A Syllabus for Usability Engineering in Multi-Project Courses

Jan Ole Johanssen, Dominic Henze, and Bernd Bruegge
Technical University of Munich, Munich, Germany
jan.johanssen@tum.de, henzed@in.tum.de, bruegge@in.tum.de

Abstract

Usability engineering is an important activity in today's application development, which raises the need to focus on its teaching efforts. However, in contrast to theoretical concepts, learning usability engineering is most successful in a hands-on environment. Furthermore, a challenge exists in efficiently applying usability assessment techniques to carry out usability engineering over time. We developed the UE4MP syllabus, a four meeting-based teaching approach for agile multi-project courses using cross-functional teams. The syllabus enables students to gain usability engineering knowledge and experience through hands-on learning and to make use of a platform for usability engineering. The overall goal is to relate usability concepts closely to applications that are developed by the students themselves. We applied the UE4MP syllabus over the course of one semester and report on our observations and challenges.

1 Introduction

With the availability of software for a great range of platforms, reaching from desktop computers to mobile devices, usability engineering gained high importance.

Typically, software engineering classes focus only on teaching development practices and engineering theory and miss out on the usability of an application, which results in a situation in which developers design user interfaces [26]. Nevertheless, students usually have a general understanding of *usability engineering* (UE) that raises from different sources: (a) common knowledge, intention, or personal interest in the topic, or (b) lectures, that either are focused on teaching usability concepts to the full extend or discuss usability engineering as a subtopic [17]. In any case, the students usually lack a real-world scenario in which they can apply new knowledge.

Problem 1: Students are barely able to apply usability engineering in real-world applications.

In addition, many concepts require a well-defined environment to be applied, which is not easy to provide. For example, the *Thinking Aloud* protocol by Nielsen [18] might require a time-consuming setup; this does not scale and is not easy to apply in the given time frame of a semester, even when the students know how to do it. Furthermore, it requires more steps beforehand, such as creating an understanding of the feature under test, how it can be measured, and delivering the feature to the end user.

Problem 2: Usability engineering concepts cannot easily be applied by students due to high setup efforts.

To approach both problems, our hypothesis is that students require a well-aligned teaching concept and tool support to enhance the effectiveness of teaching usability engineering. Furthermore, we argue that such a teaching concept needs to be set within a project course that provides a hands-on environment to apply usability engineering concepts over a considerably longer timeframe in a practical manner as already suggested by Wohlin and Regnell [29].

We describe a teaching syllabus and present an experience report, in which we show how the teaching concept is interweaved with the usage of a usability engineering platform. We build upon the assumption that the combination of theoretical concepts and practical elements helps to make the usability engineering more tangible for students [18].

This paper makes the following contributions:

- Description of a cross-functional team for usability engineering that allows to transfer knowledge into project teams, which reduces the need do this with every individual team member.
- A teaching syllabus that can be applied in a multi-project course to promote the hands-on application of theoretical usability engineering concepts. The syllabus incorporates the use of a usability engineering platform.
- Results from the application of the syllabus over the course of a semester, which allows to derive useful insights for further teaching efforts.

The presented work and its contributions are motivated by the idea of encouraging interaction and collaboration when teaching usability engineering as suggested by [6]. Furthermore, we foster transparency by using collaborative tools and interactive teaching units that have a high dependency on applications that are developed by the students themselves.

This paper is structured as follows. Section 2 presents related work in the context of teaching usability engineering. Section 3 provides the description of a multi-project capstone course. This course forms the environment for a syllabus consisting of four meetings that are described in Section 4. Section 5 reports on an experience report which we discuss in Section 6 through highlighting observations and challenges. Section 7 concludes the paper.

2 Related Work

We follow Nielsen's advice on teaching usability engineering by "bas[ing] the course firmly in the laboratory" [18] and thus teach in applied, project-based courses to prepare students for their later careers [9, 23, 29]. Nielsen stresses that—besides learning a required set of skills through theoretical lessons—the hands-on approach is indispensable for students to not only learn and understand the concepts of usability engineering, but also to trigger a "revolutionary change in [their] attitudes" [18]. This hands-on approach is also supported by others, such as Basili *et al.* and Ghezzi *et al.*, who stress that students need to apply their theoretical knowledge in practical projects [2, 11] to benefit from them [3, 8, 10, 28].

Offering students the chance to acquire the basic skills motivated our theory sessions at the beginning of the syllabus to be followed by hands-on exercises. According to Nielsen, this experience is in particular valuable if the students investigate the usability of applications they developed on their own [18]. Chan *et al.* present research on how to integrate teaching human-computer interaction aspects into the curriculum of master classes and highlight the importance of real customers and ongoing discussions [7]. As a result, we describe a syllabus that integrates usability engineering in a project course in which students develop applications for real customers from industry.

The need for teaching usability engineering was highlighted by Perlman in 1988 [25]. The fact that he continued to work on teaching usability engineering in 1995 [26]—almost 10 years after the initial work—shows that teaching usability engineering needs both continuous refinement and adjustments over time. We achieve this by adding agile concepts and incorporating state-of-the-art technology into our syllabus.

Newer research of teaching usability engineering is presented by Ovad *et al.*, who try to get developers in an agile industry setting to perform basic usability tasks, such as A/B tests, on their own [24]. Bruun *et al.* use a similar approach in an industrial setting to teach developers usability evaluations. With only three participants, the level of generalizability is limited, but nevertheless the results of developers quickly learning the core concepts of usability engineering look promising [5]. As shown by these researches and supported by others [20, 21], experienced developers are often the audience to teach usability engineering.

3 Course Environment

The syllabus targets an agile, multi-project course environment, the iPraktikum, which we describe in this section. We further outline the role of cross-functional teams and the CUU platform.

3.1 The iPraktikum

We describe the capstone course *iPraktikum*, which provides the environment for teaching usability engineering in agile project courses. The iPraktikum is a multi-project course which takes place every semester with 70-90 student developers who work in up to 12 project teams to create an application with a mobile context for real customers from industry solving their real problems [4]. While the mobile context is realized using Apple's iOS platform, resulting in iPhone and iPad applications, most projects are not standalone solutions, but include application servers, sensors, actuators, or wearable devices.

As shown in Figure 1, each *Project Team* consists of a *Customer* from industry, a *Project Management Team* consisting of a *Project Lead* and a *Coach*, as well as the *Development Team*. Both the project lead and the coach fulfill a role similar to a scrum master. While the project lead is a teaching assistant who is experienced in leading projects, the coach is a student who already participated as a developer. Thus, they are familiar with the infrastructure and teaching methodology. The *Developers* are students from second Bachelor semester up to their last Master semester.

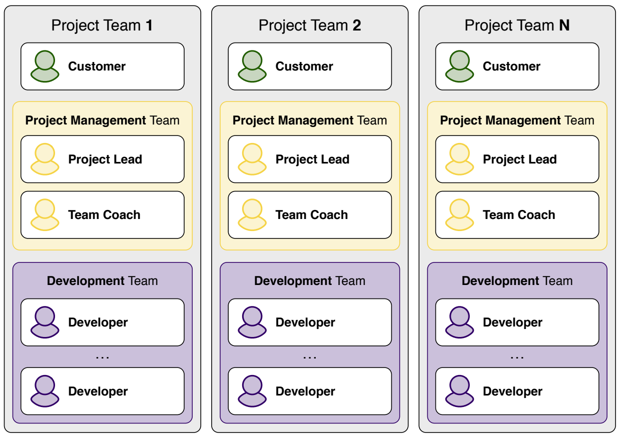


Figure 1: Overview of the iPraktikum's team structure, which consists of multiple project teams that are composed of different roles and sub-teams.

The iPraktikum fosters an event-based methodology, which enables the students to continuously interact with the customer [15]. Furthermore, the iPraktikum puts a special emphasize on the development of early prototypes and their vivid presentation to customers [31]. Over the course of multiple semesters, it has been shown that the iPraktikum's format and processes contribute to increased skills of students regarding both technical and non-technical aspects [4].

3.2 The Cross-Functional Teams

The iPraktikum uses cross-functional teams [4, 15, 31] to focus on special topics. Each of these teams is run by multiple cross-functional coaches and led by a cross-functional instructor (Figure 2). Cross-functional coaches are students experienced in the respective field. A cross-functional instructor is a teaching assistant who ensures the teaching methodology.

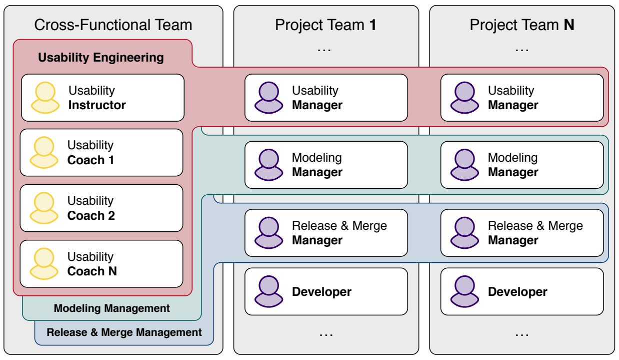


Figure 2: The iPraktikum cross-functional team structure focusing on the usability engineering team.

The cross-functional coaches work almost independently with the cross-functional teams: They elaborate on relevant topics and spread that knowledge to the cross-functional managers. These managers are developers from each team that have—in addition to their development tasks—the responsibility to share the knowledge they acquired in the cross-functional teams with their fellow team members.

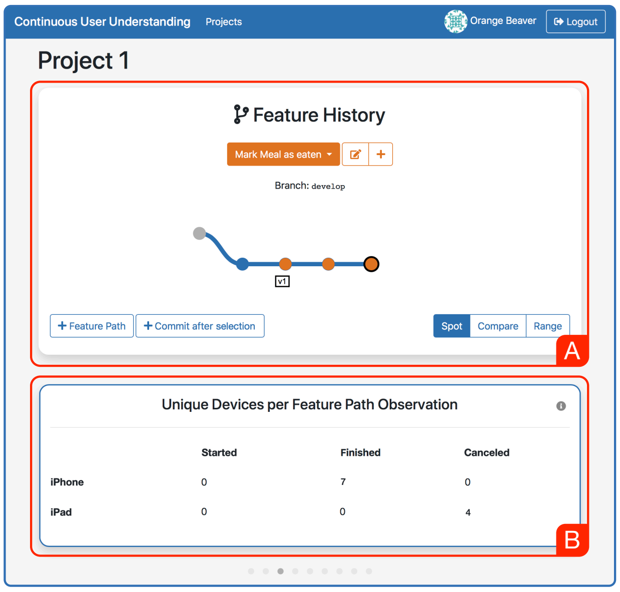
While most of the work is concentrated at the beginning of the course, support and review sessions are conducted until the end of the projects to ensure an adequate quality. This approach allows the course organizers to spread knowledge from instructors to the cross-functional coaches, followed by an information sharing via the cross-functional managers to all other students of the course. Major challenges are the timing and coordination between project and cross-functional teams, as well as with course-wide events, such as course-wide lectures, or intermediate and final presentations, in which the project teams present their current status to the whole course and all customers.

So far, the iPraktikum addressed a variety of cross-functional teams. For instance, a modeling management team was established to support the project teams with creating, reviewing, and refactoring software models created during the iPraktikum [1]. Likewise, a release & merge management team helps development teams to setup the basic infrastructure for their team, including continuous integration and delivery as well as ensuring the branching model and the code review process introduced in [16]. With this work, we introduce a syllabus that extends the iPraktikum by adding a dedicated usability engineering cross-functional team.

3.3 The CUU Platform

To enable *Continuous Software Engineering* (CSE), the iPraktikum makes use of tool support—one of which is the *CUU* platform: It aims for supporting developers in understanding user behavior [12]. The overall vision of the platform is to enable usability engineering in a rapid development environment such as CSE. It is available as an open source project.[1]

CUU does not impose a particular feature definition to usability engineers: everything that is developed on a feature branch can be understood as a *feature*. We further address the definition of a feature and its meaning in the syllabus in Section 4.2. Generally, CUU analyzes whether users started, finished, or canceled the usage of a feature and visualizes this information in *widgets* [13], as shown in Figure 3.



Unique Devices per Feature Path Observation

	Started	Finished	Canceled
iPhone	0	7	0
iPad	0	0	4

Figure 3: A screenshots of the CUU dashboard. Users select one or more commits for inspection in a graph-like representation (**A**). Related usage information is then presented in one or more widgets below (**B**).

This is enabled by the *Feature Crumbs* concept [14], a lightweight approach to mark feature specifics for detection during run-time. Feature crumbs are a single line of code, similar to breakpoints during debugging of code. Feature crumbs form the basis for referencing a variety of usability testing techniques, such as an automated implementation of the *Thinking Aloud* protocol [18]. In particular, we plan to add a widget that displays a summary of processed user feedback in relation to a feature crumb.

Besides the extract shown Figure 3, CUU hosts a *Project Overview* screen; a CUU project acts as the counterpart of a code repository. Furthermore, a *Services* screen allows UE manager to connect external services; we elaborate on this aspect in Section 6.

[1] `https://github.com/cures-hub`

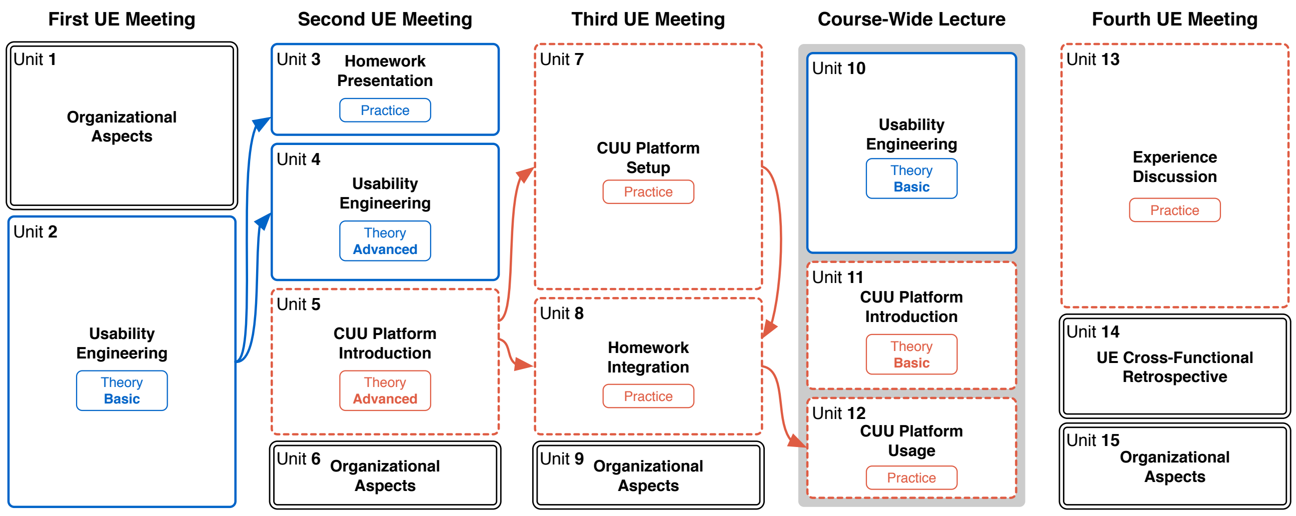


Figure 4: Visual overview of the UE4MP syllabus, a schedule for teaching usability engineering in cross-functional teams. A column represents a full meeting, which lasts for approximately 90 minutes; the UE content in the course-wide lecture might be shorter. Every box represents a Unit **U**, which encapsulates related content and is presented as one block within a meeting. The order of units suggests their actual position within the meeting and the size correlates with the time required for the unit. Arrows indicate constrains: for instance, Unit **2** must precede Unit **3** and **4**, while Unit **10** can be hold at any time. Units with blue, **single-stroked lines** address usability engineering content (4 units), while units with orange, **dashed lines** deal with the CUU platform (6 units). Units with **double-stroked lines** related to organizational aspects of the course. The **Theory** and **Practice** tags distinguish content presentation from hands-on units; theory units vary in their difficulty level: **basic** content includes overviews and topic introductions, while **advanced** content extends basic unit knowledge.

4 Teaching Usability Engineering in Multi-Project Courses

In this section, we introduce the **UE4MP** syllabus, which provides a guideline for teaching usability engineering in cross-functional teams within multi-project courses. The UE4MP reads *Usability Engineering (UE) for multi-project (MP) teams*, while the number 4 also represents the number of meetings during the semester. A broad overview is provided in Figure 4.

The goal of the syllabus is to step-wise introduce important concepts and to reduce the theory parts in favor of practical elements over time. A course-wide lecture, which all students of the course attend, serves as a general point of reference and milestone for the usability managers. More details about this lecture is provided in Section 5.1, while in the following, we detail every meeting with its content and goals.

4.1 First UE Meeting

The focus of the first UE meeting is the theory of usability engineering, which we base on fundamental work [18, 22]. The meeting should further clear up any organizational questions the UE managers might have at the beginning of the course.

The first UE meeting consists of many organizational aspects (Unit **1**) and starts with an introduction round of all usability managers. Furthermore, they are asked to include a brief overview of the projects they are working on. This should prepare them to provide feedback to any of the developed applications during a later point in time of the project. Hereafter, we introduce them to the role of the usability engineering manager. We summarize them as follows:

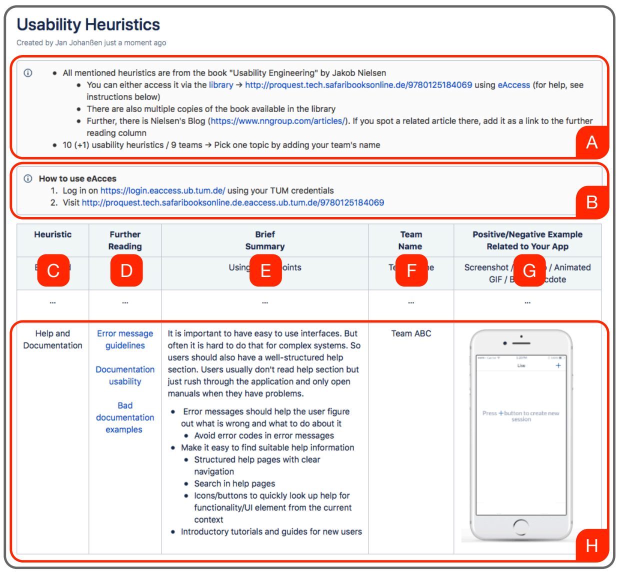
- Learn, repeat, and consolidate the general concepts of usability engineering.
- Integrate a framework for user understanding in teams' project and promote its application.
- Drive the realization of new insights that they gained from usage data and usability testing.

As the major element of the first UE meeting, the UE coaches prepare and hold a presentation of the usability engineering basics. This is roughly based on the book *Usability Engineering* by Jakob Nielsen [18]. The coaches are asked to focus the presentation on the following key aspects:

- overview of usability slogans;
- introduction to the usability engineering lifecycle;
- different forms of prototyping and their benefits and challenges, including tools and best practices;
- overview of usability heuristics to prepare the homework for the next meeting;
- how to use the Thinking Aloud protocol as one example of usability testing;
- idea of discount usability engineering.

The presentation should be interactive, and the coaches are encouraged to invite the managers to contribute ideas and descriptions while presenting the slides; usually, the managers already have a common sense of the concepts, however, they cannot formally refer to it (e.g., the Thinking Aloud Protocol.)

The homework is clarified at the end of the meeting; it reflects a first step of applying theoretical concepts to the team's applications. We ask the managers to pick and research on one usability heuristic and share their results within the second UE meeting as part of a two-minute talk. In Figure 5, we outlined a wiki page that we prepared for that purpose, in which each manager has to fill in one row.



Usability Heuristics

Created by Jan Johanßen just a moment ago

- All mentioned heuristics are from the book "Usability Engineering" by Jakob Nielsen
    - You can either access it via the library → http://proquest.tech.safaribooksonline.de/9780125184069 using eAccess (for help, see instructions below)
    - There are also multiple copies of the book available in the library
    - Further, there is Nielsen's Blog (https://www.nngroup.com/articles/). If you spot a related article there, add it as a link to the further reading column
- 10 (+1) usability heuristics / 9 teams → Pick one topic by adding your team's name

How to use eAcces
1. Log in on https://login.eaccess.ub.tum.de/ using your TUM credentials
2. Visit http://proquest.tech.safaribooksonline.de.eaccess.ub.tum.de/9780125184069

Heuristic	Further Reading	Brief Summary	Team Name	Positive/Negative Example Related to Your App
E d		Using oints	Te e	Screenshot / / Animated GIF / B cdote
...	...	...	...	...
Help and Documentation	Error message guidelines Documentation usability Bad documentation examples	It is important to have easy to use interfaces. But often it is hard to do that for complex systems. So users should also have a well-structured help section. Users usually don't read help section but just rush through the application and only open manuals when they have problems. • Error messages should help the user figure out what is wrong and what to do about it • Avoid error codes in error messages • Make it easy to find suitable help information • Structured help pages with clear navigation • Search in help pages • Icons/buttons to quickly look up help for functionality/UI element from the current context • Introductory tutorials and guides for new users	Team ABC	Live Press + button to create new session

Figure 5: Wiki page created for the Usability Heuristics homework; figure modified for visualization. (**A**) Description of homework. (**B**) Help on how to access supportive material. (**C**) Heuristic name. (**D**) Link to further reading. (**E**) Brief summary of the heuristic. (**F**) Name of the team that worked on the heuristic. (**G**) Concrete example of the heuristic within the team's application. (**H**) An instance for an entry of a usability heuristic within the table.

We are particularly interested in the UE managers' results on (**G**), the application of the usability heuristic to their application. We ask them to create mockups of good and bad examples of the heuristics. Therefore, it is important to ensure that the UE managers have access to the book, either by lending a copy from the university library or having access to an online version of the book. We enriched the wiki page, on which UE managers were asked to add their homework, with a brief introduction on how to access the online library. The homework fits the setup of this course, since there are usually 10 teams and 10 usability heuristics; however, other allocations will also work.

The meeting closes with more organizational aspects, i.e., discussion on the meeting time of the next meeting and an overview of the upcoming meetings, as well as clarification of questions.

Before and after the first UE meeting, the coaches are asked to publish additional content that is relevant for the managers, such as information pages about prototyping and links to popular tools. Not only do they create and maintain these pages, they actively inform the usability managers about the availability through the chat messaging platform. The coaches further ensure that the usability managers work on their homework; this is achieved by issues that are part of the issue tracking system. This should prevent UE managers from forgetting the homework, which would reduce the learning effect for all of the others during the second UE meeting.

4.2 Second UE Meeting

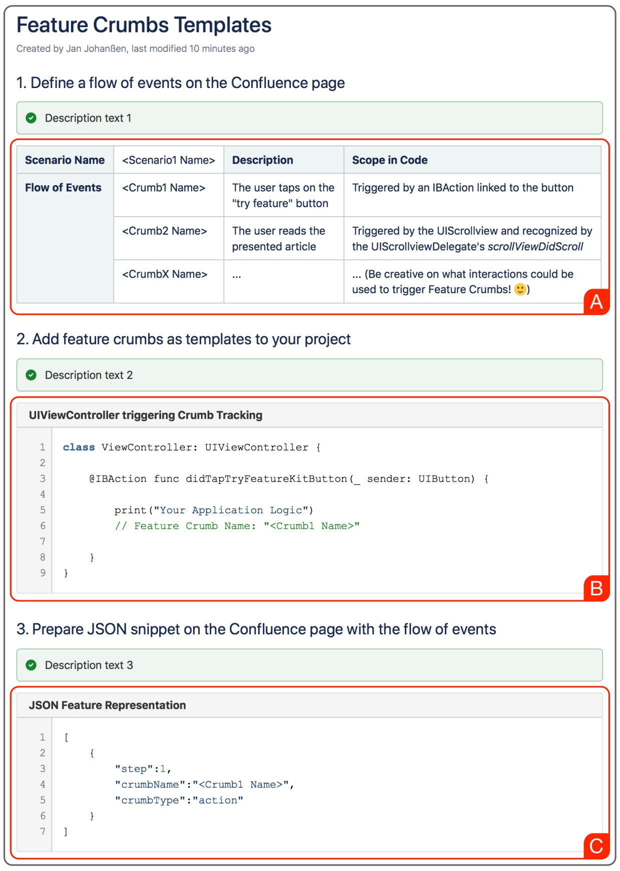
The second UE meeting starts with a short presentation of the usability heuristics homework by each UE manager (Unit **3**). During the presentation of this first homework, the UE instructor facilitates the presentations of the usability managers. This idea is based on the assumption that the usability instructors have a *broader perspective* on the topic; they can assess the heuristic and know how to apply it and therefore can ask for typical problems and situations in which they occur, in case the UE manager did not mention that aspect as part of their summary. Furthermore, the UE instructors help to put the heuristic into context, e.g., by noticing relationships between them. Overall, the goal is to encourage discussions about the topics and thereby sharpen the usability managers' senses and understanding for the topic.

As part of Unit **4**, the UE coaches again provide a theory session on other usability engineering topics. This time, however, the information is notably shorter (approximately half of the time invested as in Unit **2**), focusing on the following topics:

- small repetition of the topics of Unit **2**;
- an introduction to scenarios as an instrument;
- an open discussion on the definition of a feature;
- common practices, with minor focus on UE testing approaches different from Thinking Aloud and usability evaluation through heuristics.

The second and third aspects of these topics are the most important ones. Typically, the UE managers are familiar with scenarios as a way to capture and describe requirements of a system. They also receive a repetition of this topic in a proceeding course-wide lecture, which is not solely focused on usability engineering. However, in this unit, we focus on the usage of scenarios for usability engineering [19]. This prepares their understanding of feature crumbs as described in Section 3.3. The open discussion intends to strengthen their understanding of features, how they might be represented, and to prove the point that the feature understanding is often a question of definition. They are defined by the way they are managed, e.g., user stories, scenarios, epics, and that they vary in size as well as other characteristics, such as the involved systems, stakeholders, criticality, and more.

The last major unit of the second UE cross-functional meeting is an introduction to the CUU platform (Section 3.3) following these three aspects:


Feature Crumbs Templates

Created by Jan Johanßen, last modified 10 minutes ago

1. Define a flow of events on the Confluence page

Description text 1

Scenario Name	<Scenario1 Name>	Description	Scope in Code
Flow of Events	<Crumb1 Name>	The user taps on the "try feature" button	Triggered by an IBAction linked to the button
	<Crumb2 Name>	The user reads the presented article	Triggered by the UIScrollview and recognized by the UIScrollviewDelegate's *scrollViewDidScroll*
	<CrumbX Name>	...	... (Be creative on what interactions could be used to trigger Feature Crumbs! 🙂)

2. Add feature crumbs as templates to your project

Description text 2

UIViewController triggering Crumb Tracking

```
class ViewController: UIViewController {

    @IBAction func didTapTryFeatureKitButton(_ sender: UIButton) {

        print("Your Application Logic")
        // Feature Crumb Name: "<Crumb1 Name>"

    }
}
```

3. Prepare JSON snippet on the Confluence page with the flow of events

Description text 3

JSON Feature Representation

```
[
    {
        "step":1,
        "crumbName":"<Crumb1 Name>",
        "crumbType":"action"
    }
]
```

Figure 6: Wiki page created for the feature crumbs homework; figure modified for visualization, i.e., the description text has been removed. (**A**) List of crumbs to describe the UE managers' rationale when designing the crumbs. (**B**) Example of how to add the crumbs on a code level. (**C**) A JSON file that allows them to formally specify the feature path.

- Present the platform's functionality theoretically.
- Introduce the feature crumbs concept, in particular walk through the feature path and status. Feature paths represent a concatenation of feature crumbs [14], which allows statements about the feature state using the *Unique Devices per Feature Path Observation* widget as shown in Figure 3.
- Explain the platform's goals and align them with the previously presented approaches for usability testing and evaluation. In particular, this includes an introduction of how feature crumbs can be used to perform *Heuristics Evaluation* [18], based on the knowledge that the usability managers gained through the homework.

To be able to respond to the managers' questions about the platform and the crumbs, we ask the coaches to carefully inform themselves using relevant literature [13, 14]. We also provide a short, informal summary of the two papers in form of a wiki page.

We close the meeting with Unit **6**, organizational aspects, which, for the most part, introduces the homework: For the next meeting, the UE managers are asked to prepare a feature path of a specific scenario in their application as a preparation for the third UE meeting. Therefore, we provide a wiki page, as depicted in Figure 6, which we ask every UE manager to copy into their team's space and provide a URL for reference in a shared table.

The managers start by creating a table that is similar to a formal representation of a scenario as shown in Figure 6 (**A**). Each row describes a feature crumb, split by its name, a short description and the rationale of the crumb, and under which circumstances the crumb is triggered. Then, we ask them to add comments to their source code where they think the individual crumbs from the table should be triggered. We use the comments as a first step toward the actual tracking of features. This *dry run* requires to put a focus on designing the feature, rather than starting with the tool that is able to track the execution of the feature. We provide an example of how we expect this comment to look like in Figure 6 (**B**); we put an emphasize on the naming, since it should follow the exact same spelling as in Figure 6 (**A**). This should highlight the requirement that the same name of one row needs to be exactly spelled as in the code. The same holds true for the JSON file that needs to be provided in Figure 6 (**C**). Here, the UE managers design a machine-readable version of the feature representation that they previously described as a wiki page. As with the comment-styled crumbs, this prepares an easy transition into using the tool in the next meeting.

4.3 Third UE Meeting

The third UE meeting is mainly hands-on. It enables each team to use the platform independently. We provide step-wise guidance on how to setup the platform and integrate the needed framework for an example feature (Unit **7**), followed by the integration of their first own application-related feature (Unit **8**) that they prepared in their homework from the previous meeting. As a result, Unit **7** forms the basis for Unit **8**.

As this meeting brings results from multiple units, synchronization becomes a major challenge when preparing this meeting. The teams require a code basis which can be released and run on a device; this puts out requirements for both, the code and the workflow for its processing. Furthermore, in practical terms, the UE managers require the hardware (computer and mobile device) to run the tasks described in the following to setup and use the platform.

For Unit **7**, we prepare seven wiki pages that describe everything required to set up the platform and make use of it. Since they contain potential pitfalls, we walk the UE managers through them step by step:

Step 1 *Login and Navigate to Project.* Since the managers have never used the platform before, it needs to be ensured that they know where to find the platform and how to access it.

Step 2 *Define new Feature and link Feature Branch.* This step ensures that the UE managers adhere to the iPraktikum's development process: Every feature and its related branch is based on an issue. Therefore, we repeat the steps to formally create a branch. Hereafter, the managers need to switch back to their CUU project and create a feature representation in the platform. This can be done by providing a feature name and the initial commit on which the branch is based on.

Step 3 *Initialize Client in Code Project.* The CUU framework requires a client-side software development kit (SDK) that observes the execution of the application. The managers integrate this SDK into their projects. They need to register the SDK with a *Tracking Token* and *Project ID*; this information is extracted from the services screen (Figure 8) and ensures that the SDK can push information to the platform.

Step 4 *Add Feature Crumbs to Code Project.* In this step, we introduce the managers to the notation of triggering a feature crumb. At the bottom line, this is a method call with the feature crumb name as a string parameter. We ask them to seed a feature crumb called *My First Crumb* in the startup sequence of the application. Hereafter, the managers need to push all the latest changes to the remote server.

Step 5 *Add Commits to Branch.* The latest changes lead to a new feature increment, which needs to be registered with the platform. Therefore, they need to copy and paste the latest commit hash to their feature in the CUU platform.

Step 6 *Add new Feature Path.* Finally, to associate a commit with a feature path, they select the commit in the CUU platform and use a popup window to add a feature path in the JSON format. In this case, we ask them to only use a single-step path, with step information *1* and feature crumb name *My First Crumb*.

After they performed these six steps, the feature is ready for usability assessment. To test whether the setup was correct, they need to release the latest commit. We introduce them to the different widgets and how they can be harnessed to derive information about the feature's performance. By asking the managers to add only one crumb with the same name and at the same position, we eliminate any possible confounding factors that might distract the UE managers from their actual goal: setting up the feature for tracking. Only after they were successfully able to follow steps 1 to 6 and see a change in feature usage, we can be sure that everything works as expected.

To bring the focus back to their projects, we return to their prepared tasks from the homework in Unit **8**. This is usually much more complex, but of actual relevance for the UE managers. As a side effect, they get to know the inner workings of the platform even better. They learn how to add new crumbs to a feature path and what this means for the analysis, i.e., a new feature version, by replacing the prepared crumb comments with actual method calls.

At the end of this meeting, every team has created at least one feature that can be fully tracked using the CUU platform. This serves as the input for a hands-on exercise (Unit **12**) in the course-wide lecture.

The meeting is closed by another round of organizational aspects (Unit **9**). In particular, we explain to the managers how they can invite their team members to the project using a user management screen that is part of the CUU platform. As this is an important requirement for the course-wide lecture, we track this progress in order to ensure the effectiveness of the lecture. Eventually, we encourage the managers to continue using feature crumbs from this meeting on, collect and note down any observations, feedback, and lessons-learned, as a preparation for the fourth UE meeting.

4.4 Fourth UE Meeting

The fourth UE meeting serves as an opportunity to stimulate the discussion between the UE managers. The goals of this meeting are manifold, however, the focus lies on Unit **13**, in which every UE manager presents a feature in more depth. This includes the presentation of individual feature paths. Since this UE meeting is set at a later point of the project, we expect that the managers not only present the feature composition, but also report on how the tracking helped them in terms of usability evaluation and assessment. Ideally, the managers elaborate on knowledge gained from the feature crumbs observations and how it supported them to solve a problem. These in-depth descriptions of actual insights in real applications can be beneficial for the UE managers of other teams.

Unit **14** is a retrospective in which the managers provide feedback on both the work with the platform and the overall composition of the UE cross-functional role and the composition of the syllabus.

Eventually, Unit **15** closes the meeting with a set of information regarding the remaining semester, such as their responsibilities for future deliverables or how they can approach us if questions remain.

5 Experience Report

We applied the UE4MP syllabus from Section 4 during the summer term of 2018 in an instance of the iPraktikum, as described in Section 3. We followed the intention of a case study as defined by Wohlin *et al.* by "provid[ing] a deeper understanding of the phenomena under study in its real context" [30].

Overall, we set out to ensure the applicability of the UE4MP syllabus and in particular the acceptance of tool usage and theory concepts by the usability managers. Therefore, this section reports on the temporal instantiation (Section 5.1), the results of the practical units, i.e., the homework, (Section 5.2), and the number of feature crumbs usage (Section 5.3).

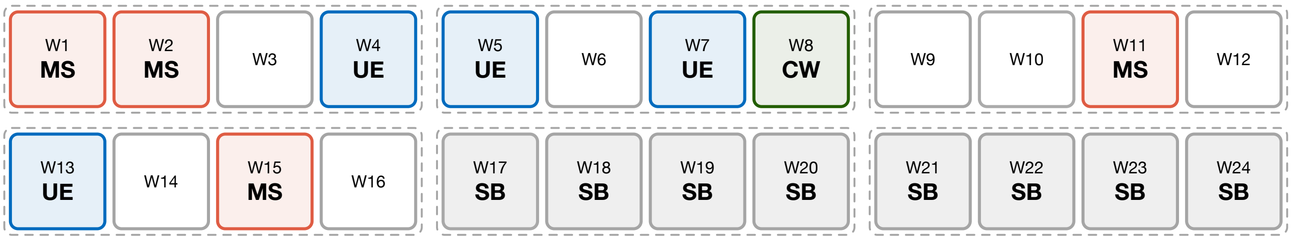


Figure 7: Chronological sequence of the iPraktikum in the summer term 2018. W1 is the first week of the semester, while W24 the last one. Supported by abbreviations, different box colors are used to map iPraktikum milestones, semester events, and UE meetings of the syllabus to semester weeks as follows. Blue boxes indicate usability engineering meetings (**UE**) and red boxes iPraktikum course milestones (**MS**). The green box highlights the usability engineering course-wide lecture (**CW**), while grey boxes relate to the semester break (**SB**).

5.1 Temporal Instantiation

To make the UE4MP syllabus transferrable to other courses, we mapped the meeting units to a typical semester of six month as shown in Figure 7.

As stated in Section 4, the syllabus builds up on cross-functional teams. These cross-functional roles, as described in Section 3, require project teams and therefore can only be assigned after the initial iPraktikum student assignment to project teams has finished. As a result, the first two weeks of the semester *W1, W2* were reserved for the iPraktikum setup, while *W3* got the projects started and the cross-functional managers of each project team assigned. In *W4*, we ran the first usability engineering meeting (Section 4.1), in which we focused on the organization, the usability engineering basics and a first homework assignment. In the following week *W5*, the second usability engineering meeting took place (Section 4.2), containing the first practical exercises and the introduction to the platform. In the third usability engineering meeting (Section 4.3) in *W7*, the tool support in form of the CUU platform was integrated in each team's project by the usability managers, followed by a first contact with the platform within each project.

After finishing the setup and explaining all concepts to the usability managers of each team, in *W8* all course participants were introduced to the concepts of usability engineering and the platform as part of a course-wide lecture. The course-wide lecture is an optional addition to the UE4MP syllabus; it intends to provide an overview of the first three usability engineering meetings. As shown in Figure 4, Unit **10** ensures a short theoretical introduction that sets the stage. In Unit **11**, we introduce the platform, how it can be used, and what it aims for. Finally, with Unit **12**, we make use of the prepared features from the third UE meeting, which allows all students to benefit from a pre-defined feature within their own application to experience the usability assessment and evaluation methods of the platform.

The timing of the course-wide lecture left the teams enough time to use the offered features until the second milestone of the iPraktikum in *W11*, which is an intermediate presentation of their work. In *W13*, two weeks after the intermediate presentation, but before the final presentation in *W15*, the fourth usability engineering meeting (Section 4.4) took place, gathering feedback for the UE cross-functional team and discussing feature paths and the corresponding feature crumbs.

5.2 Practical Unit Results

In this section, we report on the results from the homework of the usability engineering meetings, since they can serve as an indicator whether the UE managers understood the topic and were able to successfully apply the concepts.

The first homework regarding the usability heuristics (presentation in Unit **3**) was well perceived by the managers. Everyone was able to fill in the core components of the heuristic table. However, applying the heuristic to their actual application was only successfully performed by two teams; the other teams chose to use examples from well-known applications such as Microsoft Word.

In the second homework, we asked the managers to define a feature that could be tracked using feature crumbs and the platform (Section 4.2). We were able to derive more quantitative data as shown in Table 1, which allows to derive further assumptions in the discussion section (Section 6).

Table 1 shows that every team successfully defined crumbs for one of their application's features. At minimum, two feature crumbs were required, while 14 was the maximum amount of feature crumbs used to describe a feature. On average, a feature consisted of more than five crumbs. Except for one team, all teams successfully described the feature path within a JSON file. However, as it can be seen in the following section, team 4 was able to track their feature at a later point in time, which suggests that they forget to define the feature path, but did it within the platform. Out of the twelve features, ten can be considered as a proper, meaningful feature, which makes sense to be tracked and assessed for usability assessment. However, this was not the case for two features, highlighted with a red background in Table 1.

Table 1: Each team described at least one feature as part of their homework; every row depicts a feature.

Team	Number of Crumbs	Feature Path Correct?	Feature Meaningful?
1	3	Yes	Yes
2	12	Yes	Yes
3	3	Yes	No
4	14	No	No
5	3	Yes	Yes
6	3	Yes	Yes
	6	Yes	Yes
	6	Yes	Yes
7	6	Yes	Yes
8	3	Yes	Yes
	2	Yes	Yes
9	3	Yes	Yes

Team 3 implemented the feature path as three independent actions, that all started a new feature on their own. Team 4, on the other hand, described a massive feature execution, which depicted almost all of the application's features. This led to a situation in which multiple consecutive features were described as one feature, which hinders the individual assessment.

5.3 Platform Usage

To provide more quantitative data, we counted the number of features that were created by the teams throughout the semester. Table 2 shows the results.

Table 2: The number of features created in CUU separated by team; 20 features were created in total.

Team	1	2	3	4	5	6	7	8	9
Features	3	3	1	1	2	2	2	3	3

Comparing the numbers with the features that were created during the third UE meeting, which can be derived from Table 1, eight additional features were added over the course of the semester. Furthermore, we collected more data: Of the 20 features listed in Table 2, approximately 2200 feature executions were recorded successfully, more than 850 were stopped with the possibility to be successfully finished, and more than 1500 were actually canceled. In total, more than 6500 feature crumbs were triggered.

Overall, the number of features created within CUU fall short of our expectations. On average, every team created two features in CUU—one of which originated from the third UE meeting. We discuss this aspect in *The Effect of Tool Support* in Section 6.

5.4 Threats to Validity

This section briefly outlines the threats to the validity of the reported experiences centered around the four dimensions of validity [27], namely the construct, internal, external, and reliability validity.

Regarding the disparity between the intended and actual study observations, there might be the chance that the UE4MP syllabus and the introduction did not contribute to the gain of knowledge. Since there has not been a dedicated focus of usability engineering in the course before, we can assume that any information to that topic is beneficial to the students. In addition, to ensure that the concepts are straightforward, easy-to-understand, and consistent, we involved multiple instructors from other cross-functional teams to mediate the risk of ambiguous knowledge transfer.

The correlation between investigated and other factors might become visible in the way the managers made use of the taught concepts. In particular other duties, such as the actual coding or other courses, might have affected the extent to which they were working in the role of a usability manager. We tried to mediate this effect by reducing extra efforts, as well as providing help whenever needed.

The degree of generalizability of the study is low, since we can only report of one instantiation in the agile multi-project course iPraktikum. However, we argue that the presented syllabus is in general highly related to the structure of the iPraktikum, which makes generalization difficult. Still, we think that many observations and results can help other lectures to improve their teaching efforts regarding usability engineering. To be able to report more generalizable results, we plan to repeat the application of UE4MP to gain more reliable insights.

With respect to reliability, the fact that only one instructor supervised the cross-functional team might have biased the application of the UE4MP syllabus. However, since this cross-functional team was integrated in a broader course, we argue that the effect was rather low.

6 Discussion

This section discusses the results from the experience report, provides interpretations of the presented numbers, and points out challenges regarding various aspects of teaching usability in cross-functional teams.

The Effect of Tool Support. In general, we can summarize that the combination of the UE4MP syllabus and the tool support in form of the CUU platform enabled the students to perform actual usability engineering in their real-world development setting.

At the same time, the quantitative analysis of the platform usage indicates that—after an initial phase of using the platform—the teams reduced their efforts in designing feature paths and working with feature crumbs. This might have various reasons; for one, they have additional tasks which can hinder them from doing usability engineering. Other reasons might be found in the challenge of designing features, as described in the next paragraph.

The numbers suggest that we need to further support the teams in making use of the platform, pointing out the benefits which can encourage them to invest more time in usability engineering. We continue our efforts as part of our future work (see Section 7).

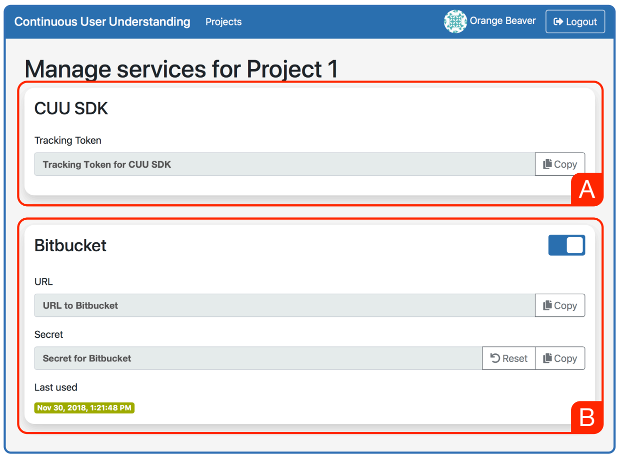
Usability Engineering is Difficult. A more detailed analysis of the homework illustrates that the students struggle in applying usability concepts to their own applications. Regarding the first homework, the fact that only two teams were able to create applied examples of good and bad usability heuristic cases within their own application might be an indicator that these heuristics are difficult to understand. The challenges in defining and understanding features become more obvious when analyzing the two cases highlighted as *"No"* in the column *"Feature Meaningful?"* of Table 1. The creation of a well-defined feature representation in form of feature crumbs requires effort.

Notably, in case a team provided a name for their feature, they always correctly defined the feature. This might explain the reason, why two teams created wrong feature path descriptions: They had a different mental model of a feature definition. This allows us to draw the conclusion that it is not enough to only provide a platform, which strengthens our goal of aligning the syllabus next to the platform usage.

Synchronizing Efforts. Synchronizing the UE4MP can become an issue regarding the following aspects:

- communicating the teaching materials and homework between the project leader, coaches, usability managers, and the individual team members;
- aligning the four meetings across the semester, as described in Section 5.1;
- ensuring that the teams' progress is mature enough that it fits to the UE4MP syllabus;
- connecting UE managers with other cross-functional managers, such as the release and merge managers, to ensure that they have access to the repository when setting up CUU, or are allowed to update third party repositories, which is a requirement to make the CUU platform's client-side SDK work; Figure 8 provides more description concerning this aspect.

Enable Collaboration. Besides setting up an instant messaging channel for every project team, we also created a channel for the usability engineering cross-functional team. It turns out that this is a very interesting place for discussions, which is important for usability engineering. Usability engineering is a collaborative process, which requires multiple participants, both experts and users. The students of other teams can act as proxy users; while they only have a rough idea of the topic of the other applications, they know just the right amount of details to assess the usability of the application from the perspective of a user.



Manage services for Project 1

CUU SDK

Tracking Token: Tracking Token for CUU SDK

Bitbucket

URL: URL to Bitbucket

Secret: Secret for Bitbucket

Last used: Nov 30, 2018, 1:21:48 PM

Figure 8: Screenshot of the *Services* screen of the CUU platform. Usability managers need to add the *tracking token* to their mobile application to enable the flow of usage data from the SDK's usage monitoring component to the CUU web platform (**A**). This requires a mature code basis, which might need several weeks until it is available. Furthermore, the credentials for connecting the code repository with the CUU project can be obtained from the services screen (**B**).

In general, we conclude that usability engineering can be taught better with hands-on examples, which are shared with others to collect different opinions. An instant communication channel promotes the collaboration, while wiki pages—that we used for the homework to collect usability heuristics as described in Figure 5—represent another point of interaction.

However, we noticed that many UE managers still contacted us directly in case of questions via a direct message and refrained from using the "public" channel to ask the questions to all the other students. We actively encouraged them to change this and continue to work on this in future semesters.

Acknowledging Privacy Aspects of Users. Usability engineering inherently relies on the interaction with users. With the CUU platform and SDK, the students obtain access to a tool that allows for learning more about the way a user interacts with an application. This includes, besides the ability to determine whether a feature has been started, completed, or canceled, additional knowledge sources that provide fine-grained information about the application usage. Generally, no personal data is collected in a way that would allow for conclusions toward an individual user. Within the units of UE4MP, we intend to sensitize the students for these aspects and the responsibility as a consequence thereof. In particular, we ask the students to let their end users, i.e., the customers, know about the addition of CUU before they start using it. Technical mechanisms inform the users about the data collection and provide the ability to opt out.

Providing the Environment. One challenge arises in providing the environment that is required to make use of the UE4MP syllabus. To take full advantage of the syllabus, an extensive set of tools in form of an infrastructure is required:

- an issue management system to enable tracking of tasks and development progress;
- a wiki system to share information, as well as enable a space to collaborate on the homework;
- a version control, continuous integration, and continuous deployment system that covers the full development process before the CUU platform can be utilized;
- an instant messaging service.

Providing and maintaining this environment represents a major challenge and is only feasible for large-scaled project courses.

7 Conclusion and Future Work

Usability engineering is an important activity during software engineering. However, usability engineering can only be taught successfully in a hands-on environment, using real-world projects in which students use their own applications for usability assessment. Therefore, additional, hands-on teaching approaches need to be developed to further support students besides their general software engineering curriculum.

In this paper, we introduced the UE4MP syllabus, a teaching concept based on four meetings that aims to integrate usability engineering into multi-project courses. This is enabled by establishing a cross-functional role, in which students take over the role of usability managers. Furthermore, it incorporates the CUU platform for usability engineering, which reduces the effort for setting up usability assessment activities.

We applied the UE4MP syllabus during one semester in the iPraktikum. Our observations suggest that the syllabus can be applied to teach usability engineering through cross-functional teams in a multi-project course. We were able to apply the syllabus as described and in particular the theory aspects were well-perceived by the students. Likewise, students successfully applied the CUU platform. However, it requires major efforts to teach the tool and point out benefits.

Therefore, as part of our future work, we plan to extend both the UE4MP syllabus as well as the CUU platform. Regarding the syllabus, we plan to add one-on-one meetings after or as a replacement of the fourth UE meeting, in which the usability coaches meet with the managers to discuss their work. Regarding the CUU platform, we intend to continue its development to offer more usability assessment functionalities that could be provided through additional widgets. This should increase the platform's overall usefulness and thereby increase its usage. Furthermore, we are working on simplifying the setup steps described in Section 4.3.

Acknowledgements

This work was supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution (CURES project). We thank the iPraktikum '18 students for their feedback and Simon Lang, Jan Philip Bernius, and Lara Marie Reimer for their support.

References

[1] Lukas Alperowitz, Jan Ole Johanssen, Dora Dzvonyar, and Bernd Bruegge. Modeling in agile project courses. In *MODELS (Satellite Events)*, pages 521–524, 2017.

[2] Victor R. Basili. The role of experimentation in software engineering: past, current, and future. In *International Conference on Software Engineering*, pages 442–449. IEEE, 1996.

[3] Barry Boehm, Alexander Egyed, Dan Port, Archita Shah, Julie Kwan, and Ray Madachy. A stakeholder win-win approach to software engineering education. *Annals of Software Engineering*, 6(1/4):295–321, 1998.

[4] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. Software engineering project courses with industrial clients. *ACM Transactions on Computing Education*, 15(4):17:1–17:31, 2015.

[5] Anders Bruun and Jan Stage. Barefoot usability evaluations. *Behaviour & Information Technology*, 33(11):1148–1167, 2014.

[6] John M. Carroll and Mary Beth Rosson. A case library for teaching usability engineering: Design rationale, development, and classroom experience. *Journal on Educational Resources in Computing (JERIC)*, 5(1):3, 2005.

[7] Susy S. Chan, Rosalee J. Wolfe, and Xiaowen Fang. Issues and strategies for integrating hci in masters level mis and e-commerce programs. *Int. Journal of Human-Computer Studies*, 59(4): 497 – 520, 2003. ISSN 1071-5819.

[8] David Coppit and Jennifer M. Haddox-Schatz. Large team projects in software engineering courses. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, pages 137–141, New York, USA, 2005. ACM.

[9] Norman Fenton, Shari L. Pfleeger, and Robert L. Glass. Science and substance: a challenge to software engineers. *IEEE Software*, 11(4):86–95, July 1994.

[10] Christopher K. Hobbs and Herbert H. Tsang. Industry in the Classroom. In *Proceedings of the Western Canadian Conference on Computing Education*, pages 1–5, New York, USA, 2014. ACM.

[11] Paola Inverardi and Mehdi Jazayeri. *Software Engineering Education in the Modern Age: Software Education and Training Sessions at the International Conference, on Software Engineering, ICSE 2005, St. Louis, MO, USA, May 15-21, 2005, Revised Lectures*, volume 4309. Springer, 2006.

[12] Jan Ole Johanssen. Continuous user understanding for the evolution of interactive systems. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '18, pages 15:1–15:6, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5897-2.

[13] Jan Ole Johanssen, Anja Kleebaum, Bernd Bruegge, and Barbara Paech. Towards the visualization of usage and decision knowledge in continuous software engineering. In *2017 IEEE Working Conference on Software Visualization (VISSOFT)*, pages 104–108, September 2017.

[14] Jan Ole Johanssen, Anja Kleebaum, Bernd Bruegge, and Barbara Paech. Feature crumbs: Adapting usage monitoring to continuous software engineering. In *Product-Focused Software Process Improvement*, pages 263–271, Cham, 2018. Springer International Publishing. ISBN 978-3-030-03673-7.

[15] Stephan Krusche, Lukas Alperowitz, Bernd Bruegge, and Martin O Wagner. Rugby: an agile process model based on continuous delivery. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, pages 42–50. ACM, 2014.

[16] Stephan Krusche, Mjellma Berisha, and Bernd Bruegge. Teaching code review management using branch based workflows. In *International Conference on Software Engineering - Companion Volume*, pages 384–393, 2016.

[17] Stephan Krusche, Nadine von Frankenberg, and Sami Afifi. Experiences of a software engineering course based on interactive learning. In *Software Engineering im Unterricht der Hochschulen*, SEUH '17, pages 32–40, 2017.

[18] Jakob Nielsen. *Usability Engineering*. Interactive Technologies. Elsevier Science, 1994. ISBN 9780080520292.

[19] Jakob Nielsen. Scenarios in discount usability engineering. In John M. Carroll, editor, *Scenario-based Design*, pages 59–83. John Wiley & Sons, Inc., 1995.

[20] Jakob Nielsen and Rolf Molich. Teaching user interface design based on usability engineering. *SIGCHI Bull.*, 21(1):45–48, August 1989. ISSN 0736-6906.

[21] Jakob Nielsen, Rita M. Bush, Tom Dayton, Nancy E. Mond, Michael J. Muller, and Robert W Root. Teaching experienced developers to design graphical user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 557–564, New York, NY, USA, 1992. ACM.

[22] Donald A. Norman and Stephen W. Draper. *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1986. ISBN 0898597811.

[23] Tom Nurkkala and Stefan Brandle. Software studio: Teaching professional software engineering. In *Technical Symposium on Computer Science Education*, pages 153–158. ACM, 2011.

[24] Tina Øvad, Nis Bornoe, Lars Bo Larsen, and Jan Stage. Teaching software developers to perform ux tasks. In *Proceedings of the Annual Meeting of the Australian Special Interest Group for Computer Human Interaction*, OzCHI '15, pages 397–406, New York, NY, USA, 2015. ACM.

[25] Gary Perlman. Teaching user interface development to software engineers. *Proceedings of the Human Factors Society Annual Meeting*, 32(5): 391–394, 1988.

[26] Gary Perlman. Teaching user interface development to software engineers. In *Conference Companion on Human Factors in Computing Systems*, CHI '95, pages 375–376. ACM, 1995. ISBN 0-89791-755-3.

[27] Per Runeson, Martin Host, Austen Rainer, and Björn Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.

[28] Mary Shaw, Jim Herbsleb, Ipek Ozkaya, and Dave Root. Deciding what to design: Closing a gap in software engineering education. In *International Conference on Software Engineering*, ICSE '05, pages 607–608, 2005.

[29] Claes Wohlin and Björn Regnell. Achieving industrial relevance in software engineering education. In *Conference on Software Engineering Education and Training*, pages 16–25. IEEE, 1999.

[30] Claes Wohlin, Per Runeson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wessln. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 2012. ISBN 3642290434, 9783642290435.

[31] Han Xu, Stephan Krusche, and Bernd Bruegge. Using software theater for the demonstration of innovative ubiquitous applications. In *Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, pages 894–897, 2015.