Summary of the Educator's Symposium

Holger Giese[1], Pascal Roques[2], and Timothy C. Lethbridge[3]

[1] University of Paderborn, Germany
hg@uni-paderborn.de
http://www.springer.de/comp/lncs/index.html
[2] Valtech Training, France
pascal.roques@valtech-training.fr
[3] University of Ottawa, Canada
tcl@site.uottawa.ca

Abstract. This first Educators' Symposium of the conference on Model Driven Engineering Languages and Systems (MoDELS - formerly the UML series of conferences) was intended as a forum to foster discussion and the exchange of information on education and training concerning model-driven engineering. This summary reports about the workshop and the results of the discussions during the workshop.

1 Introduction

Model-driven development approaches and technologies for software-based systems, in which development is centered round the manipulation of models, raise the level of abstraction and thus, improve our abilities to develop complex systems. A number of approaches and tools have been proposed for the model-driven development (MDD) of software-based systems. Examples are the UML, model-driven architecture (MDA), and model-integrated computing (MIC).

Initiating the model-driven development vision into common practice requires not only sophisticated modeling approaches and tools, but also considerable training and education efforts. To help developers adopt MDD, its principles and applications need to be taught to practitioners in industry, incorporated in university curricula, and probably even introduced in schools for primary education.

The educator's symposium at the MoDELS conference, the premier conference devoted to the topic of model-driven engineering of software-based systems, served as a forum in which educators and trainers met to discuss pedagogy, use of technology, and to share their experience pertaining to teaching modeling techniques and model-driven development. The symposium also facilitated the sharing of project ideas, exemplar models, and other teaching materials.

The symposium had 16 submissions from which only 9 papers were accepted. All papers presented during the symposium have been published in a technical report [1]. The areas addressed by the papers of this symposium include experience reports from academia, industry, and primary schools regarding issues related to teaching modeling and model-driven development in particular. The covered topics were MDD with

UML [2,4,5,6,8,9,10], MDD in general [3,7], course design issues [3,4,5,6,7,8,10], and design patterns [4,8,9]. Additionally, methodology issues as well as the integrating of modeling and model driven development into the curriculum are discussed. Extended versions of the two best papers [5, 10] are included in this volume.

In addition to the presentations, the symposium included time slots for working groups. Two possible working group themes were presented: Timothy C. Lethbridge proposed to discuss the role that modeling should play in the curriculum for software engineering and Pascal Roques proposed to discuss the differences and commonalities of teaching students and training professionals in the field.

In an initial discussion these two working group themes were slightly adjusted to also include other topics of interest. In the following we will outline the results obtained during the discussions.

2 Results of the working group on modeling in the curriculum

Tim Lethbridge first presented some information about the modeling content in SE-2004 [11]. The group members in this session then brainstormed for answers to two questions. The following are edited versions of the answers they produced:

The first question was: "What should be the goals and outcomes of modeling aspects of curricula?" The conclusions were as follows, most important first:

- Students should **be able to communicate effectively using abstractions**: They should be able to understand abstractions (beyond those found in programming languages), create new abstractions and validate abstractions.
- Students should **be able to model heterogeneously**: They should be able to find the right abstraction for the problem at hand, create different models for different audiences, and be able to work with different views of the same system. They should know the properties of each type of model, and should be able to choose the level of formalism so as to be cost-effective and balance costs and quality
- Students should **be able to model in at least one "real" domain**. They need to have knowledge of both the domain and ways to model in that domain. Since it is impossible to educate students in a large number of domains, they need to be able to have the flexibility to choose their specialty. Finally, they should be able to develop and work with domain-specific models and languages in their domain(s).
- Students should **be able to apply a wide variety of patterns** in their models, particularly design patterns. Patterns are widely recognized as effective expressions of expert knowledge. Applying patterns will result in better models.
- Students should **have a deep understanding of quality**. They need to understand how their modeling work influences the quality of the final product, and they need to be able to certify that models have certain properties.
- Students should **be able to create 'models that count'**: i.e. not just diagrams, but models that are formally analyzable, executable and/or used to generate final code of the system. Students should also be able to transition models to design and code, extract a model from code, and understand what a compiler and model compiler do.
- Students should **know the importance of keeping models updated.**

- Students should **know the basics of creating modeling tools and metamodelling**.

The second question addressed was, “What do we need to do to improve model education?” The answers fall into the following three themes:

- We should **confront students with complex models they have to change**, rather than having them create models from scratch. This will teach by example and help students learn about scalability. One strategy is to have students build systems using frameworks, and with models of the frameworks. Models should also be used as building blocks to build other models: i.e. students should re-use models as components. Students should also model with COTS components.
- We should **expose students to modeling in a variety of domains**, including in other types of engineering. For example, we can demonstrate electrical, mechanical and software modeling in the automotive and aeronautical industries, and teach about the types of analysis these models permit – such as performance, safety, etc.
- We should **ensure students learn the benefits of modeling**. In particular we must demonstrate that good modeling makes systems easier to change and can lead to improved performance. At the same time students must understand the limitations of modeling. This can be accomplished using well-designed case studies.

3 Differences and commonalities of teaching students and training professionals in the field

To start the discussion, Pascal Roques made a short presentation of his activities as a modeling consultant and trainer for a French training company called Valtech Training. He focused on the use of adult learning theory [12] in Valtech’s courses: Key tenets of this are: a) Create a positive environment, b) Disseminate information, c) Exercise knowledge, and d) Provide feedback. The main goal is to provide trainees with confidence and the ability to apply course concepts outside the classroom.

The group tried to figure out the main differences and commonalities between teaching students and training professionals in the field. The following are some of the conclusions reached by the group:

Firstly, adult learners are *volunteers*. They need the knowledge for their daily work and their company is paying for it, so they want a concrete return on investment, and *practical* training is also required. Students don’t have the “context”, adults have: very often, professionals want to see examples in their domain. But real-life examples are very difficult to elaborate.

Adult learners are often already experts in their field. They may be even more experienced than the teacher. People who have a lot of experience and expertise are often more reluctant to change. In particular, there is an important difficulty for the teacher if the trainees have been “forced” to go to the training to reconvert (example: COBOL programmers taking UML and Java courses).

In adult courses, there is often no exam at the end. The exception is for certification courses. Sometimes adults simply want to improve their C.V., and in such cases have a similar motivation to students regarding succeeding in exams.

Other differences relate to logistics. Groups of trained professionals are usually smaller (3 to 12), but may be very heterogeneous regarding their experience in the topic, group interaction, career background, etc. Training courses are short (1 to 5 days), compared to a semester with 2 to 6 hours a week. So training for professionals must be efficient and fast. The intensity is different; there is less time to digest. One consequence is that professionals usually do not have enough time to use modeling tools, whereas students have time to master them and indeed specifically want to.

To sum up the main differences, one could say that professionals need state-of-the-art skills, while students want knowledge that will survive the next 10 years.

4 Conclusion

The symposium attracted more than 20 participants, including researchers and instructors with various interests and backgrounds in modeling and MDD. The working group discussions benefited greatly from this mixture of the two perspectives provided by the two categories of attendees. We hope that this first Educators' Symposium initiates what will become a permanent offering at future MoDELS conferences such that it can serve as a starting point for building an active community that addresses the specific problems of teaching and training issues related to modeling and the model-driven paradigm.

References

[1] Holger Giese and Pascal Roques (Editors). Proceedings of the Educators' Symposium of the ACM / IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Half Moon Resort, Montego Bay, Jamaica. October 3, 2005. Technical Report tr-ri-05-260, Department of Computer Science, University of Paderborn.
A4: http://models05-edu.upb.de/proceedings/models05-edu-proceedings-a4.pdf
Letter:http://models05-edu.upb.de/proceedings/models05-edu-proceedings-letter.pdf

[2] Jörg Niere, Carsten Schulte. Avoiding anecdotal evidence: An experience report about evaluating an object-oriented modeling course. In [1].

[3] Pádua, Paula Filho. A Model-driven Software Process for Course Projects. In [1].

[4] Kendra Cooper, Jing Dong, Kang Zhang, Lawrence Chung. Teaching Experiences with UML at The University of Texas at Dallas. In [1].

[5] Ludwik Kuzniarz, Miroslaw Staron. Best Practices for Teaching UML based Software Development. In [1].

[6] Shayne Flint, Clive Boughton. Three years experience teaching Executable/Translatable UML. In [1].

[7] Anirudha S. Gokhale, Jeff Gray. Advancing Model Driven Devlopment Education via Collaborative Research. In [1].

[8] Eduardo B. Fernandez, María M. Larrondo Petrie. Teaching a course on data and network security using UML and patterns. In [1].

[9] Claudia Pons. Basis for a Course on Design Patterns: going beyond the intuition. In [1].

[10] Gregor Engels, Jan Hendrik Hausmann, Marc Lohmann, Stefan Sauer. Teaching UML is Teaching Software Engineering is Teaching Abstraction. In [1].

[11] IEEE and ACM: Software Engineering 2004 Curriculum Recommendations: http://sites.computer.org/ccse.

[12] William A. Draves, How to Teach Adults, 2nd edition, LERN, 1997.