Teaching modelling for requirements engineering and model-driven software development courses

Arne J. Berre, Shihong Huang, Hani Murad & Hanieh Alibakhsh

Citation:
Arne J. Berre, Shihong Huang, Hani Murad & Hanieh Alibakhsh (2018) Teaching modelling for requirements engineering and model-driven software development courses, Computer Science Education, 28:1, 42-64, DOI: 10.1080/08993408.2018.1479090

Teaching Modeling for Requirements Engineering and Model-Driven Software Development Courses

Arne J. Berre, SINTEF and University of Oslo, Norway

Shihong Huang, Florida Atlantic University, Boca Raton, FL. USA

Hani Murad, University of Oslo, Norway

Hanieh Alibakhsh, University of Oslo, Norway

Abstract: This paper presents the results of observations and analyses of students' learning model-driven system development from two related courses taught at a university in Norway and at a university in the U.S. in 2015, and consequently, in an updated version in 2016. The motivation of this paper is to understand and analyse how effective the current practice of teaching and learning modeling and model driven software development is in university settings, and to offer some pedagogical insights and lessons learnt from teaching two different model related graduate courses at two different universities. Empirical data of learning was collected through interviews, observations, document analysis and a survey questionnaire. The aim of these two courses is providing students with the competence of problem solving in modeling. Topics of models in these courses cover a full spectrum of modeling techniques, from business architecture models, requirements models, system and software architecture, to design models. The courses have evolved from an initial focus on modeling for analysis and design to the current focus on using executable models for software production. The result is a complete enterprise architecture modeling approach education from business architecture to software architecture to functioning software.

Keywords: Modeling, Software Engineering Education, Model-driven software development (MDSD), model-driven architecture (MDA)

1. Introduction

The benefits of model-driven software development (MDSD) have long been proven to be cost-effective on producing large-scale high-quality software systems. Models provide abstract representations of software systems that allow software engineers to focus on high-level artefacts and their relationships while ignoring the implementation details of the system. The power of the ability to do validation and verification mathematically and seamlessly working with automated software engineering makes MDSD one of the favourable software development mechanisms in industry, particular in mission critical systems.

However, in software engineering education, teaching and using models is a challenging task for many reasons. First, students have less appreciation of the usefulness of modelling which are abstract and could be mathematical sometimes. Secondly, models are intangible comparing to coding which are fun and give students immediate feedback of their work. Thirdly, many job interviews require students to have strong programming skills with less emphasis on their modelling techniques. This is a disincentive to for students to learn software modelling.

To change students' misinterpretation of modeling, and to instill the important concepts of software modeling in students learning experience, the authors from the University of Oslo (UiO), Norway and Florida Atlantic University (FAU), USA started collaborating on teaching model-driven software development and modeling techniques courses in their own universities, respectively. This collaboration started in 2013 up to present. In spring 2015 and 2016, it was decided to use these two co-teaching courses as a case study to quantitatively evaluate the effectiveness of teaching modelling in software engineering education.

Designs of these courses are based on constructivist educational theory (Vygotsky, 1934; Dewey, 1956; Golding, 2009) where students develop their skills, understanding and problem-solving capabilities in a constructive and systematic approach (Cress & Kimmerle, 2008). Furthermore, Students' full participation and collaboration, through using different educational supportive tools in course projects enhance their active learning and collaborative knowledge building (Ludvigsen & Mørch, 2010).

The motivation of this paper is to understand and analyse how effective the current practice of teaching and learning modeling and model driven software development is in university settings, and to offer some pedagogical insights and lessons learnt from teaching two courses in requirements engineering and model-based software development.

This paper begins with a section on related work. Then it presents learning models as a foundation for later analysis. Section 4 introduces the structure and approach of the courses taught at both universities. Section 5 presents the qualitative analysis of the outcome of the courses through observations, interviews, document analysis and a questionnaire. Section 6 concludes the paper and points out future work.

2. Related work

Current technology-mediated learning approaches emphasize active participation (Sfard, 1998) and joint meaning making (Hakkarainen et al., 2004; Stahl, 2004; Engeström, 2009). Software engineering course projects are usually team-based and require students to use large scale computer-aided software collaborative tools. When teaching modelling techniques, or model-driven development, we focus on using different programs for diverse communities of learners (Nardi&O'Day, 1996; Pringle, 2002). Furthermore, meaningful communication between different students performing modeling activities predisposes the availability of common or shared information spaces depending on scope of cooperation, location and time factors (Bannon & Schmidt, 1991; Bannon & Bødker, 1997). Such collaborative learning practices and platforms support co-located and spatially distributed activities (Lundgren et al., 2015) and increase "rigor and relevance in knowledge production" (Culén, 2015) in an immersive environment (Pagano and Olbrish, 2013). Students orient their interactions through fluid structures of activities towards constructing a domain specific modeling environment and then define model transformations (Clarke et al., 2009) that provide appropriate semantics to such a user-defined model (Giddens, 1984; Nardi, 1996; Kaptelinin, 2005).

Roger Säljö (2010) argues that human learning has always involved interacting with artefacts and technology. Learning technologies support traditional classroom learning by offering *hybrid-learning platforms* and practices displaying *transformational* learning processes based on the different affordances of such technologies with respect to information access, storage and use. Säljö also points out the absence of any linear correlation between the use of instructional technology and our institutional assumptions and interpretations of improved learning. Students develop improved skills and understanding through a complex interplay between cognitive attributes, socio-cultural significance, institutional setting, individual and group practices that determine the formation of new modes of collective memory (Nickerson, 2005; Mäkitalo et al. 2009).

Different studies support the notion that functional understanding takes place prior to structural understanding, (Vessey& Conger, 1994; Stamatova & Kaasbøll, 2007; Grant et al., 2009). In a test comparing explanations with and without diagrams by means of functional and structural models, including diagrams was seen to generate better learning, for students with low verbal abilities (Fururta, 2000; Cuevas et al., 2002). Structural models normally consist of graphics and text, and they depict data structures or structures of IT concepts in a model. For example a file system provides access to the files and folders in a computer, and a relation to other file systems, implying an external structure. A functional model has an input state, one or more sequential operations trigger changes, and an output state resulting from the different operations (Dutke & Reimer, 2000; Kaasbøll, 2016). When achieving functional understanding, students are then able to explain that an operation transforms an input state to an output result. Structural understanding occurs when students can use this concept as a basis for learning new concepts.

3. Learning models

Learning MDSD is an abstract and intensive process where students need to understand, design, implement and modify software systems, according to pre-defined requirements and goals. Problem-solving provides the needed bridge between acquired learning and performance skills.

Such a learning process for problem-solving requires students to have the ability to transform their acquired skills of solving one type of problem and to apply their "know-how" into a different problem domain. This is usually achieved through developing a higher-level of problem understanding where the acquired practical skills of "knowing how" learning-phase of problem-solving is transformed into an understanding-level of "knowing why" knowledge-building phase through developing a better functional and structural understanding of the actual problem domain (Kaasbøll, 2016). For example, when pursuing different course activities in software design, and the dynamic creation of structured data that would result in new software implementation (Clarke et al., 2009). Different studies also indicate that developing a good understanding of the problem domain often leads to improved abilities to solve arising similar problems (Kiili & Ketamo, 2007; Novick et al., 2009).

Understanding the usefulness of modeling in facilitating the functional and structural development can be beneficial for understanding how different models can be designed and used to generate implementation (Kaasbøll, 2016). Functional understanding takes place when comparing input and output, and the different operations leading to change in a learning process. When learners can compare the learned concept to other concepts by addressing the structure of technology used, through its conceptualisation, then structural understanding is achieved.

Our modeling courses are designed with a strong emphasis on promoting scientific enquiry based learning and guided discovery (Prince & Felder, 2006; Banchi and Bell, 2008) where students engage in collaborative exploration, experimentation, and dialogue in small groups, thus allowing group tutors to follow the students' hierarchical structure of actions which, are implemented through lower-level units of activity, called operations (Engeström, 2009). Operations are routine processes providing an adjustment of an action to the ongoing situation, where students may re-adjust their problem-solving approach as they develop better understanding of the actual task. Both the concepts of state and type of operators define the concept of a problem space, and the term problem-solving method refers to the principles used for selecting different operators (Fikes & Nilsson, 1971; Newell & Simon, 1972; Anderson, 1987, 1990b).

Developing learning skills requires the ability to communicate and express ideas either verbally or in writing about the subject matters. Internalization of information and objectified facts from the outside environment occurs through assimilation, perception, encoding and re-alignment cognitive processes, and then incorporating it into their own knowledge repository. New knowledge needs to be tested and validated to assess its real value. This may be done during an externalization process, where internalized knowhow is shared and objectified into the real world (Berger& Luckmann, 1966).

Kaasbøll (2016) presents a three-level model of competence building that can be useful to understand the different perspectives involved in learning. Table 1 illustrates these three levels in the context of our course design, and used further in Sections 4,6 and 7.

Syntax in this model refers to the rules that how individual symbols could be combined through a command sequence, ensuring that specific symbols are followed through with numbers (Kaasbøll, 2016). Semantics relates to a term and the meaning it conveys. For instance, ID cards are representative of the relationship between an individual and their name, and reflect the semantics associated with the name.

Table 1: Learning competence model

Learning Competence	**Syntax**	**Semantics**	**Business/ Context Fit**
1.Skill	Using tools and methods according to correct modeling language notations	Proper use of tools and methods for the creation of good quality meaningful models	Skills for use of tools in business/context

2.Understanding	The student can explain the principles in tools and methods according to correct modeling language notations from a functional (How models work) and structural (How models are built-up) perspectives	The student can explain the principles in tools and methods – related to the creation of meaningful models with good quality, from a functional and structural learning perspectives	Understanding Models in business/ context activities
3.Problem solving	Understanding syntax- Research cycle competence- Precise observations- Help seeking	Applying tools and methods to create new meaningful models related to a real world- case study	Solving problems of model fit in business/context. Understanding the creation and fit of a model related to the business/context requirements

1. Skill – What is important for the students to get good syntactic and semantic skills for MDSD To which extent does learning background of students in modelling or/and programming effect the transmission of requirements specification into realization phase?

2. Understanding – What is important for the students to get good syntactic and semantic understanding of MDSD? Are there any dependencies between having a modeling background and a programming background among students for developing structural and functional understanding of modeling languages?

3. Problem Solving – What is important for the students to get good syntactic and semantic problem-solving capabilities of MDSD? Is it an advantage to use more individual exercises versus group exercises to advance the learning of modeling techniques for each student?

4. Course details

The educational context for this paper is two different graduate level courses related to model driven software development taught simultaneously from 2013 – 2016 at University of Oslo (UiO) and Florida Atlantic University (FAU), which were Model-Driven Software Development at UiO and Requirements Engineering at FAU. These two courses had a set of synchronised lectures and projects. The course on Model-Driven Software development evolved from an earlier version of the course called "Modeling with Objects" given by the first author since 1996 and extended to the current introduction of Model-Driven Architecture (MDA) and Engineering (MDE) since 2003. The course on Requirements Engineering has been given by the second author since 2008, and later introduced the perspective of executable models to facilitate agile requirements engineering.

The requirements engineering course at FAU comprised a live classroom section and a distance learning section where students watch recorded lecture videos via Blackboard. The model-driven software development course at UiO used Devilry as course ware platform.

4.1 Different modeling tools used at different phases of the courses

System models may be prescriptive in their nature, where the type and scope of problem at hand are identified, to find a suitable solution (problem solving). System modelling can also be descriptive, where they are used to explain the reality of a system by describing its use-context and implementation platform (Brambilla & Wimmer, 2012). In Modeling and Model-Driven Software Development's courses (MDSD), students usually utilize a wide array of available sets of software engineering models, tools and modeling approaches to describe information systems architecture at the desired level of detail (Krogstie, 2012).

Utilizing supportive tools in model driven development, we adopt technologies supporting information sharing and collaborative learning in our two courses, including the integration of generic collaboration tools, such as Google Hangouts and UpWave for synchronous remote dialogues and screen sharing. Domain specific tools were used for modeling different phases of MDSD, including Strategyzer, Smaply, Balsamiq and MagicDraw/Enterprise Architecture). These suites of tooling provide a solid platform for local and remote collaborative learning. All students have equal synchronous and asynchronous access to the same model and modelling tools. Balsamiq is a wire-framing and mock-up tool for the GUI interface design. Its components contain various GUI controls, for example, windows, buttons etc., to design a web interface. BPMN is a flowchart based notation for describing different scenarios, defining business processes and identifying different computer-assisted user tasks, which can be linked to use cases. Other system modeling tools used in the course include Strategyzer, a Business Model Canvas tool; Smaply, a service design tool; MagicDraw Enterprise, an architectual tool, For web application modeling and development, our students use Information Flow Modeling Language (IFML) Eclipse SIRIUS is a domain specific meta-model and language editor (Berre et al, 2013).

4.2 Course settings and pedagogical structures

In spring 2015, 15 students with different backgrounds had taken the course at UiO , and 14 students at FAU. In Spring 2016 the number of students were 19 and 22, respectively. The assignments were all done in groups consisting of three to four students. The pedagogical goal is to engage all students in modeling through a complete case from business architecture to implementation. During these two years the overall projects had been related to actual business cases by two different startup companies. The related course projects were an app supporting UV sensors in 2015 and an app for citizen science monitoring of biodiversity observations in 2016. Students were encouraged to use Upwave.io or Trello.com in combination with Google Hangout for communication and Scrum project support with these tools for project planning and progress monitoring.

The two courses aim at teaching modeling techniques that can be applied directly in an industrial setting based on our philosophy of engaging the students in modelling through a complete case from business architecture to implementation. Furthermore, the courses aim to be practical by using modeling tools for the full development lifecycle from business architecture, requirements models to system/software architecture, design and implementation.

Both courses comprised lectures, teaching assistant lab hours, individual projects, group projects and student presentations. Such pedagogical approach supports internalization, externalization knowledge conversions to learning-by-doing practices (Dougherty, 2012) involving knowledge acquisition, assimilation and application when developing functioning system models.

4.3 Mapping course material to learning modeling techniques

Table 2 shows the different learning perspectives introduced in the Learning competence model in section 3, and how these are supported through teaching methods, elements and learning materials in the courses. The skill development is supported through active use of modeling tools. The functional understanding is supported by the tools' user guides, while the structural understanding is supported through the lectures and written materials with theoretical foundations and illustrative examples. The problem-solving competence is achieved through the practical exercises centered on a larger real-world case study. The course is structured into three areas with the following modeling techniques:

Table 2: Course teaching material related to elements in the learning model

Learning Competence	**Syntax**	**Semantics**	**Business/ Context Fit**
1.Skill	Teaching BA tools, SA tools, MDE tools, RE tools with demonstrations, videos, user guide and teaching assistant guidance	Teaching by showing relevant example models- as scaffolds for exercises	Teaching how to use the different tools as part of an overall development cycle

2.Understanding	a. Functional understanding (user guides) b. Structural understanding (Lectures, Books, Articles- modeling techniques)	a. Functional understanding (user guide examples) b. Structural understanding (Lectures, Books, Articles-meta models and principles)	Teaching Models in business/context activities through the teaching with relevant examples and case studies
3.Problem solving	Teaching through guidance and feedback on practical exercises	Guidance, inline help and feedback on practical exercises, assignments and main project	Requiring large project, from business need to implementation, in MDE-Domain specific language-from concept to editor and transformation

Main modeling concepts used in the teaching materials and supporting tooling are described below:

I) Business Architecture– Business Architecture focus on Business Model Canvas and Value Proposition Canvas. Service Design is modeled by using Smaply diagrams for Personas, Relationships models and service journey models. Domain models use UML Class diagrams, Business Process models use BPMN, Requirements models use User stories and UML Use cases/templates and UI Mockups use Balsamiq.

II) Software Architecture and Design with user interaction modeling used IFML (Interaction Flow Modeling Language) and supported by WebRatio, a development platform for web and app development, and support for executable BPMN models. Between these two courses, there were two special modeling requirements for each course, respectively:

> IIIa) Model Driven Engineering and system architecture – only for the model-driven software development course at UiO. It used Eclipse EMF and Sirius for development of domain specific language editors and model transformations. UML 2.0 Composite diagrams and UML collaboration diagrams within SoaML were used for modeling software architecture and design.
>
> IIIb) Requirements engineering course at FAU used additional requirements modeling techniques on goal-oriented requirements and on nonfunctional requirements.

Students worked in teams for the course projects. Project planning and actual project work were encouraged to use UpWave, a collaborative team work tool. Other tools that support modeling techniques all provide collaborative access for sharing models among team members during the model development, thus acting as an effective collaborative learning tool.

5. Experiments

The course projects and experimental approach that spanned over two years were based on an actual business case by two different startup companies' applications. One application was an app supporting UV sensors in 2015 and the other was an app for citizen science monitoring of biodiversity observations in 2016. For collaboration support within project groups for distributed work, including Scrum project support.

5.1 Setup

The empirical approach was designed to answer the main goal of the experiments performed, namely, "to what extent our teaching methods, course material and structure are effective according to the learning model elements: skills, understanding and problem-solving competence – with respect to learning syntax, semantics and business/context aspects for modeling".

The experiments were conducted at both University of Oslo (UiO), Norway and Florida Atlantic University (FAU), USA. Three participant observations on UiO students' collaborative activities were performed at the end of the project exercise in 2015, when the groups were working with their IFML app implementations.

The UiO's course was Model-driven software Development (INF5120)[1] whereas FAU's course was Software Requirements Engineering (CEN6075)[2], with 15 and 14 students, respectively. Students were divided into groups of three to four students in each group. All students worked on the same project using the same methodology (model-driven software development) in planning and design phases.

These two courses structures comprised lectures, lab hours, individual projects, team projects, and final project presentations.

5.2 Experiment methods and data collection

We used qualitative and quantitative methods for data collection, classification, and analysis including passive participant observations, audio recorded semi-structured interviews, documents analysis and a survey questionnaire (See appendix 2-4). The number of students that participated were 14 at UiO and 15 at FAU in 2015, and 19 and 22, respectively in 2016. Data classification made use of thematic analysis augmented by research questions, argumentation models (Scardamalia & Bereiter, 2003; Stahl, 2006), and the assignments. Assessments of students' newly developed skills, their functional and structural understanding of subject matter, and their problem-solving capabilities are analyzed as scaffolds of different operations, based on the three-level model of competence building developed by Jens Kaasbøll (2016). In our study, we apply this model to help us understand how students develop their new skills, achieve higher levels of understanding and improve their problem-solving capabilities with respect to the courses' concepts of syntax and semantics use, as a representation of their evolving learning processes.

5.2.1 Observations

Three participant observations were conducted in April 2015 at the University of Oslo. A week after handing out the first part of the project as an assignment 1 to students, the fourth author held a presentation to clarify the project and requirements specification and discussed the tools to be used during workshops and presentations before the final delivery. The same author had two workshops with all groups of students in Oslo. Each group consisted of four students with different backgrounds. During the workshops in two sessions, she simulated being a group member and worked with them on the same computer. She had a discussion with all group members about their problems related to the subject, the tools and use of learning materials. Students discussed more how and what challenges they had while using new tools and how they have figured out the learning materials for both the subject and tool guides.

All three observations were challenging, especially that the UiO group were familiar with the student assistant. Observations were made as objectively as possible within the context of our study and results presented here are only valid to the actual groups observed. To generalize our findings, we need to perform additional observations at neutral grounds, have a larger randomly selected sample of participants and be aware of the "*Hawthorne effect*" or "*observation bias"*, which is a well-documented phenomenon that may affect research experiments in social sciences, where people under observation in field settings tend to alter their behavior from normal, simply because they are being studied. It is however difficult to estimate the size of any such effect (French, 1953; Adair, 1985; McCambridge, et al., 2012).

[1] http://www.uio.no/studier/emner/matnat/ifi/INF5120/index-eng.html
[2] http://www.eng.fau.edu/directory/faculty/huang/ (Replace with FAU course catalog reference)

5.2.2 Interviews

The interviews were held at the end of the course in 2015, when students had submitted their final project. Eight semi-structured interviews were conducted, four from UiO and four from FAU.

The interview guide contains questions on the following: Student's background, comprehension level, functional understanding, structural understanding and problem solving approach. (See Appendix 1).

5.2.3 Document analysis

Document analysis was done for the student exercises and incremental project deliveries as well as for the exam papers in both 2015 and 2016 for the MDSD course, the RE course was graded only by project deliveries. Students submitted two assignments, and one main project. At the end of the course they took a written exam. An evaluation of assignments, the main project and exam results was done by scoring all parts of the assignments and exams questions. The written exam consisted of three questions. Question 1 is composed of five sections, while questions 2 and 3 has three sections each. Each section had a separate score, and the summative evaluation was based on the sum of all scores in the exam paper. The score-scale is 1-100.

5.2.4 Questionnaire

Questionnaires were handed out to all students in the courses in both universities in 2016. Questions related to students' background, their level of understanding of modeling, programming and of system development at the start of the course compared to those at the end of the course. Questions include, for example, "which of the modeling techniques they found most useful as input for the implementation phase and for their potential future use", "their views on the course structure and learning methods", and other comments on "what they liked and what potential improvements they suggest for future courses".

6. Results and analysis

In this section, we present the results obtained from observations, interviews, document analysis and questionnaires.

6.1 Observations results

The observation results are presented in terms of the nature of software tooling use and documentation, and the extent of which students could carry models to execution, as seen in Table 3. below

Table 3. Observations related to the Learning Model used

Observations- IFML for apps	**Syntax**	**Semantics**	**Business/ Context Fit**
1.Skill	IFML video tutorials were effective in providing tool skills	IFML example Project/ templates were effective in providing scaffolding for IFML projects	Implementing apps with IFML based on the business architecture models showed good business fit
2.Understanding	IFML serves as a good example of a domain specific language with practical use, and the lectures and reference material seemed adequate	IFML OMG standard is hard to read, IFML published book provides a better foundation. Lack of detailed user manual for IFML app platform,	The fit between IFML and the UI mockups in Balsamiq was occasionally challenging because of the fixed availability of

	for the needs	compared to web platform was a challenge	UI element options in WebRatio
3.Problem solving competence	Group organization resulted easily in some group members becoming IFML experts, while others were not	There were some occasional challenges relating to connecting to servers, etc. – with a new project that was different from the provided examples	IFML as supported by WebRatio provides a rapid approach for executable models resulting in apps or web solutions to fit business requirements

To summarize our observation results as the following:

- The observations on skill shows that the tool video tutorials with example templates work well.
- The observations on understanding shows that IFML is a complex language to learn in the context of developing mobile and web applications.
- The observations on problem solving competence shows that different groups were working differently with respect to how they divided work and responsibilities.

A recommendation after these observations is to introduce modeling for execution with IFML earlier on in the course. Also, rotate roles in the groups to allow everyone to develop relevant modeling competence is also important.

6.2 Interview results

Most of the students who attended the interviews have programming background and have worked with software engineering for some years. Moreover, they have studied and worked in different fields such as design and development of information systems and some of them have worked in network and administration field. On the question of which modeling techniques that provided most value with respect to providing a foundation for a transformation models to the implementation phase, the highest rank was given to UML, User stories and Use cases according to the interview results. The service design techniques received the lowest rank, while Business Model Canvas, UI Mockups with Balsamiq and BPMN received a medium rank. Scrum was viewed more as an overall methodology. These interview results seem to be consistent with the results obtained from the questionnaire.

6.3 Document analysis results

Analysis of the results from the written MDSD exam (4 hours practical case) showed that the students generally scored well on the business architecture (average score of 86 of 100) and system architecture and IFML part (average score of 73 of 100) . The questions on model driven engineering and meta modeling got lower scores(average score of 67 of 100) indicating students had less problem-solving competence in that area (i.e., creating a meta model for a described language).

A document analysis of the delivered exercises showed that some students were not clear on the difference between domain modeling and meta- modeling. For 2015 the MDE part on creating a DSL editor was only done as a group exercise at the end of the course. The recommendation is to postpone metamodeling and DSL exercise later as an individual exercise – or as "pair modeling» earlier on in the course.

The exam evaluation from 2016 showed the same pattern. The scoring in the MDE part had improved slightly (average score of 70 of 100) but was still lower than the other parts. A question on SoaML modeling, which had not been practiced in the project nor in any exercises, got a low score (average score of 52 of 100). This result indicated the importance of providing practical exercises and projects for all important learning topics.

6.4 Questionnaire results

Closer analysis of student's background from questionnaire, revealed that many students have relatively little experience with modeling (many the students were international exchange students and students from local industry). A conclusion is thus that we need to elaborate even more on basic principles for good modeling practices in future versions of these courses.

On the question on likelihood of using the techniques in the future, Scrum was reviewed the highest with 75% and Service journeys in service design was the lowest with 12.5 %. Also, domain specific language development was marked as low – indicating that this was not on the agenda for those with a requirement engineering focus and not the main focus for those in the course on model based system development. Clearly preferred techniques were use cases (62%) and user stories (56%) as well as UML class modeling (38%). Business model canvas was appreciated (50%) although comments indicated that this could be part of the overall context. BPMN was of general interest (50%) but comments indicated that it was not clear how to use it in practice for executable models.

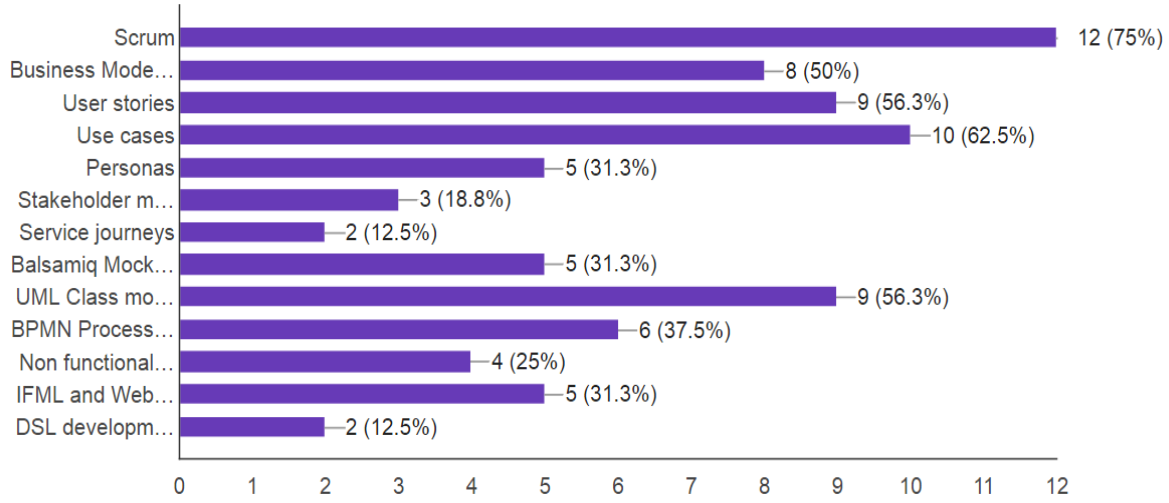

Figure 1 – Questionnaire on assumed usefulness of various techniques for software implementation

7. Discussion of results and course recommendations

7.1 Discussion related to the learning model

All empirical data collected in this study indicate the lack of dedicated tools used for teaching model-driven software development (Seidl & Clarke, 2011). Higher levels of skills, understanding and problem-solving competence are supported through active participation, as being emphasized by Sfard(1998), teaching methods, workshop-elements and learning materials in both courses. The overall skill development is supported through active use of modeling tools and proper syntax and semantics applications in business context. Furthermore, modeling skills are strengthened through *repetition,* and learning new skills is eased through *imitating* scaffolds in the form of practical IFML examples provided by Webratio, instructions or *instruction sheets* provided by trainers. This resonates well with what Roger Säljö (2010) termed as “hybrid- learning platforms” used to support *transformational learning.*

The three following tables summarize all the result and analysis from the learning model perspective.

Table 3 – Results for learner´s skill

Skill	**Syntax skill**	**Semantics skill**
From Interviews	Attending the Lectures, learning to work with BA/SA/MDE/RE tools with user guide and teaching assistance guidance	Attending workshops related to tools demonstrations, video tutorials and exercises and working by relevant simple examples
From Questionnaire	Using guidance BA/SA/MDE/RE tools, and discussion with teaching assistance	Working with relevant examples, demonstrations, video tutorials and individual assignments
From Observations	IFML video tutorials and provided learning materials by Webratio group were effective	Learning from relevant IFML examples and implementing apps with IFML exercises
From Document analysis	Provided assignments and learning materials were helpful but the lack of demonstration of individual capabilities by no individual assignments. The exam showed good individual skills for BA/SA and less for MDE.	The quality of the tool usage seems to have been adequate, but the usage of many tools implied less sophisticated usage of some tools. The exam was in written form without use of tools, and the basic techniques seems to be mastered by the students

During workshops, students were seen to develop deeper understanding when they are able to explain basic principles of methodologies and MDSD tools used according to usability- rules governing how individual syntax symbols could be combined through command sequences, and where the semantic meaning conveyed in a specific business context is clarified.

Table 4 – Results for learner´s Understanding

Understanding	**Syntax understanding**	**Semantics understanding**
From Interviews	Functional and structural understanding by attending the lectures, reviewing provided learning materials and exercises	Functional and structural understanding by attending the lectures, reviewing provided learning materials, examples, working with provided exercises and structural models

From Questionnaire	Functional and structural understanding attending the lectures, provided learning materials and exercises were effective	Functional and structural understanding attending the lectures, reviewing provided learning materials, examples, individual/group assignments and structural models were sufficient, the challenge was to learn about different tools
From Observations	Functional and structural understanding of a domain specific language by working with practical IFML examples provided by Webratio group and discussion either with group members or assistance	Functional and structural understanding by using IFML published book. Lack of time was a challenge for group works
From Document analysis	The understanding has been documented through the provided documents and associated presentations. The exam result analysis showed that the MDE part was less understood.	There were quality variations between the different groups in terms of the elaboration of their understanding and level of quality. The MDE assignment was less elaborated compared to the main project.

The modeling understanding at the beginning of the courses ranked from 2 to 8 (on a scale from 1 to 10) with the average of 4.3 and the modeling understanding at the end of the courses was ranked between 5 and 10 – with the average of 7.5. The programming understanding at the end of the course showed a similar pattern.

The functional understanding is supported by the tools' user guides, while the structural understanding is supported through the lectures and written materials with theoretical foundations and illustrative examples. Our observations indicate that understanding the usefulness of the technology used for one's own work is a driver for learning it, and through their interactions with the different modeling tools, students are able to construct and monitor their own learning /self-regulated learning (Pintrich and Zusho, 2002) through getting feedback in the form of a "dialogue" with each other, thus triggering peer discussions that may regulate the performance gap (Boyle & Nicol, 2003) and complete the understanding- loop (Sadler,1989) for individual students within the context of the learning environment.
Scaffolds (tools and methods) should therefore motivate for usefulness.

Table 5 – Results for learner´s problem-solving

Problem Solving	Syntax problem solving	Semantics problem solving
From Interviews	Related Inline help regarding models, reviewing provided learning materials like	Discussion with group, IFML inline help provided by Webratio, feedback on the main project, and

	presentations notes, teacher and assistance' feedback on practical exercises in lab hours	repeating the same steps until the problem gets solved.
From Questionnaire	Inline help, reviewing provided learning materials, feedback on practical exercises in lab hours.	Discussion in group, using learning materials, feedback on individual/group assignments and the main project by assistance.
From Observations	Not all group members worked at the same level or got the same level of skill while working with IFML	Some parts of the main project like connecting to the server was challenging and time consuming for groups which could have affected on the other parts as well
From Document analysis	Group assignments - The main project deliveries confirmed good problem-solving competence in the group but by exam sheets the MDE part of the courses needs to improvement related to providing the problem-solving capability.	Group assignments - The main project deliveries confirmed good problem-solving ability, but there were quality variations between the groups. The simplicity of the MDE assignment implied less quality on this.

Problem-solving competence is achieved through the practical exercises focusing on a case studies. Tensions arise when there is a clear mismatch between students' performance, and the expected learning outcomes outlined by the course teachers. The challenge is how to close the gap between current performance and the performance expected by the teacher to achieve significant learning benefits in developing new skills and knowledge building (Boud, 2000) that may be applied in a real-life business context.
We found preliminary evidence of multiple levels of navigation and interaction with technological platforms and their model- representations when students propose and present different solutions *in business case workshops.*

Whereas *Syntax* represent the rules that determine how single entities can be expressed and how they can be combined, *Semantics* describe the relation between a term and it's meaning. Our preliminary findings show that syntax aspects are learned through individual efforts displayed when students attend lectures and review provided learning materials and exercises. Semantics understanding of the different systems are however, supported through using the tools available and their different representations when students work in collaboration with each other in a business- context activity. Our observations show that students acquire *Semantic skills* when they are able to read, enter appropriate data and relate it to their working domain. Students were seen to achieve *Functional semantic understanding* when they are able to express why a state in the domain is represented by a particular piece of information, while *structural semantic understanding* was achieved when students were able to express the rules and conventions governing the domain-information relation, thus completing their understanding- loop (Sadler, 1989) within the context of the learning environment.
Semantic problem-solving competence is needed when the information is found to constitute inadequate representations of the desired models.

7.2 Course recommendations

The following recommendations based on the analysis is provided for future courses:

1. Continue to structure the main project of the course around a concrete business idea suitable for a start-up company, or a new product/service from an existing company to have a concrete and real-life situation as a motivating focus in the course, and cover relevant models from business architecture, requirements models to software architecture, design and realisation models. If possible work with local/global industries/start-ups around the project. Using a concrete case end to end provides the students with a good basis understanding and evaluating the different modelling techniques.

2. Take advantage of the programming background and motivation of most of the students, by ensuring that the course ends up with a working application/service/app demonstrating a Minimum Viable Product (MVP) – showing the usage of connected models from start to finish with appropriate tools for model driven engineering.

3. Extend the modelling approach to a full Enterprise Architecture perspective by modelling support for related business and system architectures, by using Enterprise Architecture modelling, with modelling framework examples like TOGAF and ArchiMate 3.0

4. Involve links to models from the business perspective and service design/interaction perspective – preferably with collaboration with related courses/communities to reduce the need for development of models from these perspectives – and to be as close as possible to a realistic future work situation. Enhance the formal modelling of Business architecture by the introduction of the recent OMG standard VDML (Value Delivery Modeling Language) and supporting modelling tools like VDMBee for the creation of business models.

5. Ensure that the modeling tools are available both with example tutorials as well as with comprehensive reference manuals to make it easy to create working models and to avoid spending too much time debugging and searching for solutions.

6. Introduce modeling exercises with quality review and feedback during the different steps of project development – focus on individual learning and related discussions for instance through pair modeling and group/class discussions. *(To get a good balance between group work and individual work).*

7. Check the background (programming/modeling) and preferred learning style (abstract/concrete) of the students at the start of the course. With a historical majority having mainly a programming background ensure an early start in the course with tools that creates actual working code and implementations from models.
8. Consider taking more advantage of learning management systems, online analysis tools and tools for collaborative learning.

9. The execution environment should take more advantage of the growing set of comprehensive software development platforms – and thus combine model-driven software development with platform-based software development.

For the joint courses for spring 2018, for joint case for the 2018 course editions will be around a complete Smart Home/Office system development.

8. Conclusions

Related work in this area has also pointed out the need to relate more explicitly to students with a good programming background and to teach how modeling also can support code generation and implementations through executable models (France, 2011; Clarke et al., 2009; Lano et al., 2015). Based on the results of the analysis we provide the following guidelines for the future course offering that we plan for:

- Continue the usage of collaborative tools both for project team work planning and execution and for collaborative development of models.
- Motivate the students, by ensuring that the course ends up with a working application/service/app demonstrating a Minimum Viable Product (MVP). These working applications show the usage of connected models from start to finish with appropriate tools for model driven engineering. The motivation can help students to get better skills and better understanding in the learning process. In addition, the student's background on programming/modeling at the beginning of the course could be evaluated.
- Introduce executable models (IFML or others, like Node-RED) at an early stage in the course for those who do not have modeling background. Modeling earlier in the course could cause lower cognitive load for learners later in learning, understanding and using the modeling in software development.

With a course approach where all the material from the course lectures is available online, and where the interaction with the students is supported by systems like Canvas (Blackboard), it is easier to provide access and inclusion to a wider group of students through the distance learning possibility for participation. The fact that the courses are heavily using computer supported tools with collaboration functionality (by sharing of models etc.), as well as the UpWave/Trello team collaboration tool for the team planning and collaborative synchronization of their work, and through the possibility to be involved through distance learning, there is support for inclusion and access to the course learning experience for a wider group than in other similar courses, with less usage of collaborative planning and model sharing tools. As a next step we will also use the learning model foundation introduced here to plan and evaluate future course offerings – both before, during and after our course delivery. The usage of collaborative tools for team work and model sharing will be continued. The next joint sequence of the courses has been executed for the spring of 2018.

The courses for spring 2018 have taken the experiences and recommendations into account and continue with the objective of keeping a balance between theory and practice as well as between individual, pair and group work. The same Learning Model for the planning, analysis and evaluation of the next version of the courses in spring 2018, both for course optimization according to the learning model as well as for collecting empirical basis for future improvements. We have now a second study in progress for the courses of 2018 and 2019, doing this also with a combination of different empirical techniques like survey/questionnaire, interviews, observations and document analysis.

9. References

Adair J.G. The Hawthorne effect-a reconsideration of the methodological artifact. J Appl Psychol. 1984;69(2):334–345.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192-210.

Anderson, J. R. (1990b). *Cognitive psychology and its implications* (3rd ed.). New York: Freeman.

Banchi, H. & Bell, R. (2008). The Many Levels of Inquiry. *Science and Children*, 46(2), 26-29.

Bannon, L. and Bødker, S. Constructing Common Information Spaces. *Proceedings of the European Conference on Computer Supported Cooperative Work ECSCW'97*. (1997), Kluwer, 81-96.

Bannon, L. J., & Schmidt, K. (1991): "CSCW: Four Characters in Search of a Context.", in J. M. Bowers and S. D. Benford (eds.): *Studies in Computer Supported Cooperative Work. Theory, Practice and Design*, North-Holland, Amsterdam, 1991, pp. 3-16.

Berger, P. L., and Luckmann, T. 1966. The social construction of reality. Garden City, NY: Anchor.

Berre, A.J., de Man, Lew, H.Y., Elvesæter, B., Ursin-Holm,B.M. "Open Business Model, Process and Service Innovation with VDML and ServiceML" in M. Zelm, M.v.Sinderen, L. Ferraira Peres, G. Doumeingts (Eds), Enterprise Interoperability, Proceedings of the Workshops of the Fifth International IFIP Working Conference, IWEI 2013, Enschede, Wiley.

Brambilla, M., Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers (2012)

Boud, D. (2000) Sustainable assessment: rethinking assessment for the learning society, *Studies in Continuing Education*, 22(2), 151-167.

Clarke P.J., Wu Y., Allen A.A, Experiences of Teaching Model-Driven Engineering in a Software Design Course, Online Proceedings of the 5th Educators' Symposium of the MODELS Conference, 2009.

Cress, U., & Kimmerle, J. (2008). A Systemic and Cognitive view on Collaborative Knowledge Building with Wikis. International Journal of Computer-Supported Collaborative Learning, 3(2), 105-122.

Cuevas, H. M., Fiore, S. M. & Oser, R. L. 2002. Scaffolding cognitive and metacognitive processes in low verbal ability learners: Use of diagrams in computer-based training environments. *Instructional Science,* 30**,** 433-464.

Culén, Alma Leora (2015). HCI Education: Innovation, Creativity and Design Thinking. *International Conferences on Advances in Computer-Human Interactions*. ISSN 2308-4138. s 125- 130.

Dewey, J. (1955) *Democracy and Education: An introduction to the philosophy of education. New York: The McMillan Company.*

Dutke, S. & Reimer, T. 2000. Evaluation of two types of online help for application software. *Journal of Computer Assisted Learning,* 16**,** 307-315.

Engeström, Y. (2009). The future of activity theory: A rough draft. In A. Sannino, H. Daniels, & K. Gutierrez (Eds.), Learning and expanding with activity theory. New York: Cambridge.

Fikes, R., and N. J. Nilsson N.J., STRIPS: a new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2(3/4):189–208, 1971.

France, R. B. 2011. Teaching Programming Students how to Model: Challenges & Opportunities, invited speak at EduSymp 2011, online at: http://edusymp.big.tuwien.ac.at/slidesKey.pdf

French J.R.P. Experiments in field settings. In: Festinger L., Katz D., editors. Research methods in the behavioral sciences. Holt, Rinehart & Winston; New York, NY: 1953.

Furuta, T. 2000. The Impact of Generating Spontaneous Descriptions on Mental Model Development. *Journal of Science Education and Technology*, 9**,** 247-256.

Giddens, A. (1984). *The Constitution of Society: Outline of the Theory of Structuration*, Cambridge: Polity Press.

Golding, C. (2009) 'The Many Faces of Constructivist Discussion', '*The Journal of Educational and Philosophy of Teaching'*, Vol.43, No.5, pp.467-483.

Grant, D. M., Malloy, A. D. & Murphy, M. C. 2009. A Comparison of Student Perceptions of their Computer Skills to their Actual Abilities. *Journal of Information Technology Education*, 8**,** 141-160.

Hakkarainen, K., Palonen, T., Paavola, S. & Lehtinen, E. (2004). Communities of networked expertise: Professional and educational perspectives. Advances in Learning and Instruction Series. Amsterdam: Elsevier

Kaasbøll, J. Developing digital competence - learning, teaching and supporting use of information technology.Report,University of Oslo, 2016.
http://www.uio.no/studier/emner/matnat/ifi/INF3280/v16/pensumliste/kaasboll2016developingdigitalcompetence.pdf .

Kaptelinin, V. (2005) The object of activity: making sense of the sense-maker. *Mind, Culture and Activity*, 12(1), 4-18.

Kiili, K., & Ketamo, H. (2007*). Exploring the learning mechanism in educational games. Journal of Computing and Information Technology, 15(4), 319-324.*

Krogstie, J. Model-Based Development and Evolution of Information Systems. Springer, 2012, pp. 74–80. ISBN: 978-1-4471-2935-6 (Print) 978-1-4471-2936-3 (Online).

Lano, K., Yassipour-Tehrani, S, Alfraihi, Experiences of teaching model-based development, in Educators Symposium (EduSymp), 2015. p. 43-54., http://ceur-ws.org/Vol-1555/

Ludvigsen, S. & Mørch (2010). Computer-Supported Collaborative Learning: Basic Concepts, Multiple Perspectives, and Emerging Trends. *International Encyclopedia of Education* 3rd Edition. Edited by Eva Baker, Penelope Peterson and Barry McGaw, Elsevier 2010.

Lundgren, S., Fischer, J. E., Reeves, S., & Torgersson, O. (2015). Designing Mobile Experiences for Collocated Interaction. *Proc. CSCW'15.*

McCambridge J., de Bruin M., Witton J. The effects of demand characteristics on research participant behaviours in non-laboratory settings: a systematic review. PLoS One. 2012;7(6).

Mäkitalo Å., Jakobsson A. & Säljö R. (2009) Learning to reason in the context of socioscientific problems. Exploring the demands on students in 'new' classroom activities. In Investigating Classroom Interaction. Methodologies in Action (eds K. Kumpulainen, C. Hmelo-Silver & M. Cesar), pp. 7–25. Sense, Rotterdam.

Nardi, B., Ed. (1996). Context and Consciousness: Activity Theory and Human-Computer Interaction. Cambridge, MA, MIT Press.

Nardi, B. A., & O'Day, V. L. (1999). Information ecologies. Using technology with heart. Cambridge, MA: The MIT Press.

Nickerson R.S. (2005) Technology and cognition amplification. In Intelligence and Technology. The Impact of Tools on the Nature and Development of Human Abilities (eds R.J. Sternberg & D.D. Preiss), pp. 3–27. Erlbaum, Mahwah, NJ.

Novick, J. M., I. P. Kan, J. C. Trueswell, and S. L. Thompson-Schill. 2009. A case for conflict across multiple domains: Memory and language impairments following damage to ventrolateral prefrontal cortex. Cognitive Neuropsychology, 26(6). 527–67.

Pagano, Koreen Olbrish. Immersive Learning: Designing for Authentic Practice. Alexandria, VA: ASTD Press, 2013

Prince, J.M. and Felder, M.R. (2006) Inductive Teaching and Learning Methods: Definitions, Comparisons, and Research Bases. Journal of Engineering Education, 95, 123-138.

Pringle, R. M. (2002). Developing a community of learners. Potentials and possibilities in Web mediated discourse. Contemporary Issues in Technology and Teacher Education, 2(2), 218–233.

Scardamalia, M., & Bereiter, C. (2003). Knowledge Building. In *Encyclopedia of Education, 2nd ed*. (pp.1370-1373). New York: Macmillan Reference, USA.

Schmidt, A., Kimmig, D., Bittner, K., Dickerhof, M.: Teaching Model-Driven Software Development:Revealing the "Great Miracle" of Code Generation to Students. In: Sixteenth Australasian Computing Education Conference (ACE2014). CRPIT, vol. 148, pp. 97–104. ACS, Auckland, New Zealand (2014).

Seidl, M., Clarke, P. (2011): Position paper: Software Modelling Education. The 7th Educators Symposium at Models, Wellington, New Zealand.

Sfard, A. (1998). On two metaphors for learning and the dangers of choosing just one. *Educational Researcher, 27,* 4–13.

Stahl, G. (2006). *Group cognition: Computer support for collaborative knowledge building*. Cambridge, MA: MIT Press.

Stamatova, E. & Kaasbøll, J. J. 2007. Users' Learning of Principles of Computer Operations. *Issues in Informing Science and Information Technology*, 4**,** 291-306.

Säljö, R. (2010). Digital tools and challenges to institutional traditions of learning: Technologies, social memory and the performative nature of learning. *Journal of Computer Assisted Learning*, 26(1), 53–64..

Stahl, G. (2004). Building collaborative knowing: Elements of a social theory of CSCL. In Strijbos, Kirschner & Martens (2004). *op cit.* (pp. 53-86).

Vessey, I. & Conger S. A. 1994. Requirement Specification: Learning Object, Process, and Data Methodologies. *Communications of the ACM,* 37**,** 102-113.

Vygotsky, L., S. (1934) cited in Palmer, J., A, (ed.) (2001, pp.33-37) *Fifty Modern Thinkers on Education: From Piaget to the Present.* London: Routledge.

10. Appendix

1. The interview guide:

   https://docs.google.com/document/d/e/2PACX1vSIerwXakMuDbg9k9HK6dFQXBPCGe1GQ1qlpzH_nzuUESnlxwWi3tyGuYgvzmVLIl3Xzwapo9pOOLWe/pub

2. Interview transcription 3:

   https://docs.google.com/document/d/e/2PACX-1vTuF3zgz6XWk5z8UcN2ICX3FwTfs4RRs5iEcchqs5ojOT5iX3qP152BBsMx4jARTn1W6TZwKA_mke8j/pub

3. Interview transcription 5:

   https://docs.google.com/document/d/e/2PACX-1vTOXXJDiMeznRQIzAXFEBQ79YErkYNyoYEvzUZAKhpOSi7Ao5l8EK_uzSFsJXGlpALLVs_X5EmpZo5G/pub

4. Questionnaire:

   https://docs.google.com/forms/d/e/1FAIpQLSfr4miId6J4Htn1sGXwEz6_qQ4k3ebYGmHsBtzgwkxSg1LWfg/viewform