

VISVESVARAYA TECHNOLOGICAL UNIVERSITY  
JNANASANGAMA, BELAGAVI - 590018



## “TAIPAN”

Thesis submitted in partial fulfillment of the curriculum prescribed for  
the award of the degree of Bachelor of Engineering in  
Computer Science & Engineering by

1CR14CS139      Simran Sarawagi  
1CR14CS141      Sourabh Pandey

Under the Guidance of

Mrs. Sagarika Behera  
Associate Professor  
Department of CSE, CMRIT, Bengaluru



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING  
#132, AECS LAYOUT, IT PARK ROAD, BENGALURU - 560037

2017-18

VISVESVARAYA TECHNOLOGICAL UNIVERSITY  
JNANASANGAMA, BELAGAVI - 590018



## Certificate

This is to certify that the project entitled “TAIPAN” is a bonafide work carried out by **Simran Sarawagi** bearing **USN:1CR14CS139** , **Sourabh Pandey** bearing **USN:1CR14CS141** in partial fulfillment of the award of the degree of Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belgaum, during the year 2017-18. It is certified that all corrections / suggestions indicated during reviews have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

----- Signature of Guide <b>Mrs. Sagarika Behera</b> Associate Professor Department of CSE CMRIT, Bengaluru - 37	----- Signature of HoD <b>Dr. Jhansi Rani P</b> Professor & Head Department of CSE CMRIT, Bengaluru - 37	----- Signature of Principal <b>Dr. Sanjay Jain</b> Principal CMRIT, Bengaluru - 37
---	---	--

### External Viva

Name of the Examiners	Institution	Signature with Date
1. -----	-----	-----
2. -----	-----	-----

# Acknowledgement

We take this opportunity to thank all of those who have generously helped us to give a proper shape to our work and complete our BE project successfully. A successful project is fruitful culmination efforts by many people, some directly involved and some others indirectly, by providing support and encouragement.

We would like to thank **Dr. SANJAY JAIN** , Principal , CMRIT , for providing excellent academic environment in the college.

We would like to express our gratitude towards **Dr. JHANSI RANI** , Professor & HOD , Dept of CSE , CMRIT , who provided guidance and gave valuable suggestions regarding the project.

We consider it a privilege and honour to express our sincere gratitude to our Internal Guide **Mrs. SAGARIKA BEHERA** , Associate Professor , Department of Computer Science & Engineering , CMRIT , for her valuable guidance throughout the tenure of this project work.

Simran Sarawagi  
Sourabh Pandey

# Table of Contents

Table of Contents	ii
List of Figures	iv
List of Tables	v
Abstract	vi
<b>1 PREAMBLE</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Problem Statement . . . . .	1
1.3 Existing System . . . . .	2
1.4 What Wireshark is Not . . . . .	2
1.5 Some Intended Purpose . . . . .	2
1.6 Features . . . . .	3
1.7 Proposed System . . . . .	3
<b>2 LITERATURE SURVEY</b>	<b>5</b>
2.1 INTRODUCTION . . . . .	5
2.2 LITERATURE SURVEY . . . . .	5
2.3 PAPER 1 . . . . .	6
2.4 PAPER 2 . . . . .	6
2.5 PAPER 3 . . . . .	6
2.6 PAPER 4 . . . . .	7
<b>3 THEORETICAL BACKGROUND</b>	<b>8</b>
3.1 INTRODUCTION . . . . .	8
3.2 XMLTODICT . . . . .	8
3.3 BYTEARRAY . . . . .	9
3.4 STRUCT.UNPACK() . . . . .	10
<b>4 SYSTEM REQUIREMENT SPECIFICATION</b>	<b>11</b>

4.1	INTRODUCTION . . . . .	11
4.2	FUNCTIONAL REQUIREMENTS . . . . .	12
4.3	NON-FUNCTIONAL REQUIREMENTS . . . . .	12
4.4	HARDWARE REQUIREMENTS . . . . .	15
4.5	SOFTWARE REQUIREMENTS . . . . .	15
4.6	SOFTWARE QUALITY ATTRIBUTES . . . . .	16
<b>5</b>	<b>SYSTEM DESIGN</b>	<b>17</b>
5.1	INTRODUCTION . . . . .	17
5.2	SYSTEM DEVELOPMENT METHODOLOGY . . . . .	17
5.3	DESIGN USING UML . . . . .	19
5.4	DATA FLOW DIAGRAM . . . . .	20
5.5	CLASS DIAGRAM . . . . .	21
5.6	USE CASE DIAGRAM . . . . .	21
5.7	ACTIVITY DIAGRAM . . . . .	22
5.8	SEQUENCE DIAGRAM . . . . .	22
<b>6</b>	<b>IMPLEMENTATION</b>	<b>25</b>
6.1	INTRODUCTION . . . . .	25
6.2	Generating Binary File . . . . .	25
6.3	Display . . . . .	27
6.4	Reading XML File . . . . .	30
6.5	Convert . . . . .	33
6.6	Show . . . . .	35
<b>7</b>	<b>TESTING AND RESULTS</b>	<b>37</b>
7.1	INTRODUCTION . . . . .	37
7.2	TESTING METHODOLOGIES . . . . .	37
7.3	RESULTS . . . . .	39
<b>8</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>41</b>
8.1	CONCLUSION . . . . .	42
8.2	FUTURE SCOPE . . . . .	42
8.3	BUILD A COMPLETE SYSTEM . . . . .	42
8.4	FUTURE WORK . . . . .	42
	<b>References</b>	<b>43</b>
<b>9</b>	<b>APPENDIX</b>	<b>44</b>

# List of Figures

1.1	General Wireshark Dissector . . . . .	1
1.2	Wireshark Architecture . . . . .	2
3.1	Accessing Elements . . . . .	8
3.2	Source Parameter . . . . .	9
4.1	Software Quality Attributes . . . . .	16
5.1	Waterfall Model . . . . .	19
5.2	Class Diagram . . . . .	21
5.3	Use Case Diagram . . . . .	22
5.4	Activity Diagram . . . . .	23
5.5	Sequence Diagram . . . . .	24

# List of Tables

5.1	Symbols used in UML . . . . .	20
7.1	Unit Test Cases . . . . .	39

# Abstract

Dissectors are used for network troubleshooting, analysis, software and communications protocol development, and education. It is a data capturing program that "understands" the structure (encapsulation) of different networking protocols. It can parse and display the fields, along with their meanings as specified by different networking protocols. The existing model for dissector is Wireshark Dissector. Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. Each dissector decodes its part of the protocol, and then hands off decoding to subsequent dissectors for an encapsulated protocol.

Taipan is a python based tool. It is used for data conversion, encapsulation, information security and integrity. It is a tool used for converting the information into binary form restricting its access to normal users. It also defines the structure of actual information, known as metadata which helps user to get access on the real data. It can also be used to render the exact information back to any authorized user with the help of the available the structure. This provides confidentiality, integrity and authenticity for the recorded information.

The aim of this project is to present a design of a simple, tool for information security and packing. It can parse and display the fields, along with their meanings as specified by different networking protocols.



# Chapter 1

## PREAMBLE

### 1.1 Introduction

Dissectors are used for network troubleshooting, analysis, software and communications protocol development, and education. It is a data capturing program that "understands" the structure (encapsulation) of different networking protocols. It can parse and display the fields, along with their meanings as specified by different networking protocols. Each dissector decodes its part of the protocol, and then hands off decoding to subsequent dissectors for an encapsulated protocol.

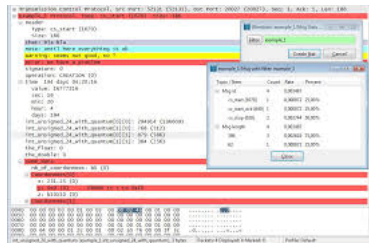


Figure 1.1: General Wireshark Dissector

The existing model for dissector is Wireshark Dissector. Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible.

### 1.2 Problem Statement

- Implementation of Packet Sniffer.
- Program should identify header of each protocol.
- Use multi-core programming.
- Filtering of Packets.

- Importing the captured packets.
- Detecting abnormal error in networking.

### 1.3 Existing System

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. You could think of a network packet analyzer as a measuring device used to examine whats going on inside a network cable, just like a voltmeter is used by an electrician to examine whats going on inside an electric cable (but at a higher level, of course). In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all that has changed. Wireshark is perhaps one of the best open source packet analyzers available today.

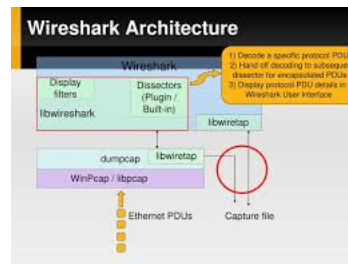


Figure 1.2: Wireshark Architecture

### 1.4 What Wireshark is Not

Here are some things Wireshark does not provide:

- Wireshark isnt an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isnt allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- Wireshark will not manipulate things on the network, it will only measure things from it. Wireshark doesnt send packets on the network or do other active things (except for name resolutions, but even that can be disabled).

### 1.5 Some Intended Purpose

Here are some examples people use Wireshark for:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- QA engineers use it to verify network applications
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals

Beside these examples Wireshark can be helpful in many other situations too.

## 1.6 Features

The following are some of the many features Wireshark provides:

- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
- Save packet data captured.
- Export some or all packets in a number of capture file formats.
- Filter packets on many criteria.
- Search for packets on many criteria.
- Colorize packet display based on filters.
- Create various statistics.

## 1.7 Proposed System

Dissector is simply a protocol parser. Wireshark contains dozens of protocol dissectors for the most popular network protocols. In case when some dissector needs to be adjusted or creation of completely new protocol dissector is desired, knowledge of dissector creation procedure might be very useful. Each dissector decodes its part of the protocol, and then hands off decoding to subsequent dissectors for an encapsulated

protocol. Every dissection starts with the Frame dissector which dissects the packet details of the capture file itself (e.g. timestamps). From there it passes the data on to the lowest-level data dissector, e.g. the Ethernet dissector for the Ethernet header. The payload is then passed on to the next dissector (e.g. IP) and so on. At each stage, details of the packet will be decoded and displayed. The amount of resources dissectors needs depends on your environment and on the size of the capture file you are analyzing. Larger capture files will require more memory and disk space. If a dissector runs out of memory it will crash.

# Chapter 2

## LITERATURE SURVEY

### 2.1 INTRODUCTION

Literature survey is mainly carried out in order to analyze the background of the current project which helps to find out flaws in the existing system and guides on which unsolved problems we can work out. So, the following topics not only illustrate the background of the project but also uncover the problems and flaws which motivated to propose solutions and work on this project.

### 2.2 LITERATURE SURVEY

Literature survey is the documentation of a comprehensive review of the published and unpublished work from secondary sources data in the areas of specific interest to the researcher. The library is a rich storage base for secondary data and researchers used to spend several weeks and sometimes months going through books, journals, newspapers, magazines, conference proceedings, doctoral dissertations, master's theses, government publications and financial reports to find information on their research topic. Reviewing the literature on the topic area at this time helps the researcher to focus further interviews more meaningfully on certain aspects found to be important in the published studies even if these had not surfaced during the earlier questioning. So the literature survey is important for gathering the secondary data for the research which might be proved very helpful in the research. The literature survey can be conducted for several reasons. The literature review can be in any area of the business.

## 2.3 PAPER 1

**Title:** Capturing Live Data Network

**Context:**

Capturing live network data is one of the major features of the dissector. The dissector capture engine provides the following features:

- Capture from different kinds of network hardware such as Ethernet or 802.11.
- Stop the capture on different triggers such as the amount of captured data, elapsed time, or the number of packets. Simultaneously show decoded packets while dissector is capturing. Filter packets, reducing the amount of data to be captured.
- Save packets in multiple files while doing a long term capture, optionally rotating through a fixed number of files. Simultaneously capture from multiple network interfaces.

The capture engine still lacks the following features:

- Stop capturing (or perform some other action) depending on the captured data.

## 2.4 PAPER 2

**Title:** Viewing Captured Files

**Context:**

Once you have captured some packets or you have opened a previously saved capture file, you can view the packets that are displayed in the packet list pane by simply clicking on a packet in the packet list pane, which will bring up the selected packet in the tree view and byte view panes. You can then expand any part of the tree to view detailed information about each protocol in each packet. Clicking on an item in the tree will highlight the corresponding bytes in the byte view. It also has the Acknowledgement number in the TCP header selected, which shows up in the byte view as the selected bytes.

## 2.5 PAPER 3

**Title:** TCP Analysis

**Context:**

By default, Wireshark's TCP dissector tracks the state of each TCP session and provides additional information when problems or potential problems are detected. Analysis is done once for each TCP packet when a capture file is first opened. Packets are processed in the order in which they appear in the packet list. You can enable or disable this feature via the Analyze TCP sequence numbers TCP dissector preference.

## 2.6 PAPER 4

**Title:** Packet Reassembly

**Context:**

Network protocols often need to transport large chunks of data which are complete in themselves, e.g. when transferring a file. The underlying protocol might not be able to handle that chunk size (e.g. limitation of the network packet size), or is stream-based like TCP, which doesn't know data chunks at all. In that case the network protocol has to handle the chunk boundaries itself and (if required) spread the data over multiple packets. It obviously also needs a mechanism to determine the chunk boundaries on the receiving side. Wireshark calls this mechanism reassembly, although a specific protocol specification might use a different term for this (e.g. desegmentation, defragmentation, etc).

# Chapter 3

## THEORETICAL BACKGROUND

### 3.1 INTRODUCTION

### 3.2 XMLTODICT

Xmltodict is another simple library that aims at making XML feel like working with JSON. An XML file like this:

```
<mydocument has="an attribute">
  <and>
    <many>elements </many>
    <many>more elements </many>
  </and>
  <plus a="complex">
    element as well
  </plus>
</mydocument>
```

can be loaded into a Python dict like this: `import xmltodict with open('path/to/file.xml')`  
as `fd: doc = xmltodict.parse(fd.read())`

and then you can access elements, attributes and values like this:

```
doc['mydocument']['@has'] # == u'an attribute'
doc['mydocument']['and']['many'] # == [u'elements', u'more
elements']
doc['mydocument']['plus']['@a'] # == u'complex'
doc['mydocument']['plus']['#text'] # == u'element as well'
```

Figure 3.1: Accessing Elements

xmltodict also lets you roundtrip back to XML with the `unparse` function, has a streaming mode suitable for handling files that dont fit in memory and supports



namespaces.

### 3.3 BYTEARRAY

The `bytearray()` method returns a `bytearray` object which is an array of the given bytes. The syntax of `bytearray()` method is: `bytearray([source[, encoding[, errors]])`  
 The `bytearray()` method returns a `bytearray` object which is a mutable (can be modified) sequence of integers in the range  $0 \leq x < 256$ . If you want the immutable version, use `bytes()` method.

#### 3.3.1 BYTEARRAY PARAMETERS

The `bytearray()` takes three optional parameters:

- Source (Optional) - source to initialize the array of bytes.
- Encoding (Optional) - if source is a string, the encoding of the string.
- errors (Optional) - if source is a string, the action to take when the encoding conversion fails.

The source parameter can be used to initialize the byte array in the following ways:

Different source parameters	
Type	Description
String	Converts the string to bytes using <code>str.encode()</code> Must also provide <b>encoding</b> and optionally <b>errors</b>
Integer	Creates an array of provided size, all initialized to null
Object	Read-only buffer of the object will be used to initialize the byte array
Iterable	Creates an array of size equal to the iterable count and initialized to the iterable elements Must be iterable of integers between $0 \leq x < 256$
No source (arguments)	Creates an array of size 0.

Figure 3.2: Source Parameter

The `bytearray()` method returns an array of bytes of the given size and initialization values.

### 3.4 STRUCT.UNPACK()

Syntax: `struct.unpack(fmt, string)` Return the values `v1, v2,..` that are unpacked according to the given format(1st argument). Values returned by this function are returned as tuples of size that is equal to the number of values passed through `struct.pack()` during packing.

# Chapter 4

## SYSTEM REQUIREMENT SPECIFICATION

### 4.1 INTRODUCTION

This chapter describes about the requirements. It specifies the hardware and software requirements that are required in order to run the application properly. The Software Requirement Specification (SRS) is explained in detail, which includes overview of dissertation as well as the functional and non-functional requirement of this dissertation.

A SRS document describes all data, functional and behavioral requirements of the software under production or development. SRS is a fundamental document, which forms the foundation of the software development process. Its the complete description of the behavior of a system to be developed. It not only lists the requirements of a system but also has a description of its major feature. Requirement Analysis in system engineering and software engineering encompasses those tasks that go into determining the need or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirement Analysis is critical to the success to a development project. Requirement must be documented, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

The SRS functions as a blueprint for completing a project. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specification, testing and validation plans, and documentation plans, are related to it. It is important to note that an SRS contains functional and non-functional requirements only.

Thus the goal of preparing the SRS document is to

- To facilitate communication between the customer, analyst, system developers, maintainers.
- To serve as a contrast between purchaser and supplier.
- To firm foundation for the design phase.
- Support system testing facilities.
- Support project management and control.
- Controlling the evolution of the system.

## 4.2 FUNCTIONAL REQUIREMENTS

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. In this system following are the functional requirements:-

- Input test case must not have compilation and runtime errors.
- The application must not stop working when kept running for even a long time.
- The application must function as expected for every set of test cases provided.
- The application should generate the output for given input test case and input parameters.
- The application should generate on-demand services.

## 4.3 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as:-

- Product Requirements

- Organizational Requirements
- User Requirements
- Basic Operational Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions. The plan for implementing non-functional requirements is detailed in the system architecture. Broadly, functional requirements define what a system is supposed to do and non-functional requirements define how a system is supposed to be. Functional requirements are usually in the form of system shall do requirement, an individual action of part of the system, perhaps explicitly in the sense of a mathematical function, a black box description input, output, process and control functional model or IPO Model. In contrast, non-functional requirements are in the form of system shall be requirement, an overall property of the system as a whole or of a particular aspect and not a specific function. The systems' overall properties commonly mark the difference between whether the development project has succeeded or failed.

Non-functional requirements of our project include:

- Response time The time the system takes to load and the time for responses on any action the user does.
- Processing time - How long is acceptable to perform key functions or export / import data?
- Throughput The number of transactions the system needs to handle must be kept in mind.
- Storage - The amount of data to be stored for the system to function.
- Growth Requirements As the system grows it will need more storage space to keep up with the efficiency.
- Locations of operation - Geographic location, connection requirements and the restrictions of a local network prevail.
- Architectural Standards The standards needed for the system to work and sustain.

## 4.4 HARDWARE REQUIREMENTS

- At least 128MB of RAM (preferably physical) available. Larger data captures require more RAM than this minimum.
- At least 75MB of disk space available. Again, larger capture files will require more hard drive space than this.
- At least a 800x600 monitor resolution with 16-bit color. Needed to display Wireshark properly.
- A supported NIC and network card driver installed.

## 4.5 SOFTWARE REQUIREMENTS

- Operating System : Windows 10 and Ubuntu.
- Coding Language : C++ / Python.
- Library : XMLtodict.

## 4.6 SOFTWARE QUALITY ATTRIBUTES

- **Functionality:** the capability of the software to provide functions which meet stated and implied needs when the software is used under specified conditions.
- **Reliability:** the capability of the software to maintain its level of performance when used under specified conditions.
- **Usability:** the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.
- **Efficiency:** the capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.
- **Maintainability:** the capability of the software to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.
- **Portability:** the capability of software to be transferred from one environment to another.

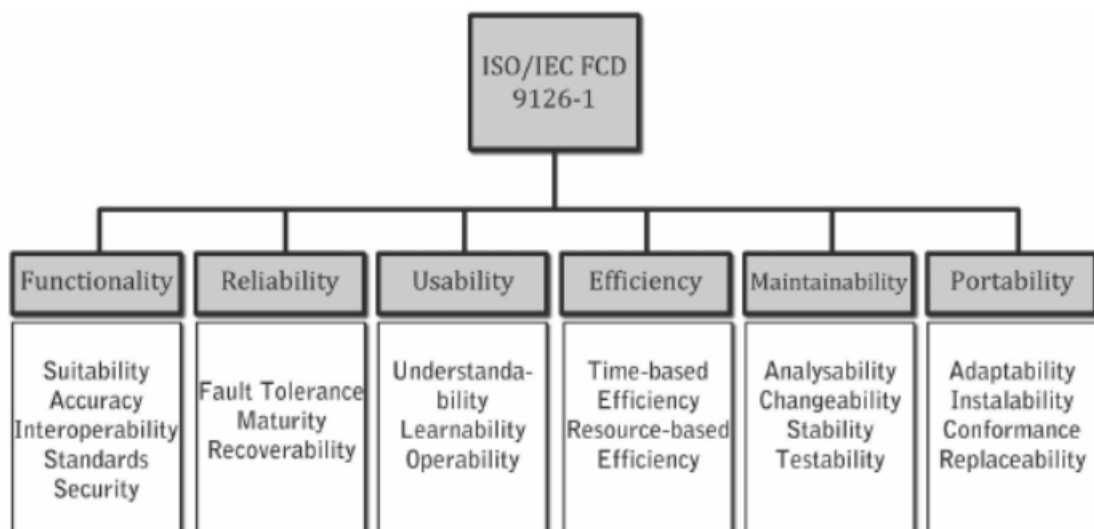


Figure 4.1: Software Quality Attributes

# Chapter 5

## SYSTEM DESIGN

### 5.1 INTRODUCTION

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed. This is the phase of system designing. Design is the perfect way to accurately translate a customers requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

### 5.2 SYSTEM DEVELOPMENT METHODOLOGY

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.



### 5.2.1 MODEL PHASES

The waterfall model is a sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Requirement initiation, Analysis, Design, Implementation, Testing and maintenance.

- **Requirement Analysis:** This phase is concerned about collection of requirement of the system. This process involves generating document and requirement review.
- **System Design:** Keeping the requirements in mind the system specifications are translated in to a software representation. In this phase the designer emphasizes on:-algorithm, data structure, software architecture etc.
- **Coding:** In this phase programmer starts his coding in order to give a full sketch of product. In other words system specifications are only converted in to machine readable compute code.
- **Implementation:** The implementation phase involves the actual coding or programming of the software. The output of this phase is typically the library, executables, user manuals and additional software documentation
- **Testing:** In this phase all programs (models) are integrated and tested to ensure that the complete system meets the software requirements. The testing is concerned with verification and validation.
- **Maintenance:** The maintenance phase is the longest phase in which the software is updated to fulfill the changing customer need, adapt to accommodate change in the external environment, correct errors and oversights previously undetected in the testing phase, enhance the efficiency of the software.

### 5.2.2 REASON FOR CHOOSING WATERFALL MODEL AS DEVELOPMENT METHOD

- Clear project objectives.
- Stable project requirements.
- Progress of system is measurable.
- Strict sign-off requirements.
- Helps you to be perfect.
- Logic of software development is clearly understood.

- Production of a formal specification
- Better resource allocation.
- Improves quality. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage.
- Less human resources required as once one phase is finished those people can start working on to the next phase.

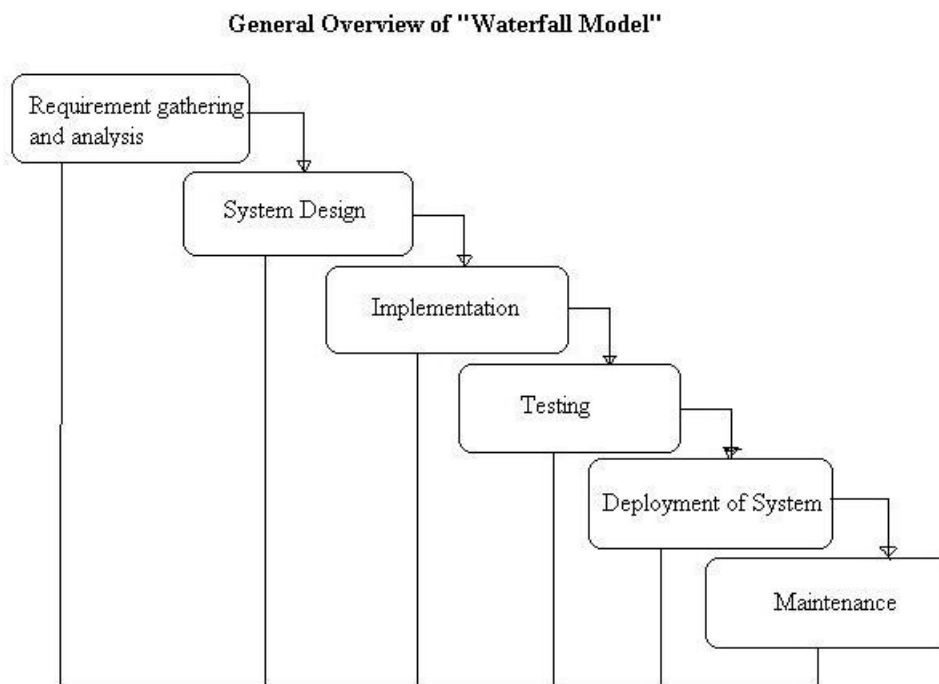


Figure 5.1: Waterfall Model

### 5.3 DESIGN USING UML

Designing UML diagram specifies, how the process within the system communicates along with how the objects within the process collaborate using both static as well as dynamic UML diagrams since in this ever-changing world of Object Oriented application development, it has been getting harder and harder to develop and manage high quality applications in reasonable amount of time. As a result of this challenge and the need for a universal object modeling language every one could use, the Unified Modeling Language (UML) is the Information industries version of blue print. It is a











<i>Line</i>	<i>Symbol</i>
Association	<u>AssociationName</u>
Aggregation	
Generalization	
Dependency	
Activity edge	
Event, transition	event[guard]/action 
Link	<u>.inkName</u>
Composition	
Realization	
Assembly connection	
Message	some code 
Control flow	

Table 5.1: Symbols used in UML

method for describing the systems architecture in detail. Easier to build or maintains system, and to ensure that the system will hold up to the requirement changes.

## 5.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) is graphic representation of the "flow" of data through an information system. A data flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. DFDs show the flow of data from external entities into the system, how the data moves from one process to another, as well as its logical storage. There are only four symbols: 1. Squares representing external entities, which are sources and destinations of information entering and leaving the system. 2. Rounded rectangles representing processes, in other methodologies, may be called 'Activities', 'Actions', 'Procedures', 'Subsystems' etc. which take data as input, do processing to it, and output it. 3. Arrows representing the data flows, which can either, be electronic data or physical items. It is impossible for data to flow from data store to data store except via a process, and external entities are not allowed to access data stores directly. 4. The flat three-sided rectangle is representing data stores should both receive information for storing and provide it for further processing.

## 5.5 CLASS DIAGRAM

UML class diagram shows the static structure of the model. The class diagram is a collection of static modeling elements, such as classes and their relationships, connected as a graph to each other and to their contents. The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed.

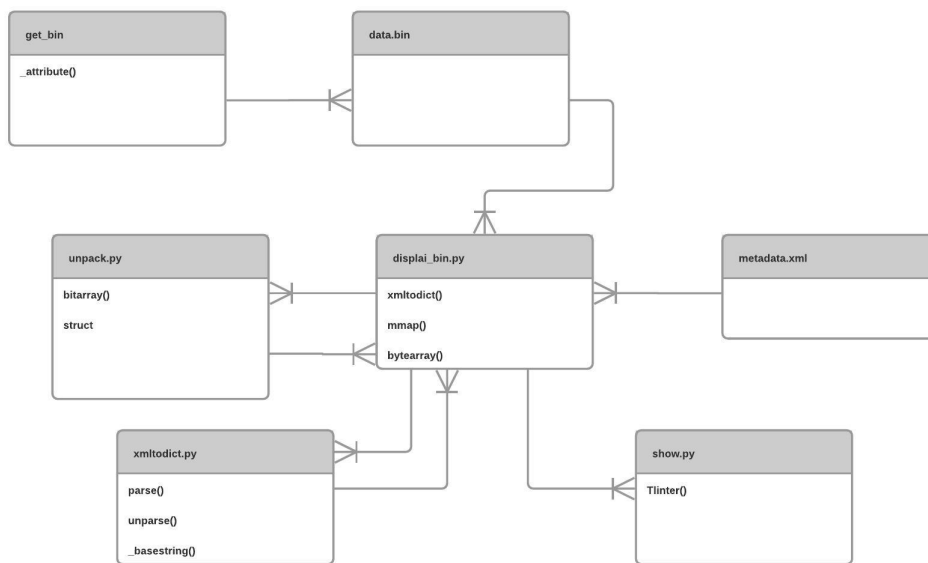


Figure 5.2: Class Diagram

## 5.6 USE CASE DIAGRAM

A use case defines a goal-oriented set of interactions between external entities and the system under consideration. The external entities which interact with the system are its actors. A set of use cases describe the complete functionality of the system at a particular level of detail and it can be graphically denoted by the use case diagram.

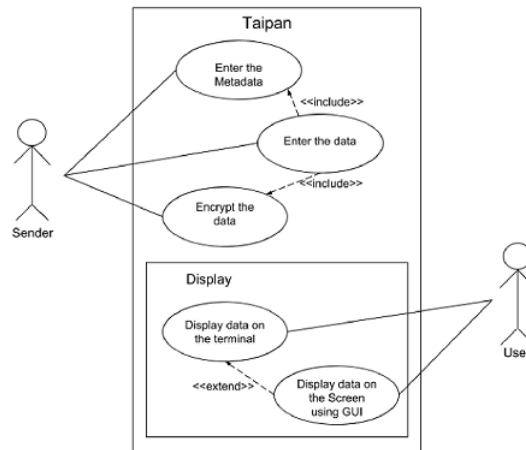


Figure 5.3: Use Case Diagram

## 5.7 ACTIVITY DIAGRAM

An activity diagram shows the sequence of steps that make up a complex process. An activity is shown as a round box containing the name of the operation. An outgoing solid arrow attached to the end of the activity symbol indicates a transition triggered by the completion.

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

## 5.8 SEQUENCE DIAGRAM

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time, the horizontal dimension

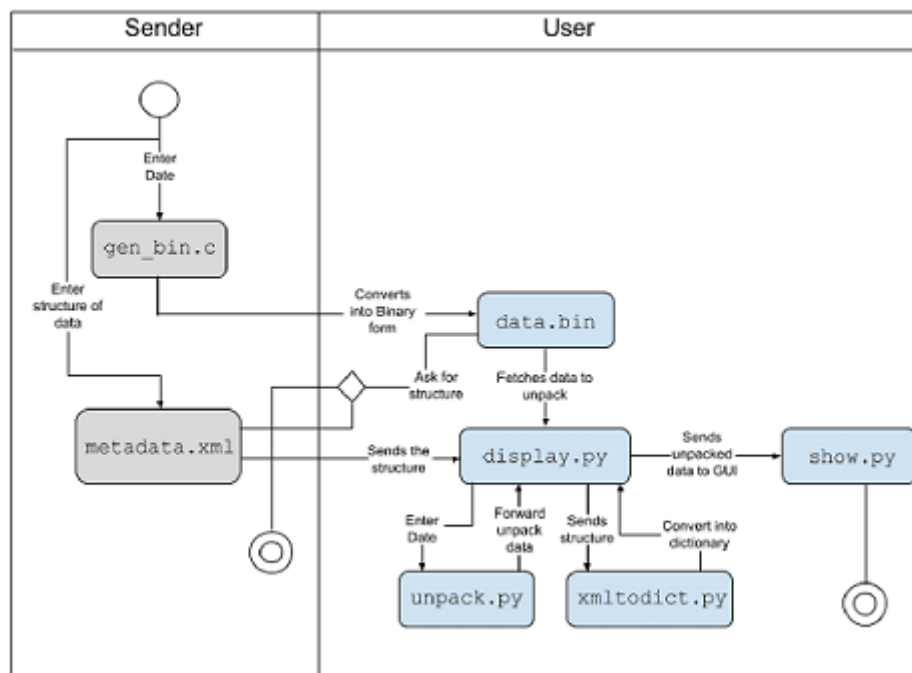


Figure 5.4: Activity Diagram

represents the objects existence during the interaction. **Basic elements:**

- **Vertical rectangle:** Represent the object is active (method is being performed).
- **Vertical dashed line:** Represent the life of the object.
- **X:** represent the life end of an object. (Being destroyed from memory)
- **Horizontal line with arrows:** Messages from one object to another.

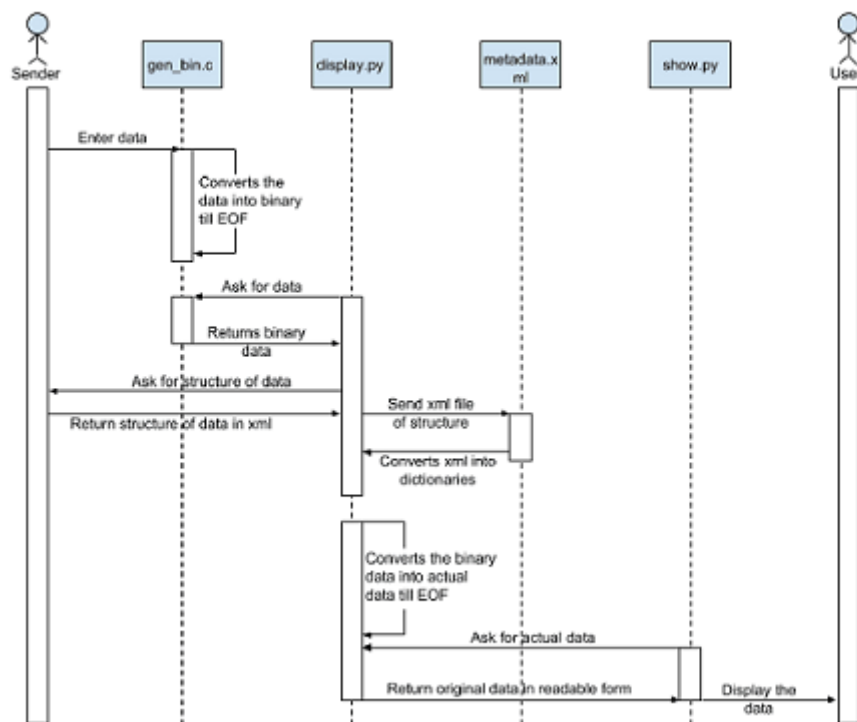


Figure 5.5: Sequence Diagram

# Chapter 6

## IMPLEMENTATION

### 6.1 INTRODUCTION

The implementation phase of the project is where the detailed design is actually transformed into working code. Aim of the phase is to translate the design into a best possible solution in a suitable programming language. This chapter covers the implementation aspects of the project, giving details of the programming language and development environment used. It also gives an overview of the core modules of the project with their step by step flow. The implementation stage requires the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.
- Evaluation of the changeover method.
- Correct decisions regarding selection of the platform.
- Appropriate selection of the language for application development.

### 6.2 Generating Binary File

```
#include <stdio.h>
struct date_entry_record_type
{
    unsigned int day    : 3;
    unsigned int date  : 5;
    unsigned int month  : 4;
```



```
    unsigned int year : 12;
} __attribute__((packed));
int day[] = {0, 1, 2, 3, 4, 5, 6};
int date[] = {10, 11, 12, 13, 14, 15, 16};
int month[] = {1, 2, 3, 4, 5, 6, 7};
int year[] = {1970, 1971, 1972, 1973, 1974, 1975, 1976};
main ()
{
    FILE *fp;
    int i;
    struct date_entry_record_type date_entry;
    fp = fopen ("data.bin", "wb");
    printf ("size: %d\n", sizeof (date_entry));
    for (i = 0; i < 7; i++)
    {
        date_entry.day = day[i];
        date_entry.date = date[i];
        date_entry.month = month[i];
        date_entry.year = year[i];
        fwrite (&date_entry, sizeof (date_entry), 1, fp);
    }
    fclose (fp);
}
```

## 6.3 Display

```
from bitarray import bitarray
import xml_read
import mmap
import tool
import convert

class MyException(Exception):
    pass

with open ('data.bin', 'rb') as bfd:
    m = mmap.mmap (bfd.fileno(), 0, access=mmap.ACCESS.READ)

    by = bytearray (m)

    print ('...._opened_data.bin_for_parsing_of_data.');
```

*#to display the actual information*

```
def value():
    l=0
    ba = bitarray (endian='little')
    ba.frombytes (str(by))
    xml_read.main()
    typ = xml_read.kind_of_type
    siz = xml_read.type_size
    mini = xml_read.min_limit
    maxi = xml_read.max_limit
    size_arr = xml_read.size_arr
    types_of_type = xml_read.types_of_type
    siz_o_struct = xml_read.struct_size
    try:
        while l < len(ba):
            try:
                if siz ['DAYENUMERATED_TYPE'] <= 8:
```

```
    day = convert.byte_conversion(b=ba[l:l+size_arr
    [0]] , sign = tool.assign_sign(types_of_type
    [0]))

else :
    day = convert.int_conversion(b=ba[l:l+size_arr
    [0]] , e='little' , sign = tool.assign_sign(
    types_of_type[0]))

if siz['DATE_TYPE'] <= 8:
    date = convert.byte_conversion(b=ba[l+size_arr
    [0]:l+size_arr[1]] , sign = tool.assign_sign(
    types_of_type[1]))

else :
    date = convert.int_conversion(b=ba[l+size_arr[0]:
    l+size_arr[1]] , e='little' , sign = tool.
    assign_sign(types_of_type[1]))

if siz['MONTHTYPE'] <= 8:
    month = convert.byte_conversion(b=ba[l+size_arr
    [1]:l+size_arr[2]] , sign = tool.assign_sign(
    types_of_type[2]))

else :
    month = convert.int_conversion(b=ba[l+size_arr
    [1]:l+size_arr[2]] , e='little' , sign = tool.
    assign_sign(types_of_type[2]))

if siz['YEAR_TYPE'] <= 8:
    year = convert.byte_conversion(b=ba[l+size_arr
    [2]:l+size_arr[3]] , sign = tool.assign_sign(
    types_of_type[3]))

else :
    year = convert.int_conversion(b=ba[l+size_arr[2]:
    l+size_arr[3]] , e='little' , sign = tool.
    assign_sign(types_of_type[3]))
```

```
        if date < int(mini[ 'DATE_TYPE' ]) or month < int(mini
            [ 'MONTH_TYPE' ]):
            raise MyException

    finally:
        print (day , date , month , year)
        l=l+size_of_struct

except MyException:
    print ("Invalid Input")

def main ():

    value ()

if __name__=='__main__':
    main ()
```

## 6.4 Reading XML File

```
import xmltodict

kind_of_type = {}
type_size = {}
min_limit = {}
max_limit = {}
size_arr = []
types_of_type = []
struct_size = 0

with open ('metadata.xml') as mfd:
    meta_data = xmltodict.parse (mfd.read ())

    print ('...._using_metada.xml_as_data_definition.');
```

```
def main_type_query (main_type_name):
    main_type_dict = None

    if len (meta_data ['msg'] ['main_type']) > 1:
        for main_type_element in meta_data ['msg'] ['main_type']:
            if main_type_element ['name'] == main_type_name:
                main_type_dict = main_type_element
                break
    else:
        if meta_data ['msg'] ['main_type'] ['name'] ==
            main_type_name:
                main_type_dict = meta_data ['msg'] ['main_type']

    return main_type_dict
```

```
def type_query (type_name):

    if len (meta_data ['msg'] ['type']) > 1:
```

```

    for type_element in meta_data['msg']['type']:
        if type_element['name'] == type_name:
            type_dict = type_element
            break
    else:
        if meta_data['msg']['type']['name'] == type_name:
            type_dict = meta_data['msg']['type']

    return type_dict

def record_read (record_dict):
    global size_arr
    global types_of_type
    global struct_size
    struct_size = int(record_dict['size'])

    for field_dict in record_dict['record']['field']:
        field_type = type_query (field_dict['type_name'])

        if field_type == None:
            print ('_..._could_not_get_definition_of_' +
                field_dict['type_name'] + '_..._existing')
            quit ()

        else:
            #print (' ... read ' + field_type['size'] + ' bits
                from buffer for ' + field_type['name'])

            type_size [field_type ['name']] = int (field_type ['size'
                ])

            kind_of_type [field_type ['name']] = field_type ['
                kind_of_type']

            for types in kind_of_type.values():
                if types == 'integer':
                    min_limit [field_type ['name']] = field_type ['
                        integer'] ['range'] ['min'] ['expression']

```

```
max_limit[field_type['name']] = field_type['integer']['range']['max']['expression']

size_arr = type_size.values()
temp = 0
for i in range(len(size_arr)):
    size_arr[i] = temp + size_arr[i]
    temp = size_arr[i]

types_of_type = kind_of_type.keys()

def main():
    main_type_name = 'DATE.RECORD.TYPE'

    main_type = type_query(main_type_name)

    if main_type == None:
        print('.....could not get defination of ' +
              main_type_name + '.....existing')
        quit()

    else:
        record_read(main_type)

if __name__ == '__main__':
    main()
```

## 6.5 Convert

```
import unpac
```

```
#to convert byte type values
```

```
def byte_conversion(b, sign):
```

```
    res = None
```

```
    bit_len=len(b)
```

```
    if bit_len > 0:
```

```
        if bit_len <= 8 and sign == 'unsigned':
```

```
            res = unpac.u_int8_val(bits=b)
```

```
        elif bit_len == 8 and sign == 'signed':
```

```
            res = unpac.int8_val(bits=b)
```

```
    else:
```

```
        print ('Invalid length %d' % bit_len)
```

```
    return res
```

```
#to convert int type values
```

```
def int_conversion(b, e, sign):
```

```
    res = None
```

```
    bit_len=len(b)
```

```
    if bit_len > 0:
```

```
        if bit_len <= 16 and sign == 'unsigned':
```

```
            res = unpac.u_int16_val(bits=b, endian=e)
```

```
        elif bit_len == 16 and sign == 'signed':
```

```
            res = unpac.int16_val(bits=b, endian=e)
```

```
        elif bit_len <= 32 and sign == 'unsigned':
```

```
            res = unpac.u_int32_val(bits=b, endian=e)
```



```
    elif bit_len == 32 and sign == 'signed':
        res = unpac.int32_val(bits=b, endian=e)

    elif bit_len <= 64 and sign == 'unsigned':
        res = unpac.u_int64_val(bits=b, endian=e)

    elif bit_len == 64 and sign == 'signed':
        res = unpac.int64_val(bits=b, endian=e)

else:
    print ('Invalid length_%d' % bit_len)

return res
```

## 6.6 Show

```
import Tkinter
import project1

optionlist = {}

class simpleapp_tk(Tkinter.Tk):
    def __init__(self, parent):
        Tkinter.Tk.__init__(self, parent)
        self.parent = parent
        self.initialize()

    def initialize(self):
        self.grid()
        lim = len(project1.strin)

        for i in range(lim):
            optionlist["Block"+"%d"%(i+1)] = project1.strin[i]
            self.dropVar = Tkinter.StringVar()
            self.dropVar.set("Choose")
            self.dropmenu1 = Tkinter.OptionMenu(self, self.dropVar
                ,*optionlist ,command=self.Click)
            self.dropmenu1.grid(column=0, row=0)

            self.labelVariable = Tkinter.StringVar()
            label = Tkinter.Label(self, textvariable=self.
                labelVariable , anchor="nw" , fg="#abccba" , bg="blue")
            label.grid(column=1,row=0, rowspan=lim , sticky='EWNS')

            self.grid_columnconfigure(0, weight=0)
            self.grid_columnconfigure(1, weight=1)
        for i in range(lim):
            self.grid_rowconfigure(i, weight=0)
        self.resizable(True, True)
        self.update()
```

```
        self.geometry('500x500')
        text = ""
        for temp in project1.strin:
            text = text + temp + "\n"
        self.labelVariable.set(text)

    def Click(self, value):
        self.labelVariable.set(optionlist[value])

if __name__ == "__main__":
    project1.main()
    app = simpleapp_tk(None)
    app.title('my_application')
    app.mainloop()
```

# Chapter 7

## TESTING AND RESULTS

### 7.1 INTRODUCTION

Testing is an important phase in the development life cycle of the product this was the phase where the error remaining from all the phases was detected. Hence testing performs a very critical role for quality assurance and ensuring the reliability of the software. Once the implementation is done, a test plan should be developed and run on a given set of test data. Each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is suppose to do. Testing is the final verification and validation activity within the organization itself. In the testing stage following goals are tried to achieve:-

- To affirm the quality of the project.
- To find and eliminate any residual errors from previous stages.
- To validate the software as the solution to the original problem.
- To provide operational reliability of the system.

During testing the major activities are concentrated on the examination and modification of the source code. The test cases executed for this project are listed below. Description of the test case, steps to be followed; expected result, status and screenshots are explained with each of the test cases.

### 7.2 TESTING METHODOLOGIES

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important types of testing are:

### 7.2.1 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level. Using white box testing we can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Execute internal data structure to assure their validity.

### 7.2.2 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot see into it. The test provides inputs and responds to outputs without considering how the software works. It uncovers a different class of errors in the following categories:

- Incorrect or missing function.
- Interface errors.
- Performance errors.
- Initialization and termination errors.
- Errors in objects.

#### Advantages:

- The test is unbiased as the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.
- Test cases can be designed as soon as the specifications are complete.

---

Test Case ID
Purpose
Preconditions
Inputs
Expected Outputs
Postconditions

Table 7.1: Unit Test Cases

### 7.2.3 UNIT TESTING

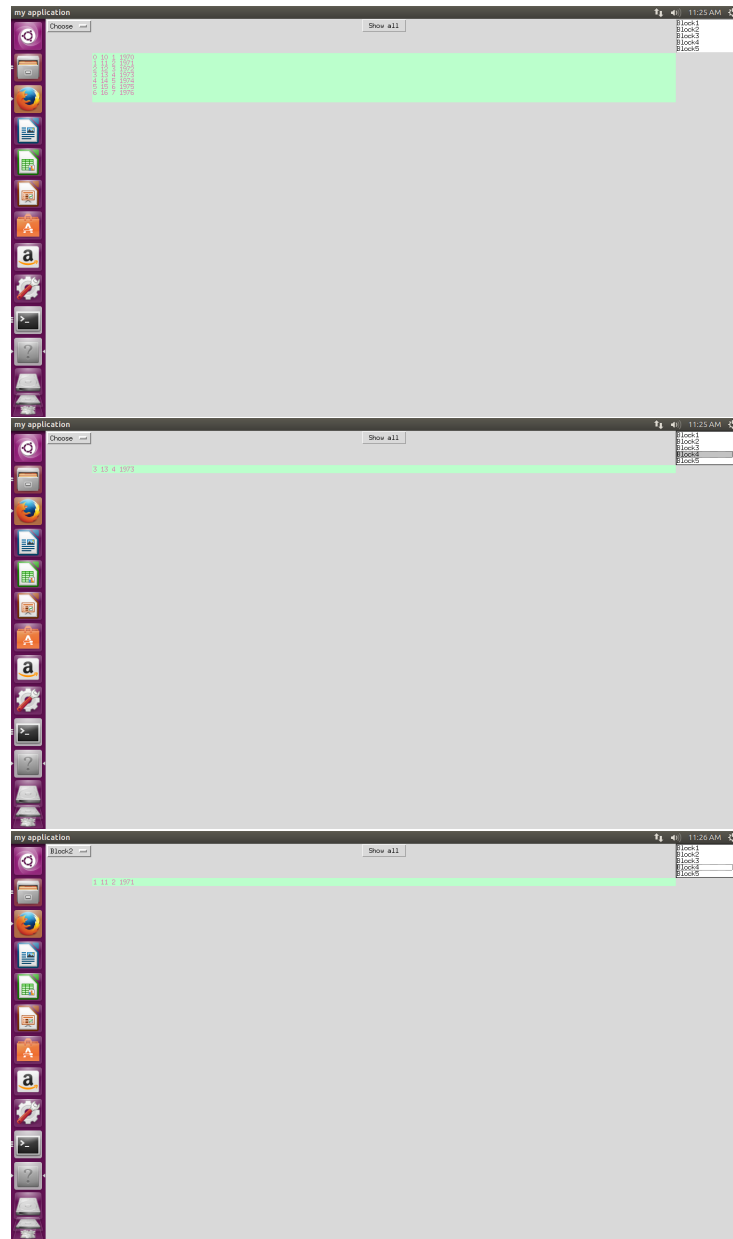
Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases. **Test strategy and approach**

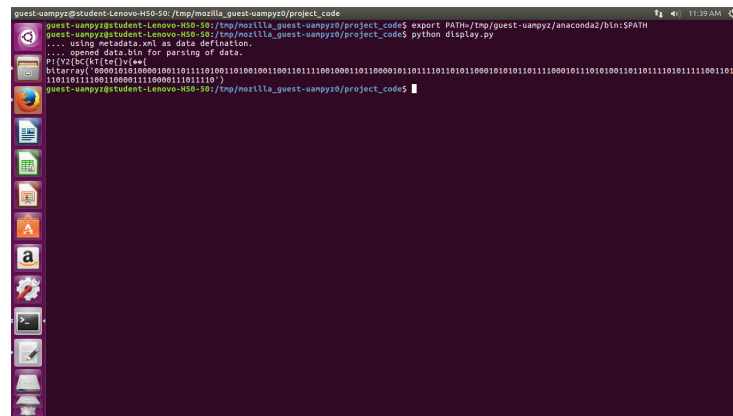
Field testing will be performed manually and functional tests will be written in detail.

**Test objectives:**

- All Components must work properly.
- Proper coordinates should be sent by the Android app to the Arduino
- The entry screen, messages and responses must not be delayed in the Android app.

### 7.3 RESULTS





```
guest-uampyz@student-Lenovo-H50-50: /tmp/mozilla_guest-uampyz0/project_code
guest-uampyz@student-Lenovo-H50-50: /tmp/mozilla_guest-uampyz0/project_code$ export PATH=/tmp/guest-uampyz/anaconda2/bin:$PATH
guest-uampyz@student-Lenovo-H50-50: /tmp/mozilla_guest-uampyz0/project_code$ python display.py
.... using metadata.xml as data definition.
.... opened data.bin for parsing of data.
display[0x10]{}{}{}{}
bitarray('0000101000001001101110100110100100110011011100100011011000010110111011010100010101011100010111010011010111010111001101
10110111001100011110000111011110')
guest-uampyz@student-Lenovo-H50-50: /tmp/mozilla_guest-uampyz0/project_code$
```



## Chapter 8

# CONCLUSION & FUTURE SCOPE

## 8.1 CONCLUSION

This project was the first attempt to develop a system of this nature. We identified from the beginning that producing a complete result would be impossible within the given time frame. We viewed the project as a journey where we learnt many lessons and gained insights to the subject which we tried to share in this report and summarised in this chapter. We tried to look at the problem from many points of view which generated some new ideas that could be explored in future. We suggested formal approaches for modelling and analysing the system which are by no means complete but could become the initiation for further research. We also created a working system and algorithms which we claim to be useful and extensible. However, as we have seen in these chapter, all these achievements are only partialy successful.

Personally, we would consider this project a success if the ideas described in the report can be a useful reference for future work on the subject.

## 8.2 FUTURE SCOPE

This project was the first implementation of a system of this nature. We identify that the work done both in terms of analysing and implementing the system is by no means complete. In this section we list the things that were either left open by this project or were opened by the analysis performed and the lessons learned during our interaction with the subject.

## 8.3 BUILD A COMPLETE SYSTEM

We will give special focus to an idea generated by this project and we believe it to be a very interesting proposal.

We can magnify the overall model which will in turn nullify the precision of the data. Which also means that the data determination becomes comparatively dense.

We now give a brief overview of the rest of the future issues.

## 8.4 FUTURE WORK

Taipan as a Network Protocol Analyzer has already proven its mettle in all necessary realms. However it still has scope of improvement in it as far as alert generation and heuristic development is concerned. We are working to introduce certain utilities in the source code of Wireshark to overcome the above shortcomings by making Taipán capable of alert generations.

# References

- [1] S. rao and S. rao, "Denial of Service attacks and mitigation techniques: Real time implementation with detailed analysis", 2011.
- [2] M. Salagean, "Anomaly detection of network traffic based on Analytical Discrete Wavelet Transform", 2010
- [3] U. Banerjee, A. Vashishtha and M. Saxena, "Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection", International Journal of Computer Applications, vol. 6
- [4] <http://sharkfest.wireshark.org/sharkfest.11/>
- [5] [http://www.wireshark.org/docs/wsug\\_html\\_chunked/](http://www.wireshark.org/docs/wsug_html_chunked/)
- [6] [http://www.wireshark.org/docs/wsug\\_html\\_chunked/ChapterIntroduction.html#ChIntroFig1](http://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroFig1)
- [7] <http://www.cacotech.com/resources.html>
- [8] <http://www.riverbed.com/us/products/cascade/airpcap.php>
- [9] <http://www.wireshark.org/faq.html#q7.1>
- [10] Waqar Ali, Jun Sang, Hamad Naeem, Rashid Naeem, Ali Raza, (2015), Wireshark window authentication based packet capturing scheme to prevent DDoS related security issues in cloud network nodes, Research Paper

