Project CS 2008
Course Report

Amir Monshi   Christian Axelsson   Erdem Aksu   Jing Yao
Johan Vikman   Karl Sundequist Blomdahl   Muneeb Anwar
Niklas Ronnblom   Xiaoming Yu   Xiaoyan Ni
Yunyun Zhu   Zou Hanzheng

January 16, 2009
Abstract

During the autumn of 2008 a group of 12 computer science students at Uppsala University taking the Project CS course set out in cooperation with Mobile Arts and Ericsson to develop a next generation instant messaging system for mobile networks on top of Ericsson’s IMS or "IP Multimedia Subsystem" platform. Not did they only do that but in the end they also implemented a framework for Erlang development on this platform and developed a J2ME client. This document contains information about the development process during this course.
# Contents

1 Project description 2
   1.1 Reviews ........................................ 2
   1.2 Resources .................................. 2

2 Project methodology and organization 3
   2.1 About Scrum .................................. 3
       2.1.1 Scrum terms ............................ 3
       2.1.2 Scrum roles ............................ 4
   2.2 How we have used Scrum .......................... 4

3 Problems 5
   3.1 Absence ........................................ 5
   3.2 Scrum ........................................... 6
   3.3 Inexperienced Estimation ......................... 6
   3.4 Lack of Requirements ............................. 6
   3.5 Scrum and Absence ............................... 6
   3.6 Tools ........................................... 6
   3.7 Communication with Ericsson ....................... 7
   3.8 Available documentation on IMS .................... 7

4 What we have learned 7
   4.1 Erlang .......................................... 7
   4.2 Scrum .......................................... 8
   4.3 Group dynamics and basic rules .................... 8
   4.4 Communication with external companies ............. 8
   4.5 Evaluation of Ericsson's IMS developer portal .. 8
   4.6 Evaluation of Ericsson's IMS client point of view .. 8
   4.7 Project Name ................................... 9
   4.8 Open Source .................................... 9

5 Suggestion for the next year project 9
   5.1 Better introduction to technologies ................. 9
   5.2 Erlang/OTP ..................................... 9
   5.3 Scrum .......................................... 9
   5.4 Hardware ...................................... 9
1 Project description

The course primary focus was to give the students experience of a larger scale project than what they get through the regular courses given at the University of Uppsala. This was done by giving the students a large project in cooperation with real companies, in this case Ericsson Research and Mobile Arts. Our project had two technical goals:

- Develop an Application Server for the IMS platform using the Erlang programming language.
- Develop an instant messaging service on top on that application server (using Erlang as well).

The IMS platform was supplied by Ericsson and the specifications for the instant messaging service was provided by Mobile Arts. A requirement from Mobile Arts worth noting was that the server side software should be written in Erlang while the existing IMS software from Ericsson was all written in Java. This way developers would get an alternative to the Java environment.

1.1 Reviews

During the project development time we’ve had three external reviews. These reviews consisted of presentations and demos. The first review was held at ITC in Uppsala and the invited people where mostly people from the IT department from Uppsala University. The presentation focused on presenting IMS and our project on a quite high level of abstraction. We also presented the project methodology Scrum and how we implemented it.

The second review was located in Kista at Ericsson’s headquarters. This time the audience were a lot more technical oriented and consisted mostly of people involved in the Ericsson IMS project in one way or another. Hence we didn’t need to present IMS at all and that left us with a lot more in depth presentation that could focus on technical details. We also left out the Scrum presentation but we mentioned that we used it.

The final review was held at ITC at Polacksbacken. It had a higher degree of abstraction than in the second review and we explained how we used Scrum. The presentation was open for everyone so the audience were more mixed than earlier.

1.2 Resources

We’ve worked together with a couple of people from Ericsson that has given us the resources needed for the IMS side of the development. These resources include documentation about their IMS implementation and a server where we could actually run our application server inside their IMS test network. Mobile Arts provided us with help solving design issues, gave us some Erlang/OTP tips and with the server hardware which we could run our internal environment on. From Citerus we had help from Henrik Hindbeck who introduced the Scrum project methodology for us and provided support about Scrum. Kreditor helped us by giving us lectures about Erlang/OTP in the beginning of the project. Uppsala University helped us with a project room and workstations.

On the software side we had a Mediawiki installation running for internal communication. We used Google Calendar for keeping track of absence. We used Mercurial for version control and most people used Emacs for editing Erlang code.
2 Project methodology and organization

We were told to use Scrum as our development methodology. The companies involved acted as the product buyers, the students acted as a made-up development company.

2.1 About Scrum

Scrum is an agile development method that is iterative and incremental. This means that after each working period, usually referred to as a "sprint" (more on this later), you have a new demoable product. Then the same process is repeated again and again (iteratively) until the product is as done as it needs to be. It should be noted that Scrum leaves a lot of things open for customization. The process is visualised in figure 1.

![Scrum process diagram]

Figure 1: The Scrum process

2.1.1 Scrum terms

Sprint
A predefined period of time when the developers work on the product. This is one increment in the Scrum development process. After a sprint is done a demoable product shall be ready. A typical sprint is two weeks but this can vary.

Product backlog
This is a high level document that describes the project requirements. It contains lists of features that are required and possibly a wish list of "would be nice to have"-features. This is a business-oriented document, it’s similar to requirement specifications.

Sprint Backlog
This is a more detailed document that contains information on how the team shall complete the upcoming sprint. Tasks are assigned some sort of time estimation from the team, usually hours or man days.

Sprint planning meeting
This is exactly what it sounds like – the meeting where the sprint is planned. The scrum master comes prepared with tasks picked out from the product backlog and the team together picks which of these tasks to put in to the sprint backlog. This meeting is very important and shall be as long as it needs to be.
Scrum meeting
These are the heart of Scrum. Each day, usually in the beginning, a short meeting, between 5-10 minutes, is held. In this meeting the following questions are discussed:

1. What have you done since yesterday?
2. What are you planning to do by today?
3. Do you have any problems or is there anything that prevents you from accomplishing your goal?

During the Scrum meeting the time estimation for all tasks that have people assigned are reevaluated and updated. This helps keeping the Scrum master up to date on project’s progression, and acts like a tool for noticing if things starts to go wrong and take proper action.

Burn down chart
A public visible chart that shows progress during the sprint. After each daily Scrum meeting the scrum master updates the burn down chart to reflect the progress made so far. The actual calculation is simply the sum of all remaining tasks. The chart makes it easy for all the team members to see the progress, which is a vital part of the Scrum methodology.

Sprint retrospective meeting
This meeting is held after each sprint and shouldn’t take longer than 4 hours. The point of this meeting is to reflect on the last sprint to make continuous improvement. The following two questions should be answered during the meeting:

- What went well during the sprint?
- What could be improved in the next sprint?
- What should we never do again?

2.1.2 Scrum roles

Product owner
This is person representing the voice of the customer. The product owner writes user stories, or tasks, prioritizes them and places them in the product backlog.

Scrum master
The Scrum masters job is to make sure that each sprint runs smooth and doesn’t go off track.

Scrum team
A Scrum team consists of the developers and the Scrum master. Team sizes are kept small, usually 5-9 people.

2.2 How we have used Scrum
When we did our initial research and found out that we had to write our own application server we split the large group in to two smaller scrum teams. The project leader, who had the role as product owner, took the role of scrum master for one team and another person was elected to be scrum master of the other team. We also had one person
from each team responsible for testing and integration (both between the two teams and within the team). The burn down chart was only used correctly in the end of the development period thanks to a misconception on how it was supposed to be used. Same goes for the sprint retrospective meetings. We’ve also noted that it was hard to fit in writing of presentations and reports into Scrum and hence we did actually skip using Scrum for this in the end.

Scrum also took a role of a bug tracker for us, allowing us to add larger bugs as unplanned tasks to the scrum board.

![Scrum board](image)

Figure 2: Our Scrum board

3 Problems

During the project we ran into some problems, some small and some large. In this section we try to summarize them.

3.1 Absence

A problem when doing projects with students involved is that there often are absence problems. This can occur because of other commitments like exams, assignments, lectures and so on. At the first period of the semester (the semesters are divided into two equal length periods) the project suffered because it was only half-time, scheduled hours from 13:00 to 17:00, in this hours many lectures, labs and workshops clashed with the project hours.

This was intended to be solved through making up of the lost hours at some other time, usually by working in lunch hours or weekends. This was, for the most part,
never done.

3.2 Scrum
We did have trouble applying Scrum in the beginning of the project. Most likely because we didn’t have enough prior experience and it was hard for many of us to adapt to the method wholeheartedly. Scrum relies in the group much more than other methods and the group has to take part in planning and discuss actively to make it work. If we would have had an experienced Scrum master for the first and maybe the second sprint it might have helped us out.

3.3 Inexperienced Estimation
At the beginning of each sprint, we usually estimated the working hours on each task. Then the schedule was made based on the costs of the tasks. However, very few of us had many experiences in industry so we always either overestimated or underestimated the working time, and thus messed the schedule up, to some extent.

3.4 Lack of Requirements
Here the requirement does not mean the very detailed and specific documentation. It refers to ”some description of a task”. We usually had only one a title of the task and discussed all the details at the scrum meeting or later during the implementation. But sometimes the details might be forgotten and other times there could be some conflicting interpretations of the requirement.

3.5 Scrum and Absence
The absence affected the methodology in a very bad way. Scrum relies on many meetings, which is extremely hard when not all members are present.

The two scrum questions
- What have you been doing?
- What might stop you from completing this?

Scrum relies in the group discussing everyone’s problems and knowing each other’s progress. Some time-estimates for tasks overshot by days and one (or two) with a week only because the members didn’t discuss, didn’t attend scrums and didn’t ask for help. This isn’t a fault with Scrum, this would be disastrous using any methodology, but Scrum is more easily affected.

3.6 Tools
Hardware
The hardware provided by Uppsala University were at best antic. This means that you could not run many programs simultaneously without experiencing sluggish behaviour. There were also some problems with hardware taking up too much physical place, several students were given old CRT screens that took up half of the given desk limiting the workspace, affecting working performance and morale.
Mercurial
The choice of a decentralized version control system where confusing for some who lacked experience with these kind of systems.

Linux
Lack of experience with POSIX based systems among some of the members caused delays in the efficiency peak.

Coffe Machine
The lack of freshly brewed java juice impacted work morale deeply. The machine was smashed to pieces one morning, the new machine was very noisy and slow.

3.7 Communication with Ericsson
The communication with Ericsson has caused some problems. Long communication delays have probably been the largest problem. It happened on multiple occasions that we had long stalls in implementation because we needed to wait for Ericsson to give us a response. Another problem was getting documentation about their IMS platform that we needed. For example our implementation of XCAP-support for group management in the AS was never completed because we couldn’t get our hands on the specifications that we needed from Ericsson.

3.8 Available documentation on IMS
Availability of good documentation on how the IMS framework works have been severely lacking and sometimes conflicting. The documentation available have been either standard documentation from the 3GPP group or slides used during various conventions to introduce IMS to the masses. Both of these are of different ends of the extreme, one way to complex to grasp without reading thousands of pages and the other one not giving much information. It would have been good if were had been able to find something comprehensive in between that explained the working of IMS. Much of our current knowledge is interpolated from reverse engineering of the different protocols. Ericssons didn’t provide us with a detailed specification of their IMS implementation, which caused problems when we migrated from OpenIMS to Ericssons platform. This caused resulted in a lot of wasted time trying to figure out the differencies between the two.

4 What we have learned
Throughout the project we have both learned some valuable lessons about tools, methodologies and working in a large group. In this section we present the conclusions that we’ve made.

4.1 Erlang
The amount of Erlang that we learned varies among the members within the group. But everyone has at least brushed the language and understands what the pros and cons are. Some have learned some more and can create advanced Erlang/OTP applications.
We have learned that Erlang is very suited for its main use, telecom applications. With the built-in behaviours you can speed up the implementation a lot. Building
complex finite state machines was easy and robust using the gen_fsm behaviour, fault tolerance using supervisors and gen_servers was also very helpful.

4.2 Scrum

Overall Scrum has helped us making a working product, especially in early stages. It helped us having something running before the first review which we didn’t anticipate. Also, from the project leaders point of view, it helped a lot when it came to keeping track on what people were doing and how the progression went.

Scrum proved to be a real light-weight software process with good enough potential to keep things running according to schedule. Unlike the classic software process requirements, Scrum has been extremely flexible to use. However, with this level of flexibility comes a high level of responsibility.

4.3 Group dynamics and basic rules

In a project where the members are put together with different experiences, where there are language barriers and cultural differences, its vital that you set up ground rules early on which everyone can rely on. If you are clear on the rules, print them on paper and everyone has taken part in the rule setup there is no excuses for not following them (the rules). The initial phase should contain open discussions on how to work (including coding standards), when to work and where to work.

Having rules is common and I think everyone was used to this when we started, but its hard to make everyone follow them when you have no leverage and there are no repercussions if you break them. One example of this is the coding convention we agreed on early on, everyone agreed verbally to follow the convention but we started lazily and when the sprint demo came up, we demoed and signed the sprint off. So even though the code didn’t follow the convention, we signed it off as ”finished”. So we violated both our own and Scrum’s ”rules”.

4.4 Communication with external companies

We’ve learned that communication with external companies can never be made with assumptions of fast response and that promises about resources and such can be changed without any real notice. One needs to continuously push some companies to get what you want. This seems to occur more often in big companies where there are protocols regarding security and administration.

4.5 Evaluation of Ericsson’s IMS developer portal

We didn’t have much use of the portal since its mainly focused on Java development. When we developed the clients we used the tutorials found on the portal and they were helpful, but sometimes lacking. The full potential of the SDS wasn’t realized until we paid a visit to Ericsson.

4.6 Evaluation of Ericsson’s IMS client point of view

At the beginning of the mobile client development, we assumed that everything would work fine right away. However, we quickly ran into problems:
• Connecting to OpenIMS from the client didn’t work the way it was supposed to. We figured out that there were a lot of differences between the two IMS platforms. We could do nothing but shifting our AS and MAS to the IMS platform on Ericsson.

• We also had problems adding our custom headers to the SIP messages. Luckily we got help from the IMS people who fixed the APIs for us.

4.7 Project Name
We learned that names are hard to invent, we actually had time set apart to decide a name, but we failed after lively discussions.

4.8 Open Source
We have learned that open source is often badly documented but excellent if you are willing to dig into the code.

5 Suggestion for the next year project
This section contains different subjects we think should be improved.

5.1 Better introduction to technologies
Lack of a good introduction to IMS and how it works have been a major stopping block at various points in this project. It would have been good if a proper lecture could have been arranged where the technology could have been explained.

5.2 Erlang/OTP
Erlang/OTP is pretty complex, especially the OTP part. We should have had some workshops in the beginning of the course to program in Erlang and use OTP.

5.3 Scrum
Having an experienced Scrum master available from the start of the project that can be on site during the first sprint or two. This should solve the initial problems that one run into when setting up and learning Scrum. This person would also be a focal point for the group who could steer the group in the right direction. There was a tendency to avoiding everything with Scrum and just working as if it were any other course. Many daily scrums were missed.

5.4 Hardware
As mentioned earlier having better hardware available would result in a better performance from all aspects of the project.