# Mastering Model-Driven Engineering

Håkan Burden
Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
burden@cse.gu.se

Tom Adawi
Engineering Education Research
Chalmers University of Technology
Gothenburg, Sweden
adawi@chalmers.se

## ABSTRACT
The challenge of transforming the understanding of a problem into a validated solution is not a trivial task. Using the conceptual framework of cognitive apprenticeship we show two ways to guide novices towards becoming masters in model-driven engineering.

## Categories and Subject Descriptors
K.3.2 [**Computer and Information Science Education**]: Computer Science Education

## General Terms
Design,Theory

## Keywords
Cognitive Apprenticeship, Pair lecturing, Executable models

## 1. COGNITIVE APPRENTICESHIP
In software development the process of understanding a problem and defining a solution implies interacting with other stakeholders as well as validation of the implementation. Mastering this process is not necessarily an easy task. In the context of teaching Model-Driven Engineering, MDE, it is not obvious for all students how to develop and validate different solutions. Cognitive apprenticeship [1] aims to verbalize how masters step-by-step formalize their understanding into a solution. In this way the tacit knowledge as well as the alternative routes are made explicit to the apprentices who learn from imitating and reflecting on the practice of the master.

## 2. TEACHING MDE
The only way to validate a sketch or blueprint model is by manual inspection which requires the skills of a master. We therefore employed an executable modeling language that supplied the students with continuous validation [3]. Using the terminology of cognitive apprenticeship teacher-student time was used for *modelling* and *coaching* while *scaffolding* and *exploration* distinguished the work the students did on their own. The project demonstration at the end of the course was an opportunity for us as teachers to ensure the students reflected on their own ability. In total 43 out of 50 projects delivered executable models that met the design criteria within the designated time frame. Our own evaluation of the models gave that the quality in terms of details and consistency had improved in general and in some cases went beyond what we thought possible, given the context.

While the introduction of executable models improved the students modeling we still found that the students struggled to apply the lecture content to their project. Lectures tend to present a neat solution to a problem and in the case the process for obtaining the solution is given that is also given as a straight-forward process. What we wanted to do was to increase the interaction with the students in order to adjust the lecture content to their needs since students learn more when they are actively involved during lectures.

Pair lecturing lets us do just this [2]. In the context of cognitive apprenticeship we used the lectures to *model, articulate, reflect* and *explore* different ways of modelling. In this way we make explicit our own individual cognitive processes in interaction with the students. Regarding the pros and cons of pair lecturing the students found that too many opinions could be confusing while they appreciated that complicated concepts were explained twice and in different ways by the two teachers. As teachers we found that we were out of comfort zone since we could not predict where the student interaction would take us. The most important change for us was the new possibility for *reflection*.

## 3. REFERENCES
[1] J. S. Brown, A. Collins, and P. Duguid. Situated cognition and the culture of learning. *Educational Researcher*, 18(1):32–42, Jan-Feb 1989.

[2] H. Burden, R. Heldal, and T. Adawi. Pair Lecturing to Model Modelling and Encourage Active Learning. In *Proceedings of ALE 2012, 11th Active Learning in Engineering Workshop*, Copenhagen, Denmark, June 2012.

[3] H. Burden, R. Heldal, and T. Siljamäki. Executable and Translatable UML – How Difficult Can it Be? In *APSEC 2011: 18th Asia-Pacific Software Engineering Conference*, Ho Chi Minh City, Vietnam, December 2011.