International Journal of Engineering Education Vol. 29, No. 5, pp. 1088–1099, 2013
Printed in Great Britain

0949-149X/91 $3.00+0.00

The Role of Collaborative Capstone Projects—Experiences from Education, Research and Industry*

ANNE HESS[1,2], DIETER ROMBACH[1,2], RALF CARBON[2], DANIEL F. MURPHY[3], MICHAEL HOEH[4] and CHRISTIAN BARTOLEIN[4]

[1] Software Engineering Chair, Computer Science Department, University of Kaiserslautern, Gottlieb-Daimler Straße 47, 67663 Kaiserslautern, Germany. E-mail: {anne.hess; dieter.rombach}@cs.uni-kl.de

[2] Fraunhofer Institute for Experimental Software Engineering, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany. E-mail: {anne.hess; dieter.rombach; ralf.carbon}@iese.fraunhofer.de

[3] John Deere, Moline Technology Innovation Center, One John Deere Place, Moline, IL 61265, USA. E-mail: MurphyDanielF@JohnDeere.com

[4] John Deere GmbH & Co. KG, Intelligent Solutions Group, Strassburger Allee 3, 67657 Kaiserslautern, Germany. E-mail: {HoehMichael; BartoleinChristian}@JohnDeere.com

An integral part of software engineering curricula at universities are practical classes or projects that enable students to apply theoretical knowledge gained in lectures on concrete practical examples. Practical projects, in particular, defined as university-industry collaborations provide the potential of being very beneficial especially in graduate education: in such realistic project settings, students can experience real-life software engineering challenges and achieve learning objectives that go beyond typical learning objectives of practical assignments during classes or even practical projects defined by faculty members. However, such collaborative projects have to be planned carefully and also come with various challenges. In this article, authors from academia and industry share their experiences gained during a history of successfully conducted collaborative projects. These experiences comprise objectives, benefits, challenges and lessons learned both from an educational viewpoint (i.e., students, supervisors), research viewpoint (supervisors), and industry viewpoint (customer). The experiences summarized in this article could serve as motivation and valuable information for other universities and industry companies intending to plan and organize collaborative projects of this kind.

Keywords: collaborative project; software engineering education; capstone project; industry-university collaboration; experiences; lessons learned; challenges

1. Introduction

Practical projects that offer the possibility for students to apply knowledge and skills gained in software engineering lectures in a realistic project setting are considered very effective in software engineering education [1]. Practical projects, in particular, which are defined as university-industry collaborations, provide the potential of being very interesting and beneficial compared to assignments provided during university classes [2]: In addition to working on a real problem provided by a real customer, students get the chance to experience how to interact with a real client, how to cope with unclear requirements or changing demands from the customer, how to deliver products of high quality within a typically short timeframe, etc.

But not only students can benefit from such projects. The industrial customers providing a specific problem to be addressed in such a project are also offered great opportunities: About 10 to 15 "engineers" work on a particular problem and develop meaningful solutions within a short timeframe (typically one semester). Such solutions might include prototypical implementations and evaluations of initial ideas, which could later on be refined and incorporated into the customer's product portfolio and be brought to market.

Finally, software engineering researchers who act as supervisors in such projects can also profit from such projects as they have the chance to use such project settings to apply and evaluate newly developed software engineering methods. Furthermore, empirical evidence collected during such controlled project settings (e.g., data about effort, defects detected during testing) and derived software development improvements can serve as a valuable feedback mechanism to the students [5].

However, in order to be beneficial, such collaborative projects have to be planned carefully. They also come with various challenges. For example, faculty members or researchers from universities and collaborating research institutes acting as supervisors in such project settings are often faced with several problems. These problems include how to support the students in transferring theoretical knowledge in such a project setting [3], or how to enable students to experience the benefits of discipline and to overcome the gap between real professional scenarios and scenarios used in software engineering university courses [4].

In this article, the authors share their experiences

* Accepted 11 March 2013.

gained during a history of successfully conducted projects over several years. These projects were offered as ''Team-based Software Development'' (in the following referred to as ''capstone projects'') in the form of collaborative projects between the Software Engineering Research Group ''Processes and Measurement'' headed by Prof. Rombach at the University of Kaiserslautern (in the following referred to ''AGSE''), the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE), as well as John Deere in the role of the customer. The experiences shared in this article are discussed from three different viewpoints, which play a crucial role in such projects: (1) students, who have the chance to apply theories learned during lectures in a realistic project setting; (2) supervisors and researchers, who support the students in applying methods during the project; (3) customers, who provide specific problems to be solved by the students.

The remainder of this article is structured as follows: In chapter 2, we introduce a typical project setting and discuss a history of collaborative projects aimed at briefly introducing particular customer problems and solutions developed within these projects. Chapter 3 is dedicated to reflecting on our project history by discussing objectives, challenges, and recommendations based on lessons learned gained during our history of collaborative projects from the three different viewpoints: the viewpoint of students (section 3.1), the viewpoint of supervisors (section 3.2), and the viewpoint of the customer (section 3.3). Finally, the article concludes in chapter 4 with a summary of our main findings.

2. History of collaborative projects

Since 2001, AGSE has conducted a capstone project once a year in cooperation with an industrial customer and Fraunhofer IESE [1]. During these projects, a team of students work together in a laboratory environment for the duration of approximately eleven weeks. Researchers from both AGSE and Fraunhofer IESE supervise the students during these projects. These projects are especially offered to students being in their master studies, i.e., the students are expected to have already a sound knowledge of software engineering processes and activities. In fact, according to the curriculum offered at the AGSE, this is the first practical project where students run through a complete software engineering process while being fully responsible for eliciting the requirements, designing the prototype, and implementing and delivering the system on time. Furthermore, each student is assigned to particular roles and assumes the corresponding responsibilities. These roles include management-oriented roles such as project manager and technical roles such as requirements engineer, architect, UI designer, developer or tester. After this capstone project the students are expected to [1]:

- know and understand the different roles and responsibilities in a software development project;
- execute a well-defined software development process;
- understand the importance of software and experience documentation for future projects (i.e., to document observations and experiences during the project);
- be aware of their own thinking and decision-making processes;
- reflect about events and changes of situations that originate from performed actions;
- communicate and interact with a real customer
- carry out project estimation (i.e., effort, time, quality).

On the customer's side, these capstone projects are typically set up within the organization as an applied research project aimed at exploring a technology or business model for future implementation or further evaluation.

In the following, three projects will be introduced including their particular goals, and the results achieved by the students.

2.1 The grower's notebook

One of the first capstone projects conducted with the industry partner John Deere was the so-called ''Grower's Notebook''. The project aimed at designing a demonstrator setup for agricultural task management. The realized demonstrator consisted of two parts: (1) a simple browser-based user interface allowing a contractor to negotiate field operation tasks with a grower and to assign these to a specific worker in the field (see Fig. 1), and (2) a mobile device application (iOS app) enabling the worker to look up assigned tasks and to report on the current task status. The reported task status could then be monitored by the contractor or grower via the web application. Synchronization of tasks between the workers' iPhones and the web application was realized by means of a cloud-based approach.

The typical development phases were supplemented by an evaluation of applicable technologies comparing suitable candidates in terms of (1) a cloud-based data storage back-end and (2) available mobile device platforms against the set of defined requirements.

Fig. 1. Data flow between grower, contractor, and field worker involving task assignment and reporting realized for the Grower's Notebook prototype.

2.2 The mobile configuration assistant

The Mobile Configuration Assistant targeted a customer challenge associated with the configuration of implements (e.g., hay, harvest, or seed equipment), via a display located in the cabin of a tractor. This is indeed a very complex task, especially for novice field workers who are often faced with complex configuration procedures and data entry tasks before any field work can be accomplished. The project demonstrated a configuration engine and a domain model editor. The domain model editor (located on the server side) was developed to support developers in creating and maintaining domain models that capture all relevant parameters and their relationships required for the configuration of implements. The models created with the domain model editor are then sent to an iPad where the configuration engine creates an easy-to-use user interface based on the information contained in the domain model. After the user has configured the system on the iPad, the configuration settings are sent to the vehicle (see Fig. 2 and Fig. 3). This project delivered an evaluation of multiple architecture concepts and a prototype relative to

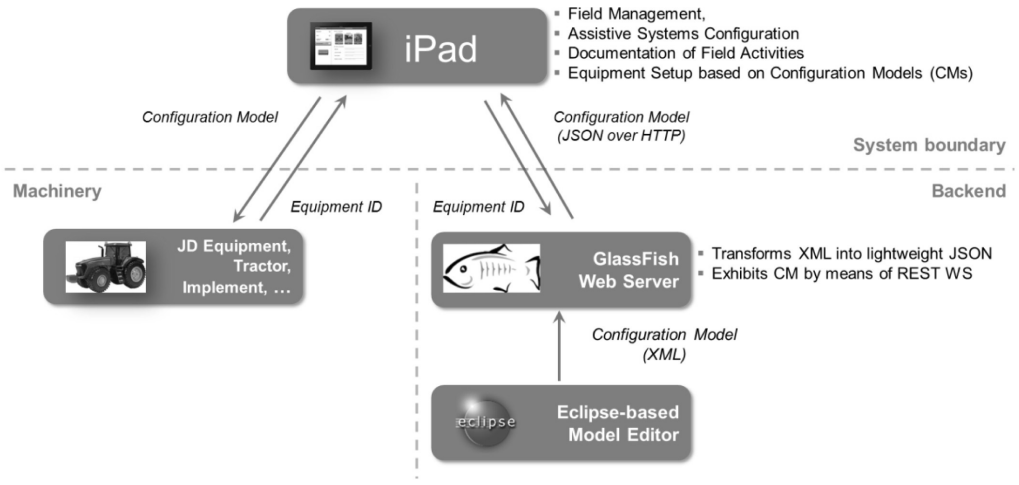

Fig. 2. High-level architecture concept of the mobile configuration assistant.

Fig. 3. User interface supporting implement configuration on an iPad and underlying domain model editor.

the business scenario, and demonstrated improvements to the industrial partner's display configuration process.

2.3 The agile dashboard

With the introduction of Agile Software Development at the Intelligent Solutions Group (ISG) of John Deere, a large number of Scrum Teams were created at several ISG locations worldwide. Agile Development and especially Scrum require certain planning activities that are typically done on whiteboards, which supported the teams in making the development status transparent (see Fig. 4).

This worked quite well for each location but the exchange of information with other locations was a challenge. Clearly, an electronic communication tool was needed and the software called "Rally" [6] was introduced at the ISG. However, simply accessing current status information stored in Rally required several interaction steps. This was quite time consuming for the users and resulted in a totally different user experience compared to a whiteboard, where the user just needs to walk by and look at the information. That is, the information on the whiteboards is absorbed at a glance, which is fast and easy. This situation was the motivation for the capstone project. So the idea was to find a solution that would combine the advantages of analog and electronic visualization: an electronic version of an Agile Dashboard.

Fig. 4. Sprint planning and status information on whiteboard.

The Agile Dashboard is a browser application developed in Ruby. A typical main screen of the application is shown below in Fig. 5. Within the application, up-to-date planning and status information is shown, comprising for instance sprint burn-down, critical defects, user stories, as well as defects and their respective status. With the exception of the build status information, which is provided from Jenkins (a tool used for continuous integration [7]), all information is retrieved from Rally. Besides displaying sprint status information, the Agile Dashboard also offers interaction possibilities to the user, such as selection of a particular section (e.g., user stories) for more details or the ability to switch views between different teams.

3. Discussion of objectives, challenges and lessons learned

In the following, experiences are discussed from the viewpoint of students (section 3.1), supervisors (sections 3.2), and customers (section 3.3), with the capstone projects introduced in the project history used as examples.

3.1 Students' viewpoint

3.1.1 Educational background

Students enrolled in computer science are offered three different practical software engineering courses and projects at the AGSE where they can apply theoretical knowledge gained from lectures in

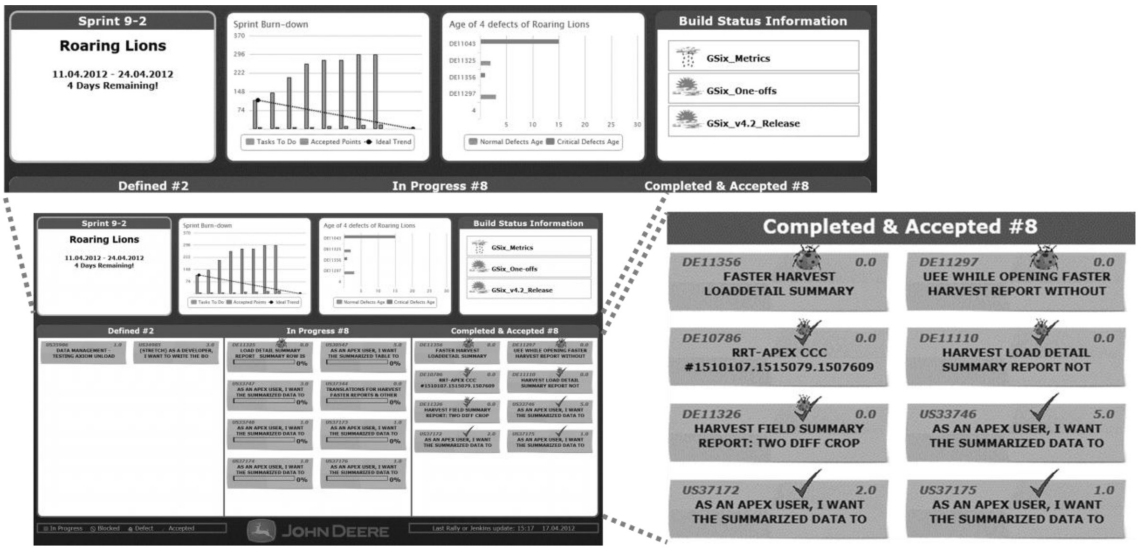

Fig. 5. Sprint planning and status information on Agile Dashboard.

a practical context. These courses differ in the complexity of the development tasks and the particular learning objectives.

Already in one of their early bachelor semesters, the students can sign up for a practical course, where they are asked to develop a component-size piece of software supplemented by some specification and verification activities. In their later bachelor studies, the students can register in a practical project. In this project, they basically have to run through all development phases, starting from requirements engineering, via architecture design, component engineering, and implementation to testing. During this course, the students (divided into teams of 5–6 students) typically follow a waterfall process model, i.e., all students are involved in all activities within these phases. The problem to be solved is typically provided by the university, often in collaboration with a third party delivering requirements, but not from an industry context. This project also has a strong focus on high-quality documentation, i.e., the students are asked to create and deliver detailed specification documents at the end of each phase.

Finally, within the scope of their master studies, the students can now sign up for the capstone project. At this stage, they already have a sound knowledge and understanding from their bachelor studies of the processes, methods, techniques, and tools that are used to develop large and complex software systems.

3.1.2 Learning objectives and expectations

As already mentioned in the introduction section, the capstone project is offered as ''Team-Based Software Development'' project, and this is, in fact, one of the main learning objectives: The students have to learn how to develop complex software in a team. To address this learning objective, the students are now assigned to particular software engineering roles and are in charge of the corresponding responsibilities. These responsibilities include the creation and delivery of high-quality specifications as well as consolidation of the achieved results and communication to the other team members. On top of that, the students experience software development in a real-life project with an industry customer providing a problem to be solved. That is, they are fully responsible for eliciting and negotiating the customer's requirements.

When talking to students that have participated in these projects within the last years they often say that ''this project was the best experience I had during my studies''. In order to get a better understanding of the particular expectations that students have when registering to these projects, we have recently conducted a survey within the scope of the current capstone project (which officially started 1 October 2012). The aim of this survey was to first collect the students' expectations at the beginning of the project and then evaluate the fulfillment of these expectations at a later stage in the project.

One week before the official start of the project, the students were asked to share their thoughts on the question ''What do you expect to learn/experience in the upcoming project?'' by filling out a sheet. These thoughts were consolidated and classified into expectations related to ''real-life project with customer from industry'', ''team work'' and ''specific software engineering disciplines'' (see Table 1).

After the second iteration of the project (i.e., 7

weeks after project kick-off), we evaluated the fulfillment of these expectations by using a questionnaire. In this questionnaire, the students were asked to rate a given statement related to the captured expectations on a scale from 1 to 5 (with 1= "I totally agree", 2 = "I agree", 3 = "neither/nor", 4 = "I disagree", 5 = "I totally disagree"). Each statement started with "The project enables me to. . ." and supplemented with the respective expectation. For example: "The project enables me to collaborate in an international team of students". The results of the captured and consolidated expectations as well as the assessment of their fulfillment in the current project are summarized in Table 1.

The collected expectations and results show, that such capstone projects offered as collaboration projects between industry and university provide great opportunities for students. Especially the interaction with the customer, both in requirements elicitation interviews/discussions and presentations of results, are really beneficial for the students. During these meetings, the students basically experience how to cope with possibly unclear requirements at the beginning of a project and learn the usefulness of certain techniques such as low-fidelity prototypes (e.g., paper prototypes) for getting a better and clearer understanding of the requirements. "Selling" their achievements to the customer in presentations, sharing their experiences, and possibly convincing the customer regarding their solution decisions is also a very positive and motivational experience for the students. This also includes the fact that the results produced for the customers are of real value to them and will be reused. This is also a very motivational aspect for the students to develop something "useful".

Depending on the scope of the project, the students also get very good insights into current software engineering processes on the customer side. A very good example illustrating this possibility was the "Agile Dashboard" project (introduced in section 2.3). During the requirements elicitation activities, the students got directly in touch with agile software development teams at John Deere. To get a better understanding of the requirements, in particular of the information to be displayed on the dashboard, the students were given the opportunity to participate in daily stand-up meetings and learned what it meant to fill backlogs for more than five Scrum teams worldwide. In addition, the other projects also enabled the students to gain very deep insights into the agricultural domain, current products, and open challenges related to their development.

3.1.3 Challenges

As stated above, one of the main learning objectives associated with the capstone project is related to experience working in a team. And this is indeed also one of the greatest challenges. As reflected in the expectations summarized in Table 1, the students form an international team of students with different backgrounds. That is, besides Computer Science students enrolled at the University of Kaiserslautern, students enrolled in the so-called "European Master of Software Engineering" program are also given the opportunity to participate in this project. These students come from different universities in Europe and spend two or more semesters in Kaiserslautern. Thus, the students typically do not know each other at the beginning of the project and have different educational backgrounds, which is also quite challenging.

To allow the students to get a good start into team

Table 1. Expectations from students' viewpoints and their fulfillment in current capstone project 2012 (average values are calculated based on the questionnaires filled out by 20 out of 24 students; 1 = "I totally agree", 2 = "I agree", 3 = "neither/nor", 4 = "I disagree", 5 = "I totally disagree")

Real-life project with customer from industry	
Experience interaction with a customer	1.85
Understand which parts of the project are more important for the customer	1.95
Experience how to elicit customer requirements	2.15
Bring software engineering (SE) concepts learned in class to a real work environment	1.95
Develop code for real-life tasks	1.80
Experience Fraunhofer IESE SE knowledge	1.95
Experience Fraunhofer IESE work environment	1.85
Team Work	
Collaborate in an international team of students	1.5
Gain the technicalities and skills to work in a team	1.9
See what appropriate communication in a team should look like	1.9
Specific software engineering disciplines	
Extend and improve programming skills and knowledge	2.35
Experience how requirements are brainstormed into design	2.05
Get insight into requirements engineering and business analysis	2.05
Learn how to create and manage priorities of tasks	2.15
Get acquainted with different tools and technologies used during a SE project	1.95

work, the capstone projects typically start two to three weeks before the other lectures start at the university in the respective semester. During these weeks, the students basically work full-time, which enables them to get to know each other, obtain a common understanding of the goals and requirements to be achieved in the project, and familiarize themselves with their assigned roles and responsibilities.

Once the lectures start, the students have to carefully coordinate and plan their activities, and communicate the current status and possible problems on a regular basis. This is really one of the most difficult tasks for them, as they are also expected to deliver good results at the end within a rather short development time.

3.2 Supervisors' viewpoint

3.2.1 Objectives and challenges

As already stated in the introduction of this article, the role of a supervisor is challenged by various questions, such as how to support the students in transferring theoretical knowledge in such a project setting, or how to enable the students to experience the benefits of such real professional scenarios compared to scenarios provided by university settings.

To overcome this challenge, these capstone projects have to be planned and organized carefully, which is one of the main responsibilities of the supervisor in charge of organizing the project from the AGSE side. The planning activities already start four months before the official project start. At this stage, the supervisor has to approve and discuss practical problems to be solved in the project together with the customer. This is important in order to announce the project at the university, but also to organize suitable support for the students. This includes both methodological support provided by a team of researchers working at Fraunhofer IESE with competencies in the various software development disciplines relevant within a particular project as well as technical support, such as preparing and setting up a suitable technical infrastructure and development environment.

Apart from these challenges, such capstone projects are also very beneficial for researchers acting as supervisors: They provide a very good setting for designing and running empirical studies such as experimental comparisons or case studies aimed at investigating particular aspects of newly developed software engineering methods. In the past, several empirical studies have been conducted during these capstone projects, for example [8, 9].

3.2.2 Recommendations based on lessons learned

Supervisors must be very talented project managers: The skills required by the supervisors being responsible for managing the overall capstone project are manifold. They need to manage an industry-style project with a team of typically less experienced software engineers. Thereby, the expectations of the students enrolled in the project are quite high as they have not yet experienced the challenges of managing a project in an industrial setting and expect an extremely smooth course of the project. Supervisors in the role of project coordinators/managers need to consider the different concerns of various stakeholders that even might be in conflict. The *customer* expects a running prototype with the functionality discussed up-front, the *students* expect to get the chance to learn without too much time pressure, *researchers* are interested in empirical results which requires a more or less strict adoption of pre-defined methods, processes, and tools under investigation. Supervisors that typically have a university background and by themselves have not led many large projects, especially in an industrial setting, are challenged by all these issues.

Iterative software development is very beneficial for capstone projects: As part of the project planning activities, decisions have to be made regarding suitable software engineering process models to be followed during the project. In the Grower's Notebook project (see section 2.1), we followed a waterfall approach. This proved to be quite efficient in this project as some of the students already started a little bit earlier than the official project kick-off and did detailed requirements analysis beforehand. Also in the Mobile Configuration Assistant project (see section 2.2), we deliberately decided to follow a waterfall approach initially, as the elicitation of current configuration processes and an understanding of the domain at the beginning of the project were very crucial for the success of the project. Therefore, the project started with detailed requirements analysis in which basically all students were involved. Afterwards we parallelized development activities in order to avoid "waiting times" for students involved in downstream activities such as coding and testing. That is, we divided the group of 13 students into two teams, with one team responsible for UI design and testing, the other one for architecture design and implementation. This resulted in the problem that the students started thinking in two different teams and did not manage to communicate and align their results.

The setting of the Agile Dashboard project (see section 2.3) was very suitable to decide upon an iterative approach. This turned out to be very beneficial as, for instance, both development and testing activities started quite early in the first iteration (i.e., after three weeks). In fact, the availability of early prototypical solutions was very

helpful for the students to get a better understanding of requirements and to get early experience with new technologies. This is in fact a risk in waterfall approaches as due to the late implementation phase the prototype might not be implemented in the end (especially if the students would have to cope with new technologies).

However, iterative development also comes with a challenge that is attributed to the typically short development cycles and interrelationships of development activities: The students have to understand all the information needs of their own and other roles of each development phase and align their elicitation, documentation, and communication activities to these information needs. In order to support this, we asked the students in the current project to elaborate an ''artifact landscape'' capturing the flow of information between the various roles involved in the development activities. This landscape is intended to visualize which artifacts are relevant for which role on what level of detail and for which task. We expect that reflecting their experience in the form of such a landscape will be very beneficial for the students and help them understand that providing the right information at the right point in time on the right level of detail is very crucial when working in a team.

Regular intermediate presentations with the customer should be planned: During all of our projects, we planned and conducted intermediate presentations to the customer. This was also beneficial for the students, as they had a chance to present and discuss initial ideas quite early in the project and to validate their understanding of the customer requirements. Furthermore, these presentations also supported the elicitation of requirements quite well, as we experienced in the Agile Dashboard project where the requirements were not really clear and detailed at the beginning. This was the chance for the students to start thinking about creative ideas. They presented their ideas in the form of early paper prototypes at the first intermediate meeting. This technology was really helpful for eliciting and negotiating more detailed requirements. Finally, early positive customer feedback turned out to be very motivational for the students.

Each student should take over more than one role: During our history of projects, we observed that the students started thinking in different ''teams'' when they were assigned to a particular role. They talked about ''the requirements engineering guys'' or ''the architects'' and had problems communicating their current ideas and status to the other disciplines. To interfere this thinking and to increase awareness and understanding of their own role as well as responsibilities of other roles, we assigned the students to a combination of two different roles. This means that a student assigned to the role combination ''requirements engineer and tester'' is mainly responsible for the coordination of requirements engineering activities and for the delivery of requirements reports. But it is also expected that the student should also be actively involved in testing activities. We consider such role assignments as very effective because the students gain a deeper insight into different disciplines, and the communication flow between different disciplines can be supported as well. In the current project, we asked the students to define and agree their individual roles and responsibilities within the team by creating a persona [10] for each student (e.g., dependent on competencies or interests). This was a nice exercise at the beginning of the project and helped the students to get to know each other a little bit.

Experiences should be discussed in retrospective meetings: As the students should experience a software development project in a real life setting, they should also face typical real-life problems, e.g., how to cope with unclear requirements or delays due to inappropriate communication. Of course, some of these problems could be avoided upfront by taking suitable mitigation strategies by the supervisors, but in fact we consider such ''negative experiences'' with problems and their consequences as valuable experience for the students. Therefore, we established retrospective meetings during the project where the students are expected to share their positive experiences as well as particular problems, and discuss possible solution or mitigation strategies.

At the kick-off meeting of the current project, we also elicited important factors when working in a team and had the students elaborate ideas on how these factors can be achieved (such as regular status meetings in the team to support communication and coordinate activities). At the end of the project, we plan to confront the students in the final retrospective with their initial ideas so that they can reflect on which methods turned out to be successful and which did not.

Short tutorials should be used to brush up key knowledge required in the project: We experienced that students cannot easily adopt the knowledge they gathered in lectures from the beginning of the project without some brush up of their knowledge. Typically, they participated in the related lectures some time ago. Furthermore, students from other universities joining the master course and the capstone project might have a different knowledge base that should be aligned with the rest of the team. Therefore, in half-day tutorials on requirements engineering, architecture, etc. key knowledge is presented again to polish up the students' minds. Short tutorials can also be used to introduce technologies to be used in the project that are not yet

familiar to the students. In our capstone projects involving iOS (such as Mobile Configuration Assistant, see section 2.2) we introduced the students to the ObjectiveC programming language and Apple's XCode development environment because the majority of the students never dealt with such technologies and tools.

3.3 Customer's viewpoint

3.3.1 Objectives and challenges

Capstone projects provide unique opportunities for industrial partners to develop competencies within the technical and business communities. The intent is to acquire students who have an interest in the subject area as well as a newly acquired knowledge of the most modern design, technology, and architecture skills. As a concept, the combination of enthusiastic and engaged students with a challenging problem yields extraordinary solutions at a price that would be hard to reproduce within the industry.

Though the organization and management of the student resources is left to the university leaders, the organization of a capstone project by the industrial partner for its own use is equally important and critical for success.

In fact, capstone projects also present challenges in the form of project organization and socialization of their output and purpose. Industrial partners should take care to select projects that are both important to the business and will provide challenging experiences for the students. It is equally import to engage the business as they need to develop competencies in the use of these technologies in parallel. Project ownership must be shared between the business and technical staff members. Both enrich the project deliverables and the student experience.

Capstone project planning should be approached differently than typical operational projects, with more emphasis on organizational learning and network development than on near-term deliverables. Technical organizations can use these investments to engage their employees and excite their business partners regarding new capabilities and opportunities. Network development between the university and the industry partner should also be a priority as it can lead to a recurrent source of new skills and innovative ideas.

Finally, the challenges faced by industrial partners are that staff members are not practiced in project selections that review multiple design alternatives. Most project managers will seek to limit the scope of a project to a single design too early in the project phase. The creation of sustaining networks outside the organization is not a practiced skill. These activities are often viewed as unnecessary and time consuming. Industrial partners are challenged to engage their employees and to excite their curiosity regarding changes in their approach to problem solutions and product content. Capstone projects offer an opportunity to do this. However, industry partners are challenged by the time it takes to involve their best resources and incrementally transfer capstone learning.

The most effective project organization will leverage a partner's critical skills, knowledge of the problem statement, and understanding of how to monetize the ideas. Successful project organization by the partner must focus on organizational learning. Projects should produce valued results.

3.3.2 Customer reflections on project history

In the following, some customer reflections on the previously introduced projects are summarized.

The *Grower's Notebook project* (see section 2.1) generally delivered the expected result. A prototype setup could be realized to serve as a test-bed for investigating upcoming technologies regarding mobile devices and Cloud-based data storage. The students were able to produce a result incorporating various technologies that were new to them including iOS and the Google Apps Engine. The key functionality to demonstrate the scenario (illustrated in Fig. 1) has been implemented. In follow-up projects, the implemented prototype was successfully used as a basis for further technology investigations.

However, some of the initially planned features of the prototype could not be realized within the capstone project. This did, for example, concern certain security features, which were initially specified and designed into the architecture, but due to time constraints were not implemented till the project's end. Multiple reasons that slowed down progress during the project could be identified. On the one hand, students were not familiar with the agricultural use case the system was intended for. For this reason, incorrect assumptions were made in some points, even though several requirements analysis meetings took place with the industry partner. On the other hand, the applied mobile device and cloud technologies as well as the corresponding development tools were new to most participants. In addition, due to the restricted amount of time, the students were divided into two groups working in parallel. It was found that due to inefficient alignment between the teams, some parts of the architecture, the UI, and the already implemented software had to be reworked, causing additional effort.

In summary, considerable lessons were learned from the capstone project in terms of how such a short-term effort can be conducted efficiently. Valu-

able insights were gained regarding aspects and circumstances that can potentially slow down the progress of such a project.

While the *Mobile Configuration Assistant project* (see section 2.2) was successful as a technical evaluation and functional demonstration, it failed in its ability to engage product program leaders who stood the most to benefit from its results. These failures were the results of incomplete project organization by the industry partner and unavailability of critical stakeholders.

During project execution, the capstone team was supported by partner product architects, customer services staff, product managers, and advanced marketing staff. These resources ensured that the project remained on target to solve the business problem, deliver a proof of concept that could be realized in production, and delivered a user experience that met the needs of both an expert and a lesser skilled user. The project addressed the complexity of detailed code structures related to creating configuration applications and represented the needs of the second stakeholder, the application developer.

It is this second set of stakeholders whose needs were addressed, but involvement was unintentionally minimized, which turned out to become a source of failure. In this case, the failure to engage and excite the production developers who were responsible for current deliverables created a disruption in the adoption of these project results into future product program planning. The approach demonstrated by the prototype created a gap between the incumbent solutions and emerging technology supporting model-based solutions. As a result, the developers rejected the concepts in part because the project was not organized to develop their knowledge and competencies with the solution.

In summary: The project was extremely successful on many important dimensions of organizational learning but failed to deliver all that it could have to improve the products program.

In the *Agile Dashboard project* (see section 2.3) the students surprised the customer in a very positive way and delivered a very nice visualization that is now in use at the ISG of John Deere in Kaiserslautern, Germany as well as in Des Moines, USA.

Initially, the software was intended to be a typical Windows C++ application. This was mainly because of the knowledge and experience of the project stakeholders at ISG. The students listened to this request but came back to ISG a few days later with a quite different approach: They clearly mentioned that a modern application is web-based using a client-server architecture that can easily be extended to mobile devices or multiple clients when more screens are needed. Also, the entire visualization should be created using HTML. This was great feedback, of course. It was clearly the better approach for the project. It showed the students' minds were not limited by the few technologies that are introduced in a department over the years and that most employees are familiar with. So in addition to the project result itself, which was great, it helped to trigger the employees to start thinking differently and to look at other technologies as well. This effect is similar to the one achieved by companies running Hackathons or having Innovation Days.

3.3.3 Recommendations based on lessons learned

The following list provides practitioner guidelines for supporting the organization of successful capstone projects by an industry partner.

Select important problems that are both needed and supported by your company. Do not make the assumption that a project of little interest to your organization will be of interest to the students. Also, you will be investing some of your most talented resources to help guide these projects. This investment requires that these problems matter.

Secure both technical and business ownership in the project. It is easier to value the technical learning of a capstone project than the development of business competencies that provide new business opportunities. Use these projects to develop an understanding and competencies in the business.

Select problems in which you want to evaluate multiple solutions or when a solution is not known. If you know how to deliver a solution, then that project should be done internally or by a commercial consultant, not as a capstone project. Such projects really just need to be done cost-effectively. Capstone project candidates should include an element of uncertainty related to the solution method. Be willing to explore the alternative solution paths provided by the capstone team and consider that the outcome you envisioned during planning may not be the outcome of the project. Experience states that project outcomes that are very different from those proposed during the planning stage are often more valuable because they take advantage of an innovation made possible through the diversity of the team thinking.

Engage the organization to ensure that there are paths for learning and adoption by both the technical and business stakeholders. Technical members should be engaged to review the technical approach and merit of the solution developed. Business stakeholders need to understand the possibilities that a capstone project offers. Together these stakeholders should be responsible for evaluating and, when appropriate, transferring the proof of concepts created by the capstone project into product roadmaps.

Consider the above requirements when selecting your industrial project lead. Consider also that your best operational project managers may not be appropriate leaders of a capstone initiative. Capstone project management should focus on organizational learning, on developing employee engagement, and on creating excitement for the solution. Project deadlines, deliverables, and project management should remain the domain of the university students and staff, as these are fundamental lessons for student participants. Industry partners should be willing to compromise on deliverables in support of advances in learning and innovative solutions.

Finally, good operational project managers are effective because they quickly reduce the scope of a project, kill disruptions that impede forward progress, and focus on tangible deliverables with immediate and clear value. These are appropriate ambitions for projects done with consultants but may reduce your ability to innovate on a solution and reduce the value of a capstone project. It also limits the partner's ability to maximize innovation, which is a key benefit of doing capstone projects.

4. Conclusion

From an educational viewpoint, capstone projects defined as university-industry collaborations provide the potential of being very beneficial especially in graduate education.

During a history of successfully conducted projects over several years between the Software Engineering Research Group headed by Prof. Rombach at the University of Kaiserslautern, the Fraunhofer IESE, as well as John Deere the authors have gained experiences comprising objectives, benefits but also challenges that come with such projects. These experiences supplemented with recommendations derived from lessons learned are discussed from three different viewpoints: (1) students, who have the chance to apply theories learned during lectures in a realistic project setting; (2) supervisors and researchers, who support the students in applying methods during the project; and (3) customers, who provide specific problems to be solved by the students.

The most valuable learning objectives from the students' viewpoint (that go beyond learning objectives of practical assignments during classes or even practical projects defined by faculty members) are related to the interaction with a real customer and working on a real problem in a realistic setting. This enables students to face typical real-life problems that are similar to those they might also experience after their university education in their jobs.

However, the success of such collaborative capstone projects is also strongly dependent on careful planning, management and organization. That is, the supervisor being in charge of coordinating these projects from the university side has to have manifold skills in order to cope with the challenges that come with these projects and to fulfill concerns of different stakeholders involved in the project.

Finally, from the customer's viewpoint, the role of capstone projects is to explore technologies and business models outside the constraints of an organization that must produce a product for profit. Thus capstone projects provide access to new thinking and increase the diversity of culture, knowledge, and curiosity of an organization when solving complex problems. While the outcome is typically a technical solution, a capstone project provides a safe environment for organizations to create global teams, gain knowledge of new technologies, and create new business competencies. Capstone projects also enable an organization to observe potential employees on a technical and cultural basis and make it easier to select those who have the needed skills and fit within the organization's culture. Finally, capstone projects provide unique opportunities for industrial partners to develop competencies within the technical and business communities.

To conclude, the concept of a capstone project is simple in nature, but the successful implementation of such a project becomes a complex activity if it is to ensure that the needs of the customer, the university, research institutes and the students are fulfilled. The experiences shared in this article could serve as valuable information and as guidelines for other universities and industry companies intending to plan and organize collaborative projects of this kind.

References

1. E. Ras, R. Carbon, B. Decker, and J. Rech, Experience Management Wikis for Reflective Practice in Software Capstone Projects. *IEEE Transactions on Education* **50**(4), November 2007, pp. 312–320.
2. E. P. Katz, Software Engineering Practicum Course Experience. In *Proceedings of the 2010 23rd IEEE Conference on Software Engineering Education and Training* (CSEET '10). IEEE Computer Society, Washington, DC, USA, 2010, pp. 169–172.
3. R. Bareiss and E. Katz, An exploration of knowledge and skills transfer from a formal software engineering curriculum to a capstone practicum project. In *Proceedings of the 2011 24th IEEE-CS Conference on Software Engineering Education and Training* (CSEET '11). IEEE Computer Society, Washington, DC, USA, 2011, pp. 71–80.
4. D. Rombach, J. Muench, A. Ocampo, W. S. Humphrey and D. Burton, Teaching disciplined software development. *Journal of Systems and Software.* **81**(5), May 2008, pp. 747–763.
5. D. Rombach, A Process Platform for Experience-Based Software Development. In *Software Reuse—Requirements, Technologies and Applications. Proceedings of International Colloquium of the Sonderforschungsbereich 501* University of

Kaiserslautern Computer Science Department Kaiserslautern, March 10–11, 2003, pp. 47–57.
6. Rally Software, http://www.rallydev.com/, last visited November 2012.
7. Jenkins Continuous Integration Server, http://jenkins-ci.org/, last visited November 2012.
8. M. Naab, Enhancing Architecture Design Methods for Improved Flexibility in Long-Living Information Systems, *PhD Theses in Experimental Software Engineering*, Fraunhofer IRB Verlag, **41**, 2012.
9. R. Carbon, Architecture-Centric Software Producibility Analysis, *PhD Theses in Experimental Software Engineering*, Fraunhofer IRB Verlag, **38**, 2012.
10. T. Adlin and J. Pruitt, The Essential Persona Lifecycle: Your Guide to Building and Using Personas, Morgan Kaufmann, Burlington, MA 01803, USA, 2010.

Anne Hess holds a German Diploma degree in Computer Science from the Saarland University of Saarbruecken, Germany. Since October 2006, she has been working as a researcher at the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE), where she is assigned to the requirements engineering group in the department "Information Systems Development". Since 2010 she has been responsible for organizing and coordinating capstone projects at the AGSE of the University of Kaiserslautern. Furthermore, she also provides methodological support in these projects in the field of requirements engineering.

Dieter Rombach studied mathematics and computer science at the University of Karlsruhe and obtained his Ph.D. in computer science from the University of Kaiserslautern (1984). Since 1992 he has held the Software Engineering Chair in the Department of Computer Science at the University of Kaiserslautern. In addition, he is the founding and executive director of the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE) in Kaiserslautern. Previous career steps included the Institute for Data Processing in Technology at the Karlsruhe Nuclear Research Center (scientist; 1978–79) and the Department of Computer Science at the University of Kaiserslautern (scientist; 1979–1984). This was followed by positions as a guest professor at the University of Maryland and at NASA (1984–1986), as a professor for computer science at the University of Maryland (1986–1991), and as a professor at the Institute for Advanced Computer Studies at the University of Maryland and project manager at the Software Engineering Labor (SEL) at NASA's Goddard Space Flight Center (1986–1991). Prof. Rombach spent the summer semesters of 1988 and 1989 as a visiting professor at the Software Engineering Institute of Carnegie Mellon University in Pittsburgh, USA.

Ralf Carbon holds a diploma (2002) and a doctor's degree (2012) in computer science from the University of Kaiserslautern. He was a member of the AGSE of the University of Kaiserslautern from 2002 to 2005. Since then he is a researcher at Fraunhofer IESE. From 2004 to 2010 he was responsible for organizing and coordinating capstone projects involving the AGSE of the University of Kaiserslautern, Fraunhofer IESE and various industrial customers including John Deere. His personal research focuses on Software Architecture and Mobile Apps.

Daniel F. Murphy is an Enterprise Architect, located at the Moline Technology Innovation Center. He holds diploma degrees in electronics, computer science, and business administration. He has worked at John Deere since 1979 holding multiple management roles in product engineering, information technology, and innovation leadership. He has sponsored and participated in capstone projects between John Deere and the Fraunhofer IESE. He remains an active contributor and is involved in various scientific activities supporting vehicle architecture and advance information technology.

Michael Hoeh is Manager Software Engineering, Decision Support in the Intelligent Solutions Group located in the European Technology Innovation Center. He holds a diploma degree in electrical engineering and is with John Deere since 1998 where he started as embedded software developer in the Precision Farming group. In 2005 he initiated the contact with Fraunhofer IESE and he is still supporting the cooperation between the institute and John Deere. He is an active contributor to capstone projects and is involved in various scientific activities around software engineering and architecture.

Christian Bartolein holds a diploma degree in computer science received from the University of Mannheim. From 2004 until 2009 he worked as a research assistant at the Automation Lab of the University of Heidelberg. Since 2009 he is with the Intelligent Solutions Group at the John Deere ETIC located in Kaiserslautern. As part of the Advanced Engineering amongst other responsibilities he worked on mobile device projects during this time.