Abstract

Most mobile phones today offer the possibility to play advanced games, and the boundary between personal computers and mobile phones are fading. However, few games make use of the fact that mobile phones are, indeed, mobile. Project Green Fox presents a set of prototypes for location based mobile games that force the player to interact with the real world. By combining the game world with reality the players get a sense of being inside the game, enhancing the experience.
Contents

1 Introduction 5

2 Resources 6
2.1 Hardware 6
2.2 Software 6
2.3 Manpower 7
2.4 Literature 8
2.5 Environment 9

3 Project Methodology and Organization 11
3.1 Organization 11
3.2 Scrum 11
  3.2.1 Introduction to Scrum 11
  3.2.2 Daily Standup 11
  3.2.3 Sprint Planning 11
  3.2.4 Sprint review 12
  3.2.5 Scrum Board 12
  3.2.6 Burndown 13
3.3 Our Implementation of Scrum 14
  3.3.1 Product Backlog 14
  3.3.2 Sub-Groups 14
  3.3.3 Sprints 14
  3.3.4 Meetings 14
  3.3.5 Demoing 15
  3.3.6 Scrum Board 15
3.4 Methods and Tools 16
  3.4.1 Log Book 16
  3.4.2 Subversion 17
  3.4.3 Wiki 17
  3.4.4 Absence List 19
  3.4.5 English Whip 20
  3.4.6 IRC 20
  3.4.7 Phone Responsibility List 20
  3.4.8 Important List 21
  3.4.9 Need List 21
  3.4.10 PHPList 21
  3.4.11 Timeline 21
  3.4.12 Calendar 21
  3.4.13 Suggestion Box 21
3.5 Reviews 21
3.6 Problems 22

4 Review 23
4.1 Technical Review 23
  4.1.1 Phones 23
  4.1.2 Opinions on the Software 23
  4.1.3 Development Decisions 23
4.2 Cooperation 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1 Green hat People</td>
<td>26</td>
</tr>
<tr>
<td>4.2.2 Ericsson</td>
<td>26</td>
</tr>
<tr>
<td>4.3 IMS Evaluation</td>
<td>26</td>
</tr>
<tr>
<td>4.4 Recommendations to Next Year</td>
<td>27</td>
</tr>
<tr>
<td>4.4.1 Recommendations to Students</td>
<td>27</td>
</tr>
<tr>
<td>4.4.2 Recommendations to the Course Administrators</td>
<td>28</td>
</tr>
<tr>
<td><strong>5 Conclusion</strong></td>
<td><strong>29</strong></td>
</tr>
</tbody>
</table>
1 Introduction

The goal of Project Green Fox was to produce game concept ideas and develop prototypes of location based games for mobile phones. A location based game is a game that you control by doing something in the real world, using, for example, GPS, the phone's built-in camera, questions or some other method. Examples could be that you should race to a specified location on a map, take a photo of something green or answer a question which would be very hard to answer correctly if you're not in the right place.

Our other goal was to implement those games using the IP Multimedia Subsystem (IMS), which is explained in detail later in this paper.

The most important goal, from our perspective, was to learn how to work in a larger scale project than normal projects in university courses. In the project we’ve learned a lot of important things that are going to be invaluable when we start working.

We worked with an agile software development method that is called Scrum. Scrum gave us the opportunity to quickly develop small prototypes where we could assess if the game is going to be good enough. The flexibility that Scrum gave us helped save time and we were also able to change large parts of the game ideas without needing to scrap much of our already written code.

We didn’t have the ambition to create commercial products, our aim was to develop prototypes and see if they had the potential to become successful games or not. We decided that we would produce as many playable prototypes as possible, instead of developing fewer, more polished ones.

There are not that many released location based games on the market today, which makes our ideas very interesting, especially from a commercial perspective. In a few years time, GPS receivers will probably be included in most mobile phones, and the market for location based mobile phone games will increase.

The project consisted of twelve master students in computer science from Uppsala University, that took the Project CS course during the autumn of 2008. Green Fox was a cooperation between Uppsala University, Green hat People and Ericsson Research. Ericsson Research helped us with tutorials, support and provided us with an implementation of IMS. Green hat People helped us with our concepts, shared their expertise about testing and have given us feedback on our games.
2 Resources

2.1 Hardware

In total, 12 desktop computers and one server computer have been used to develop the software for both client and server side. The computers had bare minimum specifications in general, unfortunately the Celeron CPU on most of the computers was too slow for running the Eclipse IDE at resonable speeds, thus few team members had to use their own laptop to develop the applications.

The server has been running Linux Ubuntu Server 8.04 and served many services like web, version control and database server. We also experienced issues that our services weren’t accessible from outside the local network due to the university’s firewall. The first solution consisted of using SSH-tunnels where we accessed services on the server through the desktop computers, but in the longer run this wasn’t feasible so the final solution consisted of using unblocked ports for our servers, for example instead of port 80 for web server we used 8081.

One other important service we set up was a backup system. We didn’t have an extra backup server so we used two desktop computers which ran rsync servers instead. With this we were able to do regular backups from the server to the desktop computers.

For testing and evaluating the actual applications we used Sony Ericsson C702 mobile phones. They had some issues that sometimes stopped the GPS unit from functioning normally and sometimes, for no discernable reason (worked fine the next time you tried), the phone froze up and only a battery removal would be able to get it to restart.

2.2 Software

The following software has been used throughout the project:

Operating Systems:

- Ubuntu Linux 8.04
- Windows XP
- Archlinux
- Windows Vista
- MacOS X 10.5

Integrated Development Environments:

- Eclipse, primarily the version embedded in SDS.
- Eclipse Plugins:
  - EclipseME: developing J2ME MIDlets for mobile phones.
  - Subclipse: providing support for Subversion within the Eclipse IDE.
Cell Phone Emulators:

- Sun Wireless Toolkit 2.5.2
- Sony Ericsson Wireless Toolkit 2.5.

Server Operating System:
- Linux Ubuntu Server 8.04

Server Services:
- SVN: Subversion support for the team projects.
- MediaWiki: hosting all the information and the progress of the greenfox team.
- MySQL Database
- Tomcat application server
- PHPList

For more details about each software, please refer to product report preliminaries.

### 2.3 Manpower

The group consisted of a total of 12 members (initially 13) with 3 international students and 9 Swedish students.

Most of us we lacked experience in both mobile development and as well as working in larger groups. We tried to avoid limiting ourselves to developing components which we had more knowledge of. Instead we attempted to satisfy everyone by letting people choose what they wished to work with, keeping the door open for people to do different parts during the sprints. We believed that this would prevent us from being stuck in a specific part of the system and increase detailed knowledge about other people’s work in different parts. In reality this didn’t work out too well, most people choose to work on the same part throughout the whole project with minor variations.

Early on in the project we had a social event where we went out and ate at a nation (a student pub); the idea was that people should know more about each other hence feel more comfortable with each other. Even though this was pleasant it wasn’t really needed. By agreeing to only speak English in the project room, eating lunch together and playing foosball\(^1\) during the breaks, people became quite comfortable with each other and any discomfort regarding asking for help disappeared.

\(^1\)http://en.wikipedia.org/wiki/Foosball
2.4 Literature

Since the initial server programming language was expected to be Erlang, mostly Erlang programming books were requested for the project. However, due to the change of language to Java, with which most were familiar enough to work with, no general programming books were needed. In the beginning all literature needed were for setting the structure of the project and how to develop efficiently.

First, old reports from the course were read through so we could see what problems they had had, and how to avoid them. Since a lot of our members were fairly inexperienced at working in larger projects, the information received from previous groups was invaluable.

Afterwards we wanted to decide on coding and commenting standards. Thus we read through JavaDoc documentation and coding standards on the Java homepage. Having tested and useful standards was easier and more secure to use than to decide on our own for the sole purpose of this project.

Some members of the group, in particular those who were working on web interfaces for their servlets, read parts of the book Stripes . . . and Java web development is fun again². This was in order for us to develop intuitive and easy-to-use web pages with the Stripes library, a lightweight framework for web application development in Java.

As for Java language references and smaller web related information, if the internet did not supply enough information other project members were more than happy to help.

²http://www.pragprog.com/titles/fdstr/stripes
2.5 Environment

Figure 1: The Green Fox team in action

In Figure 1 you can see the room that we’ve spent most of our working time. As you can see in the picture above, this room has small windows not letting much light in and it’s also quite crowded when we were all here. Of course, having 12 people and computers in a small room like this would generate a lot of heat so the air would get quite bad. In addition, the ventilation system in this room has been broken for the entire project-period. Every once in approximately two hours the ventilation system started to make a lot of noise which forced us to shut it down. But since there were no permanent solution, we had to press that button every time it started to make noise. By shutting it down it could of course not do its job, so we had to open the window to get air. But during the cold winter, we couldn’t really open a window either so we had to do with the bad air. This implied that some of us got weary while others got headaches and were unable to be productive. Another thing that also ruined concentration for some was the quite high noise level which occurred due to the fact that we had no delimiters between our workstations. But this was not only negative, as it led to us having better communication and team work in between developers since we didn’t have to move to talk to others.

We also had some problems with the door access system in the beginning. Sometimes it did not work at all, and sometimes it worked just for some people, and sometimes it actually worked for everyone, including non-project members, which was a serious security issue. When we complained about it they “fixed” it by introducing other errors. It’s annoying that we had to complain about this simple matter 4-5 times. Some good things of note were the nice flat monitors
and the comfortable chairs which we believe increased productivity compared to alternatives. Note that only some of the monitors were good. There were some that were old CRT-monitors which can generates headaches, thus forcing some students to use their personal laptops instead.

The lunch room was a good resource. It supplied us with a meeting space with a white board, refrigerator, micro ovens, coffee machine and even couches to take a comfortable break in.
3 Project Methodology and Organization

3.1 Organization

We were twelve students in the group so we split up into three sub-projects which developed their own prototypes. This almost certainly made us more productive and effective than if we would have been one single group working on one prototype at a time.

3.2 Scrum

3.2.1 Introduction to Scrum

Scrum is an agile development method where the focus is to produce results as fast as possible. The process is very iterative, where every cycle is called a sprint. The length of a sprint can be from one week to a month and the idea is that the team should have something that can be demoed after every sprint. This is very important because it gives a good indication of how well the project is going.

There are three important roles in Scrum. First we have the project owner who represents the customer. It could be an employee of the customer or someone in the team acting as a proxy to the customer. The product owner decide what the group should do, create so called “stories” (general tasks) and then prioritize them. Then we have the ScrumMaster who’s task is to make sure the team follow Scrum and make sure the team meets the sprint goal. That means that the ScrumMaster doesn’t have to be the same thing as a project leader. Finally we have the team which consist of the rest of the group who are responsible for delivering a product.

3.2.2 Daily Standup

One core element in Scrum is the daily meetings. In agile development it is very important that everyone knows what the other team members are working on. In this way, they can very quickly find out if someone needs help with something. This makes the team work more efficiently as problems are discovered early.

3.2.3 Sprint Planning

In order to have a good sprint it is necessary to do sprint planning. In a sprint planning meeting, the Scrum team meet and decide which of the stories that they will be able to finish during the upcoming sprint. The team then divides the sprint into tasks based on the stories which shouldn’t take more time than two days to finish.

All tasks are prioritized by the team and their time requirements are estimated. The time can be estimated in hours, days, or just relative points. Everybody should participate when estimating the time it will take to finish each task.

One good way to make sure that all members really think and participate is to use poker planning cards. Poker planning cars are cards with different numbers representing time. Everyone in the team get a number of planning cards and then everyone have to take out the card they think estimate the time
needed for the task most accurately. When everyone has picked a card the cards are revealed and the discussion starts. If someone has an extreme value, that person have to explain why and motivate the choice. If the team members don’t agree with the estimated time, they will do the procedure all over until no extreme values exist.

After the planning, all tasks are converted to post-its and put up on the Scrum board.

### 3.2.4 Sprint review

At the end of each sprint, the team should have a sprint review where the members have the opportunity to give feedback and comments. One common way to do this is to hand out post-its to everyone in the team where they should write down at least one good thing and one bad thing. Later the post-its are collected and written down on a whiteboard. The “bad” items are sorted by importance and the team agrees on how to resolve the most important problems.

Often the sprint review is followed by a demo of what the team have accomplished so far. This is very important because it shows everyone involved what’s going on.

### 3.2.5 Scrum Board

![Figure 2: The Scrum board.](image)

In Figure 2 you can see the Scrum board, which is the heart of Scrum. All tasks are put up on the scrum board in form of post-its. It is used to get an overview
of how the sprint is going and what everybody in the team are working on. On
the Scrum board there are five sections:

**Not Checked Out** Here are all tasks that no one has started on.

**Checked Out** Tasks that someone is working on.

**Done** All tasks that are finished.

**Unplanned** Tasks that have to be done during the current sprint but weren’t
planned in the sprint planning meeting.

**Next** Tasks that should be completed in the next sprint but could be done in
current sprint if time allows.

### 3.2.6 Burndown

![Burndown Chart](image)

**Figure 3: The burndown chart.**

In Figure 3 you can see the burndown chart. The burndown is a graph showing
how much the Scrum team have worked during the current sprint and approxi-
mately how much time that remain on the tasks. It is updated by the ScrumMaster every morning after the stand-up meeting.
3.3 Our Implementation of Scrum

3.3.1 Product Backlog
We didn’t use a product backlog as we didn’t know what we were going to do. Our task was to invent game concept ideas and prototype them, which is hard to “plan” as such. We had a rough idea about what was to come in the next sprint but we couldn’t be sure until we had the next sprint planning meeting.

This was good in a sense, as it gave us free reigns to make what we wanted and the ability to put all our effort into making the creations we ourselves had decided upon. The negative part was that it did not allow us to use the full implementation of Scrum. In retrospect, we feel that we should probably have used a product backlog in any case, and just filled it in as we came up with stuff. At some sprint planning meetings there were problems since we had to invent much on the spot.

3.3.2 Sub-Groups
There is an official recommendation that a Scrum group should contain between four and seven members. Project Green Fox consisted of twelve students so we had to divide the project into subgroups. We thought about dividing the project into one client and one server Scrum team, but instead we decided to create three Scrum groups, each of them developing their own prototypes. Every sub-project had a Scrum leader that made sure the group followed the Scrum methodology. Starting with small groups seemed to be a good way as we weren’t used to work in Scrum.

The dividing of the group was generally met by approval. It allowed for each group to develop their prototypes without being needlessly dependent on the other groups. The downsides were that the groups became slightly imbalanced in number of members and experience. Fortunately, the groups were not isolated as such, and there was still a lot of cooperation and interaction between the groups.

3.3.3 Sprints
We started out with a one week sprint to try out how Scrum worked and because we wanted to try it without losing control. After a while we increased it to two weeks and even tried a three week sprint, but we fell back to two weeks. Our observation is that the more experience with Scrum you have, the longer sprint you can handle. For us, 14 days was usually the best sprint length considering the planning and workload, with a few minor exceptions.

3.3.4 Meetings
Sprint planning and sprint review meetings were performed quite informally with everyone sitting in the lunch room. After we got the whiteboard in there we could work quite well with it. In some meetings we used the conference room, but in the end it wasn’t used very much.

The sprint planning was slightly complicated due to our inexperience, but when we all got the hang of it it became a very useful tool for the development process. The Poker Planning Cards were used with varying frequency from
group to group in the beginning, but they proved to be very nice, if not necessary, for good planning.

The daily meetings were also a helpful friend, as it gave insight to what other people were doing and what they needed help with. It also worked as a work motivator, in that it made you make sure to have something to say to the other members. If the daily meetings had been kept at specified times early in the morning each day, it would also have ensured that people were on time and knew exactly what to do during the day.

3.3.5 Demoing

At the end of each sprint we tried to have something to demo (as Scrum dictates). It could be at a course review or just among our own groups where it was very informal, but the important part was that we had accomplished something.

3.3.6 Scrum Board

The Scrum board is the physical representation of the Scrum methodology. On this board, you can see how things are to be structured, what other people are doing and what needs to be done. Therefore it has been the favorite place to hold the daily meeting for most Scrum groups, since you can remind yourself what you have done and what you intend to do until the next meeting.

It was approved by the majority to have a real board instead of a Scrum emulation software, as it would take unnecessary time to learn a system that could so easily be implemented with post-it notes. Examining an on-screen Scrum board with your colleagues becomes uncomfortable when you have to cramp together; a large board allows for people to view it and discuss its contents with each other. The downside was the quality of the post-its as they used to fall down a bit too much.
3.4 Methods and Tools

Figure 4: The white board.

3.4.1 Log Book

In the beginning of the project we had a problem keeping track of the amount of time people spent and how much work they contributed with. The solution we came up with was the log book. At first we tried to use a web-based log book but people forgot to write in it, so we decided to use a physical log book. Every sub-project had one log book in which the members of that group wrote what they did that day and how long time it took. One other thing that was supposed to be filled in the log book was the estimated time left of the task and if any problems had occurred during the day. The log book have helped us a great deal when writing the final report.

- Good to see what people are contributing with.
- Have to reflect over what you have done.
- Should have contained more opinions of how things are going and what problems we have.
- Some don’t think that it was useful when writing the report.
- Good personal motivator, it made sure you had something to write in the log.
- Good to be able to check back what you did a couple of days before.
• Not that many details were written down in the log. There should have been more in order for it to be very useful.

3.4.2 Subversion

A version control system for a project is essential for keeping track of code and other project files. We used a version control system called subversion (SVN). The SVN server was run on our project server and the tool worked good.

3.4.3 Wiki

Figure 5: The Green Fox wiki.

A wiki is a tool that lets many people edit a set of web pages in any way they want to. The system keeps track of changes so they can be restored (like subversion for our code). The users can edit the pages in a coarse-grained parallel way (i.e. they can’t write in the same paragraph at the same time but different sections on a page is fine).

We used the wiki for many things including:

• Storing specific contact and authoring info like IMS SIP addresses.

• Putting up documentation of how things work and adding tutorials and examples to share with all the group.

• Writing rules and conventions which includes Java code convention and Comment convention.

• Managing and organizing each of the subgroups, defining task and responsibilities.

• Adding feedback and comments.
- Writing the final course and product reports. The final reports was then converted from wiki format to \LaTeX (a document preparation system and markup language) which in turn gave you this nicely formatted report.

People thought that the wiki was a really good tool to spread documents, guidelines and other type of briefing information to the other people in the group. This helped collaboration among everyone but also some thought that it wasn’t used enough.
3.4.4 Absence List

In addition to the log book we also introduced the absence list for keeping track of absence. The motivation for introducing the list was mainly to make people come in time, and if they didn’t, make them work over some other time. This method gives every individual some freedom to handle their own schedule efficiently and at the same time contribute enough to the project.

You had to write, at least one day ahead, if you were going to be absent.

Figure 6: The absence list

| Monday (22/9):        | Davus 13-15           |
|                       | Stephen 21-23 (Monday 13-15) |

| Tuesday (23/9):       | David 17-19 (Monday 13-15) |
|                       | Håkan 17-21 (Friday 13-17) |

| Wednesday (24/9):     |                         |

| Thursday (25/9):      |                         |

| Friday (19/9):        | Sam 13-17               |
|                       | Frederick 13-17         |
|                       | Håkan 13-17             |
In this way we could easily keep track of valid and invalid absence. The fact that we know who is going to be absent in advance, helped us to organize our time and the tasks the way that attendance would not affect other members and development.

In the absence list we also had one field where we collected the total absent time minus overtime which gave us the total amount of time everyone had to work in order to keep up with the work. In this case if anyone was late, their lateness would accumulate during time and it becomes a motivator to make everyone come on time or at least do their best to be on time.

In general we believe that the absence board idea worked well and the execution was quite helpful to keep track of absence and keep people from being absent too much.

3.4.5 English Whip

Because of the fact that not everybody in the team spoke Swedish we had to make sure everybody understood each other. We decided quite early that the default language in the project room was English. The reason for that was that it’s very easy for someone to feel outside if he doesn’t understand what’s being said. We tried to tell people that talked non-english but it only worked temporary. We had kind of a hard time to make people remember to talk English so we introduced the “English whip”. It all started as a pretty bad joke but people started using it and it gave a positive effect. As soon as someone started talking in the wrong language we started saying “English! English! . . .” until he started to talk English again.

Opinions from the members about this whip is that it was a shame it was needed but it contributed to a better work atmosphere in the end, albeit some people got irritated and disturbed of the English whip.

3.4.6 IRC

We had an IRC (Internet Relay Chat) channel where we all could discuss project related matters without having to disturb the others. This proved to be very useful when people were sick and/or worked at home. The channel was password protected to keep unwanted people out.

3.4.7 Phone Responsibility List

We borrowed four mobile phones from Ericsson Research and needed a system for keeping track of them. The phones could easily get lost if they were lying around in the project room without anybody being responsible for them. That’s why we introduced the “Phone responsibility list”. The list had four slots, one for each mobile phone, and as soon as someone took one phone out of the safe he had to write his name in the corresponding slot. One other advantage with the list was that if you needed a certain phone you could easily find out where it was by simply looking at the list. Of course it happened that someone forgot to write their name on the list (or remove it) sometimes but it was much better than not using the list at all.
3.4.8 Important List
On one white board we had a dedicated area called “Important list”. In that area we put up important bullets of tasks to do and things to think about. The important list gave us a good idea of what was going on at the moment.

3.4.9 Need List
The need list was a part of the board where we wrote what we needed at the moment, for example more white board pens or more post-it’s. The list was a good idea and it worked okay. We had some issue that some things took awful a lot of time until we actually got them, which stalled us a bit some times. At first we hoped that our assistant could help us more with this list, which unfortunately was not the case.

3.4.10 PHPList
PHPList is a newsletter manager. We used it to define mailing list for every subgroup. The members of each subgroups could subscribe to any desired mailing list and then allowed to send mail to all the subscribed member of that lists. We didn’t use PHPList as much as we initially thought. Instead we simply wrote the mail address of all people we wanted to send the mail in the subject.

3.4.11 Timeline
The timeline was a concept we tested to get an overview of all important dates and how much time we had left. The timeline was divided into weeks, crossing over them as we went forward. In that way we could easily see where we were in the timeline.

3.4.12 Calendar
On our web server we had a web calendar using the software VCalendar. Our idea was that everybody had to type in to the calendar when they were sick or going to be absent. This didn’t work and instead we used the absence list for that.

3.4.13 Suggestion Box
This concept was introduced after the first course evaluation. Our thought was that anyone could put an anonymous note in the suggestion box with a comment or suggestion. The suggestion box was very seldom used and people gave suggestions and comments on meetings and course evaluations instead.

3.5 Reviews
Most project member appreciated the reviews, that they were good milestones and helped keep the deadline. It had a positive impact in the way that we had to make our work presentable and then we felt that we had accomplished something. Some liked that they got to know what the other project (IMS Message Office) was doing but one thought it was unnecessary to listen to that. One thought that the 1st review provided to little feedback whereas another thought
that the 2nd review was too late so the feedback couldn’t be implemented. Some though it actually was boring and there where an opinion against the early rehearsal (1 week before) for 2nd review as we had not enough content to rehearse

3.6 Problems

After the first course evaluation we got aware that people got disturbed of people running around and talking loud. Our solution was to switch seats minimizing the summed average distance between members in the same subgroup.

When working in a product it’s always a problem when people don’t arrive in time. On the absence list we wrote people when they arrived at least 15 minutes late. That turned out to be somewhat effective—people arrived mostly less than 15 minutes late. So it was changed so the start of late time became exactly at quarter past full hour. Now people started to come a bit earlier but nearly each day someone was late. It affected the scrum groups that they couldn’t have their daily stand up meeting. Usually people knew something that they needed to continue working on but it was always a feeling that the working day hadn’t really started.

We lacked a product owner but that wasn’t so impeding because we had got free hands from Green hat People to develop what we thought was interesting, which made the product owner’s executive role unnecessary regarding the product. When there was a disagreement about something we could discuss and come up with a compromise that was good for everyone.

Another problem was that estimating the time it takes to finish a task turned out to be very difficult. Oftentimes it is very hard to know the undertakings that a task imposes, and even after being very pessimistic in setting the estimated time, it was not rare for it to take much longer than expected. One of the reasons for this is that it is hard to take unexpected events into consideration, like bugs or dependencies that become apparent once development has started.

Considering the communication between each team it can be concluded that although it worked out well but we could still have more collaboration and help each other in designing and learning how each module and code works. Also occasionally prioritizing important features and aspects of our project could be done in an easier and more efficient way.
4 Review

4.1 Technical Review

4.1.1 Phones

We received four phones from Ericsson to use for testing, though they arrived a little late so we had to work more on ideas before we started deploying our first test applications. The first issue with the received phones was that they didn’t have any SIM-cards which meant that we had to wait a couple of days before we could even try to send a simple “hello world” message through the IMS network.

At the beginning of the project the IMS framework needed user confirmation for sending every message over internet on the phones which was very troublesome for evaluating and testing. Later on the newer IMS framework removed the need to confirm every message received or sent, but it still needed confirmation for sending the first message. Ericsson could provide us with certificates to overcome all this confirmation and make the process easier.

Most of the time the phones worked fine but in some cases we noticed some bugs in the phones such as a phone restarted itself and reported an invalid SIM-card while the application tried to send message over IMS network. Also from time to time the GPS stopped functioning and unfortunately it happened that the phone completely hung itself were the only solution was to hard reboot the phone by reinserting the battery. We think that there were issues with this particular model.

4.1.2 Opinions on the Software

• Eclipse

Some members disliked this IDE because of bugs while some disliked working with it because of the poor performance of the computers we had, most of them didn’t even live up to the requirements set by Ericsson to run the IDE.

• WTK

There were two WTKs available, one from SUN and one from SonyEricsson. Some members had difficulties running the SonyEricsson emulator due to conflicts with Windows XP. One more point worth mentioning is that the SUN’s WTK works in Linux while the SonyEricssons lacks that support.

• Subversion

This was one of the few plug-ins to Eclipse that worked without any big issues

4.1.3 Development Decisions

• Erlang as a server language

We attended some lectures about the Erlang programming language (primarily given to the IMS Message Office project as they were required to use Erlang) and got very excited and some of us started studying it. As
Erlang is renounced for being very parallel, supporting hot-swap of code and scales well, we were all excited about it. We heard that the other project would be given a lot that was needed to complete an application server in Erlang. But soon after that it turned out that the software the other project needed didn’t exist and they had to implement it on their own (like SIP stack, Application Server). We made the choice to go with Java and the Sailfin application server because otherwise we had to wait a very long time to get a server to run on. Also there was the problem that they (IMS Message Office) weren’t allowed to deploy it live in the IMS cloud until the end. The application servers of MJCF was in the cloud and then mobile phones could access the services, and without that only the phone emulators would have been possible to use for testing.

- Running a local server
Right from the start of the project we weren’t sure if any of the partner companies would supply us with a server machine. But we decided early on to host our own local server to supply us with various services, such as SVN, MySQL, Tomcat, MediaWiki etc. This turned out to be a good decision since we had total control over the software running on the machine. Unfortunately the hardware in the machine was pretty poor, but sufficient for our needs at least. One downside of this was that we didn’t have any proper backup (just a script copying to another computer in the project room) or redundancy of the hard drives. We also couldn’t run services such as HTTP and SSH on standard ports since they are blocked by the university gateway. We’re lucky that the hardware hasn’t failed us during this period, we’ve actually had a 100% uptime on the server – which is quite a lot better than what the MJCF server has managed.

- Code conventions
We agreed on using SUN’s Java code conventions\(^3\) for all the code in our project. We found that it’s important to keep the code as consistent as possible throughout all of the project, any other way would probably create a lot of annoyance.

- Live vs. Simulated environment
The SDS provided by Ericsson contained tools which could simulate the whole IMS network, where the client applications running on the emulators could interact with the servlets running on a local Sailfin server. In the beginning of the course we set up one such environment with the help of an Ericsson employee but due to lack of documentation this process couldn’t be repeated on a different desktop computer. At the same time we were able to deploy and run our servlets on MJCF development portal and interact with them through our emulators as well as real phones so we had chosen to disregard the simulated environment. If we had chosen to set up the simulated environment and develop our applications (until we needed to test it on a live phone) we could have spared some time regarding the issues with MJCF; issues such as bad uptime, continual reboots of the MJCF application server and so on.

\(^3\)http://java.sun.com/docs/codeconv/
• Hibernate vs. JDBC

As mentioned in the product report Hibernate is an object-relational mapping library. Hibernate gives us the opportunity to regard stored information in the relational database as objects. This library did cause problems for us, where the library behaved differently from what we specified. Most of these issues was due to the magnitude of the library, as we only wanted to use a small part of it. We also didn’t want to spend too much time studying the library, which lead to us not configuring it properly from the start. This in turn resulted in us having to spend time understanding how to express ourselves in the different configuration files, which were quite a few, and unfortunately the documentation was poorly written. Though even in hindsight our decision remains, it is a requirement to use such a tool if anyone wants to keep a good quality of source code. If we didn’t use Hibernate then we would have defined and implemented a specific data model layer which would use an object-relational mapping similar to the one used by Hibernate, which would in a sense be reinventing the wheel just to avoid reading bad documentation.

• PageMessages vs. MSRP

When we started to implement the common communication libraries, we used PageMessages as the method of communication, mostly since it seemed easier to start with, and all the basic tutorials were written with PageMessages in mind. When we realised that we would need to send pictures over IMS we tried to investigate the pros and cons of changing to MSRP instead. We did not get any straight answers from our contacts at Ericsson, they basically said that either method would work for sending large data packets, though MSRP might have a slight edge, but that it wasn’t fully functional yet. Since we were already working with PageMessages and they did not require a constant connection, we decided to stick to them.

After a couple of weeks of troubleshooting our implementation we found out that the PageMessages could silently fail if they were bigger than 400 bytes. This led to us having to change our framework to MSRP since we could not use PageMessages for non-text transfers. Luckily the transition went relatively smooth, and we gained a lot of transfer speed in the process. The Agent SQ and Tourist Guide prototypes were the only ones that changed to the MSRP version, since the other ones did not need image transfers, and MSRP tended to be slightly more prone to dropped connections due to using sessions instead of asynchronous communication.

In hindsight, it taught us a lot to use PageMessages in the initial version so it wasn’t all bad, and we still ended up using it for two of the prototypes. We could have saved about two weeks of frustrating troubleshooting though, which would have been much better spent elsewhere.

• Common framework

In the beginning of the project we experimented with simple servlet and midlet applications to learn about IMS, these applications became more complex and actually performing tasks for the games. We soon agreed on rewriting these simple applications and evolving them to a more general
platform common for all different subgroups. The common framework is explained in the product report under system description. This common framework lies as base for every group, for instance if there are any changes to how a message is sent then it’s not needed to change any other source code than just the specific communication class in the common framework and just updating the subgroups’ repositories. This gave us opportunity to separate tasks between groups; some could focus on explicit game logic and protocol as other focused on the issues regarding IMS interaction.

4.2 Cooperation

4.2.1 Green hat People
Green Hat People seemed very happy and impressed with what we were developing. At first we didn’t know that their intention were to not tell us so much about what they wanted and we felt that their participation was too small.

Later (around mid course evaluation 1) it was revealed that they had done it with intention so we wouldn’t be so affected by their ideas when we came up with our prototypes, but some of us would still have preferred some more involvement from their part.

Later on we finally got the chance to play one of their current game, *Kreativa Uppdraget* (*The Creative Mission*), which we invited the members from the other project (IMS Message Office) to join and we played at Polacksbacken. After that we had a better understanding of the game and how people could think it can be fun.

In December we visited Green hat People at their office in Stockholm and demoed the games live for them. They got very excited when playing albeit it was very cold in the air. We discussed various things about the prototypes and looked at their current administration system for watching the events that players trigger when they are playing on the phones.

4.2.2 Ericsson
Ericsson were, in the beginning of the project, very helpful. They even sent a person to help set up our IDE and software as the tutorials didn’t work. But later, the IMS project on their side seemed a lot less prioritized, as questions on the forum remained unanswered for days at times, and the fact that they did not allow us to have direct communication with them in any way which was slightly disturbing as well.

Another surprising and slightly annoying thing was the cheapness of Ericsson. They did supply cellphones for us, but not with prepaid SIM-cards. Not even during the invitation to hold our review at their head quarters in Kista did they see it fit to offer us some kind of lunch. Since our bug findings in their code must have been invaluable to them one would have expected them to repay the kindness in either support time or economically, but that was evidently not going to happen. We expected more enthusiasm and appreciation from them.

4.3 IMS Evaluation
Making a standard framework for IP connections with mobiles, i.e. IMS, is a great idea. When developing mobile applications you do not want to spend too
much time implementing the connection establishments. If the standards could be implemented by all phone companies and every phone model, even better. The same goes for any server application creation; the connection development parts should be kept to a minimum, to allow more time to be spent on program logic.

Unfortunately, a good idea was all it was. As an implementation, it left much to desire. First of all, the documentation was severely lacking in all aspects. Getting started tutorial guides were not working, not even when followed by the letter. Also there were some information that was missing, like for example data size restrictions on messages and such. Even the IMS descriptive papers were so full of buzz-words that it was hard to even grasp what its purpose was.

The system was not really ready for real life usage either. MJCF has a lot of bugs and errors that we’ve continued to stumble upon through the entire project. It took quite some time to get MJCF started and working sufficiently well, but even then it was incredibly slow and still as fragile as a card house. Since these things were pretty much out of our hands to fix, the project moved slowly as testing our applications became very difficult and time consuming. For the sake of the project, using plain socket communication instead of IMS communication may have increased the production speed and thereby improved the end results.

So in conclusion, more time and work needs to be spent on developing the Ericsson IMS. Increased speed and stability are of immense importance for any company who want to use it. For anyone to actually be able to use it at all better documentation and tutorials that really work is needed as well. Also adding functionality for sending page messages larger than 400 bytes is a must. Since one of IMS’s selling points is to be able to send all kinds of media it should be able to handle way bigger messages without problems.

4.4 Recommendations to Next Year

4.4.1 Recommendations to Students

Judging from our experience, these are a few rules that future students might want to consider when taking this course.

- **Think everything through before you start.**
  A good planning make the work easier in the development process.

- **Get people to actually be in the project room every day.**
  Don’t allow people to work at home except in special cases. It really helps if ALL team members are present in the project room, as they can be of assistance to the other members. This course is supposed to simulate a real job environment, after all. Have a time reporting system to regulate this.

- **Have fun!**
  Enjoy doing the project and being with your team mates. Try to see problems more like challenges, waiting for you to use your exquisite brain to solve them.
• Set the rules early and follow them!
   Having some strict rules gives a more professional feeling and hopefully prevents people from slacking off. This includes regulating attendance hours and when breaks can be taken. Make sure the rules are followed.

• If you have non-Swedish speaking students make sure to only talk English.
   No exceptions! If you don’t, people who don’t understand the spoken language will feel outside the group.

• Have team building sessions and get to know each other.
   We had only one team building session, which was sufficient, but a few more wouldn’t have hurt. When people know each other better it gets easier for them to work with each other and dare to ask for help, which in the end is beneficial for the entire group.

• If using Scrum, try to implement as many of its features as possible.
   Learning the Scrum methodology could be a very valuable asset when entering real life. And also useful during the project as well when you’ve gotten the hang of it. If it is much to swallow in one bite, implement it step by step instead.

4.4.2 Recommendations to the Course Administrators

• Be present at least during the first months on team meetings.
   Even if you as coordinator is not very experienced in projects of this kind, neither are the students and any kind of input you have may be useful.

• The project room was not that good.
   The ventilation started screaming every now and then so we had to turn it off every two hours. When the air is stale, working becomes much of a chore and the productivity decreases. Maybe a room that is a bit more spacious and with bigger windows and better view would improve the working conditions for the people who are expected to spend entire work days in it.

• Make the course full time.
   Hard to keep the project running when people have to go away to lectures, especially when using the Scrum methodology that requires lots of human interaction at times.

• Try to announce the project subjects before course application.
   It would be appreciated if the subject of the project was announced before you have to apply for it. In that way people can get a hint if they think the subject is interesting or not and choose thereafter.
5 Conclusion

The overall feeling about the project in the group is positive. The course has, through its interaction with companies such as Ericsson Research and Green hat People, given us the opportunity to experience what the “real” world may be like after studies. Not only have we been able to use the knowledge gained through years of studies, but we have also gotten practice in gaining information through other means than being taught by a lecturer.

We have felt the pressure of having expectations upon us, the frustration of well designed plans going terribly wrong and the joy of completing tasks and assignments that actually feel worthwhile and useful.

Unlike most educational events this course has prepared us a bit for the conditions we are likely to experience after graduation. We now have a bit of experience in how to handle work related stress that arises from obligations to be on time, having results by the end of our sprints and so forth. Any and all mistakes made during the project are now another experience to add to our respective knowledge databases.

In summary, it was a good course, and we all had a lot of fun together.