BAOBAB

PLANT US & WE'LL GROW

Document: Plan for Software Aspects of Certification(PSAC)

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

Date: May 29, 2015

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

Member details

Ephiphania Munava	10624610
Lutfiyya Razak	10198408
Lerato Molokomme	11197961
Mpedi Mello	11210754
Patience Mtsweni	11116774
Tsepo Ntsaba	10668544

Git repository link:
https://github.com/Ephiphania/Baobab

Contents

List of Figures

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

1 Introduction

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.

1.2 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

2 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.
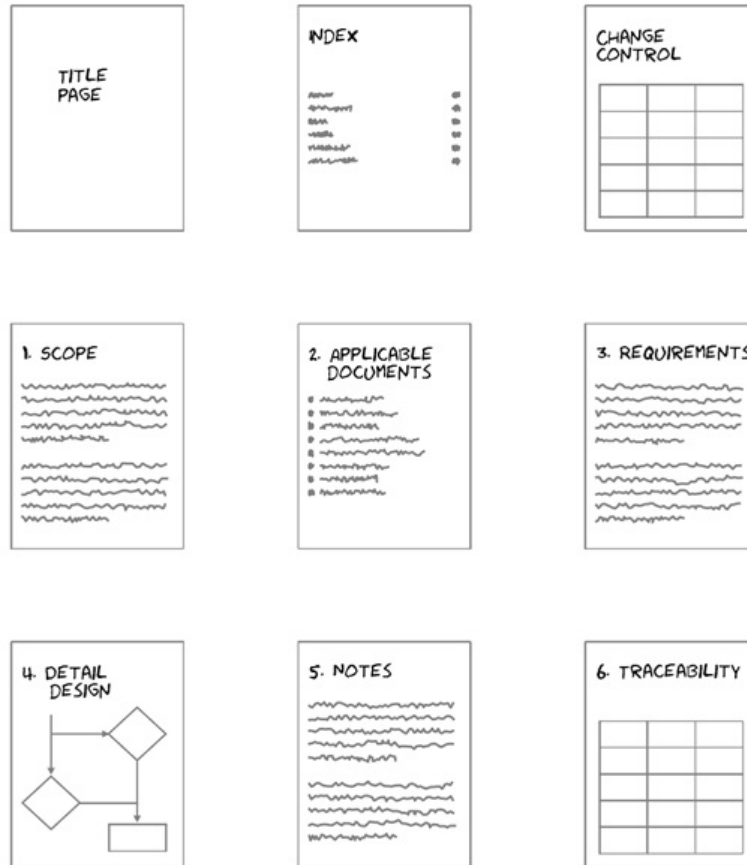
Figure 2: Agile development.

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 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.

3   System overview

3.1   Server Architecture

Linphone is a an IM application with VoIP capabilities based on the SIP (Session Initiation Protocol). The backbone of Linphone is the SIP server (Belle-SIP), we will be looking at the SIP architecture to analyse how it works and how group chat with encryption will be implemented in the Linphone application.

SIP would not be able to function on a network without the use of various devices and protocols. The essential devices are those that you and other participants would use in a conversation, allowing you to communicate with one another, and various servers may also be required to allow the participants to connect together. In addition to this, there are a number of protocols that carry your voice and other data between these computers and devices. Together, they make up the overall architecture of SIP.

3.1.1   SIP Components

There are two fundamental components that are used by the Session Initiation Protocol:

- User agents,which are endpoints of a call or session (i.e.,each of the participants in a call or session).

- SIP servers,which are computers on the network that service requests from clients,and send back responses.

User Agent
The mobile devices, laptops or computers that are being used to make the call and to receive the call or initiate sessions are known as user agents.

User Agents can alternate as a UAC (User Agent Client) and UAS (User agent Server) depending on who initiated the session and who is responding; Because the user agent will send a message, and respond to another, it will alternate back and forth between these roles throughout a session. It will act as both the client and server.

SIP Server
The SIP server is used to resolve usernames to IP addresses, so that requests sent from one user agent (mobile device) to another can be directed properly.

A user agent registers with the SIP server (currently each time the application is opened), providing it with their username and current IP address, thereby establishing their current location on the Linphone network. This also veries that they are online, so that other user agents can see whether theyre available and invite them into a session, basically indicating the user agents presence. Because the user agent probably wouldnt know the IP address of another user agent, a request is made to the SIP server to invite another user into a session.

The SIP server then identies whether the person is currently online, and if so, compares the username to their IP address to determine their location.

In performing these various tasks of serving client requests, the SIP server will act in any of several different roles:

- Registrar server

- Proxy Server

- Redirect Server

Registrar server

Registrar servers are used to register the location of a user agent who has logged onto the Linphone network. Registrar servers play a huge role in providing location services to be used by the Proxy and Redirect servers. The location service is used to keep a database of those who have registered through Belle-SIP server,and where they are located.

When a user agent registers with a Belle-SIP Registrar server, a REGISTER request is made. If the Registrar accepts the request, it will obtain the SIP address and IP address of the user agent, and add it to the location service for its domain.This database provides an up-to-date catalog of everyone who is online, and where they are located.

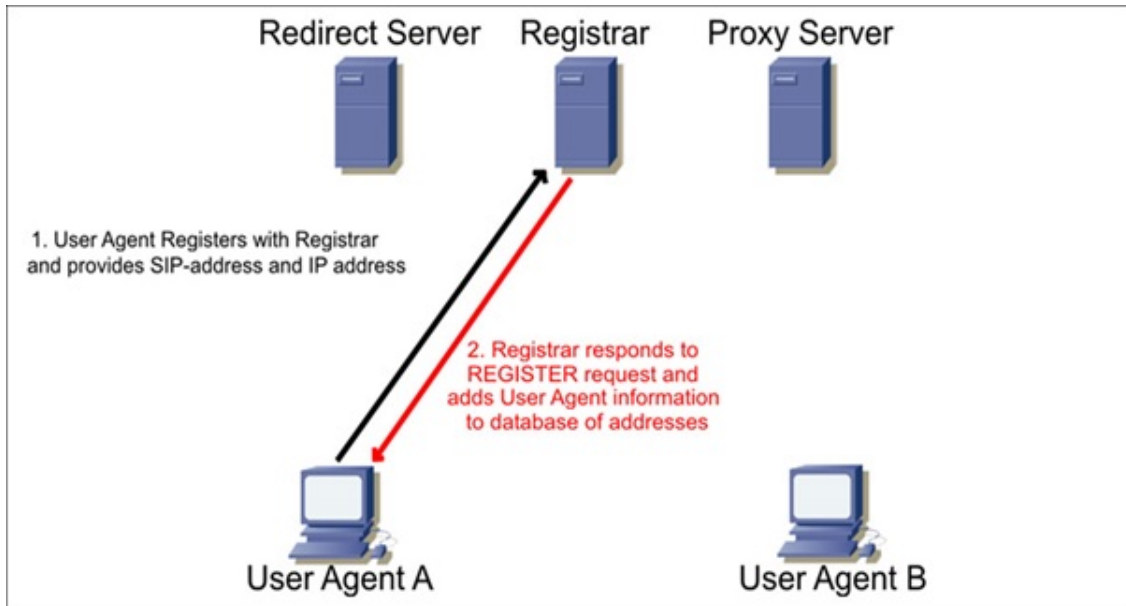Figure 3: Registrar Server

Proxy Server

It is a server that acts as an intermediary for request from clients seeking resources from servers. A client (user agent) connects to the proxy server requesting some service, such as a file. Connection or other resource available from a different server and the proxy server server evaluates the request as a way to simplify and control its complexity.
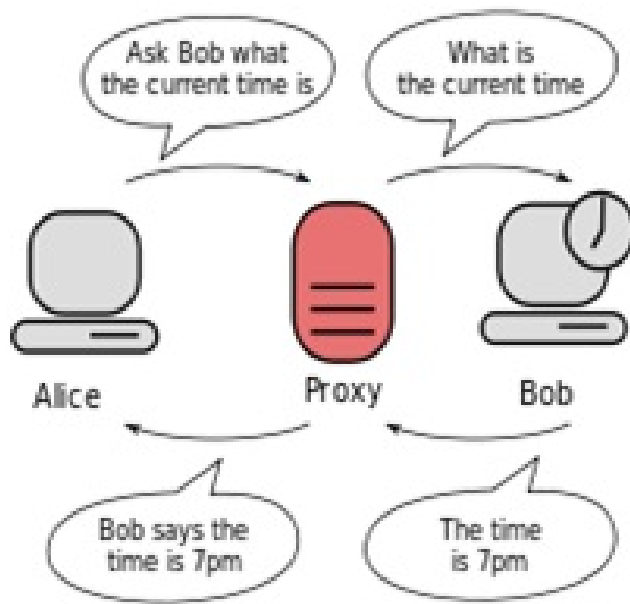
Figure 4: Proxy Server 1

Figure 5: Proxy Server 2

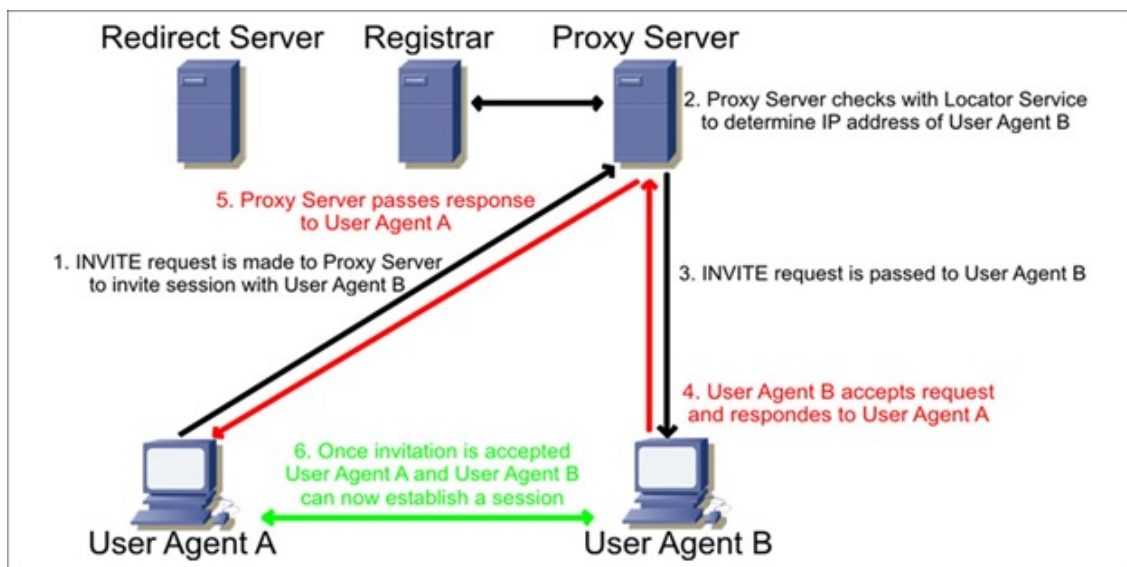**Redirect Server**

The Redirect servers are used by SIP to redirect clients to the user agent they are attempting to contact. If a user agent makes a request, the Redirect server can respond with the IP address of the user agent being contacted. This is different from a Proxy server, which forwards the request on your behalf, as the Redirect server essentially tells you to contact them yourself.
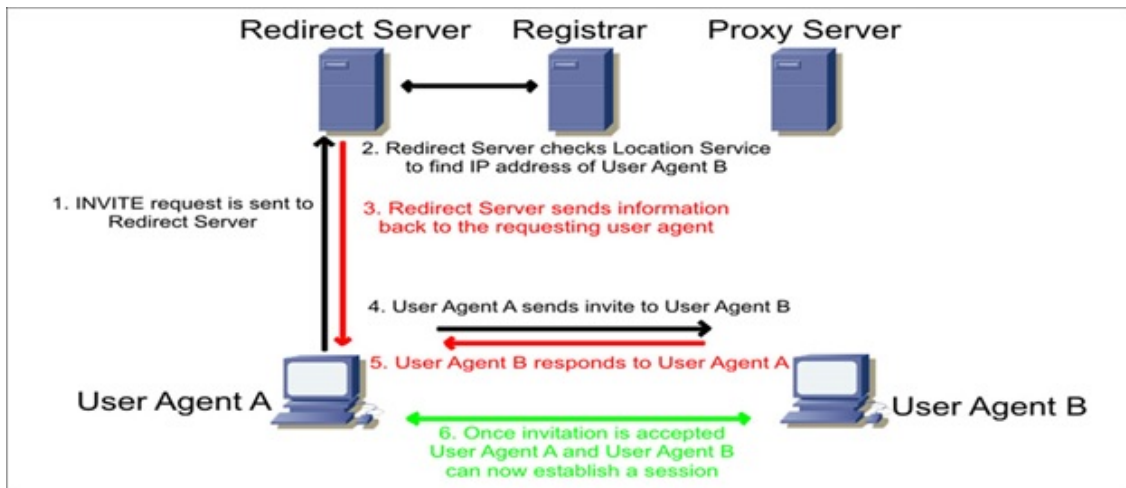
Figure 6: Redirect Server1

3.1.2 Client/Server and Peer-to-Peer(P2P) Architecture

The Session Initiation Protocol uses two different types of architectures in network communications, both the:

- Client/Server Architecture

- Peer-to-Peer (P2P) Architecture

Client/Server Architecture

- Client - requests specific services or resources

- Server- dedicated to fulfilling requests by responding (or attempting to respond) with requested services or resources.

In VoIP, a client (mobile device) sends a request to register with a Registrar server, or makes a request to a Proxy Server or Redirect Server that allows it to connect with another user agent. In all these cases, the clients role is to request services and resources, and the servers role is to listen to the network and await requests that it can process or pass onto other servers.

A server may run a number of services or have multiple server applications installed on it, a computer dedicated to the role of being a server may provide several functions on a network. For example, a Web server might also act as an e-mail server. In the same way, SIP servers also may provide different services. A Registrar can register clients and also run the location service that allows clients and other servers to locate other users who have registered on the network. In this way, a single server may provide diverse functionality to a network that would otherwise be unavailable.
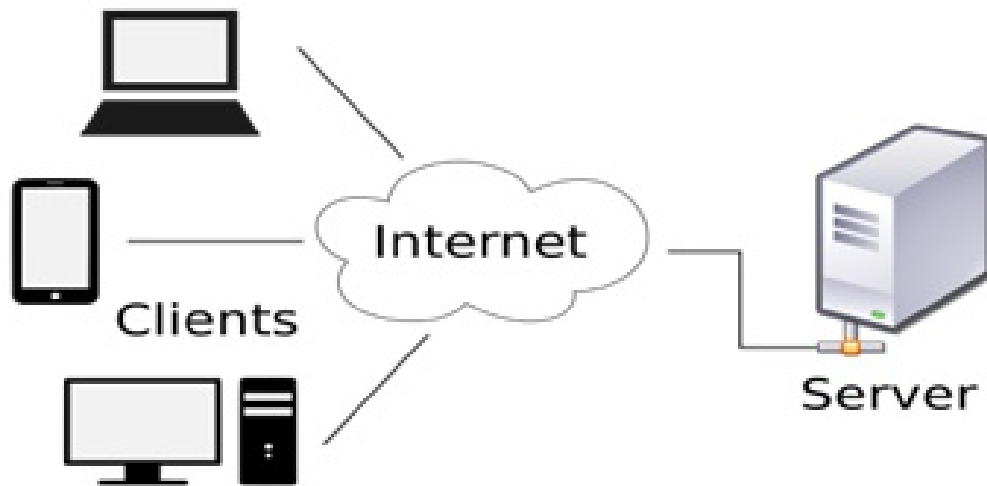
9

Figure 7: Client/Server Architecture

Peer-to-Peer Architecture

Once the user agents have completed registering themselves, and making requests and receiving responses on the location of the user agent they wish to contact, the architecture changes from one of client/server to that of peer-to-peer (P2P). In a P2P architecture, user agents act as both clients who request resources, and servers that respond to those requests and provide resources. Because resources arent located on a single machine or a small group of machines acting as network servers, this type of network is also referred to as being decentralized.

When user agents have initiated a session with one another, they become UAS (User Agent Clients) and UAS (User Agent Servers) to one another, and have the ability to invite additional participants into the session. Each of these User agents can communicate with one another in an audio or group chat. If one of these participants ends the session, or is using a device that fails during the communication, the other participants can continue as if nothing happened. This architecture makes communication between User agents stable, without having to worry about the network failing if one computer or device suddenly becomes unavailable.
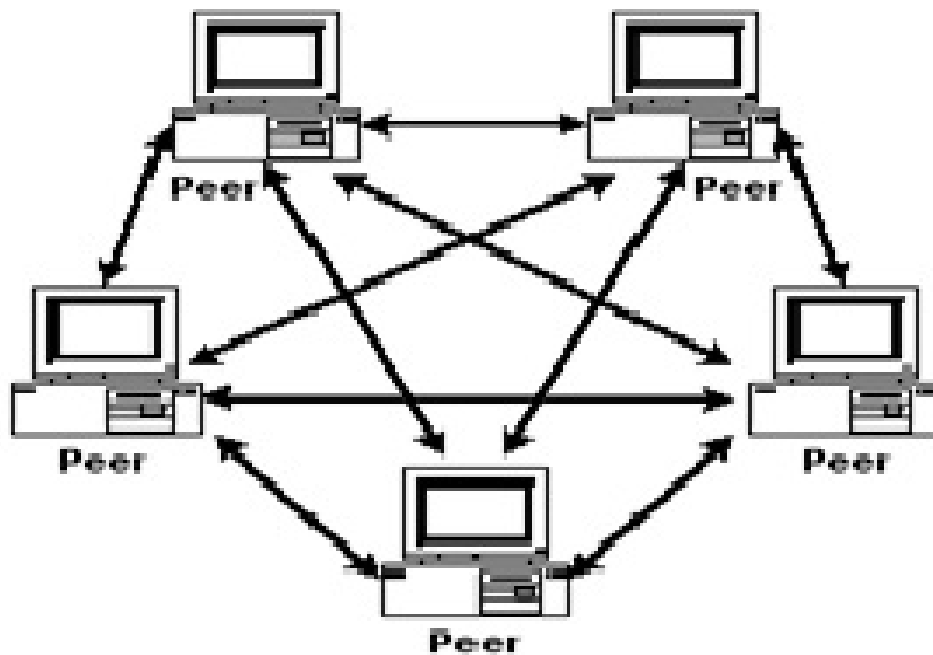
Figure 8: Peer-to-Peer Architecture

3.1.3   Protocols used with SIP

SIP works with different protocols at different stages of communication to pass data between servers, devices and participants. Without these protocols, communication would be impossible or insecure. A few protocols that will be used are User Datagram Protocol (UDP), Transport Layer Security (TLS).

User Datagram Protocol (UDP)
UDP is used to transport units of data called datagrams over an IP network. UDP is faster than Transmission Control Protocol (TCP) because there is less processing of data. It is also used because when messages with small amounts of data are being sent across the network or when data will be unaffected by a few units of missing data.

UDP shall be used in Linphones group chat implementation because it is fast for instant messaging and with Voice over IP when streaming a video, a minor loss of data will be a minor fault that generally would not affect the overall quality or performance. In this case, it is more important that data is passed quickly from one endpoint to another.

Transport Layer Security (TLS)
TLS is used with other protocols like UDP to provide security between applications communicating over an IP network. TLS uses encryption to ensure privacy, so that third parties do not tamper with the messages being sent. A secure connection is established by authenticating the client and server or the User Agent Client and User Agent Server and then encrypting the connection between them. TLS is a multi-layered protocol that consists of a TLS Handshake Protocol and a TLS Record Protocol.

The TLS Handshake protocol is used to authenticate the participants of the communication and negotiate an encryption algorithm. The client and server will have to agree upon an encryption method and prove who they are, using cryptographic keys before any data is sent between them.

11

Once this has been done successfully, a secure channel is established between them. Once the TLS Handshake Protocol is used, the TLS Record Protocol ensures security that data is not modified and others cannot access the data while in transit.

TLS may be used in Linphones implementation for a group chat to ensure that the messages are not modified and that it is securely transmitted to the other participants within the group chat therefore the messages will be encrypted.

Session Description Protocol (SDP)
SDP is used to send description information that is necessary when sending multimedia data across the network. It also provides information on what multimedia a user agent is requesting to be used and other information that is necessary in setting up the transfer of the data.

SDP does not deliver media itself but is used between end points for negotiation of media type, format, and all associated properties. It contains the name and purpose of the session, the time the session is active, etc.

Real-Time Transport Protocol (RTP)
RTP is used to transport real-time data across a network. It manages the transmission of multimedia over the IP network with SIP. Information in the packet header tells the receiving user agent how the data should be reconstructed.

RTP provides some reliability in the data sent between user agents. RTP uses the Real-Time Control Protocol which allows the receiving user agent to detect if there is packet loss and allows it to compensate for any delays that might occur as data is transport across the network.

RTP is used by Linphone for delivering data such as voice and video over IP network and will also be implemented for Linphones group chat.

Media Gateway Control Protocol (MGCP)
MGCP is used to control gateways that provide access to the Public Switched Telephone Network (PSTN). This protocol provides a method for communication on a network that supports voice calls between PSTN-IP or IP-IP networks. It is used to convert the data from a format that is used on PSTN to one that is used by IP networks. This protocol is used to set up, manage and terminate calls between endpoints.

MGCP may be used for Linphones group chat implementation where users will be able to send calls to other users.

Real-Time Streaming Protocol (RTSP)
RTSP is used to control the delivery of streaming media across the network. It provides the ability to control streaming media. Through this protocol, an application can issue commands to play, pause or perform other actions that affect the paying of media being transferred to the application.

4 Software overview

4.1 Software Architecture

Linphone uses a layered structure with separation between the user interface and the core engine (business logic) of the application. The architecture used by Linphone is depicted in the following diagram.
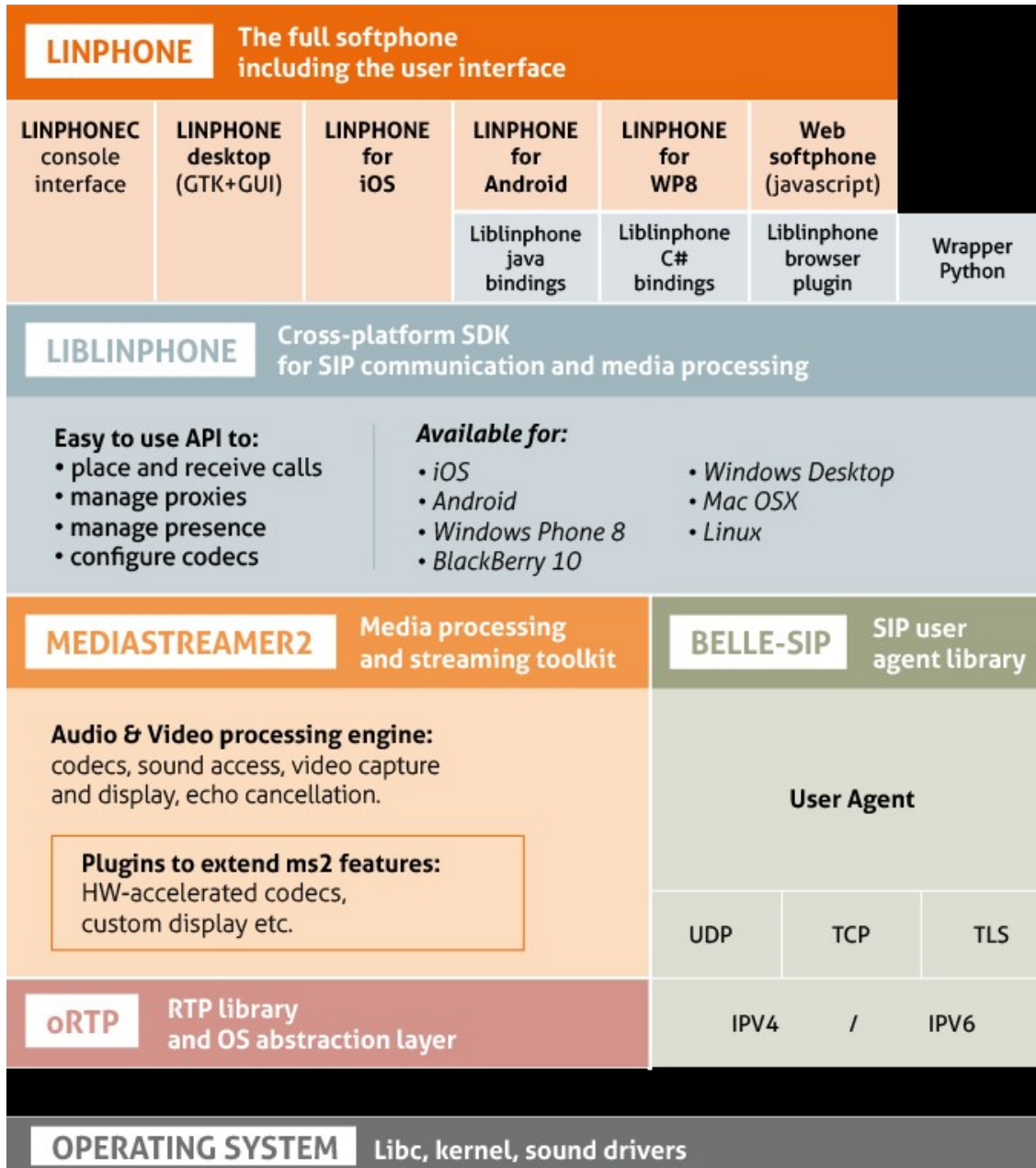

Figure 9: Linphone Architecture

Liblinphone is the core engine library written in objective C that provides all the functionalities of Linphone. The group chat functionality will be added on top of the Liblinphone stack, making

use of Liblinphone Java bindings to expose the C API in a Java friendly manner. The current package/document structure of the project looks like the following:

▷ AndroidTest
▷ AudioDeviceAndroidTest
▷ **linphone-android**
  – submodules/linphone/mediastreamer2/java/src
  – submodules/linphone/java/j2se
  – submodules/linphone/java/common
  – submodules/linphone/impl
  – submodules/externals/axmlrpc/src/main/java
  – submodules/linphone/coreapi/help/java
  – ...
  – ...
  – **src**
  – ...
  – ...
  – AndroidManifest.xml
  – Makefile
▷ Mediastreamer
▷ NativeActivity
▷ VideoCaptureModuleAndroidTest
▷ ViEAutotest
▷ WebRTCDemo

The group chat code will be added in the linphone-android package under the src/ directory (linphone-android/src).

We will use LinphoneChatRoom interface to implement the group chat. LinphoneChatMessage will include createLinphoneChatMessage which will create a LinphoneChat. LinphoneCore : ge Core() This will return a back pointer to the core managing the chat room. LinphoneChatMessage.State This class will show the state of an object. For example: Whether the file was transferred or not and whether the message was derived or not. It will keep track of certain actions that take place and track the progress.

5 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.

6 Software lifecycle

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

AGILE

⌈ WRITE PSAC AND SDP
⌊ CA APPROVES PSAC AND SDP?

⌈ REQUIREMENTS GATHERING
⊦ UPDATE BACKLOG
⌊ SETUP CI / CD / ETC

⌈ PRE-SPRINT PLANNING

⌈ SPRINT (2 WEEKS)
⊦ WRITE CODE
⊦ WRITE UNIT TESTS
⌊ WRITE AUTOMATED TESTS

⌈ SPRINT REVIEW
⊦ UPDATE SVCP
⊦ UPDATE DESIGN DOCUMENTATION
⌊ UPDATE SPECIFICATION (SRD)

⌈ MANUAL TESTING / WRITE SVR
⌊ PRODUCT FULFILLS MISSION?

⌈ WRITE SAS, SCMR AND SQA
⌊ DO-178 CERTIFICATION

Figure 10: Agile development

6.1 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 (SRD's). It will look something like this:
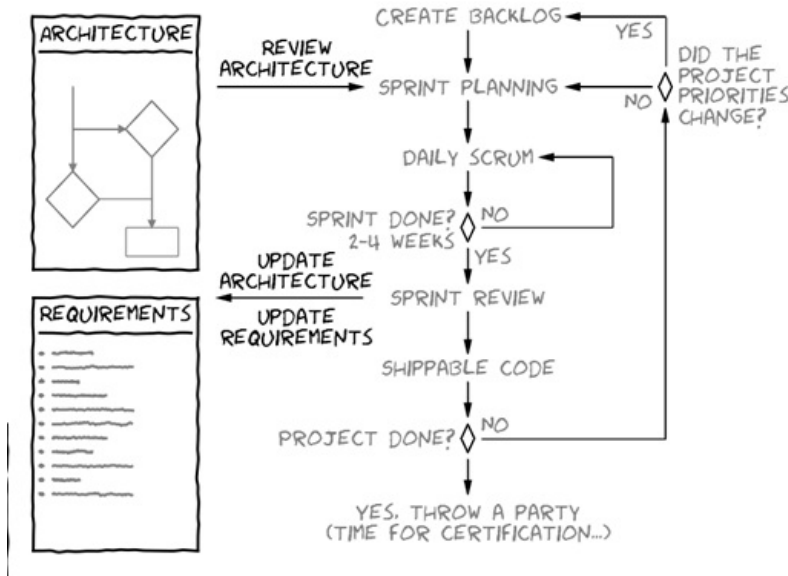
Figure 11: Scrum

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).

6.2 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.

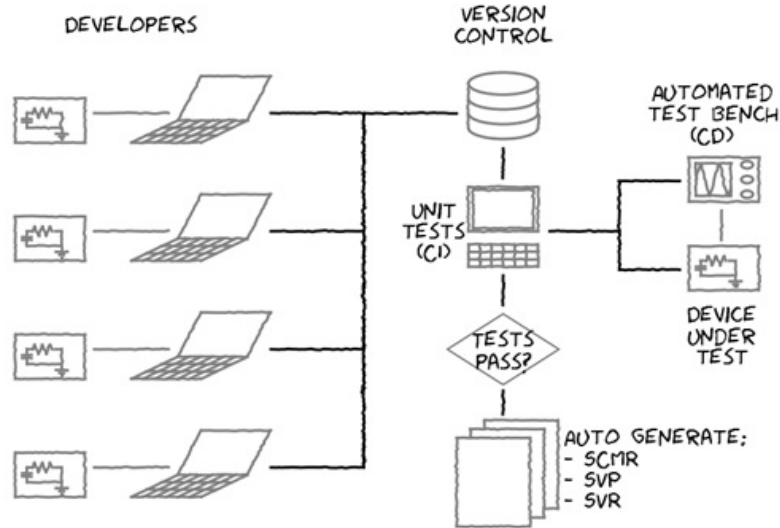This means an agile setup might look as follows:

Figure 12: CI and CD outputs

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.).

6.3   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 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.

7 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 functionality

- JUnit to test the Java code

- CUnit to test the C code

8 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.

9 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.

9.1 Task Description

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 13: Task Description

9.2 User Roles

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

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 14: Task Description with roles
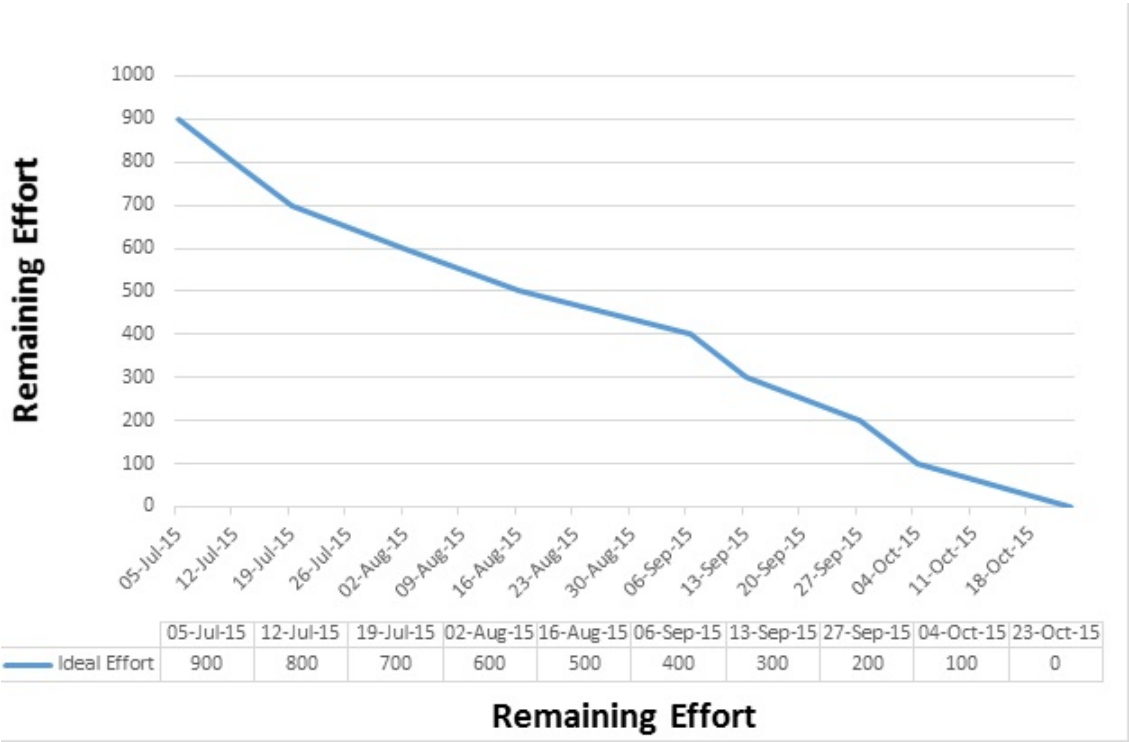
21

9.3   Burn Down Chart

Figure 15: Burn-Down Chart

10 Licencing and ownership

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