Project Specification

Remote Monitoring of Power Consumption in a Cloud Server Network

Version PS 1.2 - May 12, 2014

Customer: Patrik Arlos

Chief Executive Officer: Dragos Ilie

Developer Team Zenoss		
Student 1	Name	Sarat Chandra Pasumarthy
	E-Mail	sapa14@student.bth.se
	Social Security Number	921231-0297
Student 2	Name	Manish Reddy Guduru
	E-Mail	magu14@student.bth.se
	Social Security Number	930706-0518
Student 3	Name	Sukesh Kumar Tedla
	E-Mail	sute14@student.bth.se
	Social Security Number	940224-5352
Student 4	Name	Prakhyath Neelagiri
	E-Mail	prne14@student.bth.se
	Social Security Number	930326-1359
Student 5	Name	Hadassah Pearlyn Nagathota
	E-Mail	hana14@student.bth.se
	Social Security Number	930719-9548
Student 6	Name	Preetham Kalakuntla
	E-Mail	prka14@student.bth.se
	Social Security Number	920523-3779

Preface

The document is the third release of the Project Specification concerned with this project. This document is primarily aimed at the parties involved in this project, that is, the customer and the company involved in providing solution to the customer's needs. The company comprises of Chief Executive Officer (CEO) and the developer team Zenoss. The document starts with glossary and abbreviations which define the technical terms and abbreviations used in the document. Secondly, the document contains background which specifies the customer's business environment and an overview of the customer's needs and problems which will be addressed by the product. Thirdly, the document contains the proposed solution which meets the customer's expectation. Further, the document contains limitations which are associated with the product and a detailed time plan is mentioned about how the product will be developed and detailed work breakdown structure associated with it. These sections are shared with the Project Proposal document.

The sections pertaining only to this Project Specification document are the project organisation describing the various roles and responsibilities of the team members, the configuration management explaining the version management, system building and the release management process followed by the developer team Zenoss. Furthermore, this is followed by sections of progress tracking, quality control, risk management and system release plan describing the methods to keep track of the progress of the project, to ensure quality of the released product, procedures pertaining to identification and analysis of risks, and the plans drawn for the release of the system. Lastly, there is a references section giving a list of the references used for creation of this document.

This is version PS1.2 of the document and the version history is as follows:

Released v PS1.2 on 2014-05-12

- Updated the graph for progress tracking (see Section 8.Progress Tracking).

Released v PS1.1 on 2014-05-05

- Modified preface by adding snapshot of the previous versions of the document.

Released v PS1.0 on 2014-04-28

- Initial release

1. Glossary and abbreviations

API- *Application Program Interface*

An interface, generally specified as a set of operations, that allows access to an application program's functionality.

Cloud Computing

The provision of computing and/or application services over the Internet using a large number of commodity computers and virtualization technology to make effective use of these systems.

CN- *Computer Node*

Gantt chart

An alternative name for a bar chart.

HTTP- *Hypertext Transfer Protocol*

It is an application protocol for distributed, collaborative and hypermedia information systems.

NMS- *Network Management System*

It constantly monitors a computer network and notifies the network administrator in case of outages.

PDU- *Power Distribution Unit*

It is a device designed to distribute electric power with multiple outputs.

SNMP- *Simple Network Management Protocol*

It is an internet-standard protocol for managing devices on IP networks.

SSH- *Secure Shell*

It is a cryptographic network protocol for secure network services between two networked computers.

UPS- *Uninterruptable Power Supply*

It is an electrical device that provides emergency power when the input power supply fails.

VM- *Virtual Machine*

It is a software based emulation of a computer. Virtual Machines operate based on the computer architecture and functions of a real or hypothetical computer.

2. Background

The advent of cloud computing has given rise to multiple products and services which were unheard of before and most importantly it has eased the burden on the users. Today, users can just log in through the internet and access these services without actually installing anything. Cloud computing also enables service providers to maximise their profits by reducing their operational costs substantially. These significant features make cloud computing really lucrative for businesses on different levels.

On the other hand cloud providers also have to ensure that they reduce carbon emissions and maintain environmental standards. This can be achieved if power consumption of devices in use is monitored in an effective manner. Thus, through monitoring, unnecessary consumption of power can be detected and reduced.

The customer, a Green Cloud provider, wants to monitor the power consumption of each device (UPSs, PDUs, CNs, and VMs) in the network. Additionally, the customer also intends to monitor the total power consumption at each node of the network and the total power consumed by the network in real-time. Through this monitoring the customer can also keep a check on redundant power loss which could be due to reasons such as unauthorised or unknown connection to the power unit or existence of an unidentified device in the network that is consuming power.

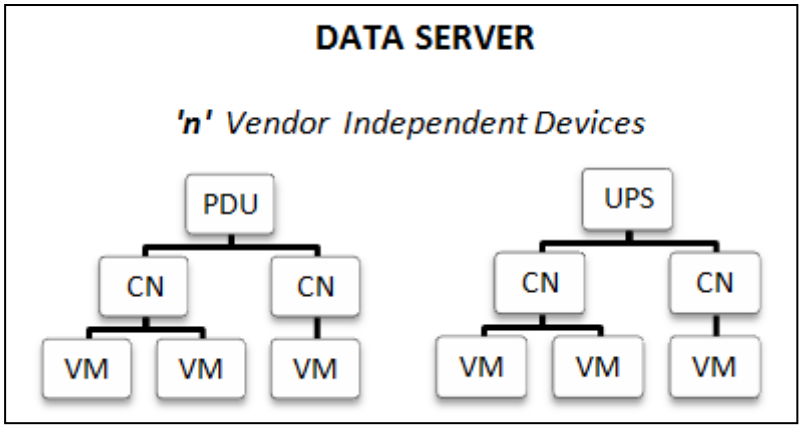

Figure1. Block Diagram of customer's network

3. Proposed solution

The basic requirement of the customer is monitoring power consumption by the network. In more detail, the customer wants this need to be addressed in the following way:

- Monitoring of power consumption with Zenoss as the front end. However, the product must also be compatible with any other Open NMS.
- Total power consumed by the network to be determined in real-time.
- Power consumption of each device (UPSs, PDUs, CNs, VMs) is to be measured as:
  - Current power
  - Average power
  - Minimum power
  - Maximum power
- These statistics should be stored in a server and be retrievable by using either SNMP, HTTP or SSH protocols.
- Statistics are to be presented as graphs on the front end in different time scales.

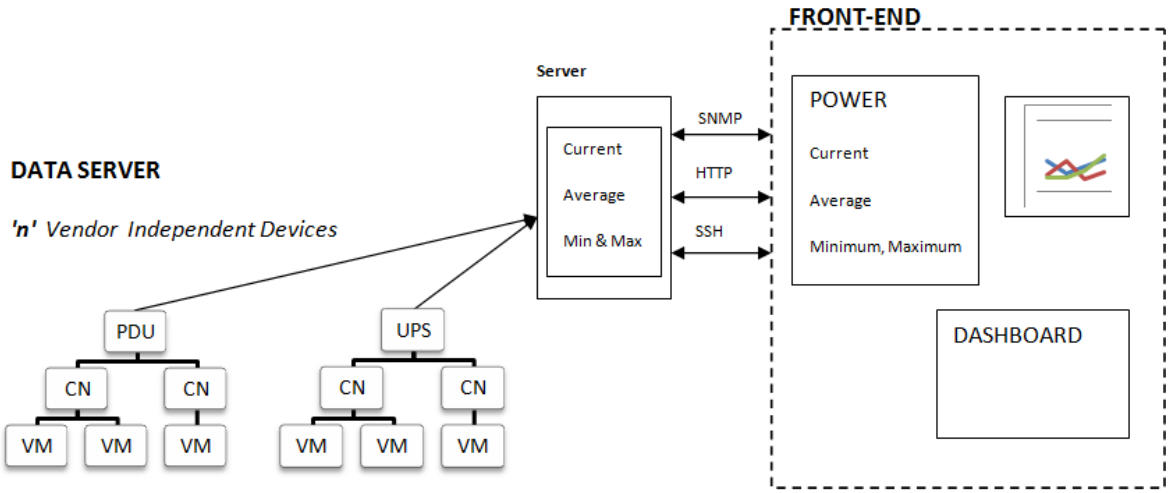

Figure2. Block diagram of the proposed solution

The tasks for which the developer team is responsible for are:

i. Create an API that will be used by the customer to monitor power consumption.
ii. Create a program/script such that it can be run for any arbitrary number of devices as per the customer's requirements.
iii. A web server that will be used to store the various statistics measured regarding power consumption.
iv. The product developed must monitor any device irrespective of the vendor.

4. Limitations

The project is limited to measure only power consumption.

5. Time plan

S.no	Task	Days Required (estimated)	Starting in
1	**Project Allocation**		**09-04-2014**
2	**Project Proposal** • Research about the topic area. • Determining the underlying aspects related to customer's requirements. • Documentation of the Proposal.	**6**	**Week 1**
3	**Software Requirements Specification** • Defining the customer's needs and their environment. • Individual research and congregation of the material (Research pertaining to the different types of architecture possible). • Defining the architecture of the system according to the customer needs. • Documentation of the software requirements.	**9**	**Week 2**
4	**Software Design** • Research on Programming tools which facilitate the project. • Selection of programming language required in each stage of the architecture. • Designing API.	**8**	**Week 3**
5	**Implementation** • Writing the code to implement the system architecture defined in the software requirements. • Code optimisation.	**6**	**Week 4**
6	**Testing and Validation** • Testing the code for bugs. • Testing the code for the customer's work environment. • Revisit and review the source code if the requirements are not met.	**10**	**Week 5**
7	**Documentation**	**5**	**Week 6**
8	**Demo for the Customer**	**1**	**Week 6**
9	**Final Product Delivery**		**19-05-2014**

Table1. Time Plan

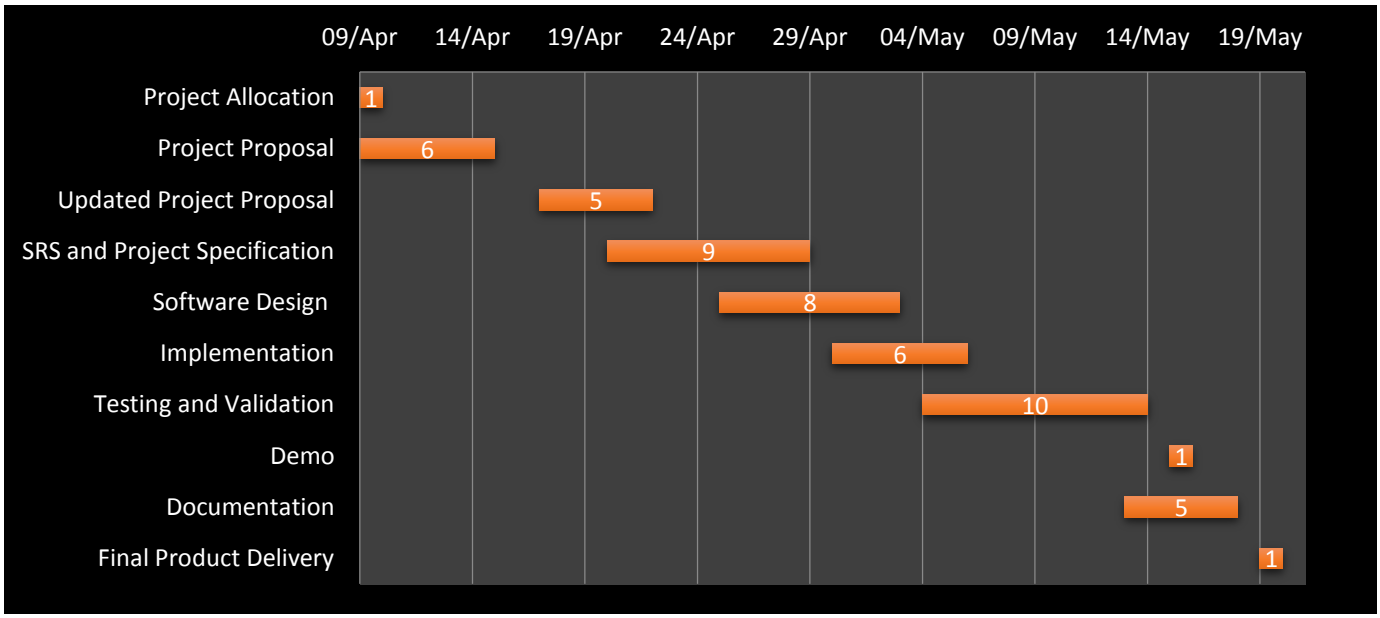

Figure3. Gantt chart

Checkpoint	Task	Parties involved	Estimated completion date
Tollgate 01	Project Proposal	Customer and CEO	14th April
Milestone 01	Project Plan	Team Zenoss and CEO	16th April
Tollgate 02	Updated Project Proposal	Customer and CEO	22nd April
Tollgate 03	Project Specification and Software Requirement Specification	Customer and CEO	28th April
Milestone 02	Software Design	Team Zenoss and CEO	2nd May
Tollgate 04	Design Document	Customer and CEO	5th May
Milestone 03	Software Implementation	Team Zenoss and CEO	8th May
Tollgate 05	Acceptance Test Plan	Customer and CEO	12th May
Milestone 04	Testing and Validation	Team Zenoss and CEO	13th May
Tollgate 06	Demo	Customer and CEO	15th May
Milestone 05	Documentation	Team Zenoss and CEO	17th May
Tollgate 07	Final Product Delivery and Documentation	Customer and CEO	19th May

Table2 : Milestones and Tollgates

6. Project Organisation

The role of the developer team members is given in the following table:

ROLE DISTRIBUTION	
TEAM MEMBER	**ROLE**
Sarat Chandra Pasumarthy	Project Manager
Manish Reddy Guduru	Testing Manager
Sukesh Kumar Tedla	Technical Manager
Hadassah Pearlyn Nagathota	Programmer
Prakhyath Neelagiri	Administrator
Preetham Kalakuntla	Programmer

Table3 : Role Distribution Table

The responsibilities associated with each role can be described in the following manner:

Project Manager For this project, the Project Manager also plays the role of the Configuration Manager and Release Manager. The responsibilities:

- Planning and execution of the project.
- Oversee the version management and system building
- Deciding, managing and documenting the system releases.

Administrator Responsible for configuring and maintaining the environment including the hardware as well as the software components.

Technical Manager Responsible for the establishing standards and to track and measure project's progression. Also, evaluates software implementation on design and task thoroughness.

Testing Manager Responsible for testing built executable systems on a target environment and report faults, if any, to the developer. The Testing Manager is responsible for quality control as well.

Programmer Responsible for programming of source code and also optimizing the source code according to test results.

7. Configuration Management

All the documents and configuration items such as source code, libraries, data files, etc. are all stored in a single repository which is accessible by the Project Manager. Working copies are provided in the repository which are accessible by the developers. The developer uses a working copy for a specific item from the repository and develops it by making desired changes to it. While doing so, that developer must provide brief description of the changes made. The developer submits this copy to the Project Manager who checks the copy and decides accordingly whether to update the master copy that is present in the repository or not. The other developers working on the same item need to update their working copies to view the changes made by their fellow-developer, if any. The Project Manager is responsible for any resolving conflicts that arise when more than one developer makes changes to the same item at the same position.

Version management

To keep track of the different versions of the documents and the system components or configuration items such as the source code, we are using a versioning scheme that can be depicted as follows:

Identifier major.minor[.build]

Identifier	To identify the type of the item. For example, PS is used to indicate *Project Specification.*
[.build]	Keeps track of the revisions made during the internal stages of development, that is, during the making process within the developer team. Upon updating, the *build* number is incremented to indicate the revision to that item.
minor	The version of the item which is a minor release for repairing reported bugs and customer problems.
major	The major release of the item which is either an initial release, a release which delivers new functionality or a release which has undergone a complete makeover from the previous version.

Figure4: Depiction the versioning scheme

System Building

This section discusses system building, the process followed for creating an executable system by compiling and linking the system components, external libraries, configuration files, etc.[1]

[1] Ian Sommerville, *Software Engineering*.

For the sake of system building, other than the repository where all the system components are stored, we have the following components:

- A development system, which includes development tools such as compilers, source code editors, etc. This is used by each developer for developing the system components.
- A build server consisting of compilers and linkers that is used to build definitive, executable versions of the system.
- A target environment on which the system is executed and tested.

Building Process

- The Project Manager sets a delivery date and time for system components.
- Developers must deliver any new versions of the components they have developed before that delivery deadline.
- A new version of the system is built from the baseline components by compiling and linking them to form a complete system.
- This system is then delivered to the Testing Manager, who carries out a set of predefined system tests on the target environment created for the project.
- Faults that are discovered during system testing are documented and returned to the system developers. They repair these faults in a subsequent version of the component.

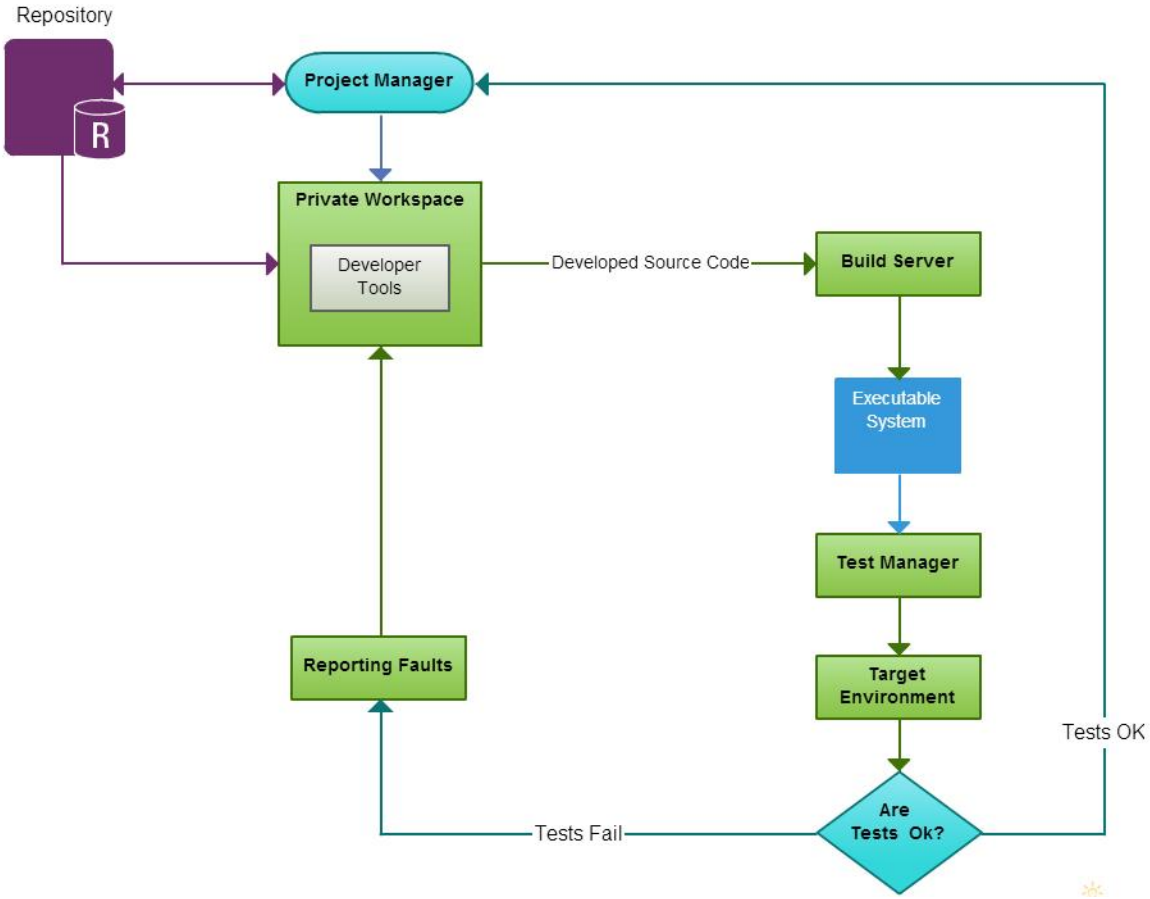

Figure5: System Building Process

Release management

The task of release management is the responsibility of the Release Manager, who in this project is the Project Manager. The responsibilities include

- Deciding when the system should be released.
- Managing the process of creating the release.
- Managing the versions of the source code files, corresponding executable codes, configuration files, libraries, compilers and tools used to build the system.
- Documenting the release to ensure it can be re-created.

Every system release consists of the following:

- Executable code of the system.
- Configuration files defining how the system should be configured for installations.
- Installation program to help install the system on the target environment.
- Documentation describing the system.

8. Progress Tracking

To ensure that the project is progressing smoothly, the Project Manager is responsible for giving deadlines to the team in accordance with the milestones and tollgates specified in proposal. These deadlines are for minor progressions such as report submission, source code development, testing etc. The main deadlines are the milestones and tollgates which must be used to compare the progress of the project with. Such a comparison not only provides a view of the progress of the project but also provides an opportunity for the team to anticipate any delays or fallbacks in the project timeline and take necessary steps to cover them.

Review of the differences between estimated and actual number of days can be shown as follows:

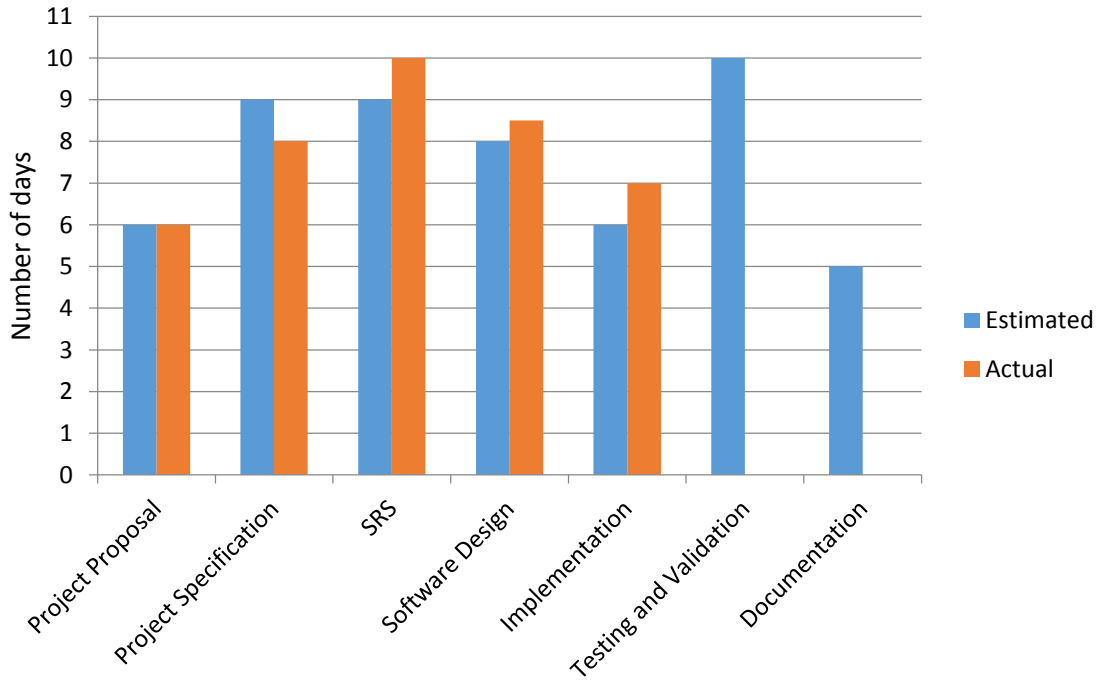

Figure6: Project progress tracking

9. Quality Control

This section discusses methods involved for monitoring the software development process and to ensure that the released product follows the defined requirements and standards. Quality control is done by performing quality reviews on the software, its documentation and the processes involved. Moreover, the product is checked for quality attributes such as maintainability, reliability, portability and usability. In a brief manner, these quality attributes look for optimum complexity, program size, length of user manual and number of procedures and errors.[2]

The key stages in this process are :

- Choosing components to be assessed such as source code, documentation, etc.
- Reviewing it by comparing by defined requirements and standards.
- Identifying any chances for improvements in the components.
- Make necessary modifications to improve component. For example, reducing size or length and cyclomatic complexity of the code.

10. Risk management

Making it easier to cope with problems and to ensure that these do not lead to catastrophic impacts of the project is the basis of risk management. This is one of the jobs of the Project Manager. Risk management comprises the following stages:[1]

1. Identification of risks whether that affect the project schedule or the quality and performance of the product being developed.
2. Analysis of the probability and impact of the risks.
3. Planning for avoiding risks or to minimize the effects of the risks.
4. Monitoring risks by continuous or periodic assessment and planning for mitigating the risks.

The possible risks according to priority have been tabulated describing the risks, their probabilities, affected components as result of the risk and impact of the risk.
Also, the risks require strategies to

- avoid the risks
- minimize the impact of the risk and
- prepare a contingency to deal with the risk

These are tabulated and presented in the following page.

[2] Crosby, *Quality Is Free*.

Risk Identification and Analysis			
Risks	**Probability**	**Affected Components**	**Impact**
Software tools cannot be integrated.	High	Project and Product	Serious
Changes to requirements that require major redesign are proposed.	Moderate	Project and product	Serious
Hardware along with configuration which is essential for the project is not delivered on schedule.	Moderate	Project and Product	Serious
Tools, which support the project, do not perform as anticipated.	Moderate	Project and Product	Tolerable
Faults in software components have to be repaired before these components are reused.	Moderate	Product	Tolerable
The time required to develop the software is underestimated.	Moderate	Project	Tolerable
Delay in delivering the software to the customer.	Low	Project	Serious

Table4 : Risk Identification and Analysis

.

STRATEGIES	
Risks	**Plan**
Software tools cannot be integrated.	Avoid this risk by checking compatibility of tools beforehand.
Changes to requirements that require major redesign are proposed.	Analyze proposed changes and required time and effort for necessary changes to be done.
Hardware which is essential for the project is not delivered on schedule.	Draw up necessary contingency plans for scheduling related tasks accordingly.
Tools, which support the project, do not perform as anticipated.	Accurate testing management and quality control.
Faults in software components have to be repaired before these components are reused.	Precise version management and system building.
The time required to develop the software is underestimated.	Investigate components and plan accordingly for covering up delay.
Delay in delivering the software to the customer.	Regular progress tracking.

Table5 : Risk management strategies

11. System Release Plan

This Section briefly explains the components involved in release plan of the project.

11.1 Testing Plan

To ensure that the customer receives the desired results, the system is finally tested as a whole after integration of independent system components.. The tests which will be carried out are as follows:

- **Diagnostic Test:** The Diagnostic test will be primarily carried out to ensure that the code performs according to planned architecture of the system. Also this test will help to detect possible errors in the code.
- **Scalability Test:** The devices in the system will be increased substantially and software will be implemented over these devices. This test will help to analyze the capability of the software in handling increased number of devices.
- **Real time Scenario test:** This test will primarily consist of two major parts. Firstly, adding new devices dynamically. Secondly, disconnecting devices from the network dynamically. This test will enable to understand whether the system is performing effectively in real time scenarios.

The Testing Manager is responsible for overseeing and implementing the test plan, using the built system. He will report errors, if any to the developers and project manager.

The time schedule for the testing plan will be as mentioned in section 6 of this document. It will be the final aspect of the testing and validation as part of milestone 4.

11.2 Packaging Plan

The software and documentation that are required for the system release will be bundled together in form .tar.gz archive.

The time schedule for the delivery of the package will be as mentioned in the section 6 of this document. It will be a part of Tollgate 7.

11.3 Documentation Plan

The documentation plan is divided into three subsections which are described below.

Installation Documentation

The document specifying the installation related to the software will be in form of a PDF file. It will cover the step by step process associated with installation and configuration of the software. Moreover the installation also mentions about installing any third party open source GNU-GPL software to be installed by the customer to facilitate the software.

The time schedule for the installation document will be in coherence with time plan which is mentioned in section 6. It will be as part of Milestone 5.

User Documentation

The user documentation will be in form of a PDF file. The user document will be primarily detailed to facilitate the end user. It will contain a brief description about the user requirements and the scope for which the software is applicable. It will explicitly mention about the different user modules present in the software. It will also describe the different steps to be followed by the user to reach the desired outcome using the modules present in the software.

The time schedule for the user documentation will be in coherence with time plan mentioned in section 6. It will be prepared as part of Milestone 5.

Developer Documentation

The developer Documentation will be in form of a PDF file. The document will clearly contain the function library, API and frameworks (if developed) during developing the software. The document will comprehensively describe about the new development, its need, implementation scope and functionality. It will also contain the related source code associated with these new developments.

The time schedule for the developer documentation will be in coherence with time plan mentioned in section 6. It will be prepared as part of Milestone 5.

12. References

[1]Ian Sommerville. *Software Engineering*. 8th ed., n.d.
[2]Crosby, P. *Quality Is Free*. New York: McGraw-Hill, 1979.