BLEKINGE TEKNISKA HÖGSKOLA

Monitorsing the Performance of Virtual Machines

TEAM: 'SHIELD'

HARSHINI NEKKANTI

HIMA BINDU NUTALAPATI

JYOTHI SPANDANA PENMETSA

NAVYA UPPALAPATI

PRIYASUBHA CHUNDRU

RAYWON TEJA KARI

SAIPHANI KRISHNA PRIYANKA KOLLURI

SASANK SAI SUJAN ADAPA

SRAVANI KANCHARLA

TULASI PRIYANKA SANABOYINA

VEERAVENKATA NAGA SOMESWARA MANITEJA DARISIPUDI

Document Type: **Project Specification**

Version 1.2

CONTENTS

1. Preface

The proposal aims at providing an outline of the project to be implemented in order to meet the requirements of the customer and CEO. The company comprises of the CEO and development team Shield.

The remainder of the document is organised as follows. Section 2 defines the technical terms and abbreviations used in the document. Section 3 gives a brief description of the customer's business environment and an overview of the customer's needs and challenges that will be addressed. Section 4 is the proposed solution which includes a proposal to meet the customer's expectations and to overcome the challenges. Section 5 states the limitations of the project. Section 6 includes a detailed time plan that states the work breakdown structure (WBS) and the time allocated for each stage in the project. Section 7 gives the Project Organization. Section 8 includes the version management, system building and release management. Progress tracking is included in Section 9. Quality control is dealt in Section 10. Section 11 includes the probable risks and strategies to manage the risks. Section 12 gives the system release plan and finally references are included in Section 13.

Customer: Patrik Arlos

CEO: Dragos Ilie

Revised version v1.2 on 2015-05-15

- Modified System Building and Release management according to the feedback from the CEO. Refer section 8.2, 8.3.

-Modified Risk1 as suggested by the CEO. Refer section 11.

Revised version v1.1 on 2015-05-08

- Changed the definitions of API, hypervisor and abbreviation of CN in Glossary and Abbreviations

- Coding assigned to all the team members in project organization.

- Configuration management updated. Refer section 8.

- Informal meetings added to Progress tracking. Refer section 9.

- Changes made to the figure in quality control. Refer section 10.

- Risk management, system release plan updated.

Initial version v1.0 on 2015-04-13

-Initial release.

2. Glossary and abbreviations

API: Application Program Interface

An API is a set of programming instructions and standards for accessing a web based software application.

CN: Compute Node

GUI: Graphical User Interface

Hypervisor:

Hypervisor is a hardware or a software that allows multiple operating systems to share a single host.

Monitoring:

Use of systems or processes that constantly oversee computer or network resources for the purpose of alerting personnel in case of outages, alarms, or other predefined events.

VM: Virtual Machine

Virtual machine is an operating system or an application environment that is installed on a hypervisor.

3. Background

Data centers today employ virtualization in order to support various applications that run simultaneously on server platforms. Virtualization creates virtual desktops which are hosted in the data centers. Virtualization helps in reducing energy consumption.

The customer is a data center provider who controls the pools of processing and network resources. The customer needs help in allocating their resources better by monitoring the compute nodes for:

- CPU load and utilization
- I/O usage
- network usage
- memory usage and
- disk usage

The customer lacks flexibility in identifying status of the compute nodes and requires a system where the status can be tracked through a simple web interface. In order to address this problem we develop a tool that enables the customer to monitor performance of the virtual machines. The product should include a dashboard that shows the status of all the devices

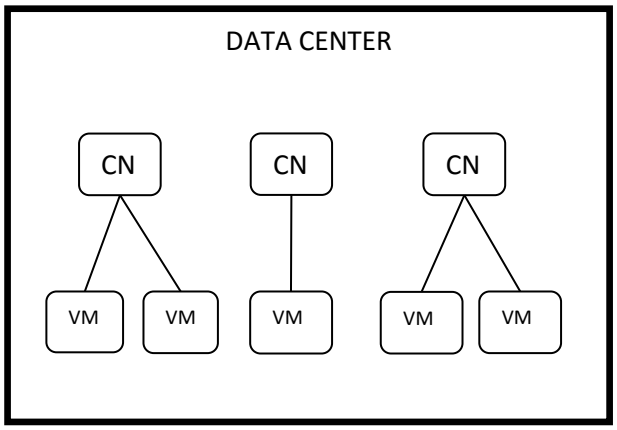


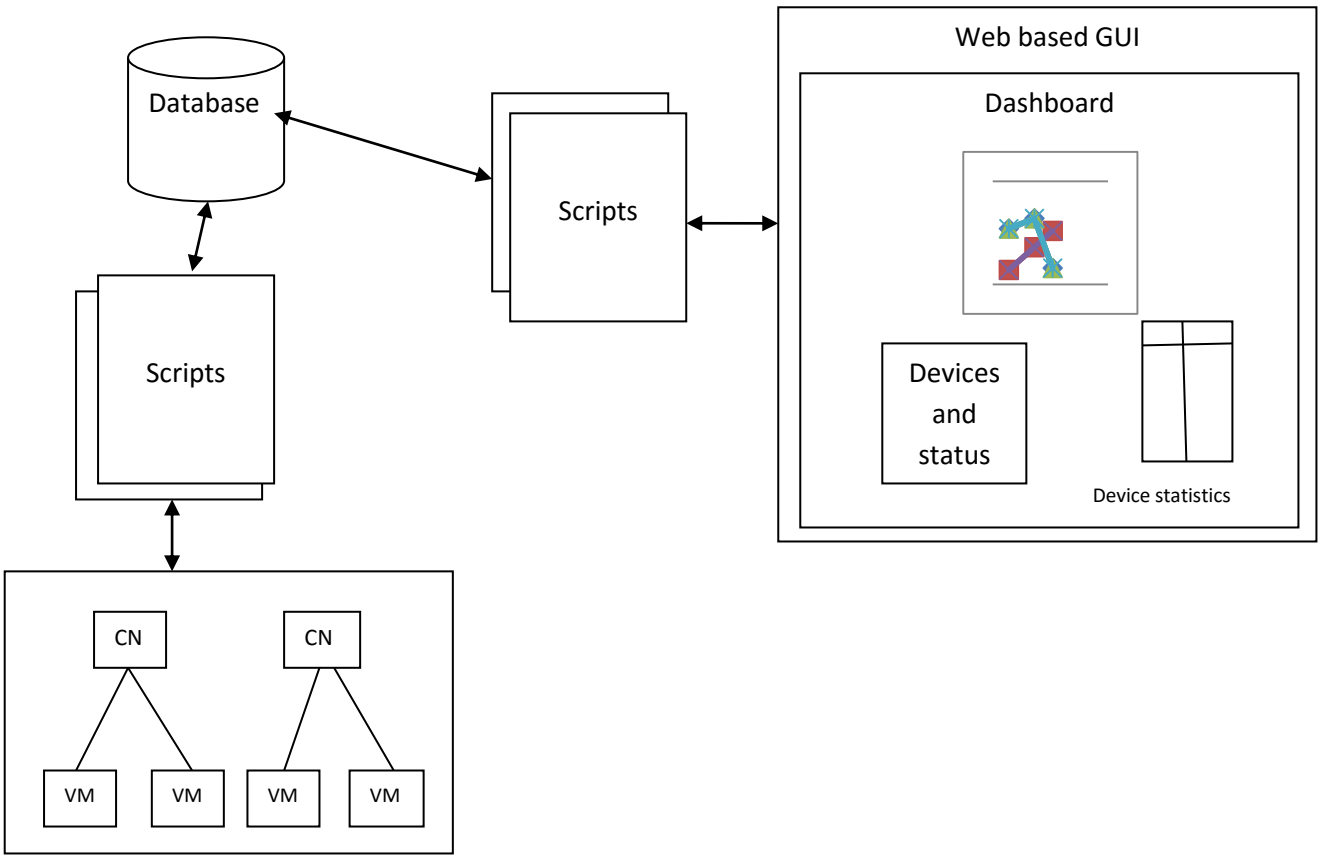
Fig. Block diagram of the customer's data center

4. Proposed solution

The basic requirement of the customer is to monitor CPU load and utilization, I/O usage, network usage, memory usage and disk usage of the virtual machines in a data center. The proposed solution would be to develop a tool to monitor resources of the virtual machines per host using hypervisors. The user will be able to

- Login to monitor the status of the devices.
- Add a network element to the monitoring list.
- Remove a network element from the monitoring list.
- View the statistics and graphs for a specific VM and a compute node.
- View alerts when any device crosses the threshold level.
- Logout.

In order to do so, the devices will be monitored through a web based interface where a simple user authentication is provided. Once authenticated, the customer can view a dashboard with the information of all the VMs and identify their status, whether normal, warning or critical. If the resource usage of any device crosses the threshold, a notification will be sent out to the customer. A time series aggregate of all the metrics will be provided which can be exported as a graph. A historical aggregate up to four weeks with relevant sampling intervals will be displayed. A restful API is used to import or export the data to be appropriate for the third party solution and graphical cross correlation between our data and the third party data can be performed. The solution should be able to handle a large number of nodes and should be vendor independent.



5. Limitations

The tool will only monitor the CPU load and utilization, I/O usage, network usage, memory usage and disk usage. Any other metrics are beyond the scope of the project.

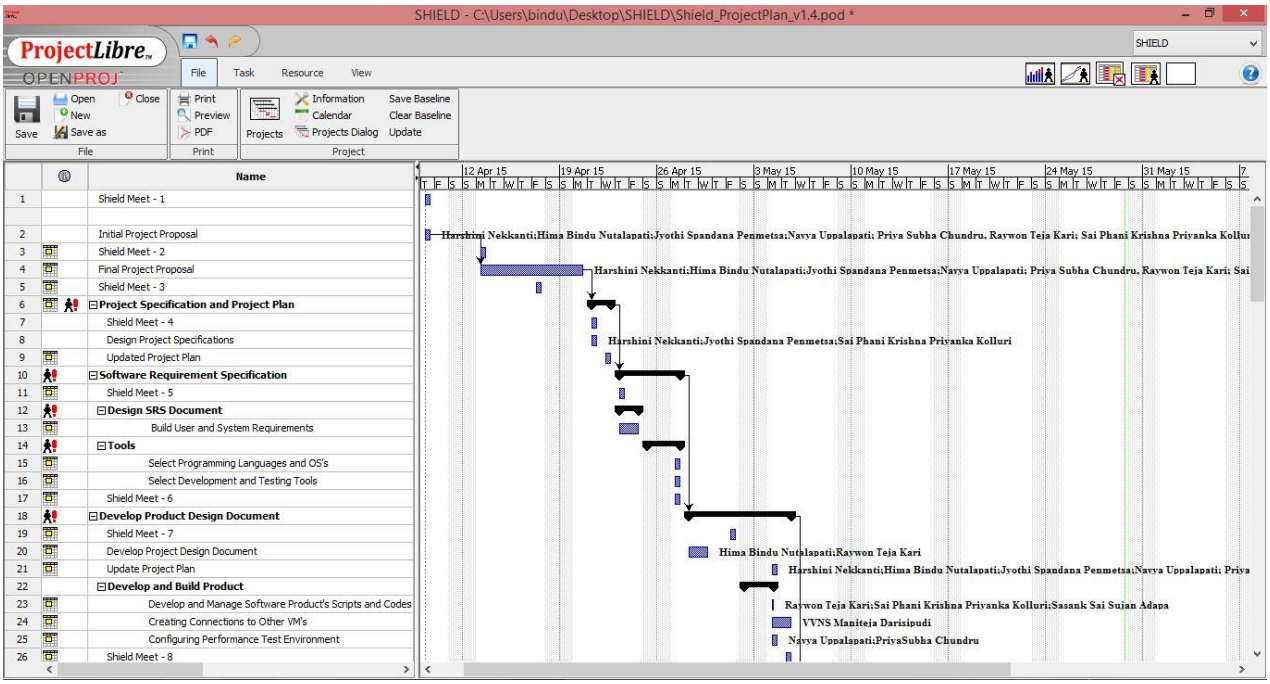
6. Time Plan

A work break down structure that includes the time allocated for each stage in the project is represented as a Gantt chart using Project Libre. Toll gates and milestones are documented.

S.No	Task	Estimated time (in days)
1	Project Allocation	
2	Project Proposal • Researching the topic • Document design based on the customer's requirement	3*
3	Theoretical Study and Project Plan • Research on the customer's requirements • Revising the project proposal	6*

4	Software Requirement Specification • Defining customer needs • Designing the system architecture according to the customer's needs	7*
5	Software Design and Implementation • Selecting appropriate programming language • Designing API • Writing code to implement the system architecture	6*
6	Acceptance Test Plan 1. Testing the code for errors and customer's work environment	7
7	Project Documentation	5*
8	Final Product Release	1

*Task completed



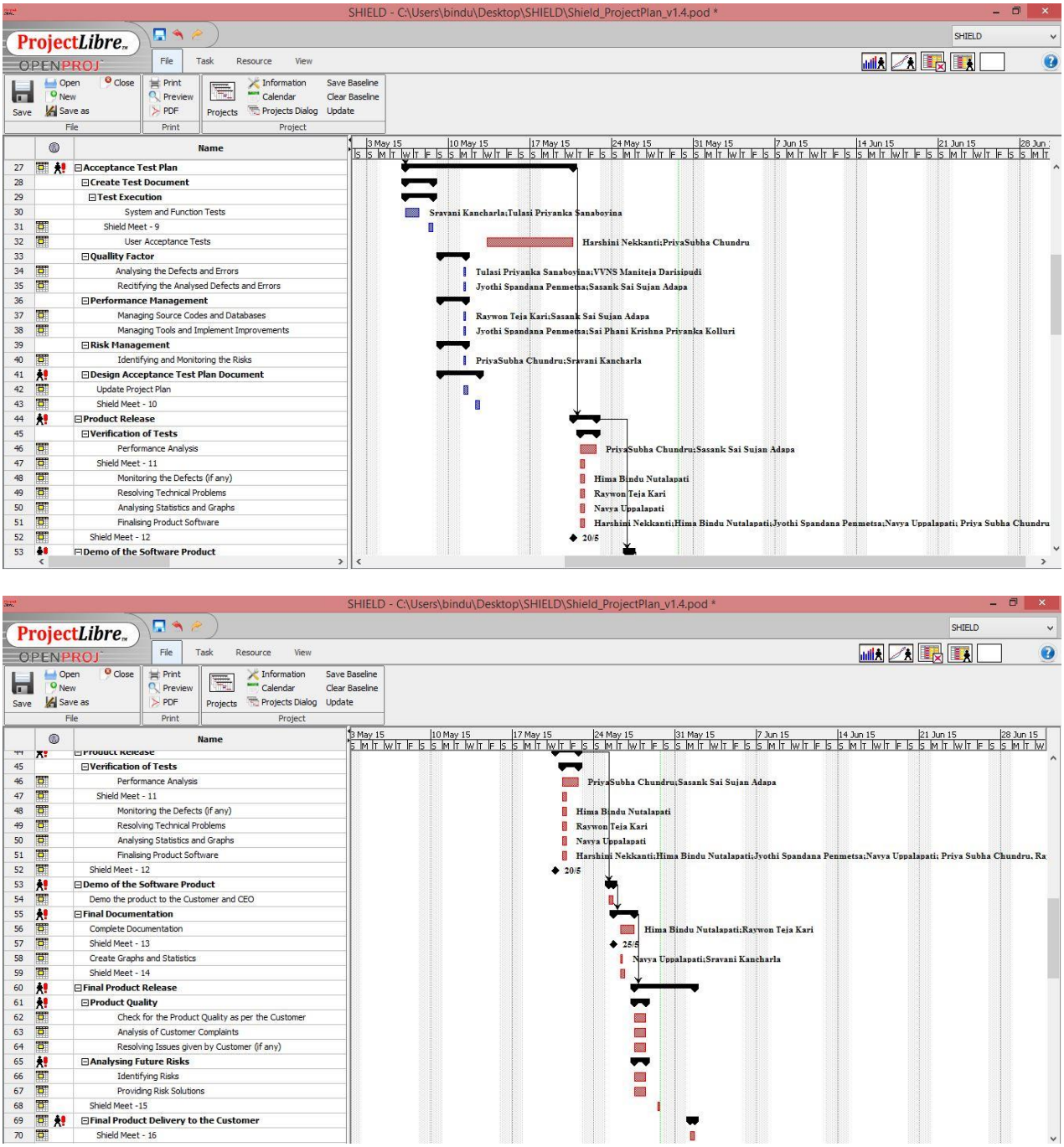


fig: Snapshot of the time plan from Project Libre

CHECK POINT	TASK	PARTIES INVOLVED	ESTIMATED COMPLETION TIME
Tollgate 1	Project Proposal	Customer and CEO	2015-04-13
Milestone 1	Project Plan	Customer and CEO	2015-04-20

Tollgate 2	Software Requirement Specification	Customer and CEO	2015-04-27
Milestone 2	Design documentation	CEO and Team Shield	2015-05-04
Tollgate 3	Acceptance Test Plan	Customer, CEO and Team Shield	2015-05-14
Milestone 3	Project Documentation	CEO and Team Shield	2015-05-20
Tollgate 4	Final Product Release	Customer and CEO	2015-05-28

table: Tollgates and Milestones

7. Project Organization

S.No	Task assigned	Team member
1.	Installations	Mani Teja Sasank
2.	Programming: a. User interface module: Displaying the CPU load and utilization, I/O usage, network usage, memory usage and disk usage. b. Data storage module: Storing the data retrieved using the data retrieval module c. Data retrieval unit: Retrieving the CPU load and utilization, I/O usage, network usage, memory usage and disk usage.	Bindu Navya Harshini Priya Subha Tulasi Priyanka Sravani Mani Teja Spandana Krishna Priyanka Raywon Teja Sasank
3.	Testing	Sasank Priya Subha
4.	Documentation	Bindu Raywon Teja

8. Configuration Management

i. Version Management:

The initial version of all the codes and documents is 1.0. Major or minor changes are made according to the customer's requirements and a revised version with incremented number is released. The changes are recorded and can be seen in the document's version history. E.g. Shield_Documentname_1.2 Shield_Documentname_2.0

Git is the version management system employed for the same. It is used to save and retrieve the various versions of the project. Git helps in keeping track of the source code when multiple developers edit the same file. Each developer has a local version of their files. When any changes are to be done, the developer uses 'git add . 'to add the local changes to Git. A description is added which indicates where the changes are made using 'git commit –m "Changes made in ... "and the changes are documented using 'git push'. When the other developers do 'git pull', the changes and conflicts (if any) with their local versions will appear.

ii. System Building:

System building is the part where the data, programming components are assembled, and compiled to get the executable system. It includes component version and system version management. The version management system manages the build process that involves checking out the component versions from repository and captures the data about the inputs and outputs of build process. Gitlab is used for version management.

Code lines give the sequence of versions of source codes that are derived from the earlier versions. This applies to components of the system as well. Baselines are the collection of component versions that build the system. The mainline has sequence of baselines, the release system, workspace that creates area where the software can be modified. After the system is build, executable tests are done.

The Software product is built using scripts, which are later merged as one single executable script, before the system release. The script is pushed onto GitLab. Git is a coding system, which is used to track the changes, push and pull from remote resources. Whereas, GitLab is a service, that provides remote access to git repositories. It makes use of the Git coding system to enable this service. Apart from the access, it also allows different users of GitLab to read the data from various user accounts.

iii. Release Management:

The system release is a version of the software system that is distributed to the customers. A system release includes executable code, configuration files and documentation. It will be tested with the tests mentioned in the acceptance test plan. The checklist includes the requirements and services as stated in the SRS. An executable file will be provided when releasing the product.

9. Progress Tracking

The development team keeps track of the progress through informal meetings, where the strategies to proceed with the task on hand is discussed. A project planning chart is prepared where the completed and ongoing tasks are listed. The chart includes the team members, tasks assigned, and status of each task. This helps the development team to be updated on the actual progress of the project.

Shield Meet 1 – 2015.04.10

Shield Meet 2 – 2015.04.13

Shield Meet 3 – 2015.04.17

Shield Meet 4 – 2015.04.20

Shield Meet 5 – 2015.04.24

Shield Meet 6 – 2015.04.27 (Meet with the customer)

Shield Meet 7 – 2015.05.01

Shield Meet 8 – 2015.05.05

Shield Meet 9 – 2015.05.08

Shield Meet10 – 2015.05.12

Shield Meet11 – 2015.05.15

Shield Meet12 – 2015.05.20

Shield Meet13 – 2015.05.25

Shield Meet14 – 2015.05.26

Shield Meet15 – 2015.05.29

Shield Meet16 – 2015.05.01

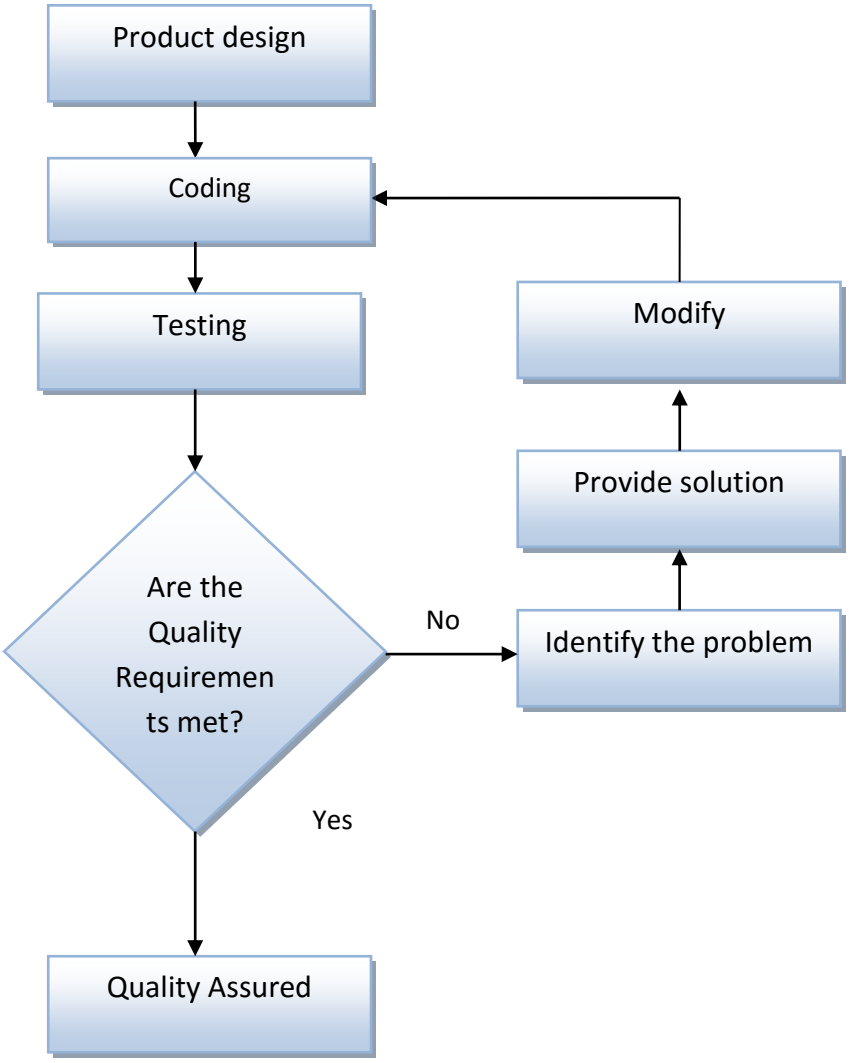
Team member	Task assigned	Deadline	Status
Mani Teja Sasank	Installing all the required software	2015-04-29	Completed
Spandana K.Priyanka Raywon Teja Sasank	Retrieving the device metrics from the VMs Storing the data retrieved into the database	2015-05-08	Completed
T.Priyanka Mani Teja Sravani	Developing the front-end GUI Linking database to the GUI	2015-05-08	Completed
Bindu Navya Harshini Priya Subha	Organizing the dashboard Producing graphs in the dashboard	2015-05-08	Completed
Priya Subha Sasank Raywon Teja	Testing the code for bugs Report the errors to the concerned developers	2015-05-13	Completed
Bindu Raywon Teja	Document drafting, formatting and reviewing	2015-05-17	Completed

10. Quality Control

Quality control evaluates whether the product meets the customer requirements. It is the responsibility of the testing team to identify failures/defects to ensure the product quality. The quality of the product can be controlled by

- Verifying if the user and system requirements are met.
- If a problem is identified, a solution will be developed to fix the problem.
- Modifications will be made in the design accordingly.

The figure illustrates the quality control of the product.



11. Risk Management

Risk management deals with identifying risks and developing the appropriate solutions to reduce their effect. The probable risks and strategies to overcome the risks are shown below.

1. Risk: Changes in customer requirements -This might affect the progress of the project.
   Strategy: The risk can be avoided by making the customer adhere to the proposal by having a signed project specification and SRS. Modifications if any can be traded for extra time.

2. Risk: Errors in the code - This might affect the product functionality.
   Strategy: This can be avoided by allotting a dedicated testing team to test the code frequently.

3. Risk: Self managing team - This might result in lack of proper progress monitoring.
   Strategy: The risk can be overcome by organizing frequent meetings where the team members can interact with each other.

4. Risk: Unexpected delays - Unexpected delays such as team member falling ill might jeopardise the progress of the project.
   Strategy: This can be overcome by sharing the work among the other team members and increasing the work time.

5. Risk: Code may not work as expected - This results in delay in the time plan.
   Strategy: This can be overcome by assigning testers to verify the code.

12. System Release Plan

12.1 Testing plan

The testing plan checks whether the developed tool meets the customer requirements. The functionality of all the components is tested before releasing the tool to the customer. The testing will be implemented in two phases.

- Build test: Build test will be carried out during the coding phase. All the modules are implemented individually and tested. Tests are performed to resolve the module interaction issues. Build tests ensure that the product can be built and installed successfully.

- Release test: Release tests will be carried out before the product release. The product will be tested for installation, GUI and other major tests on different platforms.

Time schedule:

Build test • Module tests • Module interaction test	 2015-05-11 to 2015-05-19 2015-05-19
Release test • Installation test • GUI test	 2015-05-20

12.2 Packaging plan

The final software package is delivered as a .zip archive which includes the software, installation and user documents.

12.3 Documentation plan

12.3.1 Installation documentation

The installation document is provided as a pdf file. It includes the details of

- Installation procedure
- Configuration settings
- Un-installation procedure

Time schedule:

Final documentation after testing: 2015-05-16

12.3.2 User documentation

The user document is delivered as a pdf file. The user document includes a brief description of the different modules employed to develop the tool and the step by step instructions for the user to access the tool.

Time schedule:

Final documentation after testing: 2015-05-19

12.3.3 Developer documentation

The developer document is intended for the developers to understand and improvise the existing tool. The document provides information on the API description, dependencies of the tool, source code organization, data base tables and the protocols used for establishing communication among the modules.

Time schedule:

Documentation throughout coding and testing	2015-05-11 to 2015-05-18
Documentation after testing	2015-05-20

13. References

- Sommerville, Ian. *Software Engineering,* 9th ed. Addison-Wesley, 2011