Evaluating pedagogical practices supporting collaborative learning for model-based system development courses

Ewa M. Kabza
Departments of Informatics, University of Oslo & Capgemini Norge
Oslo, Norway
ewaka@uio.no

Arne J. Berre
SINTEF Digital & Department of Informatics, University of Oslo
Oslo, Norway
arne.j.berre@sintef.no

Hani Murad
Department of Informatics, University of Oslo
Oslo, Norway
hanim@ifi.uio.no

Dervis Mansuroglu
Norwegian Labour and Welfare Administration, IT Development
dervis.mansuroglu@nav.no

Abstract

Model-based software development (MBSD) has been widely used in industry for its effectiveness of code generation, code reuse and system evolution. At different stages of the software lifecycle, models -- as opposed to actual code -- are used as abstractions to present software development artifacts. In a university software engineering curriculum, compared to other concrete and tangible courses, e.g., game and app development, these levels of abstraction are often difficult for students to understand, and further, to see models' usefulness in practice. This paper presents an evaluation of pedagogical practices supporting collaborative learning for MBSD courses from experiences of teaching them at University of Oslo. The focus is to answer two research questions: 1) What are the challenges and possibilities when using a collaborative learning approach for teaching modelling and architecture? 2) What are the challenges and benefits of having a holistic approach to MBSD courses in light of the requirements of academia and the needs of industry?
The term "holistic" is understood 1) as an approach that involves human factors (users), technology and processes, 2) as an approach to teaching MBSD courses where modelling for Enterprise Architecture is taught together with System Architecture and Model-Driven Language Engineering. Empirical data was collected through interviews, questionnaires, and document analysis. The paper's research results show that three different course perspectives (Modeling for Enterprise Architecture with Business Architecture, System Architecture and Model Driven Language Engineering) are essential parts of teaching modeling courses, and an industry field study shows that industry sees the potential of having junior architects to provide support to a team and solving basic architectural problems.

1. Introduction

Model-driven architecture and software development is a methodology within system engineering which concentrates on building and using domain models rather than computing concepts. The goal is to create conceptual representations of all matters correlated with all kinds of subjects which can be encountered during the development process, from requirements gathering to the final phase of engineering. The challenge, however, is to create those models in a way that can be understood and applied by different stakeholders; and so that they do not become a hindrance during the project development process [1][2][3].The understandability of models is an important aspect since research shows that applying modelling techniques in various stages of software development processes improves their efficiency and effectiveness [4]. However, if the given model is unclear or misunderstood, it can lead to delays or major failures.

Brambilla et al. (2012) enumerate four reasons that illustrate why creating appropriate models is an important part of the software development process. Firstly, the contemporary world depends on software. Software has become an integral part of human's life, and it requires not only constant maintenance and updates but also innovation and improvements, so it can meet people's needs. Secondly, today's systems are complex. It would be challenging and costly to replace them, so there is a necessity to efficiently maintain systems when they are expanding. These extensive systems are composed of different abstraction layers which are based on different factors, like the requirements and needs of stakeholders, the aims of industry, and technical limitations. Thirdly, the complexity and constant and rapid evolution of technology engenders a deficit of skilled professionals, because the curve of innovation

This paper was presented at the NIK-200x conference. For more information, see http://www.nik.no/.

grows faster than a curve of experts. Fourthly, the world's dependency on software and its complexity drives us to see software development as an element of a composite chain which is interconnected with the non-technical world. Non-technical actors need to understand features and usage of software.

2. Background

The background for this paper is the evolution of teaching a holistic course in model-based system development based on the analysis and evolution of the MBSD course-INF5120 given at the University of Oslo during spring semesters 2018 and 2019. It is also a continuation of a paper presented at Models 2018 [5]. The INF5120 course evolved from the earlier version of the course called Modelling with Objects given at UiO from 2016. Since 2003, an introduction to model-driven architecture (MDA) and engineering (MDE) has been available on the curriculum. The course supports students to work collaboratively in small teams consisting of 3 to 5 people. The pedagogical purpose is to interest students in modelling activities by creating those activities around a comprehensive business case that includes enterprise and software architecture and modelling. The aim is to teach modelling techniques that are directly applicable in an industrial setting based on engaging students in modelling by working on a complete case from business architecture to implementation.

To support students theoretical and practical knowledge and skills the course offers on a weekly basis a theory-based 2h lecture and two 2h hand-on teacher-lead workshops that focus on developing students' skills and problem-solving competence. Students work alone and in groups to solve tasks that are similar to those given in the obligatory assignments and the individual final exam. The obligatory assignments are thought to be a group project to improve students collaborative skills and prepare them to work in teams at work. To support collaborative work within the group, students were encouraged to use various communication tools like Slack, Google Hangouts, UpWave, among others.

The INF5120 is designed to be a holistic course that approaches teaching modelling by combining Enterprise Architecture, System Architecture and Model-Driven Language Engineering together to familiarize students with all stages of the project lifecycle with an aim to provide them a solid base for future employment.

3. Related work

Understanding learning and knowledge-building processes presents multiple challenges for developing new strategies for both classroom and remote learning and for articulating collaborative work practices through the use of different modelling tools for students with different academic backgrounds. The current learning trend for IT courses using technology-supported learning puts emphasis on active student participation [6] and collective meaning-making [7][8][9][10]. Students' commitment and performance competence are promoted through developing a self-understanding of computer science learning as a facet of collaborative work exercise [11]. Such activity is ruled by sets of learning frameworks using various representations [12], theories [8][9], methods [13][14], tools [15], structures [16], communities of interest [17], and an epistemic agency and knowledge-building discourse for uniting systematic knowledge [18][19].

Constructive alignment [21] is a didactic approach where learning activities and student practices are arranged with planned learning goals at the beginning of each semester[22][23]. By exploiting different possibilities for scaffolding collaborative methods [24][25] and interactive technologies in scholarly settings, the INF5120 course focuses on building innovative classroom and remote activities that motivate individual and group participatory exercises in collaborative learning settings. Various researchers, including

Kaasbøll, argue that functional understanding occurs before structural understanding [26][27] and using functional and structural models helps mostly those students who have lower verbal competences [28]. Kaasbøll's competence building model, was adopted for the INF5120 course, It states that functional understanding is accomplished when students can explain the input and output of their actions, while structural understanding is achieved when students are able to use a learned concept as a base for acquiring new notions.

The research also points out that the curriculum for MBSD courses should take into consideration the abilities of different groups of learners [29] and should be adjusted to their level of concept comprehension. Providing adequate tutorials [30] and curriculum supports meaningful interactions between learners when they solve modelling-related tasks. Berre et al. (2018) propose that "*students orient their interactions through fluid structures of activities towards constructing a domain-specific modelling environment and then define model transformations that provide appropriate syntax and semantics to such a user-defined mode*l". Learning MBSD is an intensive and construct-based process where students need to understand, design, implement and modify software systems, with regard to predefined requirements and objectives, by using multiple tools and modelling approaches to describe their information systems architecture at the desired level of detail [31][32]. In [5] and [10] we have discussed how the INF5120 course adopts the three-step model of competence building developed by Jens Kaasbøll [28], focusing on developing practical skills, good understanding of the subject matter , and problem solving capabilities in similar task domains.

The three perspectives of the INF5120 course also relate to three different developments of Body of Knowledge (BOK) for Enterprise/Business Architecture [33][34], Software Engineering and Model-Based Software Engineering [35][36] and Language Engineering [37]. A variety of different course approaches about teaching architecture, modelling and meta-modelling are presented in Berre [5][10]. Based on previous research, the current INF5120 course follows the strategies of MBSD and language engineering with the aim of providing a suitable framework for also teaching MBSD courses in a virtual classroom. It seems that very few other MBSD courses relate either to the use of models in Enterprise architecture and Business architecture, or to engineering for domain-specific languages for enterprise and business architecture, which is very useful as an agile basis for further model-based system and software engineering.

4. Collaboration in practice

Anchored in the different computer supported cooperative work and collaborative work principles and practices (CSCW, CSCL) [38][39][40], students develop their own collaborative ways of work and knowledge sharing using different shared information spaces and platforms. [41][42][43]. To support students' journey in their modelling practices, the INF5120 course also applies the constructivist group work and active learning approaches presented by CSCL activists[21][23][25]. By following a construct-based approach, students are able to design and implement different systems by using a variety of modelling techniques and tools and different levels of complexity [10]. Students are also encouraged to explore modelling concepts by being introduced to a holistic, active learning perspective [14] by working on a project that touches upon techniques and tools related to enterprise and software architecture and modelling that results in the development of executable models.

5. Industrial setting

Familiarising students with business settings when teaching MBSD is an important part of their learning process, because the majority of them will start working in the industry after

graduation. To illustrate the importance of understanding business modelling for IT practitioners this article briefly presents the case of Norwegian Labour and Welfare Administration (NAV) when the lack of understanding of business requirements by architects led to *Moderniseringsprosjektet* and how the organization recovered. NAV is a governmental organization and is allocated ⅓ of the national budget and is the most critical governmental agency because it provides more than 60 different welfare services that cover all the major life stages of citizens and residents in Norway.

Before 2016, the IT governance frameworks implemented were Information Technology Infrastructure Library (ITIL) and a variant of Control Objectives for Information and Related Technologies (COBIT). ITIL was used to build organizational infrastructure while COBIT provided ways to deliver software using this organizational infrastructure. NAV implemented COBIT partially by adopting a 6-stage model related to business demands. ITIL was successful in developing critical systems for the pension reform of 2015 that were delivered on time and budget with few errors. COBIT, moreover, introduced the silos-based model called Plan-Build-Run (PBR). The Plan-Build-Run model was not iterative and the previous phase had to be finished before the next one could start, which in consequence left little space for potential changes. Therefore, PBR resulted in many challenges such as: 1) Container-based architecture that was ineffective due to its complexity, 2) Complex bureaucratic design of bug tracking 3) Tracking systems, causing over-expensive bug fixing that was paid for "by citizens from taxes" (Conway's Law) 4) Lack of collaborative work due to misleading communication and many isolated teams that worked on their own, 5) Poor governance, including IT governance, and staff allocation, 6) Poor time-to-market delivery.

Jørgensen (2015) [44] gives four architecture-related reasons why IT projects in the public sector fail: 1) Not enough understanding of the complexity of the built system, 2) Low ability to describe and evaluate non-functional requirements, 3) Lack of focus on integration and good system/portfolio-architecture, and 4) Lack of risk management and underestimation of risk for projects. A combination of poor governance and cost estimation, weak architectural approach and miscommunication between teams resulted in the biggest scandal NAV was ever involved in - *Moderniseringsprosjektet*, where the organisation invested in poorly designed and architectured IT systems that failed to fulfill work requirements and work flow across the whole of the organization.

In 2016 NAV introduced organizational changes that aimed to make NAV more iteratively agile in order to improve time-to-market. IT practitioners were grouped into self-organised teams where architects became part of the teams and worked collaboratively with developers, testers and designers. Doing proper architectural work is an essential factor when running successful projects. However, it is important to balance how comprehensive this work is. Proper risk management should be ensured, but not so the work becomes a delaying factor due to its complexity [45] Bohem (2011) [46] states that the scope of architectural work depends on the size of the projects and developed systems. He seems to promote a big up front design, saying that crucial systems require safer and more stable architecture from the beginning, whilst smaller projects can adopt agile techniques and improve architecture during various iterations [46]. However, Waterman et al. (2015) [47] say that making big up front design makes it almost impossible to introduce new changes during the project, and Brown [48] advocates that architects should do "just enough up front design" so they know what their goal is and how they are going to achieve it. The big up front strategy, reduces risk and the need for costly changes in later stages. It also indicates whether architecture is implementable. On the other hand, this strategy limits the possibility to make changes during development, and it makes it very hard to implement the system with an agile

methodology [45]. The need for big up front design and architecture emerges with large and very costly projects. A project has defined start and end dates and a budget that the contractors commit themselves to. A project is staffed and financed during its duration period. When the development of the system is finished and handed over to the customer, the project (together with its financing and staffing) ends. Systems built this way have, however, considered to be very stable, with few errors. Nevertheless, this still leads to several problems for NAV, such as lost knowledge and no ownership. More importantly, at the time of hand-over, the technology stack for which the big up front architecture was targeted is often outdated and imposes tough challenges. This usually entails in very costly upgrades. When a new critical system needs to be built, the work in NAV is no longer organized as a project. Instead, the system is broken down to smaller modules and products that allow *Just Enough Architecture* [48]. This strategy is best described as a middle way between Emergent Architecture (no up front design, [47]) and Big Up Front Architecture [45]. Each module/product is split between *cross-functional product teams* that have long-term financing. These teams are staffed with very diverse groups of members that are often a blend of architects, devs, ops, testers, domain experts, security experts and UX/UI designers. Cross-functional teams do just enough up front, but for specific areas, the organization is still doing big up front design/architecture. Examples of such systems are welfare benefits for pension and health.

The NAV case illustrates the importance of developing a good awareness about the need to focus on designing and modelling robust architectural systems, with enough flexibility for future modulations, to meet the ever changing work needs of any organization in the public service sector.

6. Data collection & analysis

The data collection and analysis are based on the triangulation of various qualitative research methods including interviews, document analysis and questionnaires. The main objective is to analyze to what extent the teaching methods, the curriculum and structure of the course are effective in relation to the learning model elements: skills, understanding and problem solving with reference to syntax, semantics and business/context aspects for modelling. By examining different synergies in students' work- flow throughout the semester, the composites of our analysis point to the need of better designs of cooperative work articulation [49] for most student groups and for alternative forms of re-compositions [50] of both assignment modules and teaching practices in our seminars. Modular decomposition [51] facilitates better coupling between different project modules and supports reduced dependencies between various module components. This is highly relevant in MBSD especially that our course students have varied backgrounds in computer science and different student groups use different programming languages and modeling tools in their business software development process. An example of which are Java packages which represent modular systems consisting of well-defined manageable units with clearly defined interfaces among the units. Software modularity brings most benefit to the developing system when the modules are autonomous or independent and the degree of modular dependency reflects the extent to which each program module relies on each one of the other modules. Low coupling between different modeling tools and programs infers higher efficiency and robustness in the executable model. Furthermore, decomposing modeling practices allows for systematic work coordination between different student group members, and the handling of any software changes and updates that might take place throughout the model development process. The aim of our recomposed course design is to offer our students better modeling experiences and to help them design their modules with the goal of high cohesion and low coupling.

Interviews

Interviews with six course participants and four representatives of the industry were held during the past two semesters. The semi structured interviews with students were held during and at the end of courses in 2018 and 2019 after delivering the final exam and semi structured interviews with the professionals were run over the course of a year. Students had different IT backgrounds but most of them had some experience with programming from other courses, but they did not have significant experience with either software engineering or modelling. Interviews with students were transcribed and the interview guide contained questions about students' background, group work, tools, comprehension level and understanding, functional and structural understanding and problem-solving approaches. The interviews were analysed using Malterud's systematic text condensation [52] that resulted in identifying 4 main interview categories and 89 subcategories. Main categories as follows:

Category 1 - *Collaboration processes* were found in 42 quotes, or 18% of the total analyzed quotes.
Category 2 - *Course structure* was found in 67 quotes, or 29% of the total analyzed quotes.
Category 3 - *Tools* was found in 62 quotes, or 27% of the total analyzed quotes.
Category 4 - *Critiques & improvements* were found in 59 quotes, or 26% of the total analyzed quotes.

Category 1 - Collaboration processes

The interview included questions related to group work and collaboration in a team. The goal was to learn more about students' interactions with each other and how those interactions influenced the competence building process. The most frequent subcategories were: group work (29 mentions - 34% of total mentions), task distribution (17 mentions - 20% of total mentions), knowledge/skill/understanding (15 mentions - 17% of total mentions).

Group work: Respondents related positively to group collaboration. They worked out how to cooperate while taking into consideration differences, academic background, or personal fears related to working together. They discussed how they organized work when meeting up and challenges they faced, such as lack of contribution or communication. Some respondents solved potential disagreements by having open dialogue with team members. One of the students found an additional value in working in a team as he learnt about conflict management.

Task sharing: When discussing task distribution, groups showed different approaches. Some groups met up and worked on assignments together, other groups divided tasks among group members and condensed their parts for the final assignment, others still presented a mixed approach. Respondents identified different reasons for dividing assignments, such as time constraints , efficiency, complexity of deliverables and other university obligations.

Knowledge/skill & understanding: Respondents focused mostly on three aspects of the subcategory: information sharing processes, their own deficiencies, and the role of the more skilled colleague in the group. Sharing knowledge is a valuable practice for common growth and scoring well in the exam. Students were aware of limitations in their own competences, so when an issue arose, they tried to solve it on their own or looked to a teacher for guidance.Using other students' experience and having a good information flow had a positive impact both on group work, since it proved more productive when working on assignments, as well as on individual performance in the exam. The most skilled colleagues were also seen as a source of help.

Category 2 - Course structure

The interview included questions related to the course and its structure. The goal was to learn students' opinions about the course and pedagogical practices. The most frequent mentions were of the course curriculum (35 mentions - 34% of total mentions), knowledge/skill/understanding (24 mentions - 16% of total mentions) and hands-on tasks (15 mentions - 10% of total mentions).

Curriculum: When discussing the complexity of curriculum subcategory, students focused on the abundance of course material and meta-modelling. The course curriculum was seen as too dense. Respondents found the curriculum to be valuable and relevant yet so complex and overwhelming that sometimes they felt lost. Respondents commented that many topics are touched upon too briefly , but the nature of those topics allows students to become familiar with different domains, and in consequence, they have more fields to choose from. Students also hinted about the kind of changes that could be implemented to make the course easier to follow, such as color-coding slides or lengthening the course to one year. Meta-modelling was seen as problematic and difficult to understand . Students took previous years' exams to become fluent. Despite being the most challenging part of the course, respondents saw value in learning metamodeling. In 2018, it was hinted that introducing meta-modelling earlier on the course would be beneficial, and in 2019 this change in the course set-up was considered helpful.

Knowledge/skill & understanding: Respondents focused mostly on the importance of individual learning in addition to collaborative learning. Students noticed that collaborative learning was not enough to prepare them well for the exam; it was seen as a knowledge sharing forum.

Hands-on task: The majority of respondents noted the importance of encouraging students more actively regarding various modelling activities from the beginning of the course, since it would have improved their understanding of concepts as well as skills. In addition, it would have developed their problem-solving skills for later stages. Students found it important to differentiate theoretical lectures from hands-on focused group sessions. Practical group sessions would have increased the students' efficiency, since they would have provided students with live tutoring in case the supporting material provided was not enough.

Category 3 - Tools

The interview included questions related to tools and students' interaction with technology. The aim was to see how students used tools in order to solve tasks and support collaborative learning. The most frequent mentions were of the issues with tools (14 mentions - 11% of total mentions) and collaborative tools/cloud (11 mentions - 9% of total mentions).

Issues with tools: Students faced various problems related to tools. Respondents stated that some tools were not intuitive, problematic to be installed and not compatible with their operative systems that in consequence led to spending a significant amount of time on trouble-shooting, making tools work or giving up on a tool and finding an alternative solution. Modelio with ArchiMate was complained for not being a collaborative tool and causing various problems both with installation and use. Node-RED scored positive feedback.

Collaborative tools/cloud: Most of the respondents gave positive feedback about collaboration supporting tools. Slack was perceived as useful and easy to use. For writing students used mostly Google Drive because it was free and respondents were familiar with it. The lack of cloud functionality in case of Modelio/ArchiMate did not allow students to work collaboratively.

Category 4 - Critiques & improvements

During interviews, students were asked to provide critical remarks and discuss potential changes that could improve the course in the future. The aim was to learn what students liked or disliked, and which aspects should be taken into consideration to improve the course. The most frequent subcategories were: employment (15 mentions - 13% of total mentions), industrial setting (10 mentions - 8% of total mentions), changing structure of the course (10 mentions - 8% of total mentions).

Employment: Respondents stressed the relevance of the course for their future employment because students become familiar with various tools that the potential employer might be interested in. Interviewees put emphasis that the course was useful for them because they already use some of the tools in their daily work. Moreover, respondents remarked that finishing a course with a certification would boost motivation since it would be beneficial for them when looking for work.

Industrial setting: Students spoke flatteringly about having a real customer during the course since it gave them a glimpse of real-life situations that happen on a customer's side. It was also appreciated that INF5120 focuses on the industry as well because combining theory and practice is profitable.

Changing course structure: One of the improvements related to the course was an idea to divide it into separate parts. Some students suggested splitting it into the business part and software engineering/meta-modelling part, others saw it beneficial to have one focus on the industry with business modelling and software engineering, and other focus on academic research with meta-modelling. Additionally respondents mentioned that they would prefer INF5120 to be a project-based course because it would be a more efficient way to use techniques and tools in practice.

The interviews with students raised a couple of questions related to the usefulness of the curriculum for future employment, and it has been decided to conduct semi-structured interviews with the representatives of the industry. The interviewees were selected based on their work experience within IT (seniors) and the company they work for (large-sized companies). The reason for choosing professionals from large-sized companies is that most of the graduates will start working in those companies, so the provided information will be relevant for them. It has been learned that the business perspective is important for students to understand because all the projects are approved or rejected based on the business decisions. Moreover, it was suggested that the curriculum could focus less on meta-modelling and more on learning the concepts and software that is already used by the industry.

Document analysis

Document analysis was done for course assignments and incremental project deliveries as well as for the exams in 2018 and 2019. The 4 hours written exam contained three questions corresponding to the three parts of the course, namely Enterprise Architecture (EA), Software Architecture (SA) and MDE and meta-modelling. Analysis of exams showed that students scored well on EA and SA. The questions about MDE and meta-modelling scored lower points indicating less problem-solving competencies in creating a meta-model for a described language. In addition, students often confuse meta-modelling and domain modelling.

The analysis of project deliverables showed progress in understanding the different parts of the course. For the final delivery, students were asked to improve their first and second assignments and some groups showed an improved understanding of the basic concepts covered in the syllabus. However, we noticed that some concepts including Unique Value Proposition, Business Segment, UML and composite diagrams, User Stories were still

unclear, so we held a test exam in the lab three weeks before the final exam. Students were asked to work in groups and Group 1 (2 students) worked on a question 1 related to Enterprise Architecture, Group 2 and 3 (2 students per group) worked on a question 2 related to System Architecture and Group 3 (3 students) worked on a question 3 about MDE and meta-modelling. During the first two hours, students worked on the assigned task, for the next two hours they were asked to present their solutions and discuss them in plenum. The open discussion resulted in a better understanding of the concepts.

Questionnaires

Questionnaires were handed out to all students in both courses. The main question categories were on the student's background, their level of understanding of modeling, programming and system development at the start of the course compared to at the end of the course. 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 comments on what they liked and what they suggest improving in the course. Since these are graduate level courses, our earlier assumption has been that students have had a basic modeling competence before entering the courses. Our closer analysis of the student's background has, however, revealed that many students have relatively little experience with modeling (several of the students are international exchange students and students from local industry)– and a conclusion is thus that we need to elaborate even more on basic principles for good modeling practices in future versions of these courses.

Questionnaire in 2018 focused on the evaluation of modelling tools and techniques whilst in 2019 on a group work. The combined findings are summarized as follows:

- MDLE was the most difficult and time-consuming part of the course. Solving MDLE related tasks caused a lot of frustration among students since they did not have enough practical and theoretical knowledge to be able to make the assignments.
- Delivered assignments showed that some students were not clear on the difference between domain modeling and meta- modeling.
- Business architecture methods show that using Lean Business Canvas, Business Model Canvas – was rather easy, whereas using ArchiMate layers was rather difficult.
- Modelio was not ideally suitable for collaborative project work, since the multi user, repository support was not used in the class. Only one group member could use it at any time thus rendering asynchronous collaboration problematic without using added platforms like Dropbox or Google Drive.
- Students appreciated the industrial setting of the course and inviting IBM as a real customer since it allowed them to become familiar with industry. That experience was helpful for some students during their job searching process.

7. Summary of findings

RQ1: *What are the challenges and possibilities when using a collaborative learning approach for teaching modelling and architecture?*

Based on the data analysis, the following challenges have been defined when observing students in 2018 and 2019:

- students had different motivations for taking the course: some were interested in modelling and architecture, while others needed to take the course the course to get credit points
- many students had different cultural backgrounds and neither English nor Norwegian were their native language
- students attended different study programs and had different academic backgrounds
- students had different learning preferences. Some of them were team players, while

others favoured studying alone
- students had different work ethics that resulted in different levels of contribution
- students had different family situations that had impacts on group work
- students who were more experienced tended to take more tasks on

Similar challenges can be observed in any collaborative learning environment because they are very human. However, three more, tailored to MBSD challenges, have been observed:

1) The majority of students did not have previous knowledge about the domain
2) Some tools (f.e. Modelio, Eclipse) did not perform as expected
3) Few students did not have the possibility to work remotely

ad1) Learning MBSD is a construct-based process where students are encouraged to participate in modelling activities with a view to developing skills, understanding and problem-solving abilities related to enterprise architecture, software engineering and meta-modelling. Construct-based learning is a scaffolding process that pre-assumes that learners are able to associate a new learning concept with one they already knew. In the context of the MBSD course, it has been observed that even though some of the students had prior knowledge related to architecture and modelling, it was not enough to understand meta-modelling

ad2) Even though students understood what a model was supposed to do and knew how to make it, they did not manage to do so because the tools did not perform as expected. This was the case with VDMBee for Business Ecosystem Map (BEM). The majority of students evaluated BEM as easy to apply, whereas VDMBee was rated as difficult to use to create BEM. During interviews, students asked for more hands-on tasks so they could become more fluent in applying techniques and using tools. In these two scenarios, it is therefore important to support students with suitable pedagogical agents [57], so they can acquire a new skill by for example imitation and following guidance [29].

ad3) Collaborative learning is connected with computer-supported collaborative learning that allows learners to work together even when distributed globally. The majority of students complained that not all tools provided for the course supported remote work.

Students criticized Modelio/ArchiMate because it did not have a cloud possibility, so it was impossible to work collaboratively on a tool. Thus, a toolchain should be provided that supports collaborative work both online and in person to improve students' experience. The following benefits of applying collaborative learning principles for MBSD courses have been observed:

a) students improve interpersonal skills
b) students learn how to talk about MBSD and architecture
c) students support each other’s learning journey

a) Encouraging people to work collaboratively on a project gives them the possibility to interact and to improve their interpersonal and managerial skills. Both the survey and the interviews in 2019 showed that students favoured working in a group and consequently improved their cooperation skills.

b) As illustrated in the NAV case, the industry is leaning towards establishing cross-functional teams, where different IT professionals work together. Facilitating group work during the course allows students to become familiar with terminology and to explain concepts in an understandable way.

c) Students support each others' learning journey. Different people understand different topics better or worse and group meetings were considered a knowledge sharing forum

RQ2: *What are the challenges and benefits of having a holistic approach to MBSD courses in light of the requirements of academia and the needs of industry?*
Challenge 1: Having a focus on both business and technology in a dense curriculum of methods, techniques and tools may be overwhelming for some students. Their motivation may drop and they may just want to pass the course without focusing much on developing knowledge or skills.
Solution 1: This could be to adapt course length to the complexity of the curriculum. Many students mentioned that the course was too short in comparison to the length of the curriculum. A one-year course would give more time to students to develop skills, understanding and problem solving competences.
Solution 2: A couple of respondents mentioned that splitting the course into two or three courses could also be a possible solution. By splitting into smaller parts, the course may lose its holistic approach, so what could be done is to create a study path based on the three parts of the course and its curriculum.

8. Recommendations and conclusions

Better focus on designing and implementing novel learning spaces that work across both physical and virtual domains simultaneously. The conceptualisation of hybrid learning spaces is very relevant to today's COVID- 19 Implications and the need for revolutionary teaching and learning practices across all spectres of educational settings. Underpinned by our empirical findings, we advocate a holistic approach, embedding novel collaborative learning designs and technology supporting synchronous and asynchronous students' activities in their attempt to create meaningful architectural and functional business models.

More hands-on activities: Students pointed out that it would help them improve if they were given small, practical tasks during the lectures and/or group sessions. To achieve this, more ad-hoc exercises could be given to students during the lectures and labs when the students could work individually or in groups, solve the tasks and get immediate feedback from the teacher, who could also encourage the group discussion.

Better tutorials for software: Students said that it took them a lot of time to install, learn and troubleshoot the software and they had to look up some tutorials online. They stated that due to spending a lot of time doing things not related to the assignments, they experienced a drop in motivation when working on assignments. To prevent this, the course could provide easily accessible support materials (written or video tutorials) for students to access when needed. The support materials could give sequential instructions on how to use the software.

Better support materials for models: Students stated that sometimes they got stuck on some tasks because they did not know how to solve them due to lack of skills and understanding. They asked for more guidance, so the course could provide some example video tutorials in which the instructor solves a similar task to the one students are supposed to do.

More interactive group sessions: Students asked to have more hands-on exercises during the group sessions. Currently, the group teacher changes every year and a new one is picked from among those students who took the course before and scored a final grade of A. The disadvantage of this is that there is a lack of consistent pedagogical practices. To solve this problem, there could be a fixed group teacher (perhaps a PhD candidate) with solid

practical competence in modeling tools and good pedagogical practices.

Providing cloud-based tools: Students pointed out that they experienced issues with software installation and its collaborative aspect. That is why to ensure a good learning experience which would allow for collaborative and remote work, the University could provide cloud-based tools. This may take up to two years due to the cost of licensing. This cost has to be included in the budget, and getting approval may take up to a couple of months.

Providing certifications: Students suggested that having a certificate at the end of the course would be beneficial for them when looking for work. Professionals pointed out that having TOGAF, BMPN, ArchiMate and technology certification (either Azure or AW), would land students a job right away.

Creating a study path: In the long run, the course could be converted into a case-based study path that could prepare students to work as architects after graduation.

These concluding recommendations will be related to in the planning of future versions of the considered courses.

Acknowledgements

We would like to express our appreciation to Prof. Shihong Huang from Florida Atlantic University for her earlier collaborations on course offerings and valuable and constructive suggestions during the writing of this article.

REFERENCES

[1] Frankel, D. S. (2003). Model Driven Architecture: Applying MDA to Enterprise Computing, John Wiley & Sons, ISBN 0-471-31920-1.
[2] Poole, J.D. (2001). Model-Driven Architecture: Vision, Standards And Emerging Technologies.
[3] da Silva, A.R. (2015). Model-driven engineering: A survey supported by the unified conceptual model. In Computer Languages, Systems & Structures, Volume 43, October 2015, Pages 139-155.
[4] Brambilla, M. & Cabot, J. & Wimmer, M. (2012). Model-Driven Software Engineering in Practice. Morgan and Claypool Publishers, ISBN: 9781608458837.
[5] Berre, A.J., Huang, S., Murad, H. & Kabza, E. (2018). Evolution towards teaching a holistic course in model-driven system development: modeling for enterprise architecture with business and system architecture and platform-based development. Pages 62-66. 10.1145/3270112.327013.
[6] Sfard, A. (1994). Reification as the Birth of Metaphor. For the Learning of Mathematics, 14(1), 44-55.
[7] Hakkarainen, K., Paavola, S., & Lipponen, L. (2004). From Communities of Practice to Innovative Knowledge Communities. Lifelong Learning in Europe, 2, 75- 83.
[8] Stahl, G. (2013). Theories of collaborative cognition: Foundations for CSCL and CSCW together. In S. Goggins & I. Jahnke (Eds.), CSCL@work. (Vol. #13 Springer CSCL Book Series). New York, NY: Springer.
[9] 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.
[10] Berre A.J, Huang S., Murad H., Alibakhsh H. (2018). Teaching modeling for requirements engineering and model-driven software development courses, Computer Science Education, 28:1, 42-64.
[11] Cole, M., & Engestrom, Y. (1993). A cultural-historical approach to distributed cognition. In G. Salomon (Ed.), Distributed Cognitions: Psychological and Educational Considerations (paperback 1997 ed., pp. 1 - 46). Cambridge, UK: Cambridge University Press.
[12] Ainsworth, S. (2006). DeFT: A conceptual framework for considering learning with multiple representations. Learning and Instruction, 16, 183-198.
[13] Prince, M., & Felder, R.M. (2006). Inductive teaching and learning methods: Definitions, comparisons, and research bases. Journal of Engineering Education 95 (2): 123–38.
[14] Linn, M. C. & Eylon, B.-S. (2011). Science Learning and Instruction – Taking Advantage of Technology to Promote Knowledge Integration. New York: Routledge. Ch. 8: 186-217.
[15] Rasmussen, I., & Ludvigsen, S. (2010). Learning with computer tools and environments: a sociocultural perspective. In K. Littleton, C. Wood & J. K. Staarman (Eds.) International Handbook of Psychology in Education. (pp. 399- 435). UK, Emerald.
[16] Brambilla, M. & Fraternali, P. (2014) Interaction Flow Modeling Language: Model-Driven UI Engineering of Web and Mobile Apps with IFML, 1st Edition. ISBN: 9780128001080, The MK/OMG Press.
[17] Mørch, A.I., Andersen, R., Fugelli, P., Ponti, M. & Lahn, L.C. (2013). Communities of interest: Newcomer participation in open online collaborative software development and learning communities. Technical report, Department of Education, University of Oslo.
[18] Bereiter, C., & Scardamalia, M. (1989). Intentional learning as a goal of instruction. In L. B. Resnick (Ed.), Knowing, learning, and instruction: Essays in honor of Robert Glaser (pp. 361-392). Hillsdale, NJ: Lawrence Erlbaum Associates.
[19] Bereiter, C., & Scardamalia, M. (2013). Self-organization in conceptual growth. Practical implications. In S. Vosniadou, (Ed.), International handbook of research on conceptual change (2nd ed.,) (pp. 504-519). New York: Routledge.
[20] Säljö, R. (2010). Digital tools and challenges to institutional traditions of learning: technologies, social memory and the performative nature of learning. J. Comput. Assist. Learn. 26(1), 53–64.
[21] Biggs, J. & Tang, C. (2007). Teaching for Quality Learning at University. McGraw Hill Education, England, third edn.
[22] Bloom, B. S.; Engelhart, M. D.; Furst, E. J.; Hill, W. H.; Krathwohl, D. R. (1956). Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain. New York: David McKay Company.
[23] Armarego, J. (2009). Constructive Alignment in SE Education: Aligning to What? In: Ellis, Heidi J. C, Demurjian, Steven A & Naveda, J. Fernando, (eds.) Software engineering: effective teaching and learning approaches and practices. Information Science Reference, Hershey, PA, pp. 15-37.
[24] 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.
[25] Golding, C. (2009). The Many Faces of Constructivist Discussion. Educational Philosophy and Theory. 43. 467 - 483.
[26] Stamatova, E. & Kaasbøll, J. (2007). Users' Learning of Principles of Computer Operations. Issues in Informing Science and Information Technology. 4. 10.28945/951.
[27] Horne, R., Grant, T. & Verghese, K. (2009). Life Cycle Assessment: Principles, Practice and Prospects. 10.1071/9780643097964.
[28] Kaasbøl, J. (2013-2018). Developing digital competence - learning, teaching and supporting use of information technology. Report, University of Oslo.
[29] Nardi, B.A. & O'Day, V.L. (1998), "Application and implications of agent technology for libraries", The Electronic Library, Vol. 16 No. 5, pp. 325-337.
[30] Thomas, P. & Carswell, L. (2000). Learning through Collaboration in a Distributed Education Environment.
[31] Krogstie, J. (2012). Model-Based Development and Evolution of Information Systems. A quality approach, Springer, ISBN 978-1-4471-2936-3.

[32] Berger, P. L., & Luckmann, T. (1966). The social construction of reality. Garden City, NY: Anchor.
[33] BIZBOK Business Architecture Body of Knowledge.
[34] EABOK – Enterprise Architecture Body of Knowledge.
[35] SWEBOK – Software Engineering Body of Knowledge.
[36] MBEBOK - Model-based Software Engineering Body of Knowledge - Ciccozzi F., Famelis M., Kappel G., Lambers L., Mosser S-, Paige R.F., Pierantonio A., Rensink A., Salay R., Taentzer G., Vallecillo A., and Wimmer M.. . 2018. "Towards a Body of Knowledge for Model-Based Software Engineering". In ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems (MODELS '18 Companion), October 14–19, 2018, Copenhagen, Den-mark. ACM, New York, NY, USA, 8 pages.
[37] SLEBOK – Software Language Engineering Body of Knowledge.
[38] CSCL In: Seel N.M. (eds) Encyclopedia of the Sciences of Learning. Springer, 2012.
[39] Layzell, P., Brereton, O. P. & French, A. (2000). Supporting collaboration in distributed software engineering teams. Proceedings of the Seventh Asia-Pacific Software Engineering Conference. Los Alamitos, CA: IEEE Computer Society.
[40] Grudin, J. (1994). CSCW: History and focus. IEEE Computer, 27(5): 19–26.
[41] Guzdial, M., Ludovice, P., Realff, M., Morley, T., Carroll, K., & Ladak, A. (2001). The challenge of collaborative learning in engineering and math. In Proceedings of IEEE/ASEE Frontiers in Education (FIE) 2001 Conference. IEEE, Reno, NV.
[42] Sharp, J.J. (1991). Methodologies for problem solving: An engineering approach, The Vocational Aspect of Education, 42:114, 147-157.
[43] Salomon, G. (1992). What does the design of effective CSCL require and how do we study its effects?. SIGCUE Outlook, 21(3): 62–68.
[44] Jørgensen, M. (2015). Suksess og fiasko i offentlige IT-prosjekter: En oppsummering av forskningsbasert kunnskap og evidensbaserte tiltak. Oslo: Simula Research Laboratory/Universitetet i Oslo/Scienta.
[45] Dørun, I.K. (2017). En smidigere arkitekturprosess - fra «hviskeleken» til felles arkitekturforståelse? Master thesis at UiO.
[46] Boehm, B. (2011). Architecting: How Much and When. In A. O. & G. Wilson (Ed.), Making Software - What Really Works, and Why We Believe IT (pp. 161–185). Sebastapol, California, USA: O'Reilly Media.
[47] Waterman, M. & Noble, J. & Allan, G. (2015). How Much Up-Front? A Grounded Theory of Agile Architecture. Victoria University of Wellington.
[48] Brown, S. (2018)., Software Architecture for Developers. Volume 1. Technical leadership and the balance with agility.
[49] Schmidt, K. & Bannon L. Taking CSCW Seriously. Supporting Articulation Work, 1992. Computer Supported Cooperative Work: The Journal of Collaborative Computing, vol. 1 no. 1, pp. 7-40.
[50] Grinter, R.: Recomposition: Coordinating a Web of Software Dependencies, 2003. Computer Supported Cooperative Work: Journal of Collaborative Computing, 12(3): 297-327.
[51] Parnas, D.L. (1972): On the Criteria to be Used in Decomposing Systems into Modules. Communications of the ACM, vol. 15, no. 12, pp. 1053–1058.
[52] Malterud, K. (2012). Systematic text condensation: A strategy for qualitative analysis. Scandinavian journal of public health. 40.