Electronic Communications of the EASST
Volume 52 (2011)

7th Educators' Symposium @ MODELS 2011: Software Modeling in Education (EduSymp2011)

Position Paper: Software Modelling Education

Martina Seidl, Peter Clarke

10 pages

Guest Editors: Marion Brandsteidl, Andreas Winter
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Position Paper: Software Modelling Education

Martina Seidl[1,2]*, **Peter Clarke**[3]

[1]Institute for Formal Models and Verification, Johannes Kepler University, Austria,
[2]Business Informatics Group, Vienna Technical University, Austria,
[3]Florida State Univiersity, USA

Abstract: Model-driven engineering (MDE) is a promising paradigm to deal with the ever increasing complexity of modern software systems. Its powerful abstraction mechanisms allow developers to focus on the essential challenges hiding away irrelevant aspects of the system under development. Within the last few years, noticable progress has been made in putting the vision of MDE into practice, where the activity of textual coding is substituted by modeling. With matured concepts and stable tools available, MDE becomes more and more ready to be applied in software engineering projects.

Nevertheless, the best available technology is worthless, if it is not accepted and used by the developers. Also in MDE profound training is needed to fully exploit its power. In this paper, we discuss the efforts taken in educational environments to promote the application of modeling and MDE technologies for the software development process and discuss several challenges which still have to be faced.

Keywords: Software Modeling Education, Computer Science Education Research

1 Introduction

The way how large software systems are systematically developed has been steadily improved over the last decades in order to deal with the increasing size and complexity of such systems [Boe06]. One important driving force for the progressive advancements achieved in the discipline of software engineering is the concept of *abstraction* [Jac06] as it can be observed in the step from machine language to assembly, from assembly to structural high level languages which finally paved the way to modern object-oriented languages [GJ08]. The object-oriented paradigm aims at representing domain knowledge about the real world in a manner which is both natural for the human and easy to process for the computer. Hand in hand with the successful application of object-oriented programming languages goes the introduction of object-oriented analysis and design methods (e.g., [Boo86, Jac92, RBP+91]) relying on the notion of software models expressed in languages like the *Unified Modeling Language* (UML) [BM99]. However, the gap between the design of a software system and the actual realization introduces the problem of co-evolution. When both tasks—modeling and coding—are performed, an additional source for errors is introduced, because both sets of artifacts have to be kept in a synchronous state during the complete software life cycle. In consequence, the solution is to merge modeling and

* Parts of this work have been funded by the Vienna Science and Technology Fund (WWTF) under grant ICT10-018.

programing and to provide the means to generate executable code automatically from models as proposed by model-driven engineering [Sch06].

Based on this idea, modeling has experienced an enormous momentum not only in academia but also in industry. With the concretization of the vision of *model-driven engineering* (MDE), models are lifted from mere documentation artifacts to full first class citizens in the software development process offering a tool which is expected to change the way how software is developed drastically. Still, it is a far way to go before the complete software development engineering process can be cover by according techniques. In fact, it is not a revolution that from one day to another traditional coding is not any more, but slowly the new model-driven methods are incorporated to support programmers doing their work. Since developing software according to the MDE paradigm is different than traditional software engineering, programming knowledge and experience cannot directly be transfered to modeling [Fra09] and novel training methods are indispensable.

With this fast evolution of the development methodology, academia is confronted with a severe problem. On the one hand, the students shall be offered profound knowledge on well-established technologies and approaches, on the other hand they shall have a competitive knowledge on the state-of-the-art when they leave university. So the question educators are confronted with is to judge which approaches have a longer perspective of practical application and which ones are only interesting in a short term perspective and will be forgotten when the first hype is over. This challenge is certainly not new for people who are responsible for the development of curricula for studies on computer science. Still there are several areas like theoretical computer science which are relatively stable, and other ones which are still experiencing an extreme progress like in the area of software development. Novel technologies can only be applied if people are available knowing how to handle them, i.e., how to use them properly, because otherwise the benefits are quickly relativized and additional complexity is introduced instead of achieving an ease of work.

In this paper, we give an overview about the activities necessary to train developers effectively in modeling. Therefore, we first consider the different of areas which cover the wide spectrum of modeling. Then we review the actions taken in the Edusymp, the premier forum for educators working in the field of modeling. Finally, we derive a set of challenges which have to be tackled in modeling education.

2 Software Modeling Education

Software modeling education is a very broad topic. The range of teaching areas is very heterogeneous when the full spectrum of modeling shall be covered and a general view of the application possibilities shall be given to the students. Since there is not only one phase in the software development process, but several which depend on the abstraction power of models, deep understanding of how to take advantage of the application of models is required. Without proper education, the danger is very high that models are either not applied in the correct manner, causing more costs then achieving benefits, or they are not used at all, because developers who are trained in traditional programming consider modeling only as additional, even useless task. In the following, we discuss topics which shall be covered by a curriculum in order to assure that models are applied in the complete software development process.

Introduction to Modeling Languages. Before students are able to apply modeling techniques in the context of practical software engineering projects, they have to understand the concepts of modeling languages, i.e., they have to gain profound knowledge on the syntax and semantics of modeling languages. For this first step, several different approaches can be taken [EHLS06]. In fact, this shows some similarities with programming education, where a big point of discussion is the language (or even the language paradigm) with which beginners shall be confronted. Modeling may be applied for different purposes like for the description of software systems, for the description of business processes, database design, etc. One language which covers a very wide spectrum is the Unified Modeling Language (UML), a general purpose language which was developed with the goal to support as many users' needs as possible. Therefore, UML is extremely flexible with respect to not only the provided syntactical concepts but also with their semantics defined over numerous variation points to adapt UML to specific purposes. This overflow of concepts and this unsharpness of the standard usually is a great challenge for beginners who have to deal not only with a new material, but also with context sensitive concepts. A solution to this dilemma would either be the education of UML to a set of well-defined core concepts as proposed in [SHMA08], or to start with a Domain Specific Language (DSL), where the students get a more focused introduction.

Dealing with Abstractions. One major issue about modeling is to comprehend how to abstract. With modeling techniques it is possible to focus on aspects of the systems which are relevant and to neglect these which may be omitted for the moment. With this complexity of huge systems can be grasped in a more direct way, because otherwise it would quickly overwhelm the human intellect. In fact, already any programming activity demands the ability to abstract from reality, but the way how it is abstracted is different than in modeling. In programming, abstraction is necessary in order to describe a problem in such a way that it can be stated by the means of the used of the applied programming language such the problem become processable by a computer. Modeling languages, in contrast, provide the means to describe the world as it is seen by humans, the step to executable code comes much later. Therefore, novices are easily tempted to put everything into a model or mix the levels of abstractions, making the model harder to read, harder to use and harder to maintain. Therefore modelers have to be trained to choose the applied level of abstraction with care [Rob09].

Practical Application. Software modeling is usually done with the intention to build a software systems which then is practically used by the end users. For the educations, this implies it is not sufficient to learn how to model on paper only, but to put the modelling skills into practice in terms of realistic projects. With this experience, the students are prepared for working in industry and to realize real software projects. Models can be applied in many different parts of the software development life cycle, and the best way of learning is doing it with hands-on-experience. In a first step, special projects tailored for didactical purposes shall be performed by the students, in later phases projects with industrial partners under the supervision of an experienced trainer are certainly the best preparation for future jobs.

Model-Driven Engineering. Models can not only be used as design artifacts, but also as substitution of traditional textual code. Then coding activities are replaced by modeling activities. The executable system is then (semi-)automatically derived from the models. This way of building software requires a new way of thinking from the developers, because again the level of abstraction is shifted. Now only limited knowledge about computer architecture is necessary, because such optimizations are performed by the code generators. In order to generate functional code, good understanding of the MDE tools is necessary, and in some cases customizations of the generated code may become necessary for realizing the required functionalities. If the code is also modified by hand, techniques supporting the co-evolution of code and models have to be applied, otherwise modifications get lost when the system is rebuild. Furthermore, since MDE techniques are at a relative early stage, a profound understanding is necessary for deciding when the application of which software development paradigm is more appropriate.

Model Engineering. For applying the techniques of MDE special environments are required. With the Eclipse modeling framework, a lot of tools are available out of the box, but for customization purposes, it may be beneficial to develop dedicated code generators or to implement model transformations which support the conversion between different modeling languages or rewrite certain modeling concepts. Therefore, education in model engineering may not be neglected. With a deep understanding, how the applied tools work—which is certainly more easily gained when students develop small MDE environments on their own (as proposed in [BKSW09]) —it is also easier to apply available tools in the correct manner. Also concepts like metamodeling and metametamodeling become more natural when a one has designed and implemented a modeling language and the behavior of code generators is easier to understand when they have been used.

Ideally, industry and academia collaborate for training and education. Whereas industry could provide practical application scenarios to the university students giving them both a good motivation as well as valuable experiences, the theoretical background could be directly promoted from research institutions to people working at companies offering the state-of-the-art knowledge.

3 The Educators' Symposium

The Educators' Symposium (Edusymp) is an annual event collocated with the International Conference on Model-Driven Engineering (MODELS)[1]. The goal is to provide a platform where modeling educators meet, exchange experiences, and develop new approaches for promoting the MDE paradigm in education. In this section, we first outline the typical organization of this symposium by analyzing the former six editions as far as the data is available. Each Edusymp has the focus set on a special issue and some peculiar organizational highlight. We exemplarily review the last edition of the Edusymp 2010, held in Oslo, Norway. This analysis of the Edusymp allows us to identify issues which have not been handled so far and to derive challenges which shall be considered in future editions.

[1] http://www.modelsconference.org

3.1 Organization

The Edusymp is always held as a satellite event of the MODELS conference, so it is usually in parallel either with tutorials and with workshops and may use the infrastructure of the conference. The Edusymp has no budget on its own, so no funding for speakers is available. In order to successfully hold the symposium, a couple of people are necessary for the organization as discussed in the following.

Organizers. The Edusymp is organized by one or two researchers/educators working in the modeling area. The organizers of the next edition are usually announced at the closing session of the MODELS conference, so the designation usually takes place during the Edusymp. Usually very active participants are invited to do the organization for the next year. The tasks include the formulation of the call for papers, the advertisement of the call for papers, the selection of the program committee members, the guidance of the reviewing process, the notifications of the authors, etc. In short, the organizers have the typical responsibilities as any chairs of an academic event. In the past six editions of the Edusymp, the organization has been done by one or two professors or associate professors. Additionally, assistants have been involved in some years. So far, no two editions of the symposium have been organized from the same person indicating that there is high interest in the community for such an event.

Program Committee. The diagram on the left in Figure 1 shows the ratio between people from industry and from academia. The majority of the people are working at universities which is on the one hand no surprise, because reviewing activities are the daily business of academic researchers. On the other hand, many calls of the Edusymp foster investigations on the synergy between industry and academia in education and therefore, opinions from people of a practical background would be extremely valuable. The diagram on the right of Figure 1 indicates the distribution of the program committee members. In general, people from Europe are dominating. Here it has to be mentioned, that it rarely happens that more than two people from the same university are involved. The PC members from America are not only from the US, but also from Canada and countries of South America like Argentina and Brazil. Since the modeling research efforts are taken all over the world as it is shown the participants of the main conference and software engineering is part of almost all computer science curricula no matter at which university it is located, a higher distribution heterogeneity should be possible and is preferable because then different viewpoints could be collected.

Participants. On the participants, no statistical data is available. From our experience we can conclude that the Edusymp is mainly visited by the people who actively contribute to the event, i.e., who present papers (cf. next subsection). Furthermore, there is a small core of professors who regularly attend the event. For parts of the symposium where well known speakers are invited or for the panels with renowned researchers discussing, the number of participants increases drastically.

Agenda. The Edusymp is a full day event usually consisting of four sessions. In the past, the symposium usually started with an invited talk, followed by the presentations of the accepted pa-

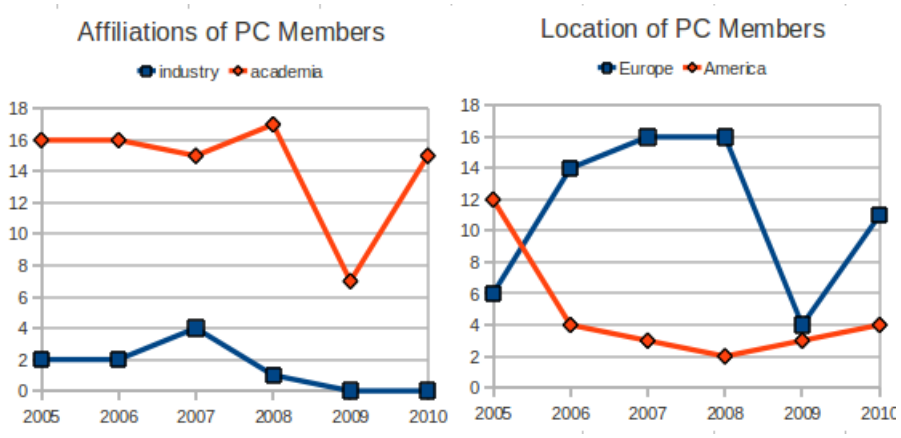

Figure 1: Statistics on the program committee.

pers, both of short and long papers. Short discussions with the authors are possible. Highlighting the high interactivity of the symposium, either panels or working group or both are organized offering room for discussing pedagogical related issues with other experts.

3.2 Contributions

The presentations of the peer-reviewed papers constitute an important part of the Edusymp. In the following, we review the topics for which contributions are invited, the topics of the actual contributions, and finally we analyze who submits the papers to the Edusymp in order to get an impression on the audience.

Call for Papers. In general, the Edusymp is intended to provide a forum for educators and trainer to exchange approaches and experiences on pedagogy for modeling education, experiences with various technologies and tools including

- experience reports and case studies on teaching modeling in academia and industry
- synergy between industry and academia in modeling education
- teaching methodologies for modeling education
- approaches to teach the tool support
- integration of modeling in the computer science curricula

In general, the accepted papers are published in terms of technical reports, which provides a way to archive the contributions. In order to increase the visibility, the electronic journal of the EASST[2], an indexed, open access journal, offers a publication channel since the last Edusymp. The best two papers as well as a short summary are included in the postproceedings of the MODELS.

[2] http://journal.ub.tu-berlin.de/eceasst

	2005	2007	2008	2009	2010
A: Object-Oriented Modeling and UML	3	1	4	2	3
B: Modeling and Software Engineering	3	1	1	1	1
C: Model Engineering Techniques	0	0	1	1	2
D: Abstraction, Design Patterns, Formal Methods	2	1	0	1	0
E: Research on Modeling Education	1	1	2	0	0
F: Collaboration between Industry and Academia	0	0	1	0	1

Table 1: Types of contributions.

Topics of Papers. Table 1 summarizes the types of contributions presented at the various editions of the Edusymp. Please note that for the year 2006 no list of accepted papers was available. The first three categories (A, B, C) contain the papers on descriptions and experiences on courses and curricula, whereas the later three categories (D, E, F) cover the papers on general methodologies and research on modeling education. Experience reports pose the majority of the presented contributions, where lecturers report on how they do the teaching. Only two papers discuss collaboration between industry and academia in modeling education. Topics like tool support are treated very sparsely. Interestingly, in first years of the Edusymp almost all papers were about UML related issues which changed over the years. This might be seen as an indicator how trends in research also influence education.

Authors. The geographical distribution of the authors is shown in Table **??**. The majority of the authors are from Europe, even if the symposium is hosted by a non-European country. Since the organizers and the PC members are mainly from Europe, maybe the promotion is better in Europe then in other parts of the world. If between 10 and 15 papers are submitted to the Edusymp, then it might be considered as a success which might be easily beaten by good workshops. The problem is that the educators do not have a specific subcommunity in the modeling community as it is the case for example for model evolution, model verification, etc. Somehow everyone is involved in teaching, but only very few feel obligated to push education research. It also happens not very often that the same author submits more than once to the Edusymp.

3.3 Case Study: Edusymp 2010

The first highlight of the Edusymp 2010 was the room filling keynote “Formality in Education—Bitter Medicine or Bitter Administration” held by Thomas K”uhne where it was illustrated very vividly how an inadequate presentation of formal methods frightens off students. In the following sessions seven papers have been presented covering innovative education approaches. Especially remarkable was the presentation of[BWH11] where the usage of novel media like document cameras in modeling education was shown in a live demo.

A online survey conducted in the prefield of the Edusymp served as the input for the interactive afternoon sessions consisting of working groups and a panel. The survey consisting of 18 questions was launched two weeks before the symposium and promoted via relevant mailing lists. More then 60 persons participated stating how and when they teach modeling. We used

	2005	2007	2008	2009	2010
Europe	3	4	8	4	5
America	5	0	0	1	0
Asia	0	0	1	0	2
Australia	1	0	0	0	0

Table 2: Geographical distribution of paper authors.

the answers for the categorization discussed in Section 2. Most of the participants use UML or Ecore in their courses, but also EER, BPMN and divers DSLs were named. The dominating tool in modeling education is the Eclipse platform. It showed that educators are in general happy with the tool support but there are several points of improvements like poor usability, poor support of standards, poor interchange facilities, etc. This issues gave a good starting point for the discussions led during the Edusymp. Details can be found in the summary [SC11]. Finally, a steering committee was established in order to watch and influence the development of the Edusymp.

4 Challenges

We conclude this paper with a set of challenges which are derived from the previous chapters. These challenges will have to be handled in order to promote modeling education properly.

Promotion. Unfortunately, teaching has not the same impact as research. Concerning the career in academics, papers on new research results are highly renowned and are important milestones for the CV of an individual researcher. For the teaching, on the other hand, it is sufficient to have a list of courses where some involvement has been taken, so no evaluation criteria are available for this thing. This is also the reason why teaching sometime comes very short and is often passed to junior researchers. In consequence, courses are often repeated and innovations are kept at a minimum in order to reduce effort necessary to spend on teaching. This is also one thing the Edusymp has to fight with. Although the majority of the MODELS' attendance are involved in teaching, only very few people participate in the Edusymp. Discussions on teaching are kept very private and so no community collected knowledge is possible. Often very dedicated workshops are better attended then the Edusymp which on should expect to be of interest for a broad audience.

Repository of Teaching Artifacts. One of the most time consuming tasks in course preparation, is the design of exercises and questions for the tests. The time for this task can be drastically reduced if good samples are available which have to be adapted to the current context only. Over the years individual teachers obtain huge local repositories with their exercises, for example if an exam has to be offered six times per year.

Archivation of Knowledge. One of the highlights of the Edusymp are the interactive sessions with working groups, panels, and open discussions. There, the participants discuss teaching

related topics and develop novel strategies. Unfortunately, the ideas and thoughts are hardly collected, usually only a short summary is provided by the organizers published on the three pages included in the collection of the postproceedings of the MODELS satellite events. In this way, the results of the Edusymp get lost and interesting ideas are never put into practice. To encounter this problem, it would require to have precise protocols which are stored in a central place which are accessible

Dedicated Tool Support. One of the hugest challenges educators as well as students are confronted with is the handling of the tools. In software engineering hands-on-experience is of particular importance in order to understand the techniques applied to build software of high quality. Since modeling is a quite young research area, tools work often well enough to use them in interesting projects, but the user has to know how to treat them. So the tools introduce additional complexity which shift the attention away from the actual problems to solve. Furthermore, the tools offer often too many features distracting the attention from the essential parts and the documentation is often insufficient. For the teachers the support is very hard to provide. Therefore, it would be nice to have dedicated tools support for education where the environment is tailored and configured to the specific needs of the specific course. This can be partly be achieved by providing complete Eclipse bundles.

Evaluation Criteria. The output of teaching activities are very hard to evaluate and to compare. In the papers on courses, typical measures are the grades of the students, some questionnaires where the students had to provide some feedback, and very often the subjective impression of the teacher. In order to seriously obtain some results, work has to be spend on the way how the quality obtained in a software product by the means of modeling can be measured. Therefore, works on measuring the quality of models have to be considered as well.

Industrial Commitment. Education performed at universities is not only done to produce researchers but also to prepare software engineerings and developers for industry. Without close cooperation with industry, the danger is inherent to produce academic who are able to use most recent concepts theoretically, but who have no experience for practical tasks.

Student Involvement. The Edusymp is only attended by educators and trainers, giving students no chance to present their experiences and ideas when they have been recently trained on modeling. In fact, after a course they should have a profound understanding on the techniques and tools and they could give a good feedback on pros and cons of the taught content, especially when they did some advanced work in the context of practicals or thesis. One reason, why a student discussion group is not so easy to organize is that for traveling the budget is often very low and that only in very rare cases funding for sending students to conferences is available.

Bibliography

[BKSW09] P. Brosch, G. Kappel, M. Seidl, M. Wimmer. Teaching Model Engineering in the Large. In *Educators' Symposium @ Models 2009*. 2009.

[BM99] J. Bézivin, P. Muller. UML: The Birth and Rise of a Standard Modeling Notation. In *Proc. of the Int. Conf. on the Unified Modeling Language (UML 1998)*. Pp. 514–514. Springer, 1999.

[Boe06] B. Boehm. A View of 20th and 21st Century Software Engineering. In *Proc. of the 28th Int. Conf. on Software Engineering (ICSE 2006)*. Pp. 12–29. 2006.

[Boo86] G. Booch. Object-Oriented Development. *IEEE Transactions on Software Engineering* 12(2):211–221, 1986.

[BWH11] M. Brandsteidl, K. Wieland, C. Huemer. Novel Communication Channels in Software Modeling Education. In *Workshops and Symposia at MODELS 2010*. LNCS 6627, pp. 40–54. Springer, 2011.

[EHLS06] G. Engels, J. H. Hausmann, M. Lohmann, S. Sauer. Teaching UML Is Teaching Software Engineering Is Teaching Abstraction. In *Proc. of Satellite Events at the MoDELS 2005 Conference (MoDELS 2005)*. LNCS 3844, pp. 306–319. Springer, 2006.

[Fra09] R. France. Why Johnny cant model. *Software and Systems Modeling* 8(2):163–164, 2009.

[GJ08] C. Ghezzi, M. Jazayeri. *Programming Language Concepts*. Wiley, 2008.

[Jac92] I. Jacobson. *Object-Oriented Software Engineering*. ACM, 1992.

[Jac06] D. Jackson. Software Abstractions. *The MIT Press*, 2006.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.

[Rob09] P. Roberts. Abstract thinking: a predictor of modelling ability? In *Educators' Symposium @ Models 2009*. 2009.

[SC11] M. Seidl, P. J. Clarke. Software Modeling in Education: The 6th Educators' Symposium at MODELS 2010. In *Workshops and Symposia at MODELS 2010*. LNCS 6627. Springer, 2011.

[Sch06] D. Schmidt. Guest editor's introduction: Model-driven engineering. *Computer* 39(2):25–31, 2006.

[SHMA08] J. Sourrouille, M. Hindawi, L. Morel, R. Aubry. Specifying consistent subsets of UML. In *Educators' Symposium @ Models 2008*. 2008.