Course report
GSM Call Service

1DT054 Project CS

Ebby Wiselyn Jeyapaul
Egemen Taskin
Erik Grafström
Fartash M. Nejad
Fredrik Pasanen
Max Morén
Mohammadreza Taghilu
Moritz Rogalli
Praveenkumar Bhadrapur

February 16, 2012

Abstract

The course report describes how the team worked during Project CS to develop a *GSM call service*. In this report we describe how we organized our work, we document our progress, we describe and explain the decisions we took and the problems we overcame. We concentrate on how we cooperated, the tools that helped us the most and the lessons we learned during the course to help both future students and teaching staff to improve on future iterations and enhance the experience and result of the course.

Glossary

API	Application Programming Interface
BC	Bearer Capability
BNT	B-Number Type
BSC	Base Station Controller
BSS	Base Station Subsystem
BSSMAP	BSS Management Application sub-Part
BTS	Base Transceiver Station
CC	Connection Confirm
CM	Connection Management
CR	Connection Request
CIC	Circuit Identification Code
CI	Cell Identifier
CS	Connection Service
DNS	Domain Name Server
DTAP	Direct Transfer Application Part
DT1	Data form 1
DT2	Data form 2
G-MSC	Gateway MSC
GUI	Graphical User Interface
GSM	Global System for Mobile Communication
HLR	Home Location Register
IMSI	International Mobile Subscriber Identity
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
IT	Inactivity Test/Timer
KVM	Kernel-based Virtual Machine
LAI	Location Area Identifier
MAP	Mobile Application Part
MA	Mobile Arts
ME	Mobile Equipment
MGC	Media Gateway controller
MGCP	Media Gateway Control Protocol
MG	Media Gateway
MO	Mobile Originating
MM	Mobility Management
MSC	Mobile-service Switching Centre
MSISDN	Mobile Subscriber Integrated Services Digital Network Number
MS	Mobile Station

MT	Mobile Terminating
MUS	Mobile User Service
NP	Numbering Plan
OBA	Origin for B-Number Analysis
OTP	Open Telecom Platform
PC	Point Codes
PLMN	Public Land Mobile Network
PS	Packet Switching
PSTN	Public Switch Telephone Network
RLC	Release Complete
RLSD	Released
RR	Radio Resource
RTP	Real Time Protocol
RTP-TA	RTP Termination Agent
SCCP	Signalling Connection Control Part
SS7	Signalling System
SIM	Subscriber Identity Module
TCP	Transmission Control Protocol
T-MSC	Terminating MSC
TMSI	Temporary Mobile Subscriber Identity
TM	Transit Module
V-MSC	Visiting MSC
VLR	Visitor Location Register

Contents

Chapter 1

Introduction

The main purpose of this project was to develop a basic ***GSM Call Service*** [19]. This document describes the course, for a detailed system description see the product report [28].

The purpose of the GSM call service is to serve as a testing framework for mobile phone network services. The project took place as part of a 30 credit full-time course titled Project Computer Science, covering one full semester. The project was proposed by the company ***Mobile Arts*** which provided the project specifications.

The team consisted of nine Computer Science students from Uppsala University, who developed the product using ***Erlang/OTP*** [17] and a ***Scrum*** [21, 23] process.
The two main goals of the course, according to the course description, are to "give insights into how a big project is run (from planning to realization), how to construct a complex distributed system" [27]. The document describes in detail the process and the methodology used and which the goals were realized.

Chapter 2

Resources

2.1 Hardware

Every member was provided with a desktop PC. We also got two workstations for server systems and a projector for common use from the university. Mobile Arts provided a ip.access nanoBTS [26] picocell for testing.

2.1.1 Servers

One was used to run services and the other for testing. Both servers ran Ubuntu server edition and we used KVM for virtualization [4]. Virtualization was used to make systems and services hardware-independent, improve flexibility and to reduce the number of physical servers.

Service Server

The Server *services.gsm* was used for supporting services for collaboration and connectivity. The following servers were used:

dev.gsm Development-related functionality. The server ran Redmine [7] for project management and a git repository for version control.

remote.gsm Connectivity services. The purpose of this server was to enable Mobile Arts employees to connect via ssh for debugging and support. We applied for an exception in the IT department's firewall rules to make the server accessible from Mobile Arts. The server also ran a DNS proxy to provide internal DNS names for our servers.

Testing Server

The Server *testing.gsm* ran all virtual machines related to testing. It consisted of several machines, with a template for new testing systems that could be cloned, started and used for testing in a matter of minutes.

bsc.gsm Originally for running openBSC [20] only, this system became our main testing system. It mostly ran all of the components of our final system.

hlr.gsm/smsc.gsm Two Red Hat Linux systems we set up in cooperation with Mobile Arts for a Tieto SS7 stack. However, since the HLR functionality was stubbed and the SMS functionality not integrated, the SS7 stack was never used.

playground.gsm A place to try out things like new versions of Erlang, etc. An Ubuntu-system that could be reset quickly as needed.

msc.gsm, others Servers that were originally created for the different components of the GSM Call Service. They were never used since working on only *bsc.gsm* was sufficient.

2.1.2 Backup

The contents of important directories on every machine were copied to the other physical server by a cronjob three times a day. The current backup was copied as a daily backup once a day, and the daily backup was copied to a weekly backup every weekend. This produced a considerable amount of overhead and used several times the space that the original data used. However, since the project was limited to 4 1/2 months and the amount of data we produced was not that big it was sufficient and simpler than incremental backups.

2.2 Software

Erlang was used as the programming language and its OTP behaviours as design principles [17].

Dia [24] is a diagram drawing tool that was a bit hard to use at first, but after learning some tricks it proved to be an efficient and helpful tool.

Redmine [7] which is a project management and issue tracker tool worked well for us. The wiki was a great way to share information though. The repository view was good for browsing the code. We did not use the rest of the functionality much.

git [25] is a powerful version control system which we used to organize ourcode and our documentation. Git involved a steep learning curve, but after sufficient practice proved very useful for version control.

eDoc [2] makes documenting the code really easy since comments in edoc-format can be automatically extracted into structured html-files. Specifications written for eDoc are used by dialyzer as well.

Dialyzer [1] is a nice tool for static analysis of Erlang code that is easy to get started with. It helped in fixing warnings inferring type checks, and testing code quality. However Dialyzer is verbose and raises non-relevant warnings.

Rebar [6] is a nice build tool and framework for creating Erlang/OTP applications. It was flexible and surprisingly easy to use. Compared to something like make/auto tools, getting a working structure for applications is a breeze.

SASL [8] is a proper tool to show various types of reports, but is a bit cumbersome to make proper configuration file in order to run report browser.

EUnit [3] is a unit testing framework for Erlang. EUnit can do too much which makes it very complex and cumbersome to use. Creating tests with it takes quite a while to learn, but we ended up using it in the end because it seems to be the de facto standard and even ships with OTP. Rebar has EUnit support and can test all the applications with a single command.

Common Test [6] is another testing tool that can do unit testing and more. Rebar has a testing keyword which compiles and runs common tests. Like EUnit, Common test also ships with OTP.

PropEr [5] is a tool for doing model based testing where you emulate your states and components using models. It is consequently more complex and harder to use than the other testing tools we tried. We did not end up using it, simply because it was too time consuming, and this level of verification was not a requirement.

Wireshark [9] is an essential tool for network related development. We used Wireshark to understand the packet format, protocols and the order of messages during call setup. Wireshark also proved useful during demos to visualize what is happening in a GSM network.

2.3 Literature

The following literature resources were useful during the project. The complete literature used can be found in the bibliography.

- GSM Specifications for understanding GSM protocols and interfaces. The bibliography lists all specifications we used.
- Books
  - [18] for Erlang programming techniques
  - [19] to understand basics of GSM
  - [15] to understand how mobile systems have evolved over the time
- Reference Manuals
  - Megaco Reference Manual [16]
  - [22] as a reference to Layer 3 GSM Signaling protocols

2.4 Manpower

In software development projects, the most important resource is manpower. In our case this was nine final year master students working full time a full semester. Students in the course had different work and cultural backgrounds and thus could contribute to the project's goals in different roles. The team members' different skills helped us with system administration, domain knowledge, and programming.

2.5 Environment

We had a large room with four white boards and a coffee machine at Polacksbacken. We used the whiteboards to organize processes such as sprint planning and similar activities. The environment was calm and usually undisturbed, which provided an excellent work atmosphere. We wanted to organize the tables in such a way that it would be easy to speak to each other and thus we decided to go with a U-style (as done the previous year). This allowed us to see and easily talk to each other.

Chapter 3

Project methodology and organization

3.1 Scrum

Scrum [21] [23] is a subset of agile methods, based on iterative and incremental development where requirements and solutions evolve through collaboration in a self-organizing, cross-functional team. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle.

3.1.1 Scrum roles

Product owner The product owner's responsibility is to manage and prioritize the product backlog.

Scrum development team Self-managed team which is responsible for delivering the product and Scrum goals to the product owner.

Scrum master The *Scrum* team's servant/leader who ensures that the team achieves the sprint goals. The Scrum master is supposed to enforce the rules and handle possible obstacles and distractions.

3.1.2 Scrum events

Sprint A sprint is a time-boxed period in which a product increment, based on the definition of *"Done"* is produced. It contains sprint planning, daily Scrum meetings called Standup, development work, a sprint review and a sprint retrospective. A sprint is usually between two and four weeks long.

Sprint planning A meeting in which all team members cooperate to make a plan for the next sprint by defining what tasks will be done during the upcoming sprint.

Daily standup A time-boxed event of roughly 15 minutes, done in the beginning of each working day. Its purpose is to keep track of what each team member has done and what their plan for the day is.

Sprint review An event at the end of sprint at which the team together with the product owner reviews the result of the sprint.

Sprint retrospective A meeting after the end of each sprint. During a retrospective the team discusses how the sprint went, points out what worked well and identifies ways to improve the team's work.

3.1.3 Scrum artifacts

Product backlog A list of requirements ordered by priority. It contains attributes and functionality of the product defined by the product owner.

Sprint backlog A list with tasks and requirements for the next sprint. The sprint backlog is defined by the team during the sprint planning and is a subset of the product backlog's contents.

Increment A working version of the product's state at the end of a sprint.

Sprint goal Defines what the team is planning to achieve by the end of the sprint.

3.1.4 Definition of *done*

A shared, transparent, and clear understanding of when a task is completed.The definition of done is the measurement for product increment, confirmed and agreed on by the product owner and the *Scrum* team members.

3.2 Customizing Scrum

We customized and used it as our software development methodology. Most of the sprints in this project took two weeks long, except for the implementation phase which was a three week long sprint. We elected a new Scrum master for every sprint. We opted for a lightweight Scrum process where we reorganised regularly during daily standups to react to changes. To document the tasks for the next sprint we used a single white board without post-its since the number of tasks was usually small. As a punishment for being late to the daily standup we agreed on home-made fika for the group.

3.3 Sprint reflections

3.3.1 Sprint 1: Praveenkumar Bhadrapur

Sprint description

The Scrum goal and definition of done was mutually agreed upon on in Scrum planning. The Scrum goal was component design, interface design and overall system design. The tasks were taken-up by specialised sub-groups we defined rather than individual persons (deviation from Scrum).

Standup meetings were mandatory and were enforced strictly at 9:00 AM every morning.

Dividing further into smaller sub-groups was helpful to organize our design work. This helped us concentrate on component design, which often is an interactive process. Meetings among the sub-groups occured regularly to discuss interface design. An architecture group consisting of one member from each sub-group helped us get the overall design in place.

Fikas were a good get-together and helped us to get to know each other and to work more efficiently.

Sprint review was conducted to revisit the design goal accomplished and documentation was suggested as improvement.

Obstacles and impediments

Impediments were initial lack of understanding already existing components which did not have documentation (OpenBSC) and a lack of knowledge about Media Gateways. Team members gained the required knowledge in time and our contact person helped us to get the required previous Master thesis work on Media Gateways, which could be re-used. As Scrum master daily standups helped me keep track of the progress we were accomplishing with respect to impediments particularly.

Our product owner could not attend the sprint review but he sent a substitute representative from Mobile Arts. Along with him and our contact person we were able to solve the lack of guidance.

The decision to form specific sub-group teams was helpful. Interface discussion meetings were facilitated to help resolve ambiguities.

Communications with product owner

Standups took care of individual work status. Sub-group discussions helped clear ambiguities and also get overall picture sooner. The product owner was present to discuss initial backlog and a substitute product owner to review the completed sprint goal.

Sprint results

Retrospectives helped us stay on track to accomplish project goal and helped us improve, identify our short-comings. Sprint results were achieved with desirable quality.

3.3.2 Sprint 2: Max Morén

Sprint description

For the first time we decided on an exact defintion of done for the sprint.This gave a more concrete feeling on what was to be done as a whole than in the previous sprint, where we only had goals and tasks.

We kept having daily standup meetings in the morning which continued to be a very good way for to find out when people had problems or nothing to do.

Obstacles and impediments

The second sprint was our first implementation sprint. We were just coming to terms with the ways things were supposed to work, so there was some confusion, especially about media handling.

During planning we first thought of implementing the full system leaving out minor features, but opted to instead implement the signalling first, pushing the media handling into a future sprint.

Implementation went well. So well in fact, that in the end myself and a few other team members ran out of work. We pulled in more items from the product backlog and I tried my best to direct idlers to new tasks. We managed to implement a lot of the media functionality in the BSC which was not originally in the sprint plan without sacrificing anything else so I consider it a successful move.

Some of us also came in late sometimes but we did not think of this as a big deal yet. We were working in teams of two, so even if someone missed the standup he could hear from his team mate what was discussed at the standup. Usually people were only a few minutes late, so I chose to ignore the problem for the time being and just enjoy the punishment fikas.

Communications with product owner

Our product owner was unable to attend our previous sprint demo and planning meeting which is perhaps the thing I'm least happy about. This meant that both planning and implementation happened without any feedback from the the product owner. We did have a representative present but he could not answer some of the questions we had about our design.

We solved this by communicating with the owner during the sprint, but it took more time than it would have had to.

Sprint results

The biggest issue that was brought up during the retrospective was code quality. We did not have a coding standard and review system in place, which meant that the code now had a large variety of indentation and naming styles. We should definitely have started earlier with the review process that we later started using in sprint 3.

We had also tested very little during programming, so we made plans to look into *Dialyzer* and *EUnit* early in the next sprint.

3.3.3 Sprint 3: Moritz Rogalli

Sprint description

As it was the third sprint already, people were already used to the Scrum process so following our lightweight Scrum process was working well. However, latecomers were an increasing problem. After addressing the problem at a daily standup and the announcement that the latecomer punishment was going to be enforced more rigorously people were less late. The goal of the sprint was to improve code quality and to implement specification compliant signalling with media.

Obstacles and impediments

As mentioned above we had a problem with latecomers. The division into subteams did not work out as planned since some tasks proved to be less or more work than expected respectively. However, we solved it by shifting responsibilities dynamically and with a minimal overhead during daily standup meetings. Testing and integration took longer than expected which impacted the integration of the call handlers. Their functionality was implemented by the end of the sprint, however integration was pushed into the next sprint to not risk the deliverable being unstable at the end of the sprint. The HLR did not arrive as promised so after checking with the product owner we worked around the problem and created a simple and stubbed version that fulfilled the most basic functionality we needed.

A major misunderstanding on media handling was cleared up by a phone call to Lars by the media team.

The coding standard that was developed in response to issues that arose in the first two sprints were followed by all team members and resulted in more readable code and improved code quality. However, deficits in testing and review became apparent during the end of the sprint and therefore review and testing were made top priority for improvement in sprint 4.

Communications with product owner

Communication in the team was working really well after we rearranged the seating order to reflect the sub-teams. I made sure to repeat important information during the daily standups to enhance the common understanding of our group. Erik did a great job of communicating with stakeholders outside our group. Communication with Mobile Arts proved to be difficult. We had to wait for answers quite a bit, this did however not impact our productivity since we planned ahead far enough.

Sprint results

The retrospective did go well. We discussed many vital issues and came up with improvements. We wanted to focus more on testing and reviewing and that proved important in making the next sprints go smoothly. The code reviewing and testing and also improved the stability of the system drastically during sprint four. The sprint result was satisfying. Unfortunately we did not get a working real HLR and integrating the call workers had to be pushed back. We presented the deliverable at the Mobile Arts office in Stockholm for all interested MA employees and concluded the sprint with pizza, beer, snacks and pool biljard at the MA office. I could not attend the mingling for personal reasons unfortunately but according to the other team members this was a big success.

3.3.4 Sprint 4: Erik Grafström

The goal of sprint 4 was to complete the product and enhance the "ilities" (availability, maintainability and usability). That meant we had to tie up loose ends like HLR functionality and system wide error handling. The sprint was short because of the *project review 2* presentation. We picked some product features like subscriber white-listing and made the session store and the TM into standalone application platform services. How to do system wide testing and the exact list of supported GSM procedures were the major discussions.

The major problem of the sprint was to test the system and figure out if fixes improved the product and if new features could be added without major impediments. To be able to add features easily was something the product was designed to handle.

Testing was hard and we made it even harder by not branching properly in our git repository All team members worked on different parts in the same branch. This created unnecessary regression bugs. The bugs were easy to fix but it interrupted the whole team since responsibilities in the code base were not completely clear. I tried to coach the team and branching improved to the end of the sprint.

We ended the sprint with a product presentation and a more stable product which we could run for demonstration without impediments. A big let down was that the HLR was not delivered.

In retrospective the team did not blame the Scrum master for the impediments since he made sure someone was tasked to fix impediments without major impact on the sprint goals.

3.3.5 Sprint 5: Fredrik Pasanen

Sprint 5 was a short sprint — less than two weeks — with a lot to do. The goal of the sprint was to complete the course report, product report, hold the final presentation and create the deliverable CD/DVD.

The problems during the sprint were for people to know what to do each day. The problem of people not knowing what to do was solved by discussing what had to be done with the person responsible for their respective report during standup. This way what needed to be done could be made clear and it was easy to assign work to people. Muneeb was unfortunately busy with other tasks during the sprint.

In the end the goal of the sprint was completed successfully and we created a complete product and course report. There was no sprint demo this sprint, but we had the final presentation in the course which went fine.

There was no retrospective after this sprint.

3.4 Working agreement

As proposed during our team dynamics coaching in the beginning of the course we agreed on a working agreement to define the rules for our project:

3.4.1 Times

- Baseline: 08-17
- Core hours: 09-16
- Daily standup: 09
- Other meetings: Can only be planned and occur during core hours.
- Lunch: 12-13 +-15 min in 4411 or individual preference.
- Other breaks: Individual preference, at most 40 minutes per day.
- Flex: Allowed, flex has to be communicated and balanced every week.
- Tracking: Individual tracking of 7 hours worth of work per day.
- Weekends: Weekend work is not banned but not encouraged.
- Sick: Tell the team and stay at home.

3.4.2 Food

- Food is not allowed in 4408.
- Water, coffee, tea, soda, fruit, snacks (those that doesn't make strange noises) are allowed in 4408.
- Weekly Fika: Monday at 3 pm
- One person has to provide Fika (not store-bought) on Monday

3.4.3 Communication

- Contact person: Erik Grafström
- System administrator: Moritz Rogalli and Fartash Mehdinejad.
- We will setup project tracking software and and give Teachers/Mobile Arts access to certain parts.
- We will use the software for announcements, planning, all kinds of documents, and version control.

3.4.4 Misc

- Language: Use English only, even during breaks.
- Keep 4408 and 4411 clean at all times.
- Put your bags and coats on the coathanger.
- Cellphones should be quiet.
- Calls and visitors should be handled outside 4408.
- Project visitors has to be planned in advance.
- Gaming, newsreading, social networking, music listening is selfregulated, bring it up if it is a problem.
- If you see a problem with the agreement or the behaviour of a group member bring it up as early as possible.

Chapter 4

Discussion

4.1 Scrum process

We opted for a very lightweight scrum process as described in 3.1. This, and our people-centric, direct communication approach worked very well for us. See the Scrum masters' reflections in 3.3 for details.

4.2 Resource management

A red stress ball was used as a mutex for the nanoBTS, groups who wanted access to the base station had to queue for the mutex. The stress ball helped us to determine conflicts and to use the nanoBTS efficiently. Though started for fun, it proved extremely handy when working on a tight schedule.

4.3 Pair programming

We used pair programming for developing our code. It really helped in some phases as the GSM specifications are dense and inconsistent. The pairs could always solve their problems and communicate problems with other pairs when needed. We reused quite a few components we sometimes did not really understand the choices made by the original authors, discussing them in the pairs made it easier for us to solve misunderstandings.

4.4 Weekly fika

Weekly fikas were held every monday where all project members gathered in the kitchen and spent one hour socializing. It had a nice impact on team spirit and made the team members getting to know each other better. Each week it was one team member's task to provide fika, and as we worked in an international atmosphere, we got to know experience many delicacies from different countries.

4.5 Erlang workshop

Before the project started we had a two week Erlang workshop where we learned how to use the language. This proved to be helpful by providing an insight into the language, the tools and the platforms involved. We learnt the language and felt confident using it within a few weeks. One problem though was that we started our project with reading specifications and designing the product. This created a big gap between learning and using Erlang making it harder to use when we needed it.

4.6 Erlang User Conference

As part of this course, all project members were invited to the annual Erlang User Conference. It is one of the largest gathering of Erlang users both from the industry and the academic world. One of the requirements was to give a poster presentation about our project there. This was an interesting experience and helped us to understand our product better. We could also attend several presentations on the conference and mingle with potential employers and interesting people from the Erlang community.

4.7 Hardware

Setting up the nanoBTS and configuring it took about half a day, Tejas from Mobile Arts as well as the openBSC website helped us with the task.

4.8 Testing constraints and integration

Testing a entire GSM system without GSM test suite proved to be a challenge. Having only one base station also proved to a hindrance in running concurrent tests. Fully fledged functional testing was possible only with manual test cases. We wrote test cases for EUnit where feasible.

4.9 Working environment

A continuous possible distraction for us was to get used to noises and distractions made by external people. Working together closely in one room was an important factor to successfully communicate in the group. The room was a bit bare.

4.10 Working agreement

In the first sprints, up to the second individual interview, from time to time, there was this problem of people coming late. One consequence was that people would lose track of what others were doing. By enforcing Fika to latecomers at the end, the problem seemed to be solved as the late coming occurrences decreased dramatically. The working agreement worked out really well otherwise.

4.11 Previous Year

In contrast to the previous year we did not start using Scrum right away, but we did have deadlines and demos for the product owner about what we had learned. This helped us during the reading by not trying to plan how much time it would take and only focusing on reading and learning. Later on, when we started implementing, we used Scrum with three week sprints not two weeks as done by the previous year. Something that we concluded from the course report of the previous year is that the planning did not go well and because of this we focused on the planning part of Scrum, which we believe helped us a lot.

Chapter 5

Conclusion

All in all the project was successful. We achieved almost all of our goals and had a great time working together. We worked efficiently and our skills added up to a very competent team that was able to deal with all bigger problems that arose during the project.

The project course is a concept that has proved itself over the years and we had no bigger difficulties. However spontaneous ideas like t-shirts for all Project CS students at the Erlang User Conference came in too late some times and could not be realized since we were usually too busy with the project itself.

The training exercises at the beginning of the course were valuable and helped us a lot during development and to manage the group.

Having our own room with all amenities, and because the course was full time which meant that we had no other courses running in parallel helped us a lot in achieving the project goals.

Communication with the stake holders outside the group and the course assistant proved difficult sometimes. They were usually busy with other tasks and could sometimes not provide as much feedback as we hoped for. This was however not a big problem for us as we could solve most problems.

Since the course should be as close to a real work experience as possible, we suggest that all the group members be given a defined number of days off which they have to manage themselves and which could serve both as an upper and a lower boundary for absent days.

Bibliography

[1] Dialyzer User's guide. http://www.erlang.org/doc/apps/dialyzer/dialyzer_chapter.html.

[2] EDoc User's guide. http://www.erlang.org/doc/apps/edoc/chapter.html.

[3] EUnit User's guide. http://www.erlang.org/doc/apps/eunit/chapter.html.

[4] Kernel-based Virtual Machine. http://www.linux-kvm.org/page/Main_Page.

[5] PropEr webpage. http://proper.softlab.ntua.gr/.

[6] Rebar readme. https://github.com/basho/rebar#readme.

[7] Redmine. http://www.redmine.org/.

[8] SASL User's guide. http://www.erlang.org/doc/apps/sasl/users_guide.html.

[9] Wireshark webpage. http://www.wireshark.org/.

[10] 3GPP. GSM 04.08 v.7.0.0. http://www.3gpp.org/ftp/Specs/archive/04_series/04.08/0408-700.zip, 1998. Mobile radio interface layer 3 specification.

[11] 3GPP. GSM 23.002 v.7.0.0. http://www.3gpp.org/ftp/Specs/archive/23_series/23.002/23002-700.zip, 2005. Technical Specification Group Services and Systems Aspects; Network architecture.

[12] 3GPP. GSM 23.018 v.7.0.0. http://www.3gpp.org/ftp/Specs/archive/23_series/23.018/23018-700.zip, 2005. Technical Specification Group Core Network and Terminals; Basic call handling; Technical realization.

[13] 3GPP. GSM 48.008 v.7.0.0. http://www.3gpp.org/ftp/Specs/archive/48_series/48.008/48008-700.zip, 2005. Technical Specification Group GSM/EDGE Radio Access Network; Mobile Switching Centre - Base Station System (MSC-BSS) interface; Layer 3 specification.

[14] 3GPP. GSM 48.006 v.7.0.0. http://www.3gpp.org/ftp/Specs/archive/48_series/48.006/48006-700.zip, 2006. Technical Specification Group GSM EDGE Radio Access Network; Signalling transport mechanism specification for the Base Station System - Mobile-services Switching Centre (BSS - MSC) interface.

[15] Ericsson AB. *GSM System Survey.* Ericsson AB, 2008.

[16] Ericsson AB. Megaco/H.248 Users Guide. http://www.erlang.org/documentation/doc-5.4.12/lib/megaco-3.2.3/doc/html/part_frame.html, 2011.

[17] Ericsson AB. Erlang/OTP R15B. http://www.erlang.org/doc/, 2012.

[18] Francesco Cesarini and Simon Thompson. *Erlang Programming.* O'Reilly, 2009.

[19] Gunnar Heine. *GSM Networks: Protocols, Terminology, and Implementation.* Artech House, 1999.

[20] Harold Welte. Free Software GSM Protocol Stacks OpenBSC, OpenSGSN, OpenGGSN, OsmocomBB. http://elinux.org/images/6/65/Elce2010-welte-openbsc.pdf, 2010.

[21] Henrik Kniberg. *Scrum and XP from the Trenches.* InfoQ, 2007.

[22] ISEL. Sistemas de Telecomunicações III. http://www.deetc.isel.ipl.pt/sistemastele/ST3/arquivo/GSM%20Messages.pdf, 2009. Layer 3 GSM signaling protocols messages.

[23] Ken Schwaber and Jeff Sutherland. Scrum Guide. http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf, 1991-2011.

[24] Kevin Breit and Henry House and Judith Samson. Dia manual. http://projects.gnome.org/dia/doc/dia-manual.pdf, 2000.

[25] Linus Torvalds. Git User's manaual. http://schacon.github.com/git/user-manual.html, 2005.

[26] nanoBTS. The world's most deployed picocell. http://www.hexazona.com/nexwave/docs/ipaccess/nanoGSM%20Brochure.pdf, 2012.

[27] Olle Gällmo. Projekt DV (project CS) 2011. http://www.it.uu.se/edu/course/homepage/projektDV/ht11, 2011.

[28] Team Mobile Arts Project CS 2011. Product report GSM Call Service, 2012.

Chapter 6

Appendix A: Individual contributions

6.1 Praveenkumar Bhadrapur

Mainly worked and was responsible for Connection Service, Media Gateway, Media Gateway Controller and Operator Handling functionality for Media gateway.

During the initial sprint , I worked with Ebby to understand MG, Media Gateway Controller and Connection Service. and worked with other team members to understand GSM architecture and network signaling.

During the design phase of our project, I designed Connection Service with Fartash, Reza. Re-jigged the design of Media Gateway and Media Gateway Controller to suit our project. Operator functionality was designed along with Max for Media Gateway. Helped design the system architecture.

During the implementation phase, I implemented Connection Service and interface'd with workers. Implemented operator handling functionality for media gateway. Implemented modifications to have megaco terminations independent of RTP as bearer. Used *eclipse* with *erlide* and found it to be write Erlang programs.

During the testing phase, I tried *E Unit*'s ,*E Unit* was cumbersome for more interactive components like CS, MGC, MG. *Dialyzer* was run over CS, MGC, MG helped fix warnings and have better code in general. *Wireshark* was useful and Max helped me use it to get the RTP packets routed to BSC's RTP terminations. *Latex* , *dia* have been useful.

During the documentation phase, documented product report for CS, MGC, MG components. Presentation and EUC poster preparation was done with Erik, Max and Moe, Fredrik respectively.

In the 1st Sprint, I took up scrum master role to achieve the goal of designing components, interfaces and overall system architecture.

6.2 Erik Grafström

Design

To be able to design something handling a couple of MSs and calls with media I had to both understand Erlang/OTP and the message passing nature of GSM. We did have a set of requirements but those were written in telecom language which forced me to read up. I had some prior knowledge of GSM and the RAN/CN but only at the network level. My previous experience in distributed and embedded systems was helpful.

My first task was to define signaling sequences for call and location procedures. I did this together with Max and Fartash and we finished the task with great confidence since we had a solid specification collection and books/sites to use as reference.

I worked with Ebby and Egemen on my second design task, the MUS component. We defined what is session is and how messages would be passed back and forth between a MS and the corresponding procedure handler in the MUS. The design consisted of a MUS gen_server, Session Store record and gen_fsm for the procedure handlers. Inter working of the A, B, C, D and E interfaces were one of our major concerns, our internal interfaces had second priority.

The third design task was overhauling the VLR since the VLR from previous year lacked proper procedure handling, we used our experience from the MUS design and defined a VLR server with procedure handlers.

Overall we worked hard to make the product modular to make it easier to add more services like SMS, O&M and maybe data.

Implementation

The first real implementation task was done with Max, we extended previous years A interface codecs and wrote a proof of concept MUS capable of handling one signaling session. It handled signaling required for call and location procedures. After the MUS concept was running we looked into media circuits. We played around with the media capabilities of *osmo-bsc* and supervision of the MUS.

The next task was creating the procedure handlers for call setup, Max and I worked on originating and terminating while Egemen worked on the gateway role. The first rough implementation was integrated with the session, routing and connection service logic which resulted in an MUS capable of concurrent calls.

The third task was improve the location procedures. Moe, Egemen, Reza and I took the path of reimplementing the VLR and that was a commitment which I carried over a couple of sprints. The gen_server and gen_fsm approached allowed us to implement a VLR supporting specification compliant procedure handling on both the B, C and D interface.

Testing and cleanup

We did not have an overall plan for testing the system which translated into doing manual ad hoc testing. I did it throughout the project which took a lot of time but it was fruitful. I was able to fix impediments in the parts of the code I was responsible for and also discuss the inter working with other components.

Since I had been working on location, call and MUS procedures throughout the project, I had to maintain that code and later clean it up.

Documentation

I have been part of the presentation group throughout the project covering the bachelor, poster, MA, review 2 and final presentation. I have continuously been working on creating diagrams for implementation, design and presentation. One of my major concerns has been to figure out how to explain the GSM domain, our design and implementation ideas to an audience without domain knowledge.

Since we have been using latex for reports and presentation material I have introduced and helped some team members to get started with typesetting in latex.

Tools

I've extensively used the 3GPP and ITU-T specifications as reference during design and implementation since we wanted to make our product as specification compliant as possible. 3GPP 23.012, 23.018, ITU-T Q.713, Q.763 and Q.764 have been extensively used for call and location procedures. [11] [12] 3GPP 48.008, 48.006 and GSM0408 were used as reference for A interface messaging. [13] [14] [10]

Development tools like Erlang/OTP, Red mine, git, wire shark, eunit, edoc, rebar have been interesting to work with. I have used some of the tools before but learnt a lot about the tools, especially Erlang/OTP.

Methodology

Pair programming in combination with Scrum has been one of the best things about the project. Working in the same room made communication between pairs easier, but there is always noise and sometimes individual space is compromised. We made sure we spent time on fika to get to know each other.

Contact person

I have been the contact person from day one, responsible for communication with MA, examinator Olle and assistant Muneeb. I have assembled sprint reports and made sure all stakeholders have been informed about upcoming events. I've also handled the day-to-day business of relaying external communication into the stand ups and reporting sick people.

Scrum master

I wanted to try out the role of scrum master. The 4th sprint was my pick since it involved both tying the product together and coach the team to focus on the right tasks.

6.3 Ebby Wiselyn Jeyapaul

During the design phase I collaborated with Praveen, since it was the requirements sprint, we spent a lot of time reading through the specs, what could be highlighted was the effort taken to get a complete picture of the problem we were trying to solve, an hack and learn approach was impossible so we spent most of the time with the documents, trying out possibilities of design, understanding how the signalling and the components fell in place.

After the initial design and gathering we had to describe the overall system design. This was much easier and also left us with a lot of gray areas not fully understanding certain aspects of our design. The final part of the design phase was to design components and do the flowchart, to design connection service, and media gateway. I also collaborated with Erik, Egemen in designing the component design of MUS Controller, the MUS and the TCP Server. At the start of the implementation phase I initially worked a bit on the codecs library.

I was paired up with Fred for the implementation phase, it proved very handy, fred could analyse and solve the problems faster and pretty good coder too, this was good for me to pair up and implement the crucial components like the TCP Server, which was the intial component done, to test with the codecs library, we needed a simple, working, design to get to the implementation phase, and this proved handy.

Next was the implementation of the complicated MUS module, me and Fred had to run through the documents created in the requirements phase, the MUS module was the connecting bridge between the BSS and the MG team, so we had to deal with a lot of stress, accomodate many changes, and also consider effective integration, I could relate to MG because I was involved in the design, and fred could relate to BSC . MUS Controller, was the module which contained a lot of complexity and it was done in best effort way, the only change of design that we could envision now was to clear seperate layers of signalling, but it still has it's cost with extra complications in dealing with stray messages that have no associations with any layer.

I also implemented the Session Store and Transit Module with fredrik, and periodically maintain small parts of location worker, MO, MT and the gateway workers. I also watched out for commit reviews, changes in modules responsible, and also random commits which should affect functionality. Handling the integration and writing a module which was susceptible to many changes and extremely cohesive

In the final sprint, I designed the initial test case document, to test and document the entire system.

6.4 Fartash Mehdinejad

I joined the project with no background in GSM. Thus, I had to spend a lot of time doing readings and research to gain a general understanding of the field. It started with general GSM networks topic and how they work, and ended with detailed call sequences that are necessary for a call to be established.

After the research phase, along with other members of the team, I participated in the architecture and interface design of the system. In the next stage, the team divided itself into a number of subgroups, each of which implementing a part of the system. I joined the Connection Service subgroup whose task was design and implementation of an application platform for the system.

Considering that I did not have any background in Erlang, I needed to do readings about Erlang/OTP and different behaviors. The Connection Service was supposed to communicate with the MG to create contexts, so I had to understand how the MG that had already been given to us worked thoroughly, and also read the documentation of Megaco/H.248.

After implementation of the Connection Service, I reviewed all the code to make sure it was flawless.

In the debugging phase of our project, I spent time learning how Dialyzer worked, and used it to extract some of the issues of the Connection Service. Also, I worked with VLR team to resolve minor bugs in the system that had roots in VLR's behavior.

Finally, I worked as co-system administrator with Moe. Also, I and Erik presented our project to first year students.

6.5 Max Morén

Like most people, I wrote code in almost every module, although my main responsibility was media handling, most of all in the BSC.

During first sprint, I worked with Erik and made signalling sequence diagrams for all the supported sequences.

During the second sprint, in which I acted Scrum master, I again worked with Erik and we wrote the IPA and SCCP codecs. We also extended the other GSM codecs with the new message types we needed. In the second half of the sprint we implemented the first workers for basic originating and outgoing call handling.

Also during second sprint, I started working alone on implementing an RTP bridge in OpenBSC. I got basic media forwarding working, using a temporary manual UDP switching application that could connect two calls together.

During the third sprint, I finished the work on the BSC and added circuit management support to it. I also wrote the MSC RTP termination agent. During this I communicated much with Praveen who was working on the MGC, MG and connection service.

During the fourth sprint, I cleaned up and fixed code in various parts of the system. I wrote unit tests where possible and set up proper OTP application structure. I started working together with Fredrik on the supervision and robustness features of the system. Finally I made it possible to generate a combined release of all system applications and another release for the media gateway.

In the fifth sprint I continued working with Fredrik and we finished the supervision and robustness work and then tested the system thoroughly. Together with everyone else I made diagrams and wrote sections for the reports and product presentation. Me, Erik and Praveen finally presented our product during the final product presentation.

6.6 Fredrik Pasanen

During the first sprint, I — like everyone else — started with reading specifications and studying GSM-networks. I grouped up with Egemen and Reza in creating sequence diagrams for different location and call scenarios which were the deliverables for this sprint.

For the first half of sprint one, I — together with Egemen and Reza — worked on the system architecture. During the second half I worked with Max in figuring out how OpenBSC handles signalling and media.

In the second sprint, I got paired with Ebby in creating the Supervisors, MUS and MUS Controller. I were also in the group creating the poster for the Erlang User Conference poster consisting of me, Praveen, Moe and Egemen.

In the third sprint, I again got paired with Ebby in creating the Transit Module used for routing ISUP-messages. I were also in the coding and testing guidelines group with Ebby and Max. I also tried some tools for testing our code (PropEr and Common Test).

In the fourth sprint, I continued working with Ebby, but now with the task of making the Session Store and Transit Module external applications.

In the fifth sprint, I were chosen as the Scrum Master. Started this sprint with completing the crash handling for the system and the testing and fixing the system if necessary (worked with Max on this). During the rest of the sprint I helped write the course and product reports.

6.7 Moritz Rogalli

System Administrator

I had the role of system administrator together with Fartash. This meant that it was my job to provide an *infrastructure* and *services* to the group to be able to work. For that I installed several *virtual machines* for our purposes. The provided services range from development and process services like *git*, *Red mine* and *pdf exports* for wiki pages to testing infrastructure. I also created support services like an internal *DNS* system and a *backup* infrastructure. The role also gave me the responsibility to apply for hardware and resources at systems support. I also was responsible for supporting people from Mobile Arts to be able to work with us. For that I applied for external *remote access*, provided *virtual machines* and knowledge about our setup whenever needed by Tejas or Johan from MA. In that role I, together with Tejas, was also responsible for getting the *base station* up and running.

Poster for the Erlang User Conference

Together with Praveen and Fredrik I was in charge of developing the *poster* for the Erlang User Conference. After we composed the input we collected from the whole group for the content into a concept I made the *layout* and *diagrams* for the poster.

VLR Rewriting and Spec Compliance

Together with Egemen I was responsible for *redesigning* and *rewriting the VLR*. This task was bigger than originally expected since the old VLR was not suitable at all for our purposes. We were later joined by Erik and Reza who took over the location management. I mainly contributed the *incoming call handler* in the VLR and the *mt worker*, which handle a call on the *terminating side* and the *roaming number support* in the VLR.

Scrum Master Sprint 3

For the third sprint I was elected *scrum master* during which we concentrated on *improving our coding style* and rewriting and improving the system to be more *spec compliant.*

6.8 Mohammadreza Taghilu

In the very first 3 weeks of the project, our team was divided into 3 subgroups to do research in *GSM system architecture*, *components* and *terminology.* Through instructions given by *Mobile Arts* as *product owner*, I, along with my subgroup-Egemen and Fredrik- likewise other teammates, started working on *sequence* *diagrams* over different scenarios in GSM *location and call procedures* by studying GSM resources, such as specifications, books and diagrams. Meanhwhile, it was also needed to survey theses done in conjuction with the company in previous years.

Eventually, we ended up with a proposal for *system architecture* and 3 seperable parts to be done: *BSC/BTS*, *MSC/VLR* and *CS and MG.* I chose CS, which was a part of *application platform.* Hence, I collaborated with Fartash and Praveen to design, and digitize *CS call's finite state machine*, *Service Virtual Switch (SVS) flow*, *Access Virtual Switch(AVS) flow*, *release flow*, *tone flow*, *virtual port connection* and *join flow*, along with correspondent data structure and skeleton. Afterwards, I needed to spend some time over Erlang *error handling*, *behaviours(specially gen_fsm)*, and *database handling* to help on implementation phase.

In the next *sprint*, in order to improve *location management*, I joined *MUS group*, collaborating with Erik, Egemen, and Moe who had already spent considerable time over implementation and reviewing related modules in MSC/VLR.

The task was to improve modules VLR, *(MSC)location worker*, and VLR *location worker* to become *spec compliant* and also to revise them based on *code conventions*, therefore I needed to review other related Singaling modules such as *mus_con*, *tcp_server* and *codex modules* as well as *specifications* in connection with implementation details of location and call sequences, in parallel.

Along with Erik, I tried Erlang *rebar* and worked over *application* and *supervision design*, location management modules and also system architecture design. We also tried *SASL* application in order to *log errors* while running MUS application. Using *Dialyzer*, I also reviewd *mo_worker* and *mt_worker* and drew fixes over functions, code design, and terms.

At the end, I worked with Fartash on some parts of course and product report.

6.9 Egemen Taskin

During first 3 weeks I spent my time with the internal group(Fredrik & Reza) to understand whole story of how basic GSM call set-up works in different scenarios required to be handled in the product and read related specifications and I noticed that specifications are not just enough to understand basic principles of GSM then i read some parts of Ericsson's education notes and a book about GSM. Also, because of the fact that I am a newbie at writing code in Erlang, I started revising last year's code and tried to catch implementation tricks.

After everybody got the whole idea behind GSM, we had brainstorming sessions to create a system architecture. After these sessions, we are divided into 3 groups and I involved in MUS group. We as group specified state machine diagrams and decided initial Erlang message format for the communication among entities and then I digitalized all state machines for future reference and for that everybody wants to see whole story in our architecture.

I and Moritz started examining Tejas's location worker and changed former message format with the format that we had decided before. After that, I got the skeleton code of MO & MT worker ,implemented by Max & Erik, and added the support for specified release cases and timeouts according to specifications. After that, i implemented Gateway worker and release cases related to it.

As whole group, we decided to re-implement VLR according to specifications. I and Moritz designed new VLR, implemented some part of it and we noticed that it is more complex and more work than we think. Therefore, I and Moritz mostly focused on VLR without Location management part. Then, I implemented VLR's outgoing call handler worker and integrated it into MO & VLR. In documentation phase, I and Ebby wrote partly product & course report.