OCL invariants are not defined in the OCL/XMI format (extension `.oclxmi` in figure 6), the first activity consists in preprocessing (activity **OCL Parsing**). It is performed depending on the input format of the OCL invariants (extension `.ocl` in figure 6). We have used `OCLINECORE`<sup>2</sup>, a textual editor for Ecore files.
2. The next step consists in using an in-house built tool to automatically compute the metrics over the metamodel (activity **Metrics Computation**). Such tool would take as an input the metamodel composed of the domain structure expressed in Ecore, and the invariants expressed in OCL and produce a CSV file containing all the metric values for the input metamodel.

<sup>2</sup> `OCLINECORE`, cf. <http://wiki.eclipse.org/MDT/OCLinEcore/>.

3. The metric values are finally analyzed with R<sup>3</sup> (activity **Statistical Analysis**). R is an open-source language for statistical applications, which provides several functionalities to run analysis and create plots, both one-variable and multi-variable. We provide a set of generic scripts that could be used for any CSV file produced with our in-house built tool. These scripts automate the production of graphics to assist analysis.

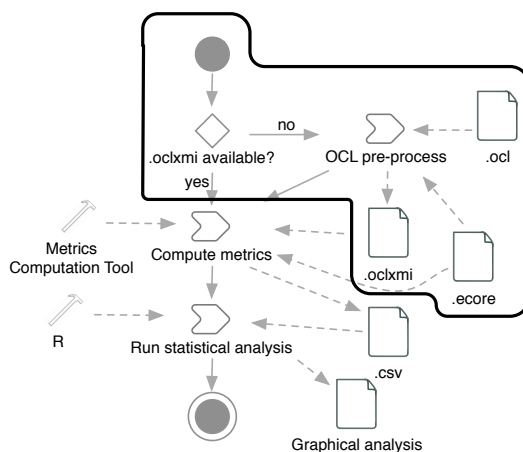


Fig. 6. SPEM Process for Metamodel Automatic Analysis

Currently our research has accomplished the encircled parts of the diagram, performing the preparation of the data of the metamodels presented in the next section.

## 5 Data setup and preprocessing

Understanding the use of EMOF and OCL requires a sample containing data from repositories in diverse backgrounds. The sample subjects must come from standard bodies, academia and industry altogether.

### 5.1 Experimental data setup

Table 1 details a list of standard specifications coming from different sources, defining domain-specific languages. The first two columns contain the name and source; the first group comes from the OMG. The following group presents metamodels taken from academic research; first a metamodel for the B language created at IMAG; SAD3 is a software architecture component model created at

<sup>3</sup> R, cf. <http://www.r-project.org/>

ENSTA Bretagne. In the last group, metamodels MTEP and XMS are metamodels created by Thomson Video Networks for encoding standards for video hardware. SAM is a metamodel from the Topcased open source software project. The third column counts the number of metamodels. In the OMG group, specifications define large modeling languages, normally structured in packages, therefore we treat each one of these as a separate metamodel. In the remaining cases, each specification contains only one metamodel. The fourth column mentions the formalism used to express well-formedness rules. As expected, we chose specifications using OCL. The fifth column shows the different standards that exist to specify the domain structure. Of our main interest, Ecore is a lightweight implementation of EMOF [5], providing equally an XMI-based persistence mechanism. The sixth column presents the format for expressing invariants in OCL. These are found either as separate .ocl text files or embedded in .ecore as annotations. Availability of the Ecore format and some of the mentioned forms of OCL invariants are necessary to enable the automation of the metrics computation.

**Table 1.** Specifications containing sample metamodels.

Name	Source	Meta-models	Expression of Constraints	Domain Structure format	OCL invariants format
UML	OMG	13	Text and OCL	Ecore	Annotations in Ecore
CCM	OMG	4	Text and OCL	Ecore	Text in documentation
OCL	OMG	4	Text and OCL	Ecore	Text in documentation
MOF	OMG	2	Text and OCL	XMI	.ocl text file
CWM	OMG	21	Text and OCL	Ecore	Text in documentation
DD	OMG	3	Text and OCL	XMI	Annotations in Ecore
B language	Academic Research	1	Language specification and OCL	Ecore	.ocl text file
SAD3	Academic Research	1	OCL	Ecore	.ocl text file
MTEP	Industry	1	OCL	Ecore	.ocl text file
XMS	Industry	1	OCL	Ecore	.ocl text file
SAM	Industry	1	OCL	Ecore	.ocl text file

## 5.2 Preprocessing data for analysis

The preprocessing step is based on an automated Java program that takes an Ecore/XMI metamodel with associated OCL invariants stored in their available format for each metamodel and its OCL invariants, according to table 1 and produces as output an Ecore/XMI metamodel with OCL/XMI invariants, where

all the OCL invariants that remain are syntactically correct (parse without errors using the Eclipse OCL parser [1]). The OCL/XMI format presents the abstract syntax tree of each OCL expression. At the end of the preprocessing step, every metamodel can be automatically analyzed for metrics computation. The metrics computation tool will be able to compute metrics on the MOF structure and the OCL invariants, and consequently analyzable data is emitted as output. Throughout this process we have observed the following issues.

**Different storage formats** Ecore is the de-facto standard based on the XMI format used to express the domain structure of a metamodel, yet there is no evidence of such a format to store OCL expressions for a metamodel. Besides OCL text files, invariants are also added as annotations; however these only consist of maps of string-to-string entries, which can themselves present different schemas. Our preprocessing program automatically detects the format and proceeds to parse and produce the previously mentioned output.

**Different OCL syntaxes** Different parsers allow or reject certain OCL constructs [7]. To enable automation analysis of the OCL expressions, such variations must be streamlined to satisfy the precise syntax required for Eclipse OCL; this was performed by replacing the unrecognized constructs by its accepted equivalents; for example, the use of the minus “-” operator to exclude elements from a collection, instead of the *exclude* operation.

**Errors in invariants** In many cases, OCL invariants are added to a metamodel with the sole purpose of documentation and might not be checked for correctness. The studied sets of invariants from the selected specifications contained incorrect OCL expressions, containing errors from syntax (invalid use of OCL constructs) or semantics (references to non-existent model elements from the domain structure). Table 2 presents trivial errors and thus capable of being corrected, as well as those that could not be fixed, since it would require knowledge from the domain expert.

## 6 Conclusions

In this article we have illustrated several issues that arise when metamodeling with the MOF and OCL formalisms. Our purpose is to learn how these formalisms are used in existing metamodels, in order to assist metamodelers in the future. The rest of the paper discusses a set of metamodels that we have gathered from different sources (OMG, open source project, industry) and the lessons we have learned while manually analyzing and fixing these metamodels to get them ready for automatic analysis. Most of the problems to automate the analysis over the metamodels are that OCL well-formedness rules first are provided in a variety of formats and secondly are often inconsistent with the MOF domain model. The next step for this work is to build a tool that can automatically analyze metamodels. This tool should compute a set of metrics about the



**Table 2.** Corrected errors in OCL invariants.

Corrected errors	Frequency
Missing parenthesis	94
Notation for enumeration literals	51
Missing variable in forAll body	30
Missing mandatory typecast (oclAsType())	22
Typos in pointers to metaclasses and properties	15
Typos in OCL operations invocation	13
Use of '->' instead of '.' for non-collection properties	10
Use of '.' as a shortcut for 'collect'	9
Use of unescaped OCL keywords	6
'if' expression without 'else' and 'endif'	5
Use of 'notEmpty' and 'isEmpty' for non-collection properties instead of oclIsUndefined()	4
Treating of boolean values as literals '#true' and '#false'	3
Use of 'union' instead of 'concat' to concatenate strings	2
Errors remaining incorrect	Frequency
Pointers to nonexistent properties/operations	133
Invariants with a context metaclass in an outside metamodel	2
Reference to undefined stereotypes	1

conjunct use of MOF and OCL. These metrics will be the basis for our empirical investigation. Such empirical work will lead to complement existent guidelines for metamodelers [8, 9, 6, 10] suggesting an appropriate use of MOF and OCL.

## References

1. Eclipse ocl, <http://www.eclipse.org/modeling/mdt/?project=ocl>
2. Omg meta object facility core, v2.0 (2006)
3. A Framework to Formalise the MDE Foundations. In: TOWERS. pp. 14–30 (2007)
4. Omg object constraint language, v2.2 (2010)
5. Budinsky, F., Merks, E., Steinberg, D.: Eclipse Modeling Framework 2.0. Addison-Wesley Professional (2009)
6. Garcia, M.: Efficient integrity checking for essential mof+ ocl in software repositories. *Journal of Object Technology* 7(6)
7. Gogolla, M., Kuhlmann, M., Büttner, F.: A benchmark for ocl engine accuracy, determinateness, and efficiency. In: MoDELS. LNCS, vol. 5301, pp. 446–459 (2008)
8. Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., Völkel, S.: Design guidelines for domain specific languages. In: The 9th OOPSLA workshop on DSM (2009)
9. Kovari, P.: Explore model-driven development (mdd) and related approaches: Applying domain-specific modeling to model-driven architecture. IBM Developerworks (2007)
10. Loecher, S., Ocke, S.: A metamodel-based ocl-compiler for uml and mof. *Electron. Notes Theor. Comput. Sci.* 102, 43–61 (November 2004), <http://dx.doi.org/10.1016/j.entcs.2003.09.003>
11. Selic, B.: Uml 2 specification issue 6462. <http://www.omg.org/issues/issue6462.txt> (2003), updates dating until 2008.

