EasyBook - Final Project Report

A booking system for the Västerås Flygmuseum

1 Background

1.1 Project description

Within the Distributed Software Development course at Mälardalen University in Sweden and Politecnico di Milano in Italy, we are to develop a new booking system for the Västerås Flygmuseum. This booking systems aims to provide an easy to use and intuitive user interface that allows to book and manage the museum's flight simulators. Also, it is going to unburden the staff, reduce bureaucratic effort and provide more comfort for the museum's customers and flight instructors.

The user will be able to select any available simulation time slot through an interface. An instructor is assigned by a coordinator through the booking system then. This instructor gives instructions to the customer while simulation and has access to all booking information. However, the user will also be able to request a time span outside the museum's opening hours. This booking request needs to be confirmed by the coordinator first. Only then it can be assigned to an instructor.

Staff members working at the cash point will also be able to book time slots for customers who are visiting or calling the museum. Apart from that, it will also be possible for them to assign an instructor right away.

1.2 Scope

Throughout the last months, several documents were produced while developing the museum's booking system. These documents describe the new system from different points of view. This document, however, gives a brief overview of the project including the team. It includes the project result, the project work and the final project experience. For particular interest, we refer to the following documents:

- **Requirement Definition** - The Requirement Definition document gathers all necessary functionalities in detail and describes each functionality from the museum's point of view.

- **Design Description** - Compared to the Requirement Definition, this document describes all functionalities more technically from the developer's point of view.

- **Framework Tutorial** - The Framework Tutorial gives a brief introduction to the Yii Framework which is used to develop the booking system.

- **Coding Standards** - The Coding Standard form a set of rules each developer has to follow in order to meet the required maintainability. Understanding the code at any later point will be much easier this way.

- **Permission Report** - The Permission Report describes ideas how to manage the interest of different users i.e. customer, instructor, etc. It was also used to communicate between the museum and the developers to further refine the functionalities and structure their accessibility.

- **Licensing Report** - This document introduces different licensing models and recommends an appropriate license based on the museum's needs.

- **Test Report** - The Test Report describes how test automation was performed during the implementation phase. However, it does not describe how reviews and walkthroughs were applied for any document listed above was carried out.

It is also planned to gather all product related documents and deliver a manual together with the software.

2 Project Result

We managed to develop a web-based booking system, that fits the needs of the Västerås Flygmuseum. We planned on having a user test under real conditions at the museum, but were not able to produce results, yet, due to timing problems.

The systems main features include the creation of bookings, management of bookings, a dynamic role system, the possibility to change almost all system parameters (including staff accounts, time slots, simulators, etc.) and an email notification system. The system is now in a state that enables the museum to actually use it. It provides simple access and overviews for customers and staff members and has been approved by our client.

2.1 Deliverables

These deliverables are all uploaded and can be found in the our project section on the website.

- **Minutes -** Throughout the project we have documented our meetings thoroughly. We have produced 23 minutes-of-meeting documents.

- **Weekly reports** - Every week we were to deliver a weekly report, summarizing our weeks work, our plannings for next week and our working hours. We have produced twelve weekly reports.

- **Presentations** - Once in a while it was necessary to present the current state of our project. The following presentations can be found on the website: Kick-off, requirements-and-design, status-report, alpha, beta and final presentation.

- **Documentation** - Other documents, which had to be delivered include: Requirements Definition, Project Plan, Design, Acceptance Test and Test Report.

2.2 Requirements satisfied

Since we have used Scrum we tracked the requirements in the form of user stories, which were altered during the process. Figure 7 will therefore show how many user stories we were able to accomplish up to this date and how many are left. It also shows our initial requirements and setup phase and how much progress we made during the single sprints.

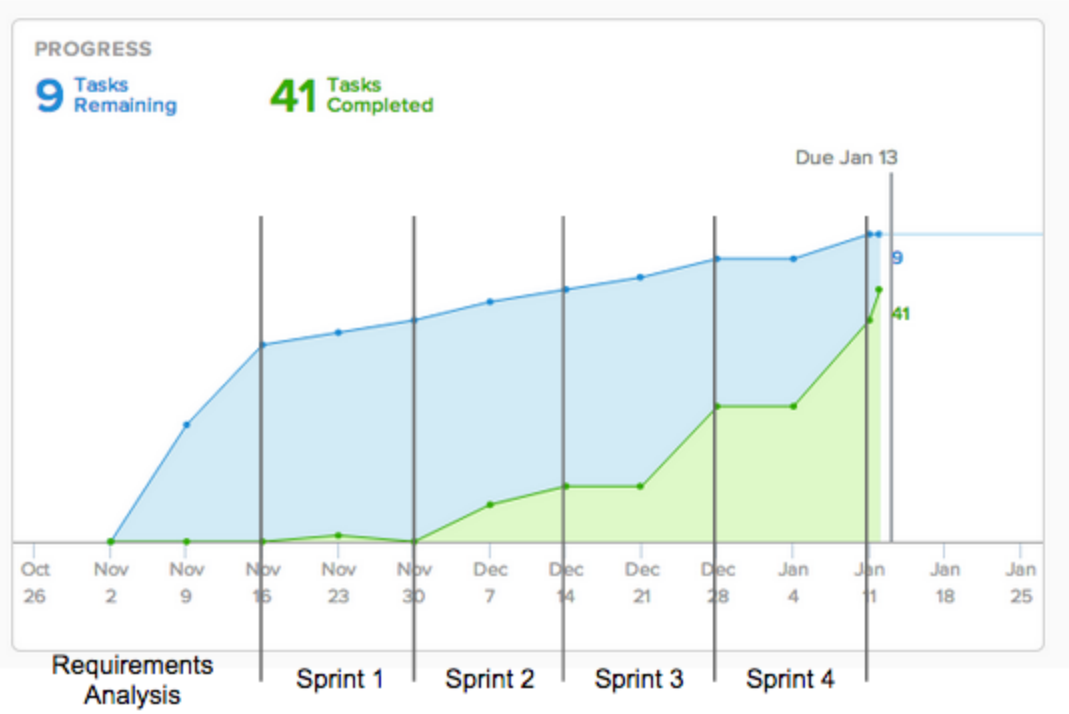

Figure 7: Backlog history

As visible in figure 7 we finished 41 user stories up to now, whereas nine user stories are not finished. Among these nine stories are six discarded ones and three, which are still worked on.

There is also another track of the requirements which has to be examined here. The requirements definition, that had to be fabricated at the beginning of the project. We updated this document in the course of the course, so it is not as far from what we have produced as would be expectable in a Scrum process. Among the implemented requirements are:

Requirements definition document	**Identified during development (selection)**
• Book simulator • Cancel booking • Log in • View booking • Assign booking to herself/himself • Assign booking • Confirm booking • Create booking • Delete booking	• Add restrictions (according to permissions) • Choose simulator • Select multiple slots for booking • Highlight bookings • Week view • Day view • Select specific dates to view • Separate between “during” and “outside opening hours slots

• Edit booking • Create account • Disable account • Edit account • Change price • Change time • Prolong booking (with alterations)	• Validate information • Create time slots • Select time spans • Send confirmation mail (coordinator and customer) • Block time slots • Manage simulators • Log out
Non-functional requirements	
1 Intuitive and simple interface (partially) 3 Accessibility 4 Maintainability and documentation 5 Future proof technologies 6 Localisation 7 Support for mobile devices 8 Licensed under BSD-3	

2.3 Under progress or missing - Future work

Even though the list of fulfilled requirements is quite long, we did not manage to finish all requirements that have been there from the start or were developed during the process.

- **Prolong booking** - It is not possible for an instructor to prolong a booking in the originally intended way. He can however alter bookings and time slots or create new bookings. It is therefore implemented indirectly.

- **Password recovery (email)** - It is not possible to recover a lost password via email at the moment, though we are still working to implement this feature. However it is possible to reset the password as admin.

- **Missing notifications** - We originally intended to send a lot more notifications. Whenever a booking was altered, whenever an instructor was assigned etc.. Currently the coordinator on duty will receive a notification if a new booking outside the opening hours is requested. Also the customer will always receive an email, which sums all his booking information up. We decided that these two notifications are enough, since the coordinator has to talk to the instructors anyway and the customers don't want to be spammed by the system. We will therefore not implement the missing notifications.

- **1 Intuitive and simple interface (partially)** - This non-functional requirement has been implemented partially at the moment. We originally said that "[o]ptions will only be shown if applicable for the current user" (Requirements

Definition v1.5). This partially true. There is a difference between logged in users and guest, but no further differentiation between staff members with different rights, e.g. enabling an instructor to see the link to the staff accounts page. Due to our dynamic permission system it is, however, difficult to hide unnecessary links, because permissions are not directly mappable on sites.

- **2 Automatic updates** - We planned to have the system automatically update any calendar view whenever the database was updated. However, we didn't put enough effort into this specific feature and it proved to be more complicated than thought. Our current solution: Set the browser tab on auto refresh.

Together with our customer we developed multiple ideas on how to further improve the system. Unfortunately we had to drop them, due to insufficient time. Additionally to the not implemented features above we thought about having:

- **Copy time slot structure** - Include a feature that enables to copy the whole time slot structure of one day, which would make it much less effort to open the museum on special occasions.

- **Frequent flyer** - Special options and discounts for returning customers. These may have a key to the museum and do not need confirmation or instructors.

- **Group bookings** - Group bookings were one of the earlier features we had to drop. It is possible to request group bookings via the comments field or by directly contacting the museum, however we thought about adding the option to the system. Groups can book more simulator time at once, get guided tours and have lunch at the museum.

- **Guided tour** - The option of booking a guided tour through the museum, additionally to the simulator flight.

3 Project Work

3.1 Organization and routines

Development was carried out using the iterative and incremental agile software development model Scrum. Compared to traditional software development approaches i.e. Waterfall-Model, V-Model, etc. it values customer collaboration and responding to change. This was of particular interest since the museum had not had an automated booking system before and requirements were rather unknown.

Because we were using Scrum, we organised biweekly Scrum Review, Scrum Retrospective and Scrum Planning meetings that typically lasted for four hours. This helped to

get the requirements right in the beginning and supported to further refine them later. Also, it gave the chance to tune the way we worked together:

- **Assign User Stories** - Right after the first Sprint ended, we realised that some User Stories got too little attention since they had not been assigned to someone. This might had originated from the missing Daily Scrum meetings. However, from the second Sprint we made sure that every User Story was assigned to a team member who kept track of it.

- **Comment User Stories** - After the second Sprint, we saw that communication through Skype was too unorganized because it was not linked to the User Story. For this particular reason, we started to use Asana's comment feature more extensively. This made communication more structural while still being transparent.

- **Estimate User Stories** - In the beginning, estimations were slightly off due to missing know-how in using the chosen development frameworks. Over time, estimations got better though.

Compared to the initial Project Plan, we did not hold a Daily Scrum meeting as formally known in Scrum. This was not possible because of colliding schedules and little progress due to other courses. Instead we used two Daily Scrum meetings per week to keep updated. Also, we planned the Sprint more agile and did not follow the initial time plan stated in the Project Plan. It was also not possible to meet the supervisors twice a week. We managed to meet them at least once a week and asked for feedback as often as necessary though. Additionally, we added GitHub's bug tracker to the list of our tools. This allowed us to separate User Stories that added value to the customer from bugs that needed to be fixed.

3.2 Effort and its distribution and working hours

First of all we can basically distinguish two phase for this project. The first one is requirement gathering phase, the design of the system and what technologies to use. This phase last from the initial week (W43) to week 46 (included). The second one is the development and testing phase that last from week 47 to 02, basically 8 weeks of developing and testing, but also further requirements gathering and altering.

Before showing the actual effort we invested in this project, we have to underline the fact that the requirements gathering also produced during 'development weeks' in order to analyze feedbacks from the customer. For this reason the hours assign to requirements plus design and to development plus testing are not the summation of the hours invested in the specific phase.

3.2.1 Communication

Since it is a distributed project one of the most important things when we speak about effort is how much we dedicate to the communication. This is something difficult to track due to the fact that we also communicate a lot with skype instant message.

In the following we have the percentage of comunication hours of the first part (72, in red) with respect to the actual requirements hours (311, in blu). We basically had 2 hours per week of communication in this phase.

It is possible to see that communication had taken quite a small part of the first phase. This is probably due to writing documentation that took quite a lot of hours.

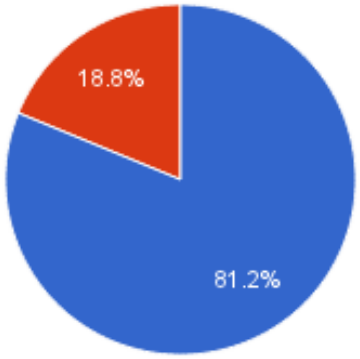

Figure 1: Communication during requirements

On development hours side, we basically increased the mean number of communication hours per week (3-4 hours per week due a more intensive SCRUM planning). On the other hand the time spent in other activities (learning and developing) also increased. For this reason we have almost the same ratio.

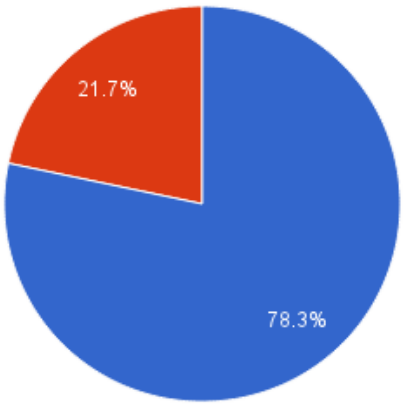

Figure 2: Communication during development

3.2.2 Effort in person day

In the following graph it is shown the effort invested in development (105 person-day) and in requirement (47 person-day) phase: basically our hours invested in development were the double of what we invested in the requirement, as it is expected.

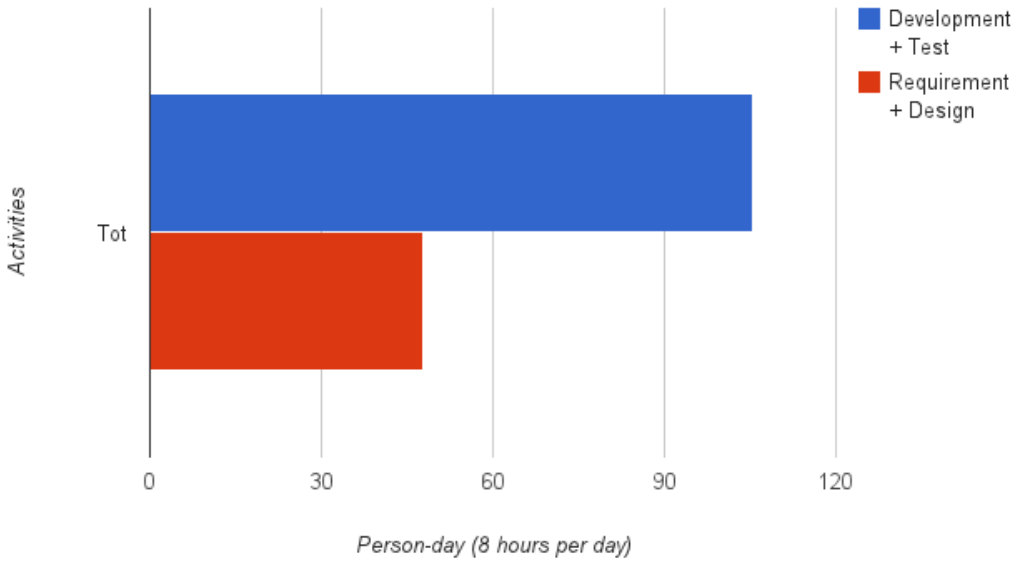

Figure 3: Working hours per member

3.2.3 Effort in hours

The following graphs shows the effort invested in this project for each of the team members, in terms of working hours for the two 'macro' activities (requirement plus design and development plus testing). The effort is distributed uniformly over the whole group. In addition note that the people most involved in required phase are Sebastian Kunze (our SCRUM

master) and Robert Engelmann, who were in continuous interaction with the customer also during the ‘development phase’, in order to always gather new requirements and feedback.

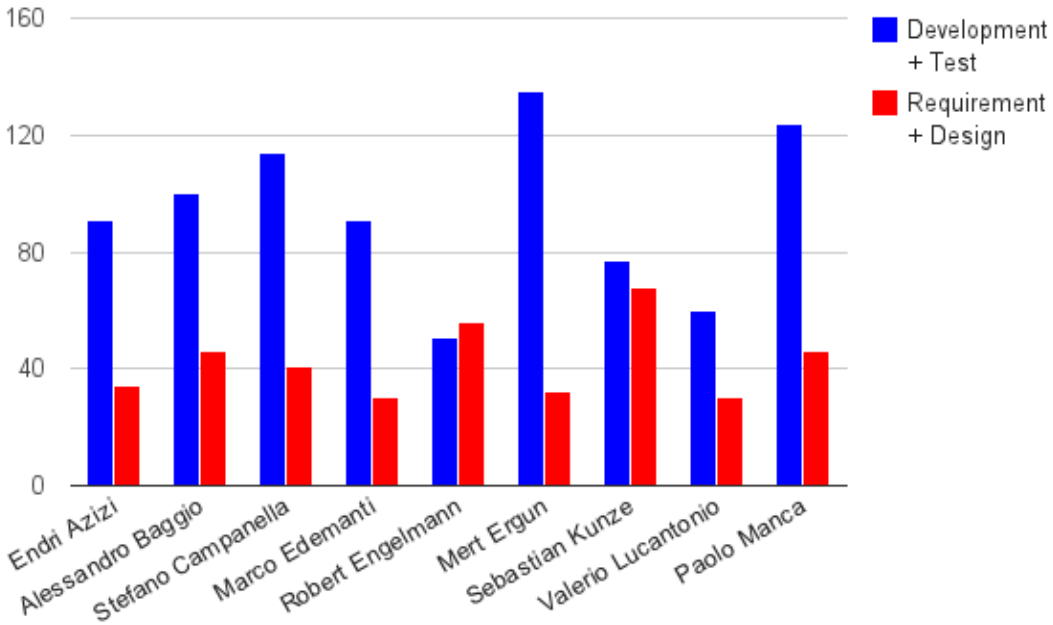

Figure 4: Working hours per member

Considering again the whole group, the trend of our working hours increased in our first period until we reached the Requirements and Design Document delivery (W46). Then the trend restart increasing again when we started the implementation for the alpha prototype (W48). After this starting point we kept an high level of working hours until Christmas holiday came (W52-W01), before came back and finalized the system (W02).

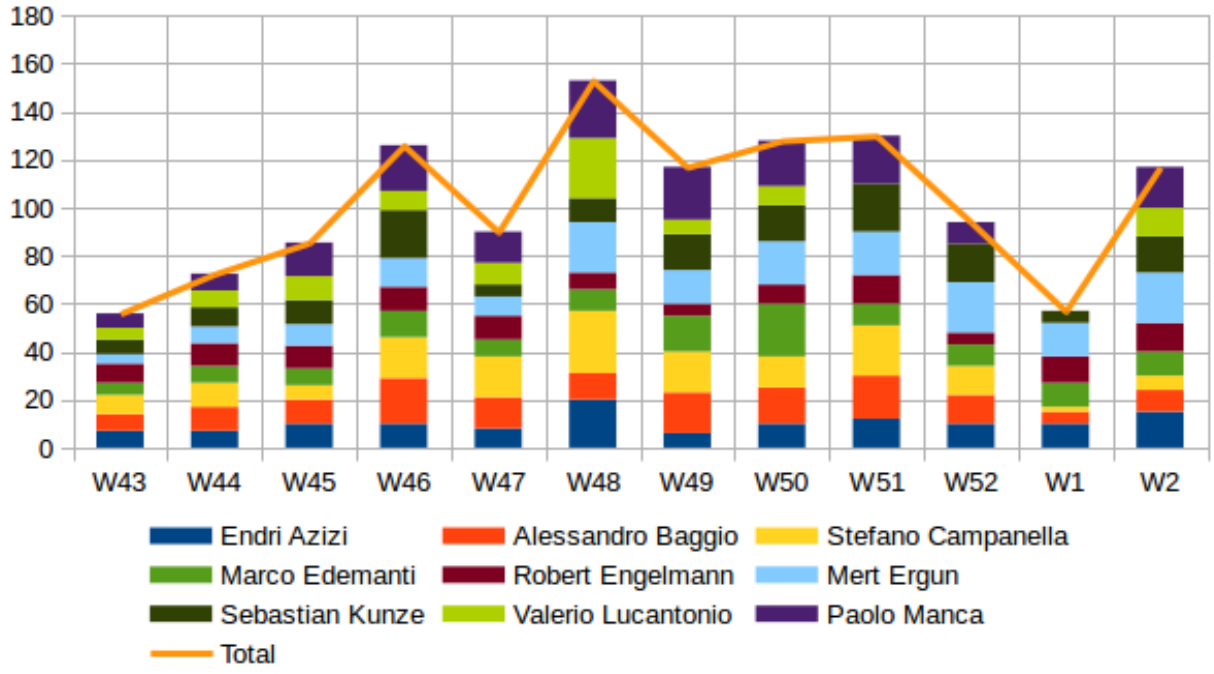

Figure 5: Overall working hours composed by team members

3.2.4 GitHub Commits

We report here the number of commits inside the team. As we already underlined during the presentation, the number of commits it is not the best metrics to measure the effort of a team member. Indeed there are some members who are used to upload their work with a lot of small pieces, and other people who preferred to upload big changes in only a commit.

We report the number of commits just to be thorough, please refer to 3.2.3 to understand the real effort.

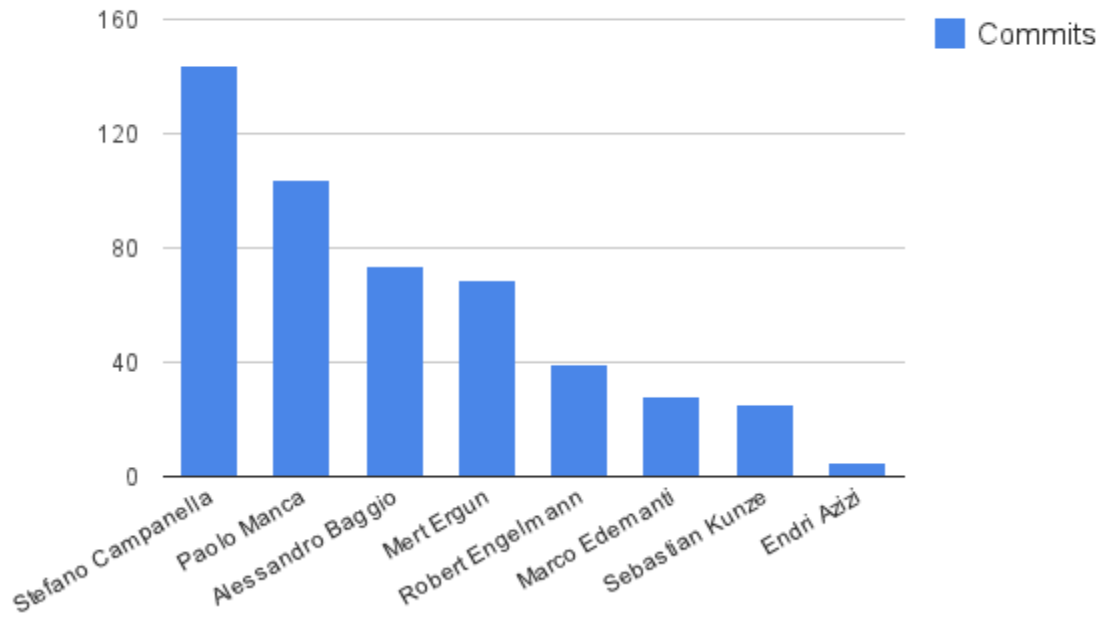

Figure 6: Commit numbers per members

4 Experience

Since you are nowadays connected to everyone over the internet anyway, we didn't feel much different in the beginning. We communicated with each other mainly over Skype, even if we were on the same campus, so everyone could always retrace the communication. We also got used to each others voices, though we didn't use video anymore after the second conference since it didn't provide real additional value. Anyway, we were missing an office christmas party and the fact of not seeing each other made verbal communication more different. None of us speaks English as his native language and not seeing each other didn't make verbal communication easier. Sometimes conversations were stretching because of missing words or due to failure in expressing oneself. Nevertheless, there were very rare cases in which parts of our team broke into their national language, thus excluding the other ones.

Overall it can be said, that Skype was a quite good compensation for the missing physical link, but we still couldn't reach the familiarity of a group that was actually meeting in the real world.

Some of us also had a previous experience in a distributed project. They stated, also with respect with the previous experience, this project was much more successful basically for two key points: communication, as we already said, and organization. Organizing a good Sprint was really important in order to succeed in the project, in particular with the decision of what user stories to do in the current sprint. To achieve a better organization the Scrum

retrospective was really useful in order to understand what we were doing wrong and how we could do better. For example at the end of the first sprint we realized that selecting user stories in order of priority wasn't the best solution, because we were not considering dependences between user stories, which slowed down our first sprint.

5 What we would like to have done different

If we had to start with the project all over again tomorrow, we would certainly change a few things. Though, we were quite satisfied with the overall experience we had during the project.

First of all, we would plan to have a dedicated learning phase before starting to code actually or to assign a few expert members of the team to "coach" less versed developers. We also realised it was nice to have physical groups to work in, even if it's only one other team member. The organisation of the documents we used for documentation and the deliverables has been organised from us into a folder structure, but it was still confusing. We would change that into a more intuitive system. Also, we had three communication channels in the end, though we decided in the beginning to use only Skype. We realised that different communication channels have different advantages and are suitable for different types of communication. So, we used Asana for user story related communications through the comments, GitHub for bugreports and Skype for all other communication.

Furthermore, we noticed that we didn't get very detailed feedback from our customer. We showed him the system after every sprint, we guided him through it and we gave him the opportunity to use it, but we did not encourage him enough to use it. Maybe we should have "forced" him to use the system himself and test it in order to produce more detailed feedback. The customer is an essential part of Scrum.