From Storyboards to Code: Visual Product Backlogs in Agile Project Courses

Lukas Alperowitz, Constantin Scheuermann, Nadine von Frankenberg
Technical University of Munich

{alperowi, scheuerm, ludwigsd}@in.tum.de

Abstract

In interdisciplinary software engineering project courses, where more than just programming skills are needed, identifying and understanding the problem in an early project stage is essential. Our experience shows that a pure scrum based agile development approach does not fit the needs of such projects. To teach agile development courses with the goal to deliver a working vertical prototype within two weeks, we have defined and used a novel visual development approach.

In this paper we show how we use storyboards as a visual product backlog to enable students to design, understand and deliver working software in a short amount of time. We use an initial version of a storyboard that describes the key events of a visionary scenario. Students use the storyboard frames to define and prioritize implementation tasks. Parallel to that, the storyboard also serves the purpose to derive the software architecture. As soon as one storyboard frame has been implemented the associated scene is filmed and placed into a demo movie. We show that small interdisciplinary projects could benefit from our process in order to rapidly understand the problem to solve and to develop the prototype video-based in an agile manner.

1 Introduction

Teaching agile software development courses in which students learn how to work on real-world problems is a common practice in software engineering education (Bruegge u. a., 2015). Within interdisciplinary project courses with students from different domains, such as mechanical engineering, textile manufacturing or electrical engineering, prior knowledge in the field of software engineering varies. In such courses, where more than just programming skills are needed, identifying and understanding the problem at an early project stage is crucial. We believe that it is important to foster the creativity of the students right from the beginning of the project. Giving students the freedom to envision their own ideas of functionality, architecture and implementation is essential to create novel ideas and innovations.

Figure 1: **Lab Environment:** Shows the prepared lab environment the instructors provided for the two weeks lab course.

Teaching students all aspects of an agile methodology like e.g. Scrum (Schwaber u. Beedle, 2002) in a short amount of time is challenging for instructors. Students need to learn not only how to express requirements in user stories or scenarios, but also how to properly estimate their complexities and priorities.

In this paper, we present an approach to teach agile software development in project courses that last up to two weeks. To achieve this, we present how we use storyboards to visualize requirements and to organize the development workflow. Our approach is based on the work of (Creighton, 2006), (Rosson u. Carroll, 2009) and (Bruegge u. a., 2015).

Storyboards are a well-established practice in the filming industry to organize and sketch out movies.

We use storyboards as a visual backlog that helps students to understand the problem that will be addressed during the project. Each frame on the storyboard describes certain functionality and the involved subsystems. Every morning it is shown to the students during a daily stand-up meeting. The team decides which frame of the storyboard needs to be im-

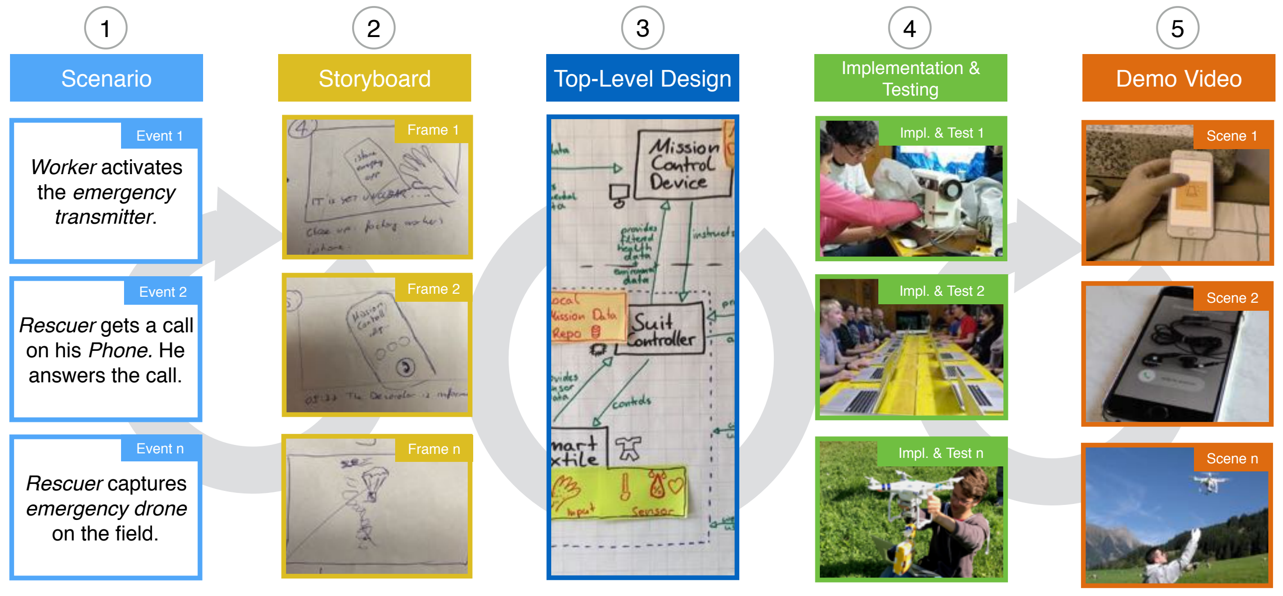

Figure 2: **Visual Backlog Process:** Example process from the written scenario with the flow of events to the demo video and final product.

plemented next. Tasks define the pieces of work that have to be accomplished to implement the functionalities described in the storyboard frame. This can be any kind of activity such as writing code, 3D-printing or crafting something. As soon as one storyboard frame has been finished, the students act the frame out and film it. The corresponding frame in the storyboard is then replaced with the filmed clip.

In this paper we describe this visual product backlog through a case study, which we conducted in 2016. We show how the approach allowed us to develop a working prototype of a Human-Centric Cyber-Physical System (Scheuermann u. a., 2016a) in a two-week course that was conducted by a group of 15 students.

In Section 2 we describe the project setup and environment. We then introduce the process of creating the visionary scenario in Section 3. In Section 4, 5 and 6 we describe the prototype creation process from storyboards to code. Finally we conclude our findings in Section 7.

2 Project Setup

In this section we present the approach we applied in an agile project course we conducted in 2016 (Bruegge u. a., 2016). The course consisted of 15 students from different domains and four instructors from the field of software engineering. We have selected students from different domains, as we wanted to address an interdisciplinary problem. The scope of the project was defined as follows: The students where told upfront that components such as e-textiles, 3D-printers, micro-controllers and mobile devices can be used. The baseline question was to find new approaches to use these components to form a Human-Centric Cyber-Physical System (Scheuermann u. a., 2016b) which supports humans in hazardous environments. We provided the students with equipment, such as stitching and soldering machines, software development infrastructure, a lab room and a 3D printer. We further provided a version control system, so that the developers and subteams were able to collaborate easily, and an analog task board together with sticky notes. Figure 1 shows the lab setup at the first day of the project.

3 Defining the Visionary Scenario

Apart from the general scope, the students were not given any details about the desired outcome of the projects. Therefore, we asked them to define a visionary scenario about helping humans in hazardous environments (see Figure 2-①).

First, students collected their ideas by means of sticky notes. Next, the students formed groups and prioritized the ideas. Finally the students selected the best ideas and began to create the visionary scenario. The visionary scenario describes an ideal system to be realized. It can also describe functionalities that may never be implemented, as they are yet too visionary or technically not feasible. At the end of the first day, the students had finished writing the visionary scenario.

4 Storyboard - Our Visual Backlog

On the second day the students started creating a storyboard as shown in Figure 2-②. The instructors gave a short introduction of how to sketch storyboard frames and provided some background to the methodology.

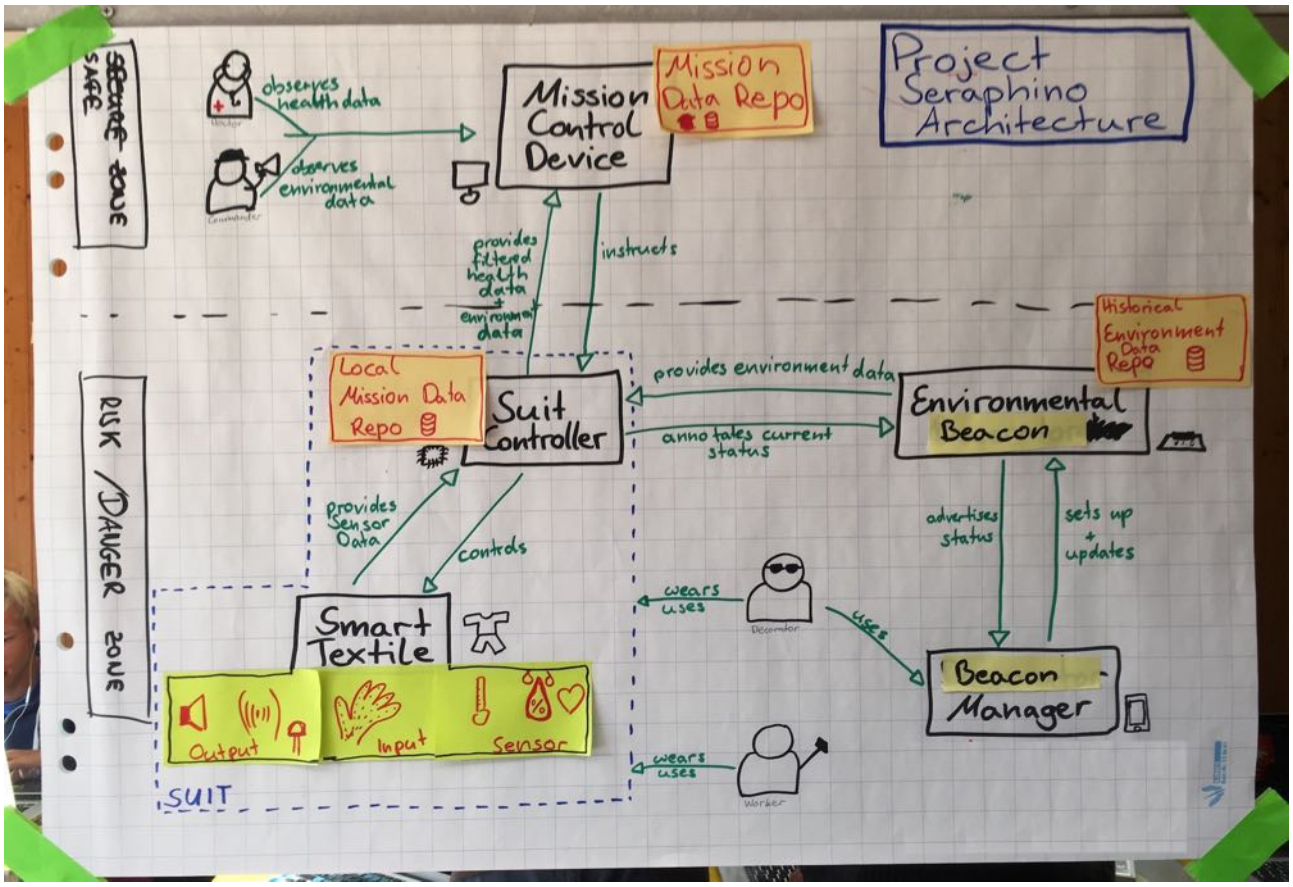

Figure 3: **Top-Level Design:** It is technology-independent, identifies the subsystems, their interactions, but does not map technologies, protocols or hardware to the subsystems.

Storyboards are a well-established practice in the filming industry to organize and sketch out movies. The storyboards consist of sequences of frames (see Figure 2-②). A frame shows a specific drawing, which represents the shots that are planned for a movie. Each frame on the storyboard describes certain functionalities of the system.

The students illustrated each event of the visionary scenario in a storyboard frame. To actually draw the frames, they revisited the events described in the visionary scenario and answered the following questions:

- What is the main focus of the project?
- What are the most important requirements?
- What do we try to accomplish with this project?

We observed the following challenges when the students sketched the storyboard:

- Emphasizing the key ideas and requirements
- Narrating a coherent story with smooth transitions among all frames
- Staying on the right level of abstraction

From then on, the storyboard served as a visual product backlog.

5 From Storyboard to Code

At this stage the storyboard consists of only rough sketches. We then asked the students to create a technology-independent Top-Level Design of the software architecture (see Figure 3 and Figure 2-③) of the intended prototype based on functionalities described in the storyboard. The instructors explained the main characteristics of the Top-Level Design.

A Top-Level Design expresses the overall architecture of a system. It shows each subsystem and the data exchange between them. The notation does not follow strict rules, but conforms to the following convention:

Each subsystem is depicted with an icon and labeled with a textual description. Arrows and a short textual description are used to describe the entities that are exchanged between the subsystems. The Top-Level Design models the problem domain technology-independently to keep the focus on the system's overall architecture and does not include implementation details. While creating the Top-Level Design, we asked the students the following questions:

- Which parts of our system are relevant for the customer?
- What can we eliminate to avoid unnecessary complexity and distraction?

During each iteration of the Top-Level Design the storyboard frames may be refined. Changes in the storyboard may influence the architecture and vise versa. The students then began the implementation process: They chose a frame from the storyboard, and defined and finished the necessary implementation tasks for the involved subsystems.

6 From Code to Demo Movie

After the functionalities of a certain frame are implemented, each subsystem is unit-tested (see Figure 2-④). Further, an integration test is conducted so that the subsystems involved in one scene are ready to be filmed. After the the unit- and integration-tests are passed, the scene is staged, filmed, and the hand-drawn frame in the storyboard is replaced with the filmed clip (see Figure 2-⑤).

This process repeats for a week. In daily meetings the current iteration of the animated storyboard is shown to the whole team and is used to prioritize the daily work each student promises to do. After several iterations, more and more frames are replaced with the actual movie-clips, showing the implemented systems. At the end of the course a working prototype and a video in which the scenario is staged is delivered.

7 Conclusion

We have defined and successfully demonstrated a novel approach to develop a prototype in an interdisciplinary two week lab course.

In advance, the instructors defined a rough scope of the project in order to find and identify new ways to support humans in hazardous environments. For the implementation of a vertical prototype, we provided the students with a version control system, as well as with hardware, including micro-controllers with sensors and actuators, mobile devices and cameras, a 3D-printer and textiles. The students defined the problem and visionary scenario on their own. Based on storyboard frames, they created a visual backlog which was iteratively turned into source code and hardware prototypes. The students followed an agile process, incrementally filmed implemented system functionalities and then added them to a demo video.

Our approach shows that students are capable of developing a vertical prototype in two weeks using a visual backlog. The demo movie supports the students to integrate each component into the system as soon as it is finished, while also showing the current progress. The storyboard and scenario movie helps participants to understand the key aspects of the project. Visual product backlogs facilitate a common understanding of the problem to be solved in an early stage of the project. We believe that the use of storyboards is a promising approach to teach agile software engineering in interdisciplinary student projects in a short amount of time. However, while our approach was successful for exploring requirements and system design, we have not yet applied visual product backlogs to projects that are aiming for more than a proof-of-concept.

References

[Bruegge u. a. 2015] BRUEGGE, Bernd ; KRUSCHE, Stephan ; ALPEROWITZ, Lukas: Software Engineering Project Courses with Industrial Clients. In: *ACM Transactions on Computing Education (TOCE)* 15 (2015), Nr. 4, S. 17

[Bruegge u. a. 2016] BRUEGGE, Bernd ; SCHEUERMANN, Constantin ; ALPEROWITZ, Lukas: *Ferienakademie 2016 - https://www1.in.tum.de/projects*. 2016. – `https://www1.in.tum.de/projects`

[Creighton 2006] CREIGHTON, Oliver: *Software cinema: employing digital video in requirements engineering*. Verlag Dr. Hut, 2006

[Rosson u. Carroll 2009] ROSSON, Mary B. ; CARROLL, John M.: Scenario based design. In: *Human-computer interaction. Boca Raton, FL* (2009), S. 145–162

[Scheuermann u. a. 2016a] SCHEUERMANN, Constantin ; STROBEL, Maximilian ; BRUEGGE, Bernd ; VERCLAS, Stephan: Increasing the Support to Humans in Factory Environments using a Smart Glove: An Evaluation. In: *2016 International IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*. Toulouse, France : IEEE Computer Society, 2016, S. 847–854

[Scheuermann u. a. 2016b] SCHEUERMANN, Constantin ; TANDON, Raman ; BRUEGGE, Bernd ; VERCLAS, Stephan: Detection of Human Limits in Hazardous Environments: A Human-Centric Cyber-Physical System. In: *The 14th International Conference on Embedded Systems, Cyber-physical Systems, and Applications, Las Vegas (USA)*, 2016, S. 3–9

[Schwaber u. Beedle 2002] SCHWABER, Ken ; BEEDLE, Mike: Agile Software Development with Scrum. (2002)