Document: Functional Requirements

Project: Group Chat For Linphone (Agile DO-178)

Date: May 29, 2015

Patience Mtsweni, Lerato Molokomme, Tsepo Ntsaba, Mpedi Mello, Lutfiyya Razak, Ephiphania Munava

Contents

Figure 1: 3 of our members have been provided with a Zest T1 Android phone.

Linphone group chat project (Agile approach)

1 Group Chat Handling

Use Cases

1.1 Introduction

1.1.1 Purpose

Description: The purpose of this project is to extend Linphone's Instant Messaging (IM) implementation on Android platforms to include group chat and to implement other minor improvements to Linphones IM capabilities and user interface. This project will give the student group exposure to:

- Android development (Java and C)
- Open source contribution
- Cryptography
- Session initiation protocol (SIP)
- DO-178 certification

This project also forms part of a larger Masters study on development methodologies for projects seeking DO-178 certification.

1.1.2 Scope

Description: We are required to develop the following functionality for the Linphone project

- Group chat (Invite additional members to a chat, all members receive chats)
- Secure group chat (AES256)
  - A basic message encryption implementation will be provided
- Creation and deletion of groups
- Voice record and send over IM
- Rework the messaging user interface
  - Spacing between words are terrible
  - Make the text bigger
  - Block indents required to better specify who said what
  - Presence indication to show a remote user is typing
  - User picture portraits

1.1.3 Documentation conventions

Description:DO-178B does not specify the documentation standard to be followed, but most projects do follow some or other documentation standard. The following figure is loosely based on MIL-STD-490A, although sometimes Detail design and Notes are changed into some other topic of discussion.

In the documentation, especially when talking about requirements and specifications, certain words convey additional meaning apart from their linguistic use. These words are usually capitalized.
SHALL and **SHALL NOT** - Indicates a mandatory requirement. **WILL** and **WILL NOT** - Indicates a declaration of purpose or an expression of simple futurity. **SHOULD** and **SHOULD**

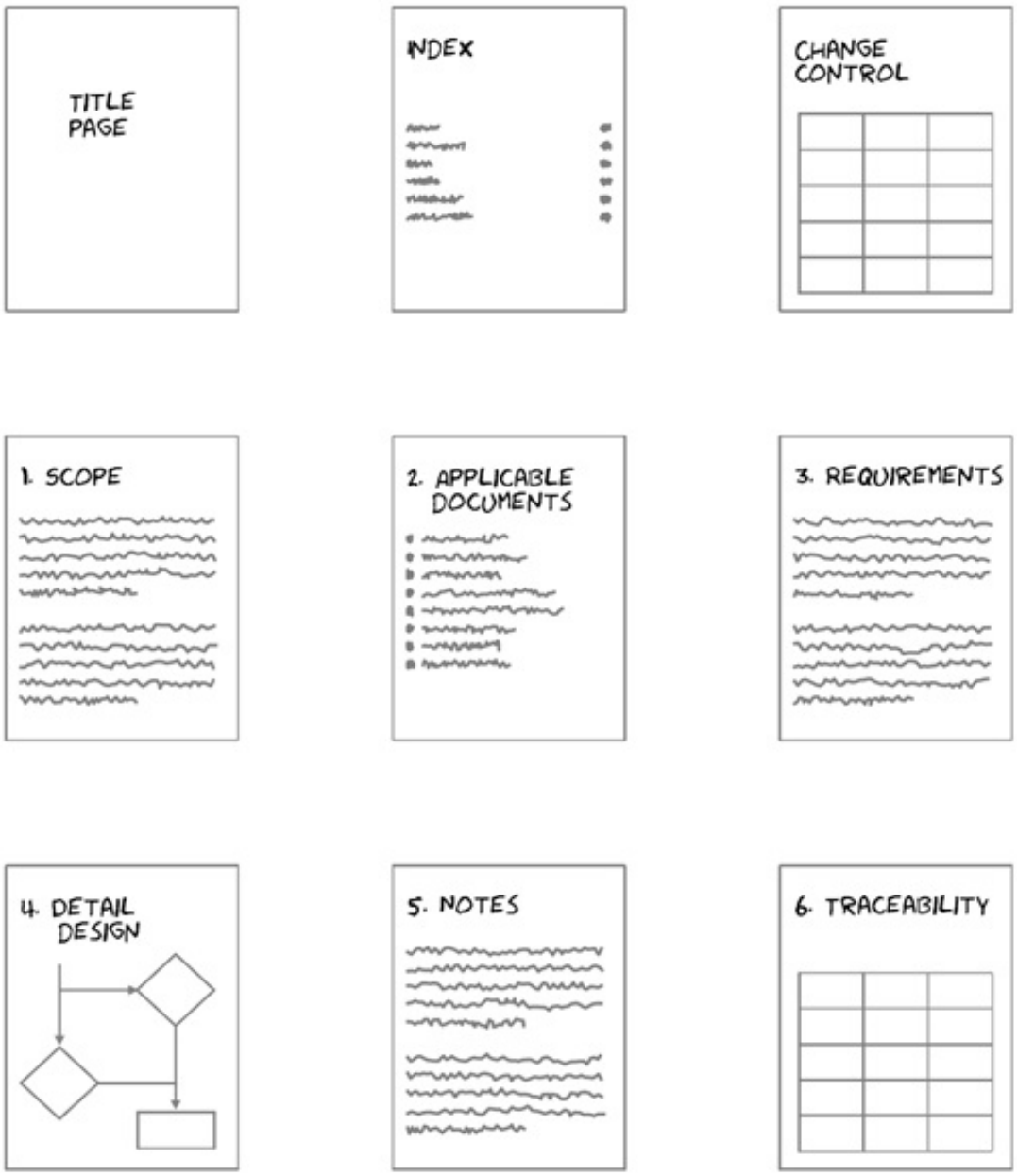

Figure 2: Agile development.

NOT - Indicates a non-mandatory desire, preference or recommendation. **MAY** and **MAY NOT** - Indicates a non-mandatory suggestion or permission. **MUST** and **MUST NOT** should be avoided as it causes confusion with the above terms.

1.1.4 System overview

Server Architecture

1.1.5 System overview

Software Architecture

1.1.6 Certification considerations

The project will be developed according to DO-178 certification requirements, for level D certification. Justification for level D is not dependant on safety considerations as is usually the case, but rather on limited time and the limited experience the team has with DO-178.
This project is not intended for avionics applications and is not safety critical, but rather serves as a case study on DO-178 software development.

1.1.7 Software lifecycle

An agile software development methodology will be followed with this project:

1.1.8 Scrum

During sprint planning and review sessions, we can review and update the Design Description (DD) document detailing the software architecture. At the end of a sprint, the implemented user stories will form the Software Requirements Data (SRDs). It will look something like this

During sprint planning we ensure that the user stories i.e. the high level requirements we are planning to implement is consistent with previous implemented requirements. During sprint review we update the requirements with the implemented user stories, and ensure the created functional tests and unit tests i.e. the low level requirements are consistent with the high level requirements (user stories).

1.1.9 Continuous integration (CI) and Continuous deployment (CD)

The CI and CD servers themselves are effectively the Software Life Cycle Environment Configuration Index (SECI), Software Configuration Index (SCI) and parts of the Sofware Configuration Management Records (SCMR) deliverables. For this to be possible, the CI server must have a copy of the version control database when duplicated for certification.
This means an agile setup might look as follows:

If all the tests pass, the CI / CD will autogenerate a snapshot of itself (VMs or some other duplication means) and the version control database to serve as the SECI and SCI. It will also generate reports of the tests run and their results to serve as the SVP and SVRs, and generate reports that can serve as SCMR items (This is baselines, commit histories etc.).

1.1.10 Test driven development

Test driven development can be used to generate the large parts of the Software Verification Cases and Procedures (SVCPs) and Software Verification Results (SVRs). High level requirements will be developed and tested with feature driven development, and unit tests will be used to develop

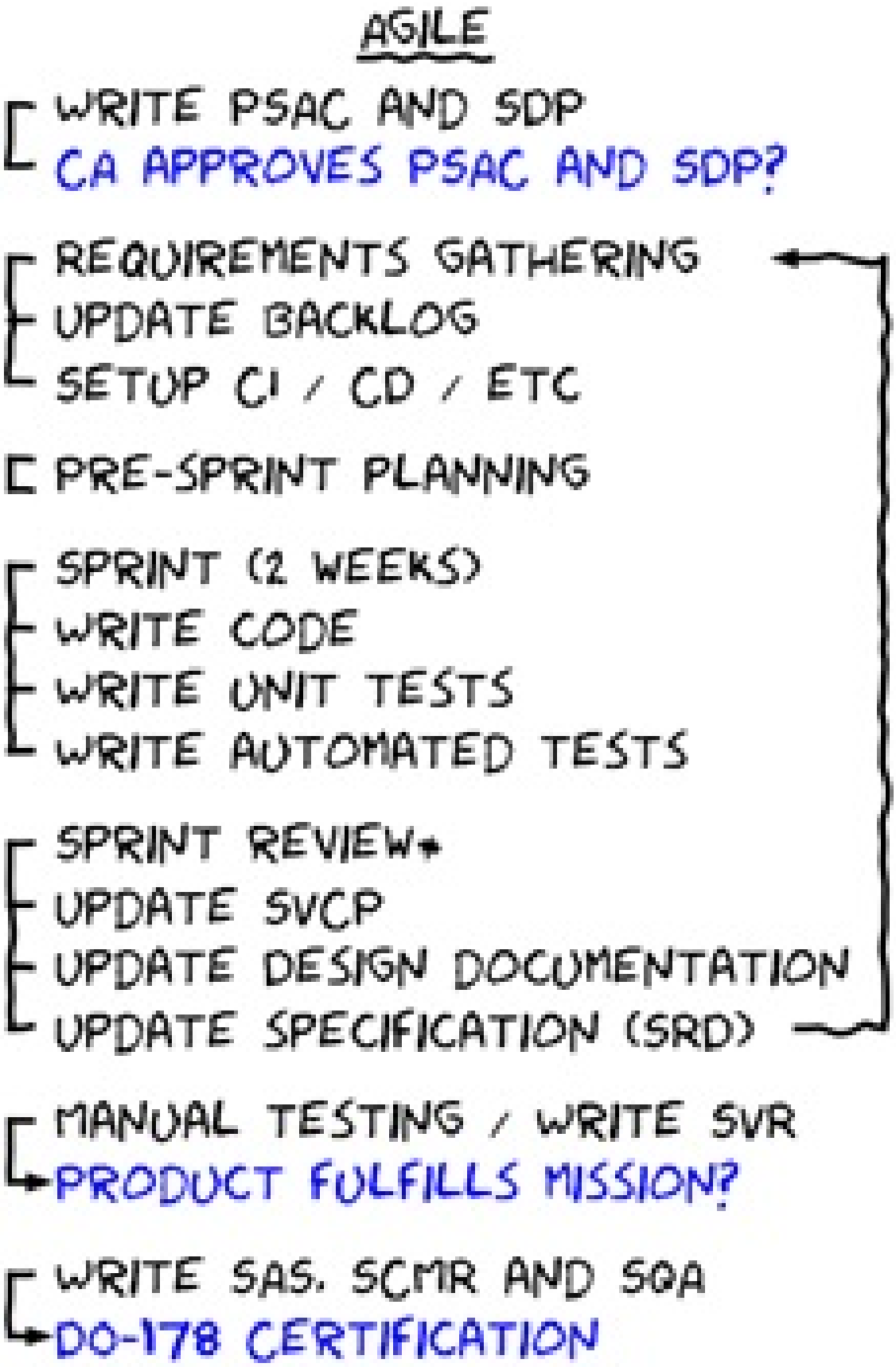

Figure 3: Agile development.

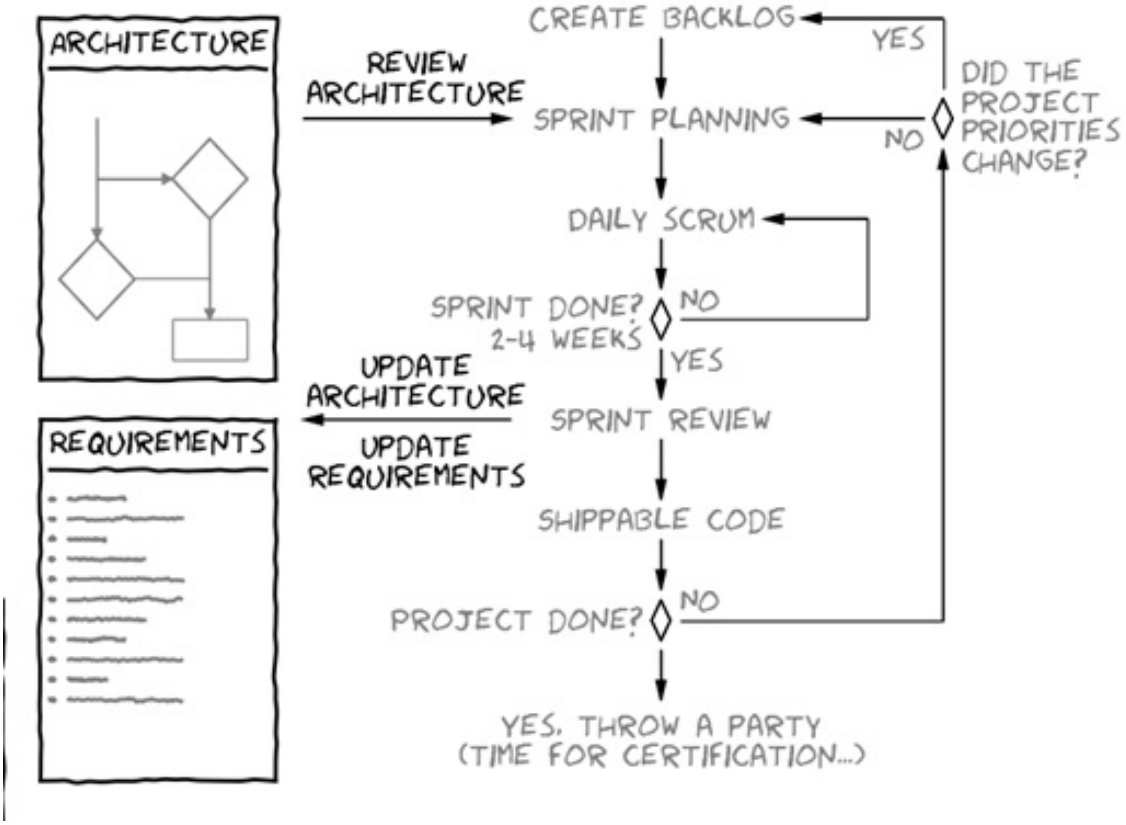

Figure 4: Scrum outputs.

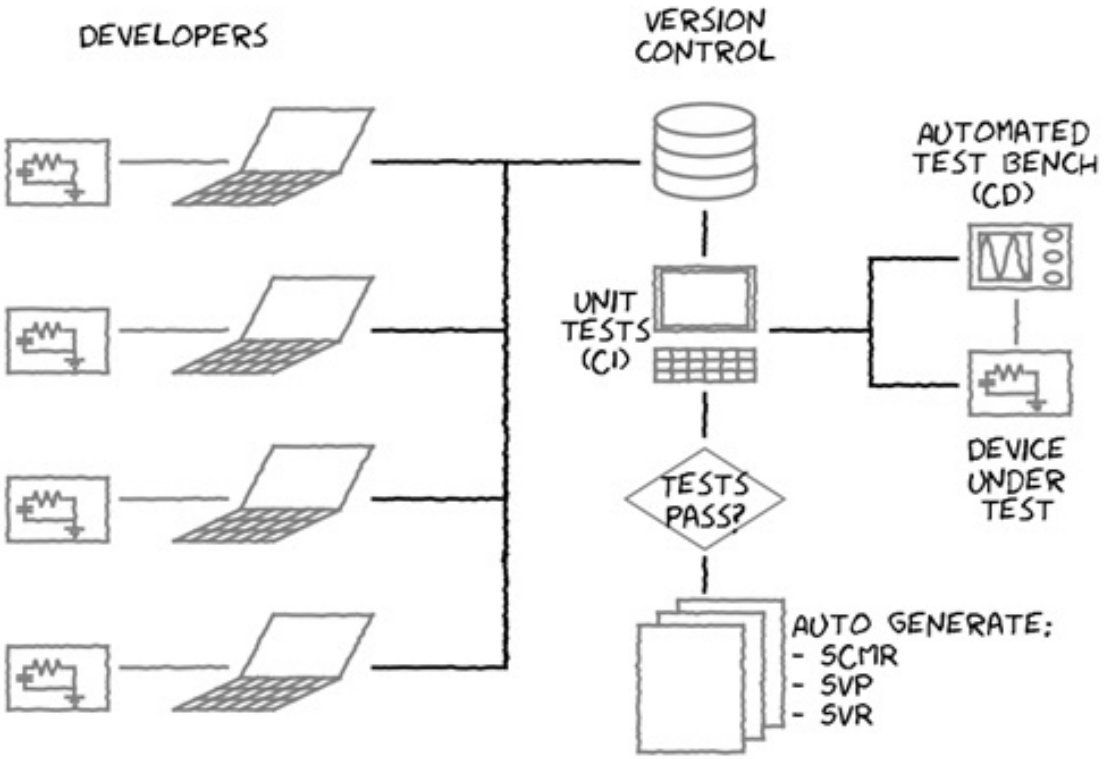

Figure 5: CI and CD outputs.

and test low level requirements. But not all unit tests are really low level requirements, for instance testing if a function can handle null pointer parameters. As such we will mark which unit tests are indeed low level requirements in the unit testing code itself.
The relationship between the Continuous integration (CI) server and the Continuous deployment (CD) server is detailed in the popular test pyramid developed by Mike Cohn, where the CI is responsible for making sure the source code compiles at all times, and passes all unit tests, and the CD is responsible for making sure the automated functional tests pass at all times. One would expect to develop a lot more unit tests than functional tests, thereby limiting (but not eliminating) the need for expensive manual testing.

1.1.11 Software development environment

We will improve and revolve this sections out when we are comfortable in the project with the development environment.

- Eclipse IDE
- Linux environment
- Java for user interface
- C for group chat functionalities
- Unit testing user interface

1.1.12 Software lifecycle data

The following deliverables will be created during this project:

- **Plan for Software Aspects of Certification (PSAC)**
  The Plan for Software Aspects of Certification is the primary means used by the certification authority for determining whether an applicant is proposing a software life cycle that is commensurate with the rigor required for the level of software being developed.
- **Software Development Plan (SDP)**
  The Software Development Plan includes the objectives, standards and software life cycle(s) to be used in the software development processes.
- **Software Verification Plan (SVP)**
  The Software Verification Plan is a description of the verification procedures to satisfy the software verification process objectives.
- **Software Requirements Data (SRD)**
  Software Requirements Data is a definition of the high-level requirements including the derived requirements.
- **Software Design Description (SDD)**
  The Design Description is a definition of the software architecture and the low-level requirements that will satisfy the software high-level requirements.
- **Source Code**
  This data consists of code written in source language(s) and the compiler instructions for generating the object code from the Source Code, and linking and loading data. This data should include the software identification, including the name and date of revision and/or version, as applicable.
- **Executable Object Code**
  The Executable Object Code consists of a form of Source Code that is directly usable by the central processing unit of the target computer and is, therefore, the software that is loaded into the hardware or system.

- **Software Verification Cases and Procedures (SVCP)**
  Software Verification Cases and Procedures detail how the software verification process activities are implemented.

- **Software Verification Results (SVR)**
  The Software Verification Results are produced by the software verification process activities.

- **Software Accomplishment Summary (SAS)**
  The Software Accomplishment Summary is the primary data item for showing compliance with the Plan for Software Aspects of Certification.

1.1.13 Schedule / Software development plan

The automated testing, setup of the CI and CD environment as well as the sprint back log will be implement and completed from the 22 June 2015 to the 5 July 2015 because that is when we conclude our exams and start implementations. Please note that for each story we will setup unit testing and automated testing as discussed later on in the user role table.

Task Description

User Role Each task in the user story is implemented concurrently and sequentially as needed.

Burn Down Chart

1.1.14 Licencing and ownership

The aim is to open-source the project which would be published on Github as forked from Linphone project.

STORY NAME	TASK NUMBER	TASK DESCRIPTION	STATUS	ESTIMATED EFFORT(weeks)	Effort Remaining	Sprint Days	
Automated testing, CI and CD environment setup, sprint back log	0.1	Sprint Back-log	Open	3	3	22-Jun-15	
	0.2	CI & CD Setup	Open	3	3	22-Jun-15	
	0.3	Automated Testing setup	Open	3	3	22-Jun-15	
STORY NAME	**TASK NUMBER**	**TASK DESCRIPTION**	**STATUS**	**ESTIMATED EFFORT(weeks)**	**Effort Remaining**	**Sprint Days**	**Ideal Effort**
GROUP CHAT	1	Create Chat	Open	2	2	5-Jul-15	900
	2	Delete Chat	Open	1	1	12-Jul-15	800
	3	Add and delete users	Open	1	1	19-Jul-15	700
	4	Members Receive Mgs	Open	2	2	2-Aug-15	600
	5	Secure Group Chat	Open	2	2	16-Aug-15	500
INTERFACES	6	Group Chat Interface	Open	3	3	6-Sep-15	400
	7	Change Message Interface	Open	1	1	13-Sep-15	300
RECORDING	8	Voice Recoding	Open	2	2	27-Sep-15	200
	9	Send Recording	Open	1	1	4-Oct-15	100
Demo	10	Testing the system to ensure that no bugs and errors are found	Open	2	2	23-Oct-15	0

Figure 6: Task Description.

USER STORY	TASKS	ASSIGNED TO
I can create a group chat and add at least 1 member, I can select members from my contact list to be a part of the group chat.	Code the create chat function	Ephiphania
	Code the add user function	Lerato
	Design UI	Tsepo
	Unit Test	Lutifiyya
	Automated Test(CI & CD)	Patience
After the group chat is completed I can (as the administrator) delete the chat, At anytime I can remove a member from the group chat	Code the delete chat function	Potego
	Delete Users	Ephiphania
	Unit Test	Lerato
	Automated Test(CI & CD)	Lutifiyya
In the group chat one member sends a message in the group all the other members will receive it	Code send messages to members	Patience
	Code receive messages to members	Potego
	Design UI	Tsepo
	Unit Test	Ephiphania
	Automated Test(CI & CD)	Lerato
The group chats have to be secure.	Secure Group Chat	**Research
As a user I should be able to send and	Send voice recording	Lutifiyya

Figure 7: Task Description.

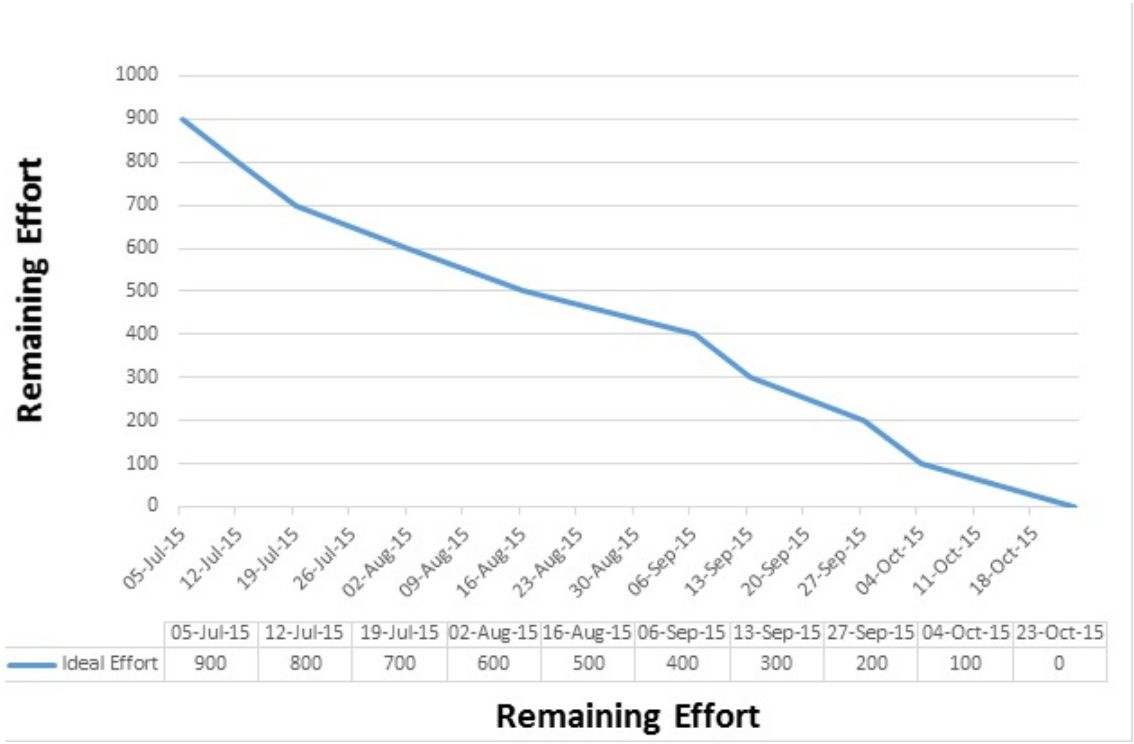

Figure 8: Task Description.