UNIVERSITY OF APPLIED SCIENCES RAPPERSWIL
DEPARTMENT OF COMPUTER SCIENCE

BACHELOR THESIS

XMPP-Grid Broker

Authors:
Fabian HAUSER and
Raphael ZIMMERMANN

Advisor:
Prof. Dr. Andreas STEFFEN

External Co-Examiner:
Dr. Ralf Hauser
PrivaSphere AG

Internal Co-Examiner:
not yet defined

Spring Term 2018

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

DON'T PANIC

"It looked insanely complicated, and this was one of the reasons why the snug plastic cover it fitted into had the words DON'T PANIC printed on it in large friendly letters."

The Hitchhiker's Guide to the Galaxy

Abstract

Fabian HAUSER and Raphael ZIMMERMANN

XMPP-Grid Broker

Management Summary

Acknowledgements

We would like to thank our advisor, Prof. Dr. Andreas Steffen, for his continuous support and helpful comments.

Contents

Chapter 1

Introduction

1.1 Motivation

In this section, we legitimate this thesis and explain the value and applicability of our proposed solution.

1.2 Background

1.2.1 XMPP (eXtensible Messaging and Presence Protocol)

The Extensible Messaging and Presence Protocol (in short *XMPP*) is an open protocol that enables the near-real-time exchange of small data between any network endpoints [7]. While it was originally designed as an Instant Messaging (IM) protocol, it can be used for a wide range of data exchange applications [6].

XMPP is made of small building blocks defined in the core protocol [7] and numerous extensions called *XEPs* [8]. The core is comprised of functionality for setup and encryption of communication channels, *XML* streams, error handling and more. Additional functionality such as *Service Discovery* [4] and *Publish-Subscribe* [1] are defined in separate extensions.

Although XMPP supports peer-to-peer communication, it is often used in a traditional client-server architecture. A client can send data to any addressable entity using *Jabber* Identifiers, hereafter called *JID*. If the *JID* of the receiver has a different domainpart than the current server, the message is forwarded to the responsible XMPP server under its domain [7].

The data exchanged over XMPP is *XML* which makes the protocol structured and extensible, but leads to some protocol overhead. XMPP communicates over unidirectional data streams with a server, which are basically long-lived *TCP* connections. The client opens a channel to the server over this connection, and the server opens one back (i.e. `<stream>` XML tags). In both streams, an XML document is opened after the connection is established. During the conversation, an arbitrary amount of *stanzas* (specified XML child elements) are written to the stream. Before a connection may be terminated, the root element is closed (i.e. `</stream>`) and both streams form valid XML documents [7][5].

The core *stanza* types are *Messages* (`<message/>`), *Presence* (`<presence/>`) and *Info/Query* (`<iq/>`). *Messages* can contain arbitrary data similar to email but are optimised for immediate delivery. *Presence stanzas* deal with network availability and the propagation of user presence information. An *Info/Query stanza* consists of a request and response (similar to the GET and POST HTTP methods), which is used for feature negotiation, configuration and general information exchange. Because of these coarse semantics, XMPP provides a generalized communication layer [7][6].

Figure 1.1 Two XMPP domains (servers), one with two users and one with one mobile user. illustrates an example setup with two servers and three clients.

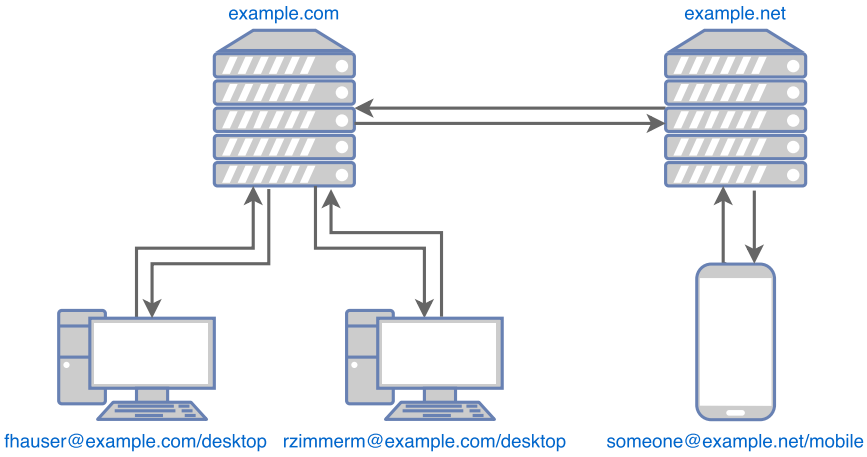

FIGURE 1.1: Two XMPP domains (servers), one with two users and one with one mobile user.

1.2.2 Relevant XMPP Extensions

The XMPP grid draft [2] makes heavy use of multiple XEPs, most notably *Publish-Subscribe*. The following XEPs are relevant as well:

XEP-0004: *Data Forms* is a flexible protocol that can be used in workflows such as service configuration as well as for application-specific data description and reporting. The protocol provides forms processing, common field types as well as extensibility mechanisms [3].

XEP-0030: *Service Discovery* enables applications to discover information about the identity and capabilities of an entity, e.g. whether the it is a server or not, as well as items associated with an entity, e.g. a list of *Publish-Subscribe* nodes [4].

XEP-0059: *Result Set Management* allows entities to manage the receipt of large result sets, e.g. by paging through the result or limiting the number of results. Result Management is often desired when dealing with large dynamic result sets from service discovery or publish-subscribe and time or other resources are limited [**?**].

XEP-0060: *Publish-Subscribe*

The *Publish-Subscribe* Extension, hereafter called *PubSub*, enables XMPP entities broadcast information via nodes to subscribed entities [1].

Nodes, also known as topics and exchanges, are the communication hubs. Entities can create nodes and configure them, e.g. set up subscription timeouts, limit publish and subscription rights. The configuration mechanism is based on data forms (XEP-0004).

The protocol defines a hierarchy of six affiliations of which only ‘owner‘ and ‘none‘ are required. The remaining four affiliations are recommended. An owner of a node can manage the subscriptions as well as affiliations of other entities associated with a given node.

To make the creation of node simpler for clients, *PubSub* defines five node access models: open, presence, roaster, authorize and whitelist. The open model allows uncontrolled access while presence and roaster are specific for IM. Using the authorize model, the owner has to approve all subscription requests. The whitelist model enables the owner to maintain a list of entities that are allowed to subscribe.

1.2.3 IETF Internet-Draft: Using XMPP for Security Information Exchange

This IETF Internet draft describes how the XMPP protocol enhanced with the previously discussed XEPs, most notably the *PubSub* extension, can be used for the exchange and distribution of security-relevant information between network devices.

One of the primary motivation for using XMPP for this task is the fast propagation of such security-relevant data. Using XMPP for such a task also comes with its downsides. Because the XMPP server (Broker/Controller) is the central configuration component in charge of managing access permission, its compromisation has serious consequences.

The draft describes the trust model, thread model as well as specific countermeasures, e.g. to use at least TLS 1.2. These countermeasures also define restrictions of the XMPP protocol and its extensions, e.g. by limiting the node access models of *PubSub* to whitelist and authorized only [2].

Bibliography

[1] P. M. andPeter Saint-Andre and R. Meijer. Publish-Subscribe. XEP-0060, Feb. 2018.

[2] N. Cam-Winget, S. Appala, S. Pope, and P. Saint-Andre. Using XMPP for Security Information Exchange. Internet-Draft draft-ietf-mile-xmpp-grid-05, Internet Engineering Task Force, Feb. 2018. Work in Progress.

[3] R. Eatmon, J. Hildebrand, J. Miller, T. Muldowney, and P. Saint-Andre. Data Forms. XEP-0004, Aug. 2007.

[4] J. Hildebrand, P. Millard, R. Eatmon, and P. Saint-Andre. Service Discovery. XEP-0030, Oct. 2017.

[5] J. Moffitt. *Professional XMPP Programming with JavaScript and jQuery*. Wrox Press Ltd., Birmingham, UK, UK, 2010.

[6] P. Saint-Andre. Streaming xml with jabber/xmpp. *IEEE Internet Computing*, 9(5):82–89, Sept 2005.

[7] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120, Mar. 2011.

[8] P. Saint-Andre and D. Cridland. XMPP Extension Protocols. XEP-0001, Nov. 2016.

List of Figures

List of Tables

Appendices

A.1 Task Description

TODO

A.2 Project Plan

XMPP-Grid Broker: Project Plan

Authors:
Fabian HAUSER and
Raphael ZIMMERMANN

Advisor:
Prof. Dr. Andreas STEFFEN

Spring Term 2018

Contents

Chapter 1

Project Overview

The goal of the bachelor thesis is to build a broker application and graphical user interface to administer XMPP-Grids according to draft-ietf-mile-xmpp-grid, as described in the task description [**?**].

Chapter 2

Project Organization

All team members have the same strategic rights and duties. Prof. Dr. Andreas Steffen is our project advisor.

2.1 Roles

Due to the small team size, most roles are performed by both team members.

Raphael Zimmermann
: project management, software engineering, quality assurance.

Fabian Hauser
: infrastructure management, software engineering, quality assurance.

Chapter 3

Project Management

3.1 Components

For a better overview and to allow us a sophisticated time assessment, we decided to group tasks into categories, i.e. JIRA components. Components represent processes, documents and products which are to be released.

Currently, tasks are separated into following components:

- Application
- Final Submission Document
- Management
- Poster
- Presentation
- Project Plan

3.2 Time Budget

The project started with the Kickoff Meeting on 19.02.2018 and will be completed after 16 weeks by 15.06.2018. The two team members are available for 360 hours each during the semester which corresponds to a weekly time budget of 20 hours per person and two weeks with a weekly time budget of 40 hours per person.

Apart from the statutory holidays, there are no further absences planned.

3.3 Schedule

The project schedule is an iterative process based on elements of SCRUM.

We decided on a sprint duration of approximately week, but allow deviations in working hours depending on statutory holidays.

3.3.1 Iterations & Milestones

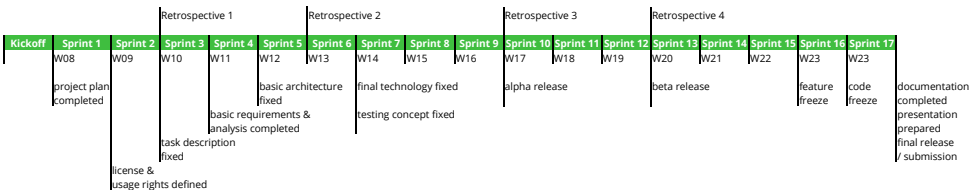

FIGURE 3.1: Overview of the 16 iterations

The goals and milestones resulting from each sprint are shown in Figure 3.2.

TABLE 3.1: Meeting Time Budget

Meeting Type	Total Duration per Person	Total Duration for the Team
Supervision Meetings	12 hours	24 hours
Standup Meetings	16 hours	32 hours
Sprint Planning Meetings	18 hours	36 hours
Retrospective	4 hours	8 hours
Total	50 hours	100 hours

3.3.2 Meetings

The team works every Monday (08:00 - 12:00) and Tuesday (08:00 - 17:00) together in the study room and remotely Fridays (08:00 - 17:00). Wednesday and Friday begin with a daily stand-up meeting taking no longer than 15 minutes. Sprint planning meetings are carried out on Tuesday at 10:00. Table 3.1 shows an overview of the total meeting time budget.

Regular meetings with the project advisor take usually place on Monday in Prof. Dr. Steffen's office.

Raphael Zimmermann will take meeting notes for every meeting. Meeting minutes are published on the project website afterwards.

Sprint	**Sprint 1**	**Sprint 2**	**Sprint 3**	**Sprint 4**
Tag	**0.1.0**	**0.2.0**	**0.3.0**	**0.4.0**
Date	20.02.2018 – 27.02.2018	27.02.2018 - 06.03.2018	06.03.2018 – 13.03.2018	13.03.2018 – 20.03.2018
Milestones	- license & usage rights defined	- task description fixed	- basic requirements & analysis completed	- basic architecture fixed
Time Budget	40	40	40	40
Tasks	- setup remaining project infrastructure - read ietf draft, XEP standards etc. - compile list of open questions - begin with chapter "initial situation" - analyze existing python proof-of-concept	- collect non-functional requirements (NFRs) - compile set of user stories - draw wireframes - research frameworks and technology. Prepare architectural decisions	- extend NFRs, user stories, wireframes - make first architectural decisions - further framework and technology research - implement proof of concepts for critical components	- make big architectural decisions - implement proof of concepts for critical components
Documents / Chapters / Artefacts		- chapter "Initial Situation" completed - signed task description		

Sprint	**Sprint 5**	**Sprint 6**	**Sprint 7**	**Sprint 8**
Tag	**0.5.0**	**0.6.0**	**0.7.0**	**0.8.0**
Date	20.03.2018 – 27.03.2018	27.03.2018 – 03.04.2018	03.04.2018 – 10.04.2018	10.04.2018 – 17.04.2018
Milestones		- final technology fixed - testing concept fixed		
Time Budget	40	40	40	40
Tasks	- implement proof of concepts for critical components - draft testing concept	- implement proof of concepts for critical components - finalise testing concept	- implement most important user stories - make further architectural decisions	- implement most important user stories - make further architectural decisions
Documents / Artefacts		- Chapter "Our Approach" completed		

Sprint	**Sprint 9**	**Sprint 10**	**Sprint 11**	**Sprint 12**
Tag	**0.9.0**	**0.10.0**	**0.11.0**	**0.12.0**
Date	17.04.2018 – 24.04.2018	24.04.2018 – 01.05.2018	01.05.2018 – 08.05.2018	08.05.2018 – 15.05.2018
Milestones	- alpha release			- beta release
Time Budget	40	40	40	40
Tasks	- implement most important user stories - make further architectural decisions - write integration tests	- implement secondary user stories - write integration tests	- implement secondary user stories - write integration tests	- implement secondary user stories - write integration tests
Documents / Artefacts	- alpha release bundle			- beta release bundle

Sprint	**Sprint 13**	**Sprint 14**	**Sprint 15**	**Sprint 16**
Tag	**0.13.0**	**0.14.0**	**0.15.0**	**0.16.0**
Date	15.05.2018 – 22.05.2018	22.05.2018 – 29.05.2018	29.05.2018 – 05.06.2018	05.06.2018 – 12.06.2018
Milestones			- future freeze	- code freeze
Time Budget	40	40	40	80
Tasks	- implement remainging user stories - improve code base - write integration tests	- implement remainging user stories - improve code base - write integration tests	- implement remainging user stories - improve code base - write integration tests	- fix eventual bugs - write documentation
Documents / Artefacts				- final release bundle - poster - abstract

Sprint	**Sprint 17**
Tag	**1.0.0**
Date	12.06.2018 – 15.06.2018
Milestones	- documentation completed - presentation prepared - final release / submission
Time Budget	40
Tasks	- write documentation - submit documents
Documents / Artefacts	- personal reports - management summary - presentations - time accounting

FIGURE 3.2: Detailed overview with tasks and milestones of all 16 sprints.

Chapter 4

Risk Management

An assessment of the project-specific risks is carried out in Table 4.2 as time loss during the whole project. The risk matrix in Table 4.1 provides an overview of the risk weighting.

To account for these risks, we reduce our weekly sprint time by the total weighted risk applicable to the planned task topics (on average approximately 13.5%). We also review the risk assessment after every sprint, adapt it and take measures if necessary.

Probability \ Severity	High ($\geq 5d$)	Medium (2-5d)	Low ($\leq 2d$)
High ($\geq$ 60%)	1		
Medium (30-60%)	6		
Low ($\leq$ 30%)		3, 4, 5	

TABLE 4.1: The risk matrix. Numbers reference to the risk assessment Table 4.2

TABLE 4.2: Risk assessment table. Time in hours over the total project duration.

#	Title	Description	Prevention / Reaction	Risk [h]	Probability	= [h]
1	Incomplete reference documentation	The reference documentation / standards are incomplete or difficult to comprehend.	Discuss missing parts with project advisor	60	60%	36
2	Communication errors	Errors due to miscommunication or misapprehension.	Maintain a high level of interaction, precise specification of tasks responsibilities, conduct meetings if ambiguities exist.	30	50%	15
3	Problems with project infrastructure	The used project infrastructure is not or only partially available, or data loss occurs within management software.	Clean setup and self-hosting of the tools to prevent third-party dependencies.	45	30%	13.5
4	Scope creep	The project's scope is extended over the project course.	Define the project scope and limitations precisely. Discuss changes with the project advisor.	45	30%	13.5
5	Dependency errors	There are errors/bugs in third-party dependencies, i.e. libraries.	Carefully select libraries and limit thid-party dependency to a minimum.	30	30%	9
6	Missing dependency documentation	Selected libraries are lacking proper documentation	The documentation quality of a library should be a selection criterion.	30	40%	12
	Total weighted risk					99

Chapter 5

Infrastructure

5.1 Project Management and Development

For project management we utilise JIRA[1]. As document/code storage, git repositories on Github are used, the continuous integration/deployment will be defined in the course of the project.

These applications are hosted on our HSR project server, which runs a standard Ubuntu Linux 17.10.

5.1.1 Development Tools

The development tools will be defined in the course of the project.

5.2 Backup and Data Safety

An incremental backup of the project server including the source code and documentation is created on an independent system every night.

As our documents and code is stored in a git repository, they are also distributed on all development systems.

Chapter 6

Quality Measures

To maintain a high standard of quality, we take the following measures:

- short sprint reviews
- four extended retrospectives
- code reviews
- automated unit and integration testing
- publish all documentation on the project website using continuous integration/delivery.
- using continuous integration for source code

6.1 Documentation

The official documents such as the final submission document, meeting minutes as well as this project plan are written in LaTeX respectively ASCIIDoc and published on the project website [1] containing all PDF documents.

The sources are in both cases kept under version control in the same repository, which allows us to use the same tools and processes for documentation and code. The continuous integration server builds and publishes the website whenever new changes are pushed to the repository.

6.2 Project Management

Because the project plan allows for an iterative process, we use JIRA with its SCRUM-features (such as sprint creation or boards) for project management.

6.2.1 Sprint Planning

Each sprint is mapped to JIRA, which allows the project advisor to trace the project progress. Sprints are represented as boards on which the current state and assignee of any issue is easily visible ("To Do", "In Progress", "Review", "Done").

6.2.2 Definition of Done

An issue may be closed if ***all*** of the following conditions are met:

- All functionality conforms to the specification. Any deviations must be discussed and decided by the team.
- The source code is reasonably documented.

[1] https://xgb.redbackup.org

- No code is commented out.
- No warnings and errors by the compiler or any other quality tool.
- A review is performed and accepted in a pull request.
- The corresponding branch is merged into the stable branch (e.g. master).
- All documents are up to date including the project website.
- Reasonable unit and integration tests exist and pass.
- The complete continuous integration pipeline works.
- All time is logged.

6.3 Development

We decided to use GitHub Flow[2], a straightforward development workflow.

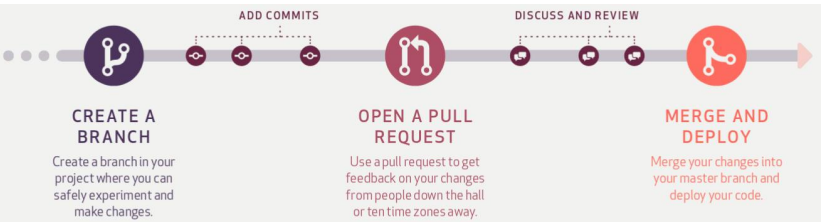

FIGURE 6.1: GitHub Flow illustrated (Source [2])

Since the effective technology will be fixed later in the project, concrete coding guidelines, tools, metrics and an error policy will be defined when appropriate.

6.4 Testing

All functionality must be automatically testable using continuous integration. Any non-trivial function/method must be verified with unit tests.

Integration tests verify extended test scenarios.

A minimal performance analysis will be carried out at the end of the project.

Bibliography

[1] Atlassian Inc. Open Source Services by Atlassian Inc. https://developer.atlassian.com/opensource/, 2017.

[2] Github Inc. Github Flow. https://guides.github.com/introduction/flow/, 2013.

List of Figures

List of Tables

List of Abbreviations / Glossary

A.3 Development Guide

Development Guide

Tools

- Git >= 2.0 for version control

Writing Guidelines

In order to have a consistent style of writing, we defined the following guidelines. These guidelines apply to all documents related to the redbackup project.

- Keep it brief, clear and objective
- Write short and straightforward sentences
- Do not use synonyms for concepts etc. (always use the same wording, e.g. 'client' or 'node')
- Abbreviations must be introduced the first time they occur in the text (except well-known ones)
- Prevent ambiguity in sentences
- Use personal style ("we") whenever appropriate; usually for the description of our work.
- When a gender-specific pronoun is required, use "he/she".
- Use present tense except for the description of (our) completed work.

Definition of Done

To maintain our high quality needs, we determined following definition of done guidelines:

- All functionality conforms to the specification. Any deviations must be discussed and decided by the team.
- A review is performed and accepted in a pull request.
    - The source code is reasonably documented.
    - No code is commented out.
    - No warnings and errors by the compiler or any other quality tool.
    - Reasonable unit and integration tests exist and pass.
    - All documentations are up to date including the project website.
    - The complete continuous integration pipeline works.
    - The code is formatted according to the guidlines (i.e. according to RustFmt)
- The corresponding branch is merged into the stable branch (e.g. master).
- All time is logged.

A.4 Time Accounting

TODO

A.5 Meeting Minutes

Meeting 2018-02-19

Attendees

- Fabian Hauser, fhauser
- Raphael Zimmermann, rzimmerm
- Prof. Dr. Andreas Steffen, SFF

Minute Taker: rzimmerm

Agenda

1. Administrative tasks
2. Where to start
3. Date and time for the next meeting
    - 2018-02-26, 09:00 in SFFs office

Discussions / Decisions

1. Administrative tasks
    - All documentation artifacts will be published on the project website.
    - We will use JIRA for project planning.
    - We will decide later, which continous integration tools we use.
        - The decision must allow the INS to take over the project after the BA.
        - reasons for the decision must be documented.
    - Decisions on the programming language and frameworks are made later.
        - Our experience and productivity in a given eco system must be considered as well.
2. Where to start
    - Read the draft XMPP for Security Information Exchange.
    - Learn more about XMPP (Read the specs and the mentioned XEPs).
    - IODEF payloads are not the main focus of this project.
    - It would be nice if a first draft is ready for the IETF Hackathon, starting on March 17.
    - Main goals of the project:
        - Design a solid architecture (Openfire plugin or standalone?).
        - Implement the requirements according to the IETF Draft:
            - Vanilla XMPP with Discovery und Publish/Subscribe XEPs.
            - Define Topics/Nodes and manage permissions.

        - Administrative utilities such as purge, list topics, show number of items, identifier etc.
    - Most is already implemented in the form of a proof of concept.
3. Date and time for the next meeting:
    - 2018-02-26, 09:00 in SFFs office

Upcoming absences

no upcoming absences

Meeting 2018-02-26

Attendees

- Fabian Hauser, fhauser
- Raphael Zimmermann, rzimmerm
- Prof. Dr. Andreas Steffen, SFF

Minute Taker: rzimmerm

Agenda

1. Open questions regarding the task
2. Date for oral exam
3. Is there a presentation during the semester?
4. Date and time for the next meeting

Discussions / Decisions

1. Open questions regarding the task
    - Some requirements can influence the chosen architecture fundamentally (e.g. to write a "bot", a component or a plugin)
        - a "normal" user (in the "bot" or the component variant) might have limited access to some node. According to SFF, this issue can be ignored because, in a real-world application, such behaviour should be limited by strict policies.
        - SFF emphasises that the authentication mechanisms use must be robust and certificate based.
    - Next steps:
        - Write User stories
        - Draw basic architecture in C4-Diagrams
        - Risk analysis (e.g. abuse cases)
        - Evaluation of XMPP servers and libraries. SFF notes that we should not spend too much time evaluating the server and assume that OpenFire fulfils most requirements.
    - Issues to address in the XMPP servers and library evaluation:
        - Can an administrator restrict users to create new topics
        - Recommended features can be checked queried using service discovery. We can also check for undesired configurations (e.g. everyone can publish)
        - Ensure that the libraries support all required functionality, especially authentication!
        - Assess the performance and usability of the libraries

    - It is desirable if the service runs is "always on", e.g. to answer subscription requests. However, this is not strictly required according to SFF.
    - Any kind of Interface is conceivable, a web interface, however, is very flexible. The core functionality does not have to be available separately.
    - The scope of the website is fine according to SFF.
2. License
    - GNU-FDL for the documentation is fine
    - The license for the code will be AGPL but might change depending on the frameworks we use
3. Is there a presentation during the semester?
    - An interim with the internal co-examiner will be carried out.
    - The primary purpose of this presentation is to get familiar with the requirements of the co-examiner (testing, documentation, protocols etc.)
    - Should be carried out if a small demo is ready
4. Date for oral exam
    - If possible, the oral exam will be carried out in early July.
    - We will continuously complete parts of the document to reduce the examination efforts
5. Date and time for the next meeting

Upcoming absences

no upcoming absences == Meeting 2018-03-05

Attendees

- Fabian Hauser, fhauser
- Raphael Zimmermann, rzimmerm
- Prof. Dr. Andreas Steffen, SFF

Minute Taker: rzimmerm

Agenda

1. Current State of the Task Description
2. Date and time for the next meeting

Discussions / Decisions

Upcoming absences

no upcoming absences

Declaration of Authorship

We, Fabian HAUSER and Raphael ZIMMERMANN, declare that this thesis and the work presented in it are our own, original work. All the sources we consulted and cited are clearly attributed. We have acknowledged all main sources of help.

Fabian Hauser

Raphael Zimmermann

Rapperswil, February 26, 2018