An Experiment in Teaching Innovation in Software Engineering

Video Presentation

Bernd Bruegge
Technische Universitaet Muenchen
Chair for Applied Software Engineering
bruegge@in.tum.de

Harald Stangl
Technische Universitaet Muenchen
Chair for Applied Software Engineering
stanglh@in.tum.de

Maximilian Reiss
Technische Universitaet Muenchen
Chair for Applied Software Engineering
reissm@in.tum.de

Abstract

The *DOLLI* project was a large-scale educational student project course with a real customer, offered to students in their second year. In the time frame of a single semester a functional system was developed and delivered to the customer. We experimented with a shift from a traditional lifecycle to an agile process during the project, and used video techniques for defining requirements and meeting capture.

Categories and Subject Descriptors K.3.2 [**Computers and Education**]: Computer and Information Science Education - Computer science education. K.6.1 [**Management of Computing and Information Systems**]: Project and People Management - Systems analysis and design, Systems development, Management techniques.

General Terms Management, Experimentation, Human Factors

Keywords Video-based requirements engineering, scenario-based design, teaching software engineering, single project course

1. Introduction

In the 1992 OOPLSA conference we first reported about a large single project course offered to 30 senior students [1]. In the meantime, software engineering has experienced some significant progress in many areas. We have moved from OMT to UML, we developed the course into a globally distributed project course [2] [3], and Wikis and agile methodologies such as Scrum have appeared. Continuous integration has turned build and release management into a routine affair, and open source tools such as Eclipse, Maven, and Cruse Control have become indispensible tools in the arsenal of many software engineers. In addition we have observed the appearance of digital video, made possible by faster computers, larger storage capacities, and powerful but inexpensive video cameras.

In this paper we report about an experiment to incorporate these recent developments into teaching. In particular, we wanted to investigate whether it is feasible to use them together in a large single project course, and offer it even earlier in the curriculum.

Copyright is held by the author/owner(s).
OOPSLA'08, October 19–23, 2008, Nashville, Tennessee, USA.
ACM 978-1-60558-220-7/08/10.

The DOLLI (Distributed Online Logistics and Location of Information) project was a student project course with over 50 participants in the winter term 2007 at the Technische Universitaet Muenchen. The students—most of them sophomores in their 3rd semester—worked together as one big team instead of in small groups, and they had to solve a real problem that was posed by a real customer: the Munich Airport.

The general problem to solve was "location tracking", and consisted of four requirements. The first was to provide a way to track luggage on its way from the belt system to the airplane. The second was to provide mechanisms to track movable resources at the airport like dollies and towing bars using WiFi tags and the existing WiFi localization infrastructure. The third was to design and build a novel visualization mechanism that merges these location data with a 3D representation of the airport using a video game metaphor and touch-based interaction. The final requirement was to provide a mobile system to the IT Field Service to inform employees about new incidents on their mobile communication devices while they are en route. Moreover, these incidents should automatically be dispatched to the employee closest to the reported problem by using traceable devices.

2. Infrastructure Setup

At the beginning of the project, a top-level design was developed by a software architect who was a member of the teaching staff together with the customer. The identified subsystems defined the structure for the team organization.

Based on this team distribution the communication infrastructure of the DOLLI project was set up featuring mailing lists, team wikis, team blogs, the teams' software version control repositories, and meeting room distribution. We also set up a mechanism based on Podcast Producer so that videos from all team meetings to be posted automatically on the team Wikis.

The teams organized themselves after they were assigned their subsystem tasks.

In addition to the subsystem teams, several cross-functional teams were defined, in particular an architecture team, a coach team, and a film team. Inter-team communication and coordination was facilitated by members of the coach team, staffed primarily by students from higher semesters. The role of the coaches was to facilitate the team meetings and to provide the communication between the teams and project management.

The task of the architecture team was to develop the overall system architecture from the given top-level design and coordinate API request from the various subsystem teams.

The task of the film team was to create scripts and shoot three films: A scenario film that visualizes the functional requirements, a making-of film that describes the interaction and the atmosphere between the project participants, and a trailer that can be used to market the project. The video described in this paper is based on the making-of film produced by the students.

The project was scheduled for a 4-month duration to fit the semester structure of the university. To deal with the workload caused by the final exams at the end of the semester, we paused the project during the exam time for 4 weeks and resumed the development afterwards for a two-week full-time block. The initial development was done at laboratories provided by the Chair of Applied Software Engineering. After the exam pause, we moved all the equipment needed for the continued development to the airport using office space of a floor in an empty building provided to us by the Munich Airport. For the demonstration of the system, we moved the equipment once more to the target environment at Terminal 1 at the Munich airport.

2.1 Different Software Development Methodologies

One of our objectives was to teach the students to work on a real software project. Another was to expose them to traditional and agile software development methodologies. We started in June 2008 with an architecture-centric software lifecycle based on the Unified Process. The inception phase focused on four workflows: the definition of the project scope, elicitation of the requirements, the development of the software architecture, and the setup of the environment, in particular the IT and communication infrastructure. The elaboration phase started with a project kickoff event at the beginning of the semester, when the customer presented the problem to the students, and the problem statement was given to them. In the elaboration phase we focused on the workflow Requirements Engineering, Analysis, System Design, and Object Design. Reviews with the customer led to several iterations, in particular in the system design. After the system design was fairly stable, we switched to *Scrum*. In Scrum terms, we entered a 2-week sprint, focusing on the delivery of the system. During the sprint we revised the acceptance scenarios, addressed changes to the object design, and performed integration and system testing tests. From a research perspective, we wanted to investigate how well a switch from a traditional to an agile process in the middle a project works. The project results demonstrate that such a switch can be done.

2.2 Continuous Integration

Another objective was to teach the students the idea of continuous integration right from the beginning, that is, we wanted to expose them to build and release management issues from the very beginning. From the unified process perspective, we started the deployment workflow when we started the elaboration phase. From a pedagogical perspective, we replaced the "Hello World" program usually taught to beginners with a program in which all subsystems of the top-level design compile successfully and can be built and run together on the first day of the project. We called it "Hello DOLLI." Our project metrics demonstrate, that this can successfully be done with 2nd year students. On some days we had more than 80 builds of the complete system. The total build count for the duration of the DOLLI project was 1373.

The project was truly interdisciplinary because the students developed software as well as custom hardware to track individual resources at the airport.

2.3 Use of Video

We used video throughout the whole project duration to visualize the requirements and to teach soft skills. All reviews (with and without the customer) were filmed to provide feedback to the students on their presentation skills. All team meetings were filmed to record decisions and to keep team members who could not be present up to speed. The teams recorded their meetings themselves with cameras attached to a Podcast Producer system. With the Podcast Producer technology, the videos were automatically converted and made accessible on the team specific web pages.

To visualize the project requirements, we used a technique called Video-based Requirement Engineering [4]. This technique is based on scenario-based design [5], but it uses multi-path video in addition to text to describe the scenarios. We painted a video picture of a scenario of a field service employee receiving incident reports on a mobile device. Such a scenario-based video eases the communication between customer and developer, and helps resolve misunderstandings and ambiguities. The use of green-screen technology enables us to create videos of the use of a device even when the user interface is not yet available (see Figure 1).

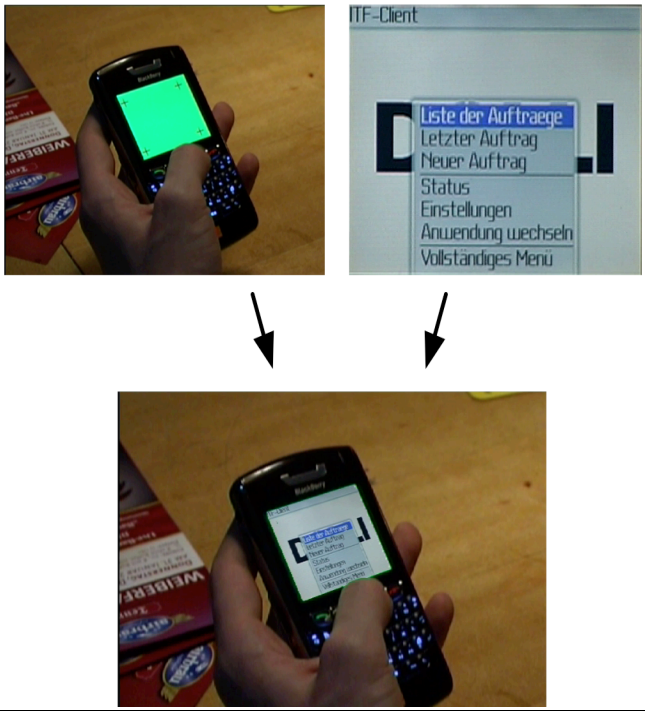

Figure 1 Green-Screening Technology is used to insert a GUI prototype developed on a desktop-based simulator into the display

3. Project Results

At the end of the project, the students presented their results in the target environment of the terminal 1 at the Munich airport with real passenger and baggage data. In the client acceptance test the participants successfully demonstrated the project requirements:

The positioning manager, containing the components under control of the data-management and the knowledge-management teams, is able to receive location update information from existing airport positioning systems as well as from newly developed DOLLI systems components. The problem of storing and serving information about traceable objects was solved in a very flexible manner. It even handles traceable objects contained in other traceable objects, e.g., a piece of luggage inside a container on a dolly. It is possible for a subsystem to register for events like "the object is moving faster than it ever could" or "is the object in an area it is not allowed to enter."

The WiFi subsystem provides an interface to the Cisco Location Appliance to receive position updates for moving objects (and if possible, also telemetry data) to feed them into the positioning manager. The Munich Airport has full WiFi coverage. But to provide more flexibility, the DOLLI WiFi-team developed a circuit board with its own programmed processor to connect a GPS receiver with a keyboard and a WiFi tag to provide a cheap and small GPS location device.

The luggage tracking team successfully demonstrated how RFID-tagged luggage could be identified when placed into containers. With this information entered into the positioning manager other subsystems could now trace the movement of luggage between the belt system and the airplane.

The IT Field Service Team successfully demonstrated the information flow from a new incident entered into the desktop-based Remedy Action Request system to a Blackberry used by the field technician. After closing the incident, a message is then sent from the Blackberry to the Action Request system. Another successful demonstration was the repeated update of the technician's position with the help from WiFi and GPS in the Blackberry. The position information, accessible via the positioning manager, can now be used by the field technician dispatcher to dispatch the nearest suitable technician to an urgent SLA (Service Level Agreement) incident.

Finally, the Visualization Team demonstrated a new approach to 3D visualization of the Munich Airport based on the metaphor of interactive video games and touch. The DOLLI interface supports multiple displays and can show all traceable objects available via the positioning manager.

Overall the DOLLI system was produced in an equivalent of 100 person months and consists of 125 KLOC new code including adapter code interfacing with the existing airport systems. The DOLLI code breaks down into 8 KLOC Blackberry code, 17 KLOC for the 3D visualization (OpenGL, Shader Language, Objective-C++) and 98 KLOC for the other Java components.

4. Conclusion

The course structure described in this paper enables us to tackle large-scale software projects even with second year students. It scales to a large number of students because they work in a self-organizing way after they are assigned their tasks. Their high motivation came not least from the fact that they knew their results would be used in production or developed further in subsequent projects.

What the students accomplished in the DOLLI project is quite impressive; it could be a fluke, but we don't think so for the following reasons. First, today's first year students have already been exposed to programming when they enter the university. In Bavaria, for example, computer science is now offered as subject in high school ("Gymnasium"), the students learn about object-orientation in the 8th grade. As a result, we can teach advanced object-oriented programming languages in the first semester and software engineering in the second semester.

Second, the top-level software design and the build server infrastructure are set up before the course starts. The same applies for the setup of the tools and the communication infrastructure. This requires a substantial amount work for the system administrators, but it can be done before the students join the project.

Third, the use of digital video has become surprisingly easy. In fact, using Podcast Producer is considered fun, not work by the students. From the videos, we can provide immediate feedback to the students without offering a separate course on soft skills. For the students this was a strong motivator.

Finally, the state of open source development tools has become quite mature, in particular Eclipse, Maven, and Cruise Control had a great impact on delivering a working system. Social software, in particular Wikis and Blogs, helped a lot in connecting the students in a very short time frame.

Acknowledgments

We would like to thank our clients, Harald Ranner, Marc Lindike, and Michael Zaddach from the Munich Airport, who had the vision to go along with us in this experiment, helping us in every phase of the project. We thank them especially for their generous provision of office space during the Scrum phase. This research was supported by a grant from Siemens Corporation. In particular, we would like to thank Klaus Beetz and Oliver Creighton from Siemens CT for their continued support of the video-based requirement engineering idea. We would also like to acknowledge the support from Gertraud Unger from Apple.

References

[1] B. Bruegge, J. Blythe, J. Jackson and J. Shufelt, Conference on Object-Oriented Programming Systems, Languages and Applications,Vancouver, ACM Press, pp. 359-376, October 1992.

[2] B. Bruegge, A.H. Dutoit, R. Kobylinski, G. Teubner Transatlantic Project Courses in a University Environment 7th Asia-Pacific Software Engineering Conference, Singapore, December, 2000.

[3] O. Creighton, A.H. Dutoit, B. Bruegge, Supporting an Explicit Organizational Model in Global Software Engineering Projects International Workshop on Global Software Development, ICSE. Portland, Oregon, May 9, 2003.

[4] O. Creighton, M. Ott, B. Bruegge, "Software Cinema: Video-based Requirements Engineering", in Proc. of the 14th IEEE International Requirements Engineering Conference, Minneapolis/St. Paul, Minnesota, USA, September 11-15 2006, pp. 109-118.

[5] J.M. Carroll (ed.), "Scenario-Based Design: Envisioning Work and Technology in System Development", Wiley, New York, 1995.