Shared Concept for Teachings in Universities and Trainings in the Industry

A Case Study for Teaching the Unified Modeling Language

Arne Noyer[1,2,3], Joachim Engelhardt[3], Padma Iyenghar[1], Florian Pramme[3], Elke Pulvermueller[1], and Gert Bikker[3]

[1] Institute for Software Engineering, University of Osnabrueck, Germany
[2] Willert Software Tools GmbH, Bueckeburg, Germany
[3] Institute for Distributed Systems, Ostfalia University of Applied Sciences, Wolfenbuettel, Germany

Abstract. For an identical topic, there are differences between trainings in industry and lectures at universities. Among other boundary conditions, they have diverse audiences and goals. Nevertheless, for teaching the same topic, of course, not everything is completely different. Therefore, this paper discusses an approach for a shared concept for teaching in universities and trainings in the industry. In this approach, the parts of a course are divided into a module-based structure. This enables to flexibly select modules based on individual boundary conditions for conducting a training/lecture. The industry and the university can share the same repertoire of modules. Furthermore, the industry benefits from up to date knowledge about theories from universities. On the other hand, universities benefit from experiences and practical applications. An evaluation of this approach is conducted on a case study based on teaching the Unified Modeling Language (UML) both at the university lecture and an industrial training.

Keywords: Teaching, Lectures, Trainings, Unified Modeling Language (UML), Theory and Practice, Shared Concept, Module-Based Approach

1 Introduction

Nowadays, there are a variety of courses offered at the universities covering the many facets of software development. On the other hand, in the rapidly growing field of Information Technology, software engineers in the industry are expected to regularly update their knowledge. To address this aspect, many training programs are offered by the industry both for beginners as well as experienced employees. While lectures and training programs often discuss similar topics, there are several differences between teaching at universities and in industrial training programs. As stated in [1], universities and the industry have different strengths for educating the audience: The universities are adept at updating knowledge through studies and research. On the other hand, the industry provides a different perspective through work experience from various groups. Hence, it is

intuitive to perceive that there exists mutual advantages in collaborating teaching in universities and training programs at the industry. In this paper, a shared concept for teaching at the university and training in industry, supported by a module-based structure is proposed.

Among the numerous topics in software engineering, the application of model-based approaches for embedded software development is a challenging and an emerging field in the recent years. This paper concentrates on evaluating the proposed approach on a case study based on teaching the Unified Modeling Language (UML), which is among the most widely used modeling language for model-based development of embedded software systems. However, creating course material for teaching the UML in the university and training in industry comes with its own challenges. For instance, a course material needs to be created for educating different audiences addressing diverse goals/challenges. Further discrepancies such as

- using different tools at industry and academia
- focus on theory and creating software for a windows-based machine in university, in contrast to focus on creating software with UML for embedded systems in industry

need to be addressed while preparing course materials.

This implies that some topics from the course materials may need to be elaborated for academia/industry and vice versa. Also, there may be a need to elaborate on specific variants of some sub-topics.

The approach proposed in this paper, with its underlying module-based structure, allows to flexibly adjust and deliver a course satisfying the various boundary conditions. An evaluation of the proposed approach is conducted on a case study based on teaching the Unified Modeling Language (UML) both at the university lecture and an industrial training. Experience reports from the case study and evaluation results are discussed towards the end of this paper.

The remaining of this paper is organized as follows. Section 2 presents related work. Experiences with independently teaching the UML at a university and in the industry are analyzed in section 3. In section 4, the courses are combined, resulting in collaboratively created and shared course documents. Therefore, a module-based approach is proposed. Section 5 analyses the usage of the concept for teaching the UML in a case study and critically evaluates the approach. The paper is concluded in section 6.

2 Related Work

The process of university-industry collaboration had already been put to practice several decades ago [2]. A collaborative effort for teaching software engineering discipline is also not anew. For instance, a simulation-based software project management course has been discussed in [3]. Similarly teaching the UML for a variety of audiences has been discussed in [3], [9]. Some of the challenges/topics

addressed are, essential aspects for teaching UML in general, modeling concepts used, different audiences being addressed, problems with tools, choosing a methodology for teaching and conducting examinations.

Further, in [7], the authors discuss the best practices for teaching UML-based software development. They analyze topics such as preparing software developers to effectively use the UML in real-life projects. An experience report for teaching the UML for a group of thousand students and its related challenges is presented in [8]. While there are several related work pertaining to the topic of this paper, addressing various interesting topics, their experience reports and conclusions seem to bolster the following opinion: University-industry collaboration helps universities improve teaching for training high quality talents for enterprises, which gives enterprises more stimuli to supporting the improvement of university teaching [2].

The focus of this paper is not only on analyzing and addressing various audiences, but also to propose a general shared concept for teaching and educating the audiences at both the university and the industry, for an identical topic. For instance, the same course materials can be shared between both the parties. However, a part of the course material, teaching and educational resources can be tailored to meet one's own individual needs and address specific directions, either in the industry or academia. To achieve this goal, a module-based approach is presented and elaborated with a case study for collaborative teaching among university and industry.

3 Experiences with Typical Lectures and Trainings

This section elaborates on the differences between typical lectures in the universities and the trainings in the industry. It first discusses the structures of typical lectures and trainings. This is followed by elaborating on the aspects pertaining to teaching the UML at a university and in the industry. This section concludes with a comparative study.

3.1 Lectures in Universities

Lectures work best when they are in the service of the appropriate learning objectives, especially to deliver cutting-edge information which supplements or enhances reading. Some of the typical characteristics for the lectures in the universities are listed below:

- Often the course materials concentrate on elaborating on the theory, rather than demonstrating a practical solution. The focus is thus to provide a base/overview for a theoretical concept, instead of providing best practices to handle common problems in real-world projects. Typically the lecturer lacks significant industrial experience and not trained for practically focused course work. Thus the lecturer aims at promoting theoretical understanding of particularly difficult concepts and synthesizing information across a range/variety of material.

– A major goal for the staff delivering the lecture at the universities is to persuade the students and motivate them to learn the material. Creating or engaging interest in a new area for the students is imperative on the part of the staff delivering the lecture. From our own experiences, it can be stated that the attention of students vary during lectures in typically predictable ways. For instance, the attention to lecture is high during the first few minutes, followed by a drop and stagnation for the remaining part of the lecture. However, towards the end of the lecture, the students tend to pick up attention and also perhaps engage in Q&A sessions [10]. Thus lecture materials should be made more interesting emphasizing relevance and citing examples.
– Typically, the presentation slides are updated with current topics, during each semester. However, the quality of slides differs between lecturers and sometimes may not satisfy standards in presentation techniques.

As an example for teaching at universities, the lecture *UML for Systems Engineering* at the Ostfalia University is analyzed. In this class, students learn basic theories about the UML as well as how to use some of the well established UML diagrams. Diagrams which are discussed are, use case diagrams, sequence diagrams, class diagrams, object diagrams and structure diagrams. The class is structured in two parts. At the beginning of the semester the students learn the mentioned diagrams and how they can be used during a software development process based on the V-Model [17]. The second part consists of a practical exercise, where the students practice the theory to develop a small software. The exercise is separated into two to three milestones. After each milestone, the results are evaluated and commented by the lecturer.

During a semester, the lecture takes place weekly (approx. 14-16 weeks). The available time each week is three hours.

3.2 Trainings in Industry

Typically, the goal of a training program in the industry is to provide the participants with the basic working knowledge of a certain topic, thereby enabling them to start working hands-on after the completion of the training. Therefore, the training must focus on certain use cases and should not solely concentrate on a theoretical stand point. Often, because of the limited time, it is not possible to discuss all the theory behind a topic. Trainings take place on a few, consecutive days. Unless a typical training program is concise, it becomes unenterprising for the participants and too expensive (both in terms of time and money).

Let us consider an example of a training program, conducted at Willert Software Tools GmbH (WST) [11]. The training program is titled "Embedded UML Startup Training" with a duration of two and a half days [13]. As the name suggests, the focus of this training is on applying the UML for modeling software for embedded systems. The course is typically suited for embedded software engineers who envision using a model-based approach for developing embedded software. In such an approach, based on the requirements, models

are specified in the design phase using modeling language diagrams (e.g. UML diagrams) from which source code is generated automatically to run on the embedded target platform. Thus, a major goal of this training program is to provide the participants hands-on knowledge on the approach described above. As a result, the participants, must be able to create executable software for embedded systems, starting with specifying design model using UML diagrams.

In some cases, it may also be necessary to deviate from the standardized theory in trainings, to achieve its goals. One goal could be to learn how to work with a certain tool, which may deviate from a standard and uses its own, proprietary meta-model [5]. Since the focus is to immediately be able to work with the new knowledge, most trainings are deeply connected to a tool.

Trainings are very practically oriented, so that participants can employ the knowledge gained in real-life scenarios, after the end of the training. Therefore, the usage of many practical exercises is beneficial. In the Embedded UML Startup Training, there are many practical exercises, which the participants do themselves, between learning theory. On the other hand, since a significant amount of time is spent on the practical exercises/examples, the time spent on discussing the theory is significantly reduced (unlike the lectures in the universities).

A major challenge for trainings is the different background of the participants. Their levels of experience vary widely. Very experienced software developers may have their own established ways of software development, some may be biased about certain ideas and may not be open-minded for new suggestions. Some people may already have experience with the topic of the training, while others may know nothing about it. For the Embedded UML Startup training, most of the participants have significant experience in object oriented programming while there are a few of them with a background only in procedural programming.

In a training, some participants have to be convinced to work with what they have learned. Different backgrounds have to be considered, which results in the need to flexibly adjusting the training. If all participants are already experienced with the topic, there may be more time for discussing theory. Nevertheless, the main goal is to provide the participants with a first-hand-experience on working with a real-life project, apart from the theoretical background. This would enable them to confidently engage in solving more complex real-world projects in their respective work environment. The approach proposed in this paper aims to address the various challenges outlined above.

3.3 Comparison of Teaching the UML at the Ostfalia University and at Willert Software Tools

In this section, a comparative study is performed on the various aspects pertaining to educating the audiences via teaching in the universities and training in the industry. The inferences obtained form this comparative study are used as a basis for proposing a combined approach and a shared concept towards collaborating the lectures and trainings.

Table 1. A comparison of various aspects in lecture and training

	Lecture in university	Training in industries
Focus on theory	standardized theory	may deviate from standard theory, notation elements used by tools
Anticipation	upcoming trends from research	can be applied immediately
Experience from audiences	mostly no prior knowledge about the topic	experience varies widely, different backgrounds have to be considered
Experience from speakers	based on theory and research	based on practice and customer's feedback
Main goal	deep knowledge about theory	practical application in real-life projects for embedded systems
Material	up-to-date theoretical content based on text books and research findings	well-proven application
Duration	approx. 45 hours distributed over 14-16 weeks (e.g. 3 hours/week)	approx. 20 hours distributed over two and half days (e.g. 8 hours/day)
Structure	approx. 1/3 theory + approx. 2/3 practice	approx. 1/2 theory + approx. 1/2 practice
Course content	UML with almost all diagrams, UML during a software development process	a selection of diagrams with most practical relevance, special subtopics regarding embedded systems, code generation, practical examples for hands-on-experience

Table 1 illustrates the differences between the lectures and trainings. While the inputs for this table are based on the author's typical experiences, they are also bolstered by experiences from teaching UML at the Ostfalia University and training at Willert Software Tools. The inferences from the comparative study are summarized below.

Universities, typically conducting cutting edge research, are often able to produce lecture materials with updates and pointers to state-of-the art theoretical methodologies. On the other hand, the training materials from industry often provide significant pointers to best practices for practical application scenarios. Furthermore, industry trainings may deviate from standards and are more focused on working with specific tools. Among other aspects, this is because both sides have different goals. The goal of universities is to educate the students with deep knowledge about the theory. Instead, trainings often aim on applying the theory on real projects with specific tools. Especially, if these tools are quite complex, trainings may even be more like tool trainings. At universities, tools are only used as an example.

Next, the average experience level of the audiences is different in lectures and trainings. Most students have no prior knowledge about the topic of a course (e.g.: an undergraduate course). Nevertheless, prior knowledge about related topics from other courses can be expected (e.g. a course on object oriented programming). The experience level of the audience at industry trainings varies

widely and they have different backgrounds. Not only the audiences, but also the speakers have different experience levels. While lecturers at universities are usually not familiar with real projects as much as trainers for professionals, the experience of lecturers is more based on theory instead of practice.

Differences can also be identified in the contents of lectures and trainings. As the university has a broader view on theory, the course contains almost all diagrams of the UML and explanations on how to use them in a software development process. The industry again focuses on practical relevance and therefore trainings contain only a selection of the diagrams, supplemented by special subtopics regarding embedded systems and code generation.

Last but not the least, the available time and structure of the courses can be compared. While lectures are organized in semesters divided into a couple of hours a week, the trainings are done in a few consecutive days. Of course, this affects the quantity of the contents on both the sides.

Although there are differences between teaching the UML for students and professionals, there can be a co-operation for creating a course to address both the audiences. This paper presents an approach (in the next section) for addressing the aforementioned aspects, resulting in a shared concept for teaching in the university and training in the industry.

4 An Approach for a Combined Concept

An approach towards developing a combined concept for creating educational materials for both lectures in universities and training in industry is discussed in this section. During the course of this discussion, relevant example scenarios, experiences and insights from teaching an identical topic (i.e., a course/training on UML) at the Ostfalia university and Willert Software Tools are cited and elaborated.

First and foremost, in order to realize a combined concept and a cooperation for creating course materials for an identical course at both the academia and industry, it is essential to have shared course materials. This would result in saving expenditure of resources such as time spent for creating and maintaining the course materials. Furthermore, such a shared concept would also provide the essential advantage of benefiting from different perspectives/view points on the same topic. On the other hand, challenges arise because of the differences between the individual courses, but much of the course contents are similar.

Next, we propose that in order to meet the individual needs of the cooperating partners, a module-based approach is suitable. This provides each partner, in the cooperation, with the flexibility to combine a selection of the modules to create a variant of the UML course, tailored to meet its specific audience and goals.

The aforementioned aspects are elaborated further in the following.

4.1 Shared Course Materials

The combined concept in general also includes sharing the course and educational material. This means that presentation slides are used in common as well as

additional material like handout papers containing more detailed information to each slide set. The main advantage is the ability to reuse documents and share out the workload for creating slides and handouts among cooperating partners. Also graphics and pictures for the slide design and theory illustration are used in common.

Even if the advantages are quite clear the challenge of creating those documents is to satisfy the needs of both parts. Some topics can be handled easily because they are among the partners in the cooperation as described in 3.3. For some significant differences a compromise has to be arrived at, in creating the course materials. Nevertheless not always a compliance can be determined, because the differences are too significant for example when using different tools to practice the UML in exercises. This requires two documents for each case that can be thought of as variants of a certain topic. If possible those variants should be avoided and instead split into different modules as described in 4.2.

Most of the material will already exist in any kind on both sides, assuming that trainings and lectures were practiced before deciding to get them together. These documents can be merged to create a solid base for the course materials. Afterwards, it may be necessary to further adapt them to satisfy all partners.

While creating and maintaining documents, both partners benefit from each other's experiences. The industry brings in the practically oriented examples, while the university cares about producing up-to-date materials based on current research results. Both already bring in their knowledge and experience on how to create the documents and slides with an attractive design. Because of team intelligence, combining this knowledge can only result in a higher quality of the results.

To organize the documents, an easily accessible storage should be chosen. For instance, this could be cloud based storage providers such as Dropbox [12], or also internal document management systems. The availability and maintainability of the documents for all the partners is very important for this aspect. Furthermore, security aspects might be considered.

4.2 Module Based Structure

In software development, the reuse of software modules is commonly employed [14], [15]. It results in better productivity, maintainability and quality. Furthermore, the modules can be combined differently to create several variants of a software. Thus, the module-based approach is also ideally suited for trainings and lectures.

The partitioning of trainings and lectures into modules is based on different aspects. Some modules are depending on the experience levels of the audience. This is particularly useful for trainings, because the experience level of the participants often varies. If a training is carried out more than once, the speaker can individually combine the modules based on the average experience level of the audience. For instance, there is a module in the Ostfalia lecture resp. the Willert UML training for discussing basics about the object oriented programming paradigm. If the audience mainly consists of software developers, who have

never used object oriented programming before, it is highly advisable to use this module. On the other hand, these software developers could have much experience with developing software for embedded systems. Then, some modules for discussing certain aspects of embedded software development could be left out. Of course, creating software for embedded systems via UML must not be the main goal of a such a course.

The goal of a course is another point that leads to different modules, which can be flexibly used during teaching. In trainings, the attendees usually want to learn how they can create embedded software by using the UML, while at the Ostfalia University, the focus is on the theory behind the UML.

Another difference is that at the Ostfalia the UML tool Merapi-Modeling [6] is used, while at Willert Software Tools, other UML tools (i.e. IBM Rational Rhapsody [16]) are used for the trainings. This leads to completely different modules or variants of a module, for discussing tool-specific behaviors as shown in Fig. 1. Furthermore, examples and getting started like handouts are tool specific. Therefore, a module in the context of this paper always consists of slides together with examples and handouts.

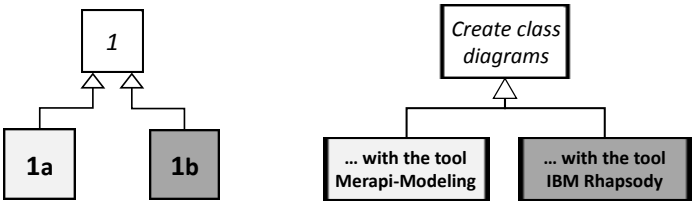

Fig. 1. Variants of a course module

The most obvious reason for creating modules is to divide a topic into subtopics. In the context of teaching UML, separate modules can be created for the different UML diagrams. Furthermore, modules can be created for discussing special topics like using the UML during a development process.

Despite the fact that there are individual modules, it does not necessarily mean that there cannot be a correlation between them. For example, while teaching the UML it has been found to be useful to create an example for realizing a software, which is developed further in different modules. Of course, modules must not be highly dependent on another module. This can be realized by using different modules to address different aspects of a system or to create different views on the same system. The benefit of this is that the participants do not have to spend as much time on analyzing different examples.

Fig. 2 illustrates the module-based approach. The modules are shared in a cloud-based system. Each partner can select which modules they want to use for their course. As shown in Fig. 2, the majority of the modules is used by both the lecture and training. From the top to the bottom, they are ordered as they may be used for courses. At the beginning of the courses, there are modules with

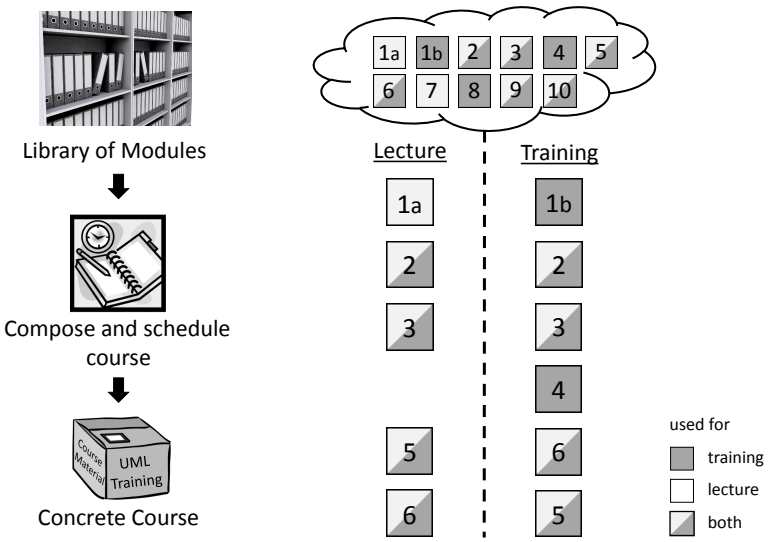

Fig. 2. Usage of modules for lectures and trainings

the identifiers *1a* at the left side and *1b* at the right side. This indicates, that there can be different variants of module *1* (Fig. 1). These may consider different backgrounds of the audiences. Afterwards, on both the sides the modules *2* and *3* are presented. Then, module *4* is only used for the industry training. This can be a module which focusses on topics for embedded software development. Afterwards, both use the modules *5* and *6*, but in a different order. The order of the modules does not have to be the same for the courses.

Because of the individuality of certain trainings and lectures there seem to be a lot of differences between them, even if the same topic is taught. Exactly this is the reason what makes a module-based approach so valuable. Modules can be flexibly selected by a speaker for each training or semester in a university to perfectly match the needs of the audience. Furthermore, if not always the same time is available for teaching a whole topic, the most important modules for achieving the goals of a course can be used.

The Structure of the Modules is divided into different parts. As mentioned above, a module for the UML courses consists of slides, examples, practical exercises and handouts (cf. Fig. 3). Fig. 3 also depicts a typical workflow for using the content parts of a course module.

In addition to the parts of the workflow handouts are given to the audience. These contain step by step manuals and everything that was discussed during the module. For instance, these can immediately be used by the audience, to get themselves used to a certain UML tool. There is a huge difference between showing the audience how they can do something and letting them do it themselves.

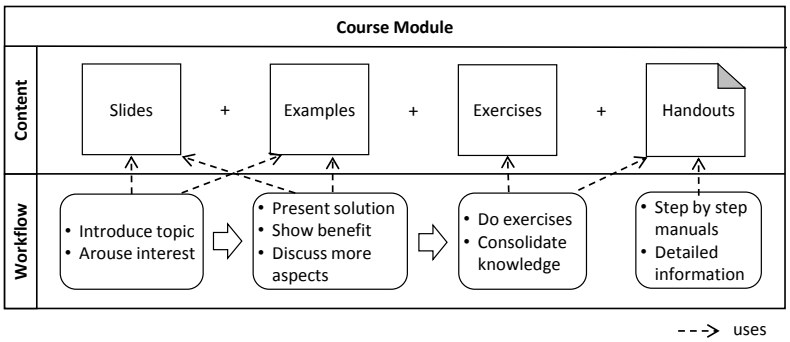

Fig. 3. The content of course modules

4.3 Practical Exercise on all Modules

At the end of a course it is useful to provide an evaluation using exercise questions, which addresses the contents of several modules. By doing this, the participants have to combine what they have learned. For the UML courses, they should be able to create a software by using the UML themselves with different diagrams - and not only to use the UML diagrams individually. Of course, if a final exercise is based on the content of several modules, all these modules had to be discussed before. In order to solve this issue, the exercise itself has to be designed with this in mind and should consist of sub-tasks, which address different modules.

Further, there can be different variants of an exercise. At the Ostfalia University and Willert Software Tools there is an exercise for creating a software by using the UML, which in one case shall be executed on a Windows machine and in the other case on an embedded system. The functionalities of the resulting software are the same.

5 Evaluation of the Case Study for Teaching the UML

The approach proposed in this paper has been applied already on an experimental basis, for about one and half years. An overview of the experience reports and evaluation results collected during this study are presented and discussed in this section.

All created modules for UML courses are depicted in Fig. 4. Many of the modules are used in common. These modules primarily contain important UML topics and diagrams without being specific to a modeling tool or addressing certain areas of application. Modules, which are specific to those aspects, are mainly used by only one partner.

Modules are flexibly combined to create courses for lectures and trainings. Furthermore, the sequence can also be adapted to the special needs of a

1	What is UML	10a	Introduction to Rhapsody	17	Introduction Target Debugger
2	OOP – Classes and Objects	10b	Introduction to Merapi	18	Interrupts
3	Overview Diagrams	11	Introduction to Keil IDE	19	Test Conductor
4	Statecharts	12	Instantiation in Rhapsody	20	Frameworks
5	Sequence Diagrams	13a	Animation in Rhapsody	21	Profiles & Properties
6	Activity Diagram	13b	Animation in Merapi	22	Eclipse integration
7	Use Case Diagrams	14	Final Exercise	23	Configuration Management
8	Composite Structure Diagram	15	Architecture Design	24	SysML compact
9	UML in V-Model Phases	16	Mixing Model & Code		

used for
training
lecture
both

Fig. 4. Modules for UML courses

course or its attendees. This is especially advantageous, while conducting trainings in industry. For example, including configuration management in the training was spontaneously necessary, when there was a question raised by an attendee in the training session. In this case, even if the module was not planned to be part of the training, it could be easily accessed and presented to answer the participant's question. An example for concrete courses during the case study is shown in Fig. 5. This figure also illustrates the different scheduling of the courses.

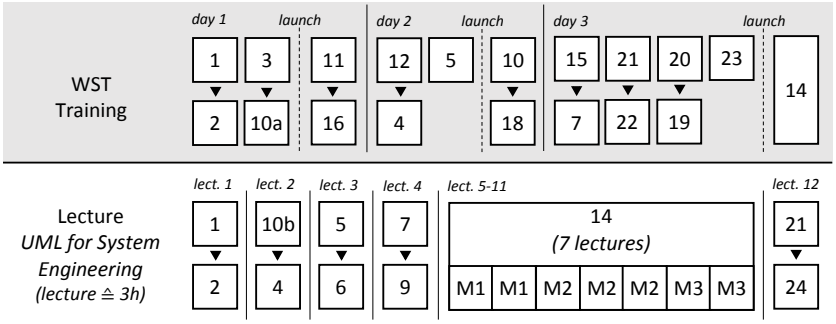

Fig. 5. Concrete module compositions and course schedules

As seen in Fig. 5, there is a module with a final exercise (module 14), which is used by both the lectures and trainings, but it is handled differently. This is mainly because of the different time schedules. In trainings, the whole exercise can be processed at one day. For lectures, the exercise is made during several weeks. Furthermore, the exercise is divided into two to three milestones. Since the exercise is influenced by the needs of trainings, it is very practically oriented. This is also appreciated by the students.

The creation of course materials was initially mostly made by merging and re-working existing documents from both sides. This increased the effort

more than initially expected. Nevertheless, the overall needed effort was limited because of splitting it to the partners.

When creating slides for the modules, it has to be considered that they should not contain too detailed textual information and rely more on using illustrations. This allows the speaker to flexibly deepen certain topics according to the audience. All the information, which is presented, can be read in detail in the handouts.

Further experiences with the shared concept have shown a positive effect on updating modules. If a module is re-worked by one partner, the other partner can also benefit from the improvements. However, changes affecting the structure or thematic content should be coordinated with the partner. For synchronizing the documents, Dropbox [12] is used. In addition, the concrete course compilations are stored in the document management systems of each party.

From the authors' point of view, the overall quality of the course materials was improved. The strength of both the partners could be combined to create documents of high scientific standards and industrial relevance. The practical samples from the professionals' experiences are appreciated by the students to easily understand the practical relevance of the theory.

One of the intentions for teaching at universities by using the same materials as industry trainings is to prepare students for real life projects. Evaluation results have bolstered this opinion significantly.

The evaluation results of the courses are very promising. In an evaluation of the Embedded UML Startup Training at Willert Software Tools from December 2013, the following question "What did you like most in the training?" was asked. The responses obtained are listed below.

- Concrete examples, which are related to embedded software development
- When the modeled software was running on the target as expected
- Support from the trainer (during the exercises)
- The course material is very good
- Practical exercises, especially with state charts and interrupts (for embedded software)
- Practice: Concrete Examples, which could be very well followed based on the target hardware
  Theory: Demonstration of problems and "bad" examples

The answers indicate that, the concrete examples, which are related to the special domain of embedded software development, prepare the audience for using the learned topics in practice.

At the university, the lecture is evaluated by a standard evaluation sheet. The overall results from the evaluation performed in May 2014 vary from "good" to "very good". Significant good ratings, which are related to the topics, were given for *understanding the theory*, the *practical relevance*, *topicality* and *usage of tools*. The students also complimented the *good explanations* and the *final exercise* with its practical relevance. Furthermore, one of the responses from a student indicated that "the practical application of the UML ... during the laboratory exercises significantly enhanced the understanding process".

However, in an earlier evaluation, students criticized the lack of a connection between the modules. Therefore, it is not sufficient to just combine modules in a certain sequence to create a course. It is also necessary to present the audience an overview about the whole course and make suitable connections between the modules. This is important for a speaker to take into consideration.

Evaluation results are always analyzed by the speakers and the result is fed back to the course material. Thus, there is a systematic improvement process of the course and its documents. Since the introduction of the shared, module-based approach, there were only about 5 to 15 students as attendees in the lectures. At the industry trainings, the number of participants varies between 4 and 8. Therefore it is difficult to estimate the acceptance of the approach for courses with many attendees. Since the practical exercises are very important for industry trainings, the maximum number of participants is limited. Of course, if more than one trainer is performing a training and can assist during exercises, the maximum number of participants can be increased. This can also be applied to exercises in lectures.

6 Conclusion

This paper discusses a shared concept for collaboration among teaching in universities and trainings in the industry, for educating the audience on an identical topic. The module-based approach proposed in this paper, allows to flexibly combine modules to suit the needs of the individual courses (i.e., both at the university lectures and industrial trainings).

An evaluation of the proposed approach, based on a case study for teaching the UML, is discussed in this paper. Some of the major challenges that arise during the application of such an approach to teach an identical course at the university and industry are: different goals, varying expectations and experiences of the audience, usage of different tools and dissimilar time frame/schedule to be met. Evaluation results from the case study indicate that, despite using the same modules for the courses, the goals of both could be achieved. For instance, at the training the participants learned how to use the UML in real-life/practical scenarios for embedded software development. On the other hand, the students at the university were educated with a strong background on the theory behind the UML. Furthermore, practical examples presented at the university were improved based on the experiences from the industry; whereas, recent trends in research influenced the creation of course materials. Thus, it is intuitive to perceive that both the partners involved in this approach (i.e., university lecture and industrial training), significantly benefit from each other while adapting the shared concept presented in this paper.

An enduring challenge towards applying the proposed approach is to take into consideration, the aspirations of all the partners involved, while preparing the course material, educational and training resources. This is imperative, to sustain the spirit of all the partners involved; thereby persevere with the proposed shared, module-based approach.

Acknowledgments. This work is carried out in close cooperation with Willert Software Tools GmbH, Ostfalia University of Applied Sciences and University of Osnabrueck.

References

1. Grice, R.A.: University/Industry Cooperation in Teaching and Cooperative Education Assignments. Professional Communication Conference IPCC '91. The Engineered Communication, International, vol.1 & 2 (30 Oct-1 Nov 1991)
2. Zhao, W., Wang, A.: University-industry Collaboration for Software Engineering Teaching. The 9th International Conference for Young Computer Scientists (ICYCS) (Nov. 2008)
3. Collofello, J.S.: University/industry collaboration in developing a simulation based software project management training course. 13th Conference on Software Engineering Education & Training (6-8 March 2000)
4. Engels, G., Hendrik, J., Lohmann, M., Sauer, S.: Teaching UML Is Teaching Software Engineering Is Teaching Abstraction. Satellite Events at the MoDELS 2005 Conference. LNCS Volume 3844, pp 306-319 (2006)
5. Noyer, A., Iyenghar, P., Pulvermueller, E., Pramme, F., Engelhardt, J., Samson, B., Bikker, G.: Tool Independent Code Generation for the UML - Closing the Gap between Proprietary Models and the Standardized UML Model. 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE). Lisbon, Portugal (April 2014)
6. Spieker, M., Noyer, A., Iyenghar, P., Bikker, G., Wuebbelmann, J., Westerkamp, C.: Model Based Debugging and Testing of Embedded Systems Without Affecting the Runtime Behaviour. 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Krakow, Poland (September 2012)
7. Kuzniarz, L., Staron, M.: Best Practices for Teaching UML Based Software Development. Satellite Events at the MoDELS 2005 Conference. LNCS Volume 3844, pp 320-332 (2006)
8. Brandsteidl, M., Seidl, M. Wimmer, M., Huemer, C., Kappel, G.: Teaching Models @ BIG - How to Give 1000 Students an Understanding of the UML. Models 2008 (Int. Conference) - Educators Symposium, Toulouse, France (2008)
9. Moisan, S., Rigault, J.-P.: Teaching Object-Oriented Modeling and UML to Various Audiences. Models in Software Engineering, LNCS Volume 6002, pp 40-54 (2010)
10. Bligh, D.A.: What's The Use of Lectures? Publisher: Jossey-Bass, 1 edition. ISBN-10: 0787951625. ISBN-13: 978-0787951627. (February 2000)
11. Willert Software Tools GmbH: Company Website. http://www.willert.de (2014)
12. Dropbox: Website. https://www.dropbox.com (2014)
13. Willert Software Tools Embedded UML Startup Training. Training Website. http://www.willert.de/embedded-uml-start-up-training/ (2014)
14. Waileung Ha, Hongyi Sun, Min Xie: Reuse of embedded software in small and medium enterprises. IEEE International Conference on Management of Innovation and Technology (ICMIT) (June 2012)
15. Selby, R.W.: Enabling reuse-based software development of large-scale systems. IEEE Transactions on Software Engineering, vol.31, no.6, pp.495-510 (June 2005)
16. IBM Rational Rhapsody, http://www.ibm.com/software/awdtools/rhapsody/ (May 2014)
17. Weilkiens, T.: Systems Engineering with SysML/UML: Modeling, Analysis, Design. Elsevier Science. The MK/OMG Press (2011)