



SNAPP CLUSTER™

SNAPPTIX

Presented to:

Professor O. Eljabiri
May 7th, 2003

Presented by:

William G. Beeck
Sriram Polepeddi
B. L. Speiser

Adebayo Browne
Mehul Shah
Ryan Tolboom

Trademark Acknowledgments

Microsoft, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Java is a trademark of Sun Microsystems Inc in the US and other countries.

HP and Compaq and the names of HP and Compaq products referenced herein are either trademarks and/or service marks or registered trademarks and/or service marks of HP and/or its subsidiaries.

Linux is a registered trademark of Linus Torvalds.

Red Hat, the Red Hat "Shadow Man" logo, RPM, Maximum RPM, the RPM logo, Linux Library, PowerTools, Linux Undercover, RHmember, RHmember More, Rough Cuts, Rawhide and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc. in the United States and other countries.

Pentium is a trademark or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Table of Contents

1.	INTRODUCTION.....	6
1.1	PROJECT SCOPE (ABSTRACT).....	6
1.2	BACKGROUND.....	7
1.3	PROBLEM STATEMENT.....	10
1.4	PREVIOUS WORKS.....	12
1.4.1	<i>Windows NT® Enterprise Edition Clustering</i>	12
1.4.2	<i>Beowulf Clusters</i>	13
1.4.3	<i>Red Hat Linux Advanced Server</i>	14
1.5	METHODOLOGY.....	15
1.5.1	<i>Waterfall Methodology</i>	15
1.5.2	<i>RAD</i>	17
1.5.2.1	Incremental.....	17
1.5.2.2	Iterative.....	18
1.5.3	<i>Spiral</i>	19
1.5.4	<i>Scrum</i>	19
1.5.5	<i>COTS Approach</i>	21
1.5.6	<i>Process Evaluation Matrix</i>	24
1.6	GLOSSARY.....	25
2.	PROJECT MANAGEMENT.....	29
2.1	PROJECT TEAM AND ROLES.....	29
2.2	RESOURCE MANAGEMENT.....	30
2.2.1	<i>Work Breakdown Structure</i>	30
2.2.2	<i>Project Milestones</i>	31
2.2.3	<i>Gantt Chart</i>	32
2.2.4	<i>Pert Chart</i>	33
2.2.4.1	All Tasks.....	33
2.3	FEASIBILITY STUDY.....	33
2.3.1	<i>Function Point Analysis</i>	33
2.3.1.1	Inputs.....	33
2.3.1.2	Outputs.....	34
2.3.1.3	Queries.....	35
2.3.1.4	Files.....	35
2.3.1.5	Interfaces.....	36
2.3.1.6	Function Point Diagram.....	37
2.3.2	<i>AFP and COCOMO Analysis</i>	38
2.3.2.1	Adjustment Factors.....	38
2.3.3	<i>Net Present Value, Return On Investment, and Break-even Point Analysis</i>	41
2.4	RISK MANAGEMENT.....	46

3.	ANALYSIS.....	47
3.1	STAKEHOLDERS (WHO ARE THEY AND WHY ARE THEY STAKEHOLDERS).....	47
3.2	REQUIREMENTS GATHERING	49
3.2.1	<i>Interview</i>	49
3.2.2	<i>Observation</i>	59
3.2.3	<i>Use-Case Scenario Diagram</i>	61
3.2.4	<i>Brainstorming</i>	62
3.2.4.1	<i>Brainstorming Prioritization</i>	63
3.3	POST REQUIREMENTS GATHERING PROBLEM STATEMENT.....	63
3.4	REQUIREMENTS DEFINITION	64
3.4.1	<i>Functional Requirements</i>	64
3.4.2	<i>Non-functional Requirements</i>	66
3.5	REQUIREMENTS SPECIFICATIONS	66
4.	REQUIREMENTS MODELING.....	71
4.1	AS-IS DFD DIAGRAMS	71
4.1.1	<i>Context Diagram</i>	71
4.1.2	<i>General Diagram</i>	72
4.1.3	<i>Network Management – 2.x</i>	73
4.1.4	<i>Problem Analysis – 2.1.x</i>	74
4.1.5	<i>Determine Server Status – 2.1.1.x</i>	75
5.	DESIGN.....	76
5.1	TO-BE DFD DIAGRAMS	76
5.1.1	<i>Context Diagram</i>	76
5.1.2	<i>General Diagram</i>	77
5.1.3	<i>Running Application – 1.x</i>	78
5.1.4	<i>Update Configuration – 2.x</i>	78
5.1.5	<i>Prepare Restart Signal – 2.3.x</i>	79
5.1.6	<i>Report Status – 3.x</i>	79
5.2	PROCESS SPECIFICATIONS.....	79
5.2.1	<i>Structured English</i>	79
5.2.1.1	<i>Process 1: Clustering Software Starting</i>	79
5.2.1.2	<i>Process 2: Acting as Beta Unit</i>	81
5.2.1.3	<i>Process 3: Consider Alpha Promotion for this Unit</i>	82
5.2.1.4	<i>Process 4: Act as Alpha Unit</i>	83
5.2.1.5	<i>Process 5: Start Clustering Service</i>	84
5.2.2	<i>Decision Tree</i>	86
5.2.3	<i>Decision Table</i>	87
5.2.4	<i>Data Dictionary</i>	88
5.3	ERM MODEL.....	90
5.4	STRUCTURED CHART	92
5.5	STATE TRANSITION DIAGRAM	93
5.6	NETWORK DIAGRAM.....	95
5.7	REPOSITORY MODEL.....	96
5.8	SEQUENCE DIAGRAM – INSTALLING A SERVICE	97
5.9	ABSTRACT MACHINE DIAGRAM.....	98

6.	TESTING	99
6.1	FUNCTION TESTING	99
6.2	PERFORMANCE TESTING	99
6.2.1	<i>Compatibility tests</i>	99
6.2.2	<i>Configuration tests</i>	99
6.2.3	<i>Security tests</i>	100
6.2.4	<i>Timing tests</i>	100
6.2.5	<i>Recovery tests</i>	100
6.3	ACCEPTANCE TESTING	100
6.3.1	<i>Alpha testing</i>	100
6.3.2	<i>Beta testing</i>	101
6.4	INSTALLATION TESTING	101
7.	WORKS CITED	102
8.	LIST OF TABLES AND FIGURES.....	104
9.	EXTRA WORK.....	105

1. Introduction

1.1 Project Scope (abstract)

It can be stated that the average small business has a need for reliable data security. The options a small business might consider would include: a Microsoft solution, which would incur a major expense for a small business, or a Linux solution in which the major expense would be the salary of the IT consultant needed to set up and maintain it.

Businesses cannot afford periods of downtime due to system failure. Because small businesses typically only have one or two file servers, the ability to insure against system failure is extremely important. If the core operations server goes down, the impact on a small business would be devastating. Given that possibility, the value of clustering becomes so apparent it makes not using one almost irresponsible. If one system in the cluster stops working for any reason, another system in the cluster can safely bear the burden of the work while the original is being repaired (or replaced).

Finding the funds for clustering is the bane of most chief financial officers of small businesses. Including the price of modern day clustering hardware, a typical cluster can cost as much as ten thousand dollars. Add to it the cost of the operating systems – particularly a Microsoft solution – and the final cost can exceed the available IT budget even before including the salary of the expert hired to run it.

In the future, a school can pull out two old unused 266 MHz Pentium® computers, and after installing the software from this project, in an hour; they

have a functioning file server cluster. With a few more clicks of a mouse, they can even set it up as an in-house mail server.

Bringing a simplified clustering solution to market may give many businesses the ability to run mission-critical systems like email safely without complex and prohibitively expensive systems. With a drastically less total cost of ownership than proprietary or even currently available open source enterprise platforms, our project could revolutionize the way older computers are used. By converting them into a powerful cluster, businesses that could not previously consider undergoing IT improvements because of the back-end costs would now have the opportunity to remain competitive.

1.2 Background

The Torah Academy of Bergen County (TABC) is an educational institution, which is becoming more and more dependent on services that are supported via a network. Email and file servers are good examples of services that are supported over a network. For email to work over a network, there is a central computer running a program that allows authorized users access to their personal email account and permits them to view and send messages. The user account could be accessed from any computer within the school or via the Internet.

File servers have directories setup on a central computer that allows people to save, view, and update files that are of interest to others in the school. Sharing a class calendar is just one example of a file that would be available via the file server service.

It is not surprising to note that files (data) stored on a computer might be lost – meaning that the data cannot be retrieved. There are a number of reasons for this - the motor that is spinning the disk drive might malfunction, a virus might infect the computer, or, a person might delete a file by accident. In all of these situations, it will take time to restore the central computer to a state where it will be accessible to everyone. To restore a deleted file could take a couple of hours while replacing a disk drive or cleaning an infected computer might take a few days. In both of these cases, TABC doesn't have the resources available to address either of these issues. For example, the CBL Data Recovery Company can copy 20G (gigs) of data from a damaged disk drive and ship the recovered data within twenty-four hours at the cost of approximately one thousand two hundred dollars.

Losing the services of a central server for as little as an hour presents an organization such as TABC with tangible and intangible expenses created by the downtime. To help reduce these costs, organizations typically implement fail-over systems, where if one server goes down another server picks up the former's responsibilities. For example, if the power supply on one email server overheats and shuts down, another server will take over providing email services. This setup tries to prevent hardware failures from causing a lapse in network services.

The fail-over concept has proven to be very reliable, but it is expensive to implement and maintain. It requires a set of computers connected together forming what is known as a cluster. Each computer in the cluster must be a clone of each other – they must have the same hardware and software installed

since if one of the computers stops working, any other computer in the cluster could take its place. A cluster also requires specialized hardware to support the many users and the fail-over concept – a large array of hard disks. Having the array of disks available via the network and separate from any one computer within the cluster guarantees that when a fail-over occurs, there is no question that the data stored on those disks is current. For example, since each computer, or node, of the cluster uses the network storage to access email information, if one node goes down and another node starts up – the new node will provide the same email data from the network storage device as the node that just went down.

Many of the utilities used to set up such a cluster and monitor performance are designed with the expectation an experienced system administrator and/or a network specialist is going to use them. The companies that have clustering solutions on the market today assume that the institutions considering their clustering solution have a full time network staff that is familiar with many of the concepts of network administration. These companies have provided a large number of options that make their clustering solutions very powerful – and the administrator must be knowledgeable enough to know what options to use and how to configure them.

Small business and schools, such as TABC, are not exempt from the responsibility of providing reliable network services to their employees/students. TABC could benefit immensely from the stability a fail-over cluster affords, but they cannot afford the expensive hardware and software required to implement any of the existing commercial fail-over clusters. The problem is to meet the

reliability needs of small organizations by implementing an easy to install and simple to use fail-over clustering system while keeping the costs of the project to the bare minimum.

1.3 Problem Statement

The goal of this project is to design a clustering system that addresses the basic issues of small organizations. They require a system that is inexpensive, reliable, easy to install, easy to maintain, and that has very little user (admin) interaction. They also require that basic services such ftp, file serving, and email will run on the cluster as seamlessly as they would on a single machine.

We aim to address the easy installation issue by writing turnkey installation code that automatically installs the required components of our system on each of two machines with little to no user prompting. We will setup one system to start up as an ALPHA server, the main system that offers services to the users. The second will be configured to start up as a BETA, which must take over the services if the ALPHA ever fails. The user will not be able to tell which of these systems is providing the services, as it will appear as one system to computers outside of the cluster.

We will provide an easy to use graphical user interface (GUI) for the person administering the cluster. This GUI will have a stoplight-type of alert which visually informs the administrator of the health of the cluster. Some key factors shown by this alert are: which server is currently ALPHA, whether or not the Heartbeat Protocol working properly, etc. The GUI will also have a services tree, which dynamically updates the status of the ftp, file serving, mail and other services running on the cluster. The GUI will also allow the administrator to

setup users authorized to use the ftp and mail services, the IP addresses used within the cluster, and to view the system logs.

We look to make the system reliable by taking widely tested open source clustering code and tailoring it for a cluster for a small organization. The benefit of using open source is that no royalties are required for use of the code, hence reducing the cost of development to next-to-nothing. We will be building our system using the open-source operating system, Linux, as this will ensure that whatever code we build upon will have already been widely tested in many environments and situations. Linux being an open source project is also well documented with a large number of people able to answer questions or fix problems related to Linux. With all of this available support, designing and implementing extensions to Linux are very straightforward. This support will be handy when TABC decides to maintain our re-designed cluster-compatible ftp, file, and mail servers. To maintain the security of the system, we will be designing our own authentication file with a special encryption and hashing algorithm.

We will be building our system on the personal computer (PC) architecture, as it is by far the cheapest hardware platform available today. Due to its widespread use throughout the world, PC hardware costs much less than proprietary hardware solutions. PC hardware is also tested by a larger market and therefore stability statistics are often more readily available. Likewise, performance statistics are also better understood and it is easier to make a cost/performance analysis to determine what kind of a system meets an

institution's needs. Also, TABC has abundance of old PCs that can adequately run our system.

The main functionalities of this system such as synchronization of the two computers and seamless transfer of services from the non-working system to the working system will be invisible to the user. The only interaction our system will have with the user are the GUI and the re-designed ftp, file serving and email services.

1.4 Previous Works

1.4.1 Windows NT® Enterprise Edition Clustering

This is a very robust solution by providing load balancing, live backup, fail-over security, remote management, active/active clustering (both alpha & beta units are contributing processing power to the cluster).

The more robust a solution is the higher the cost to implement and support year to year. In a typical solution, the servers cost the most – about twenty thousand dollars for the two of them. The lowest costing Microsoft Enterprise operating system that would be used is Windows NT 4.0 Enterprise Edition that has a list price of four thousand dollars – so the cost for the two servers is eight thousand dollars.

Assuming that the servers and operating system were purchased from Hewlett Packard – HP has a CarePac™ service that offers to bring the hardware and setup a fully functional 2-unit Windows NT 4.0 cluster starting at four thousand three hundred twenty dollars.

As an additional service, HP will evaluate the IT environment for the small business to determine the effectiveness of their backup and recovery

procedures. As a final service, HP can provide a plan to correct any deficiencies found during the evaluation phase. These additional services for a small business are listed on the HP web site as costing twenty four thousand eight hundred dollars for the evaluation and sixty two thousand five hundred dollars for the plan.

For a complete solution for any of the project stakeholders, the one time costs are one hundred nineteen thousand six hundred twenty dollars. For this reason, we are rejecting this solution; we believe a much simpler and less costly alternative exists. ^{1,2,3,4}

1.4.2 Beowulf Clusters

The Beowulf cluster is probably the best-known Linux based scientific clustering solution available today. A Beowulf cluster is actually not just a single piece of software, but many pieces that when put together comprise a clustering system. There are also numerous changes to the Linux kernel that allow for better operation of the cluster, such as Enhanced/Optimized network drivers, virtual memory management systems, and distributed inter-process communication (DIPC) services.

An advantage of a Beowulf cluster is that it supports a vast array of communications hardware. Writing programs to run on the cluster are straight forward as there are many interfaces available. Such as Message Passing Interface (MPI), and Parallel Virtual Machine (PVM). Since the Beowulf cluster is also an open source project, acquiring the various software modules and obtaining technical support is both easy and inexpensive. There are also a large

number of sources that document the use of Beowulf cluster (Books, internet sites, FAQ's, etc.).

So, why is this type of cluster not a good choice for our project? Foremost, the Beowulf cluster is a scientific cluster, meaning that it is used for crunching numbers, not for fail-over protection. While there is plenty of technical support for this type of cluster; installing, configuring, and maintaining the cluster requires a high level of Linux and open source knowledge – neither of which are readily available to the stakeholders of this project. Finally, in order to make use of the cluster, the various network services (e.g. email, file servers) have to be written to make use of the cluster and that effort is beyond the scope of this project. ^{5,6,7}

1.4.3 Red Hat Linux Advanced Server

Red Hat Advanced Server contains a high performance clustering solution. This clustering system contains many useful features for an enterprise environment. Notably, it has a fully shared storage subsystem. This means that the arrays of hard drives are separate from the computers themselves, and support the latest technologies such as Fibre Channel. Red Hat is the leading company that distributes Linux and other Linux related solutions, so support and documentation from them is of high quality.

Red Hat's Advanced Server would be overkill for what the stakeholders of this project need. Fibre Channel implementations are expensive even before factoring in the cost of the storage units that the channels will connect to. Despite being a versatile solution, it is not cost effective for the various stakeholders that require keeping hardware and software costs to a minimum. ⁸

1.5 Methodology

A large amount of research has been done over the years to improve the software development process – to bring a project in on time, within budget, with high quality, and to actually solve the customer’s problem. There are many types of methodologies a project team can use to develop an information system. Methodologies systematically implement a procedure for developing projects. Given all the variables that can accompany a project, choosing the right methodology is critical to a good final product. We have looked at the following methodologies for use in our project and appraised them in the Evaluation Matrix at the end of this section.

1.5.1 Waterfall Methodology

The waterfall methodology was one of the first methods to define how a project should be approached, thus giving the project a better chance of being delivered on time, within budget, with high quality, and meeting the customer’s needs. The waterfall is a very common methodology, which system analysts use for their projects. In waterfall methodology each phase flows naturally into the next phase like water over a series of falls. There are different types of waterfalls. The most common one is the traditional version, which starts at the top and goes to the bottom in one direction. The second one is the bi-directional type, which means anyone can go back and forth at any stage of the process. Another type is V-shaped waterfall where the processes start at the top stage, and come to the bottom level and again goes to the opposite side. These are the three main types of waterfall. There are some other less used waterfall methodologies for example: NASA, DOD, and MIS. Significant numbers of software systems have been built using this methodology and have succeeded.

Among all of these waterfalls methodology the most commonly used is traditional waterfall (Diagram Attached Below). The traditional waterfall methodology has six phases which includes project initiation, planning, analysis, design, implementation, testing and ends with maintenance. A milestone marks the end of each phase and this must be achieved in order to proceed to the next step. The traditional waterfall is very powerful, fundamental and is the simplest methodology for most projects.

While this methodology is being used today, there are some problems with the method. For instance, the amount of time it takes to get from the project identification phase to the maintenance phase can be very long since a phase cannot be started until the previous phase has been completed. Also, the amount of documentation created when using this methodology can be overwhelming and the documents cannot be used by a fourth generation tool that could help in the design or implementation phase of the project. Finally, since the documentation is not pictorial in nature, the text can be easily misinterpreted by any (or all) of the people working on the project, including the stakeholders.

Originally, the waterfall was a one-way process – always going forward to project completion. Newer waterfall methods have added feedback loops at the various phases of the project. This allows issues that are raised in one phase, to stop the project and have a previous phase address the issue. For example, the testing phase is not designed to resolve requirements issues – the testing phase is designed to test the project to make sure it meets the stated requirements.

To reduce the chance for misinterpreted requirements and to speed up the whole process, standard pictures and graphs should be used to show flows of

data, module relationships, and attributes associated with the various parts of the project. From these standard pictures and graphs, fourth generation languages can be used to create the skeleton for database structures or even the actual code.

1.5.2 *RAD*

Stakeholders have been requiring that projects be delivered much quicker than in the past – and as the amount of time between the project start to the project delivery has been compressed, the need for a development methodology to support this type of project has increased. A number of new methodologies have been created from the waterfall methodology that can be classified as rapid application development (RAD) that try to address the main issue of a compressed project timeline.

The newer methodologies accomplish this by acknowledging several factors: that an initial system need not necessarily be expected to do everything the stakeholder wishes it to; that stakeholders must be part of the design process in order to meet as many of the stakeholders' needs as possible; that an iterative process is best at accomplishing both of these factors. We'll briefly explore two RAD methodologies as examples – the Incremental Model and the Iterative model.

1.5.2.1 *Incremental*

With the incremental model, the project can be broken down into multiple smaller applications and delivered to the stakeholders at different times. Once a large project is broken into smaller applications, they can then be managed as separate projects – have their own timeline, have requirements gathered, designs written, code written and tested. And since they are smaller, it's

assumed they can be built and tested faster. Depending on the availability of resources, the small projects can overlap each other so while the stakeholder receives the final project later than originally planned, the delivery would be much earlier than if the project was done all at once. For example – if the project was a management system for a movie theater, the first application could be keeping track of the film schedule and ticket sales; the second application could be tracking inventory for the snack bar and sales at the snack bar; the third application could be an accounting package to produce employee pay checks, time cards, etc.

1.5.2.2 *Iterative*

The iterative model is much like the incremental model – where only the most important requirements are addressed in the initial release of the project and subsequent releases add more functionality. The difference between the incremental model and the iterative models is that the iterative model has the complete project delivered to the stakeholder in the initial release. This method gives the stakeholder the ability to acquire a trimmed down project more quickly, thereby allowing them to provide feedback on how the project meets or doesn't meet their needs.

Using the movie theater example from above, the project is delivered with all three applications available to the stakeholder. However, in the first release, the application keeping track of the film schedule is a manual process and only shows what films are playing now; and the snack stand application doesn't keep track of inventory, only the sales at the snack bar and the accounting package can only print checks and keep a ledger.

The second release of the application would have the application tracking the film schedule changed to support the schedule as posted by the various film distributors; the snack stand application enhanced to keep track of the inventory; and the accounting application changed to support tracking employee time cards, and calculating taxes.

The Iterative Methodology works much like the trial and error process that a programming student goes through in the implementation of a programming assignment. In this methodology, phases are repeated until the required results are reached. This type of methodology is usually used when quick prototyping and development deadlines are required for a project. (Hoffer, 18.) The Iterative Methodology, also known as the Built-And-Fix Model, is used in many consumer-end and home-based software products. The product is built, and it is reworked or re-patched as many times as needed in order to satisfy the customer.

1.5.3 Spiral

Companies building applications that have the requirement to minimize failure or loss usually use the spiral methodology. Although the spiral methodology may look very different on a graph board than that of the waterfall methodology, they are in fact very similar in the implementation process with the exception that each phase of this model it is preceded with a risk analysis. The usual case in the spiral methodology is that the risks must be resolved. If the resolution to these risks is not possible, then the choice of terminating the project is made. (See Risk analysis, section 2.4 for more details on risks.)

1.5.4 Scrum

Scrum is an iterative, incremental process for developing any product or managing any work. It produces a potentially shippable set of functionality at the end of every iteration. It is used to manage and control development work with its team-based approach to developing systems and products. It is a way to improve communications and maximize co-operation by detecting and removing anything that gets in the way of developing and delivering products, thus maximizing productivity. This process is scalable from single projects to entire organizations, and has been used to control and organize development and implementation for multiple interrelated products and projects with over a thousand developers and implementers.

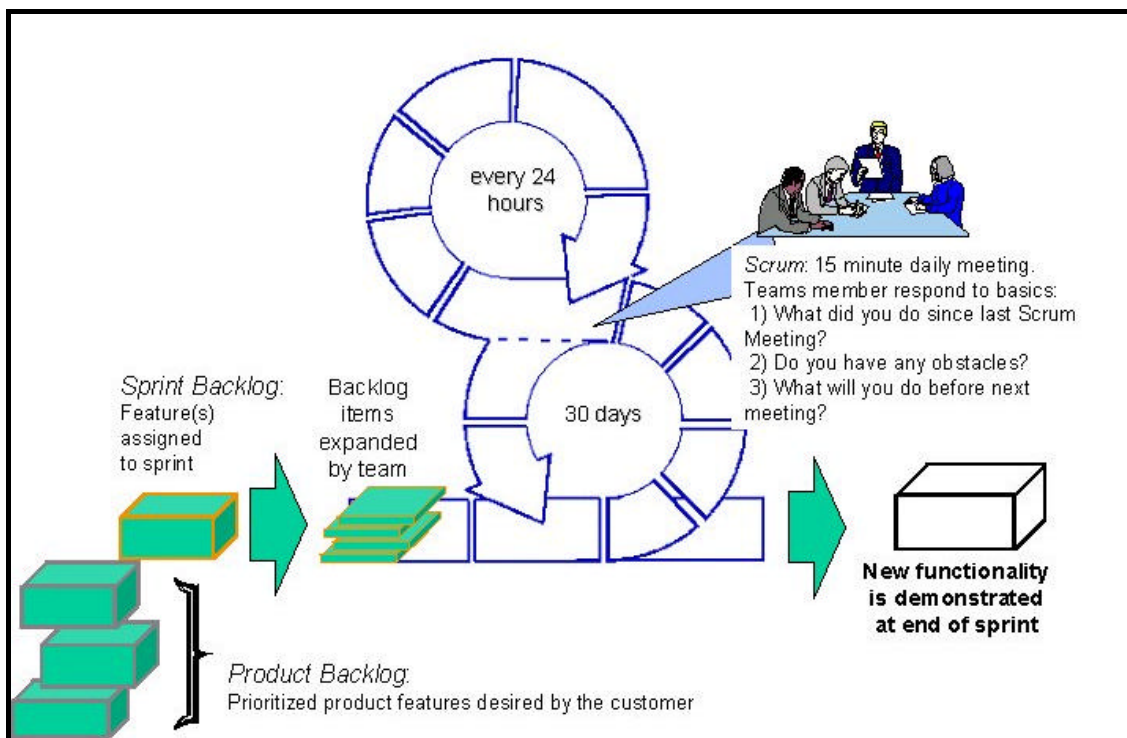


Figure 1: Scrum diagram

Scrum seeks to focus an entire organization on building successful products. Without major changes - often within thirty days - teams build useful, demonstrable product functionality. It can be implemented at the beginning of a

project or in the middle of a project or product development effort that is in trouble.

Based in modern process control theory, Scrum causes the best possible software to be constructed given the available resources, acceptable quality, and required release dates. Useful product functionality is delivered every thirty days as requirements, architecture, and design emerge, even when using unstable technologies.

Over fifty organizations have successfully used Scrum in thousands of projects to manage and control work, always with significant productivity improvements.

(<http://www.controlchaos.com/Scrumo.htm>)

1.5.5 COTS Approach

The goal of developing systems to be better, faster, and more cost-effective continues to drive software engineering practitioners and researchers to investigate software engineering methodologies. In requirements engineering, the focus has been on modeling the software engineering process and products for systems that are being built from scratch. As the size and complexity of systems continues to grow the use of “commercial off-the-shelf” (COTS) components is being viewed as a solution. Effective use of COTS components, however, requires a systematic approach that provides both a set of concepts for modeling the subject matter and a set of guidelines for using such concepts.

There are many benefits to such an approach. A COTS approach supports developing a system that satisfies the customer. The use of COTS products also impacts the architecture, or high-level design, phase of the

software development lifecycle. In more traditional software development lifecycles, the architecture is developed to fulfill the native requirements of the system without regard to the availability of COTS components. Using this approach, there may be few or no available products that fit within the chosen architecture. To maximize the use of COTS components, the architects need to consider the trade-off between the best available products on the market or the need to develop new code to adequately serve the customer's requirements.

In comparison to an approach that does not consider the use of COTS products, this approach may use more time in the requirements analysis phase. When considering COTS, the engineer needs to search through the COTS specifications to match and select among the components. The benefits of using this approach are expected to appear near the end of the software development lifecycle, in the detailed design, implementation, and unit testing phases. As a result, the evaluation of the COTS approach needs to be measured with respect to the overall development lifecycle effort.

The actors in this process model are the customer and the requirements engineer. In this description of the model, the customer is contracting the development of a large-scale system and is involved in the development work.

Variations of this model can be developed in the future to consider high volume, shrink-wrap product development. The customer's overall goal is to receive a system that solves a particular need. Sub goals for the development team include receiving product development artifacts (product goals, system requirements and software requirements) and product planning artifacts (quality plan, test plan, schedule, etc.). The customer's soft goals include receiving a

system that is delivered on schedule, within budget, and with high quality. The requirements engineer depends on the customer to validate the system as it is being developed and to 'ideally' provide complete and correct information.

(<http://www.utdallas.edu/~chung/ftp/MBRE.pdf>)

1.5.6 Process Evaluation Matrix

Evaluation Criteria	Weight	Waterfall	Spiral	Iterative	Incremental	Scrum	COTS
Risk management	8	4	6	8	4	7	8
Object oriented implementation	5	1	2	4	5	2	5
Size/Complexity Management	10	3	7	6	6	9	9
Development time	10	3	7	9	7	9	9
Cost Management	20	3	3	14	10	5	20
Reusability of project elements	12	7	8	8	6	9	9
Ease of planning	8	7	5	8	4	7	8
Handling of Customer Feedback	12	2	8	11	10	12	7
Business Value	15	10	11	13	15	15	15
Total	100	40	57	81	67	75	90

Table 1: Methodology matrix

After reviewing these methodologies we have found that there is more than one methodology that would be a good fit for our development project. The two most appropriate methodologies are the Iterative and COTS models. We will likely come up with a hybrid approach, pulling together the best aspects of these two. For example, the COTS approach works well with our intention to build on top of existing open source clustering software, while the Iterative model gives us

the proper approach for developing evolutionary prototypes through which we can get better customer feedback.

1.6 Glossary

For terms that are defined well on the Internet, a hyperlink to those sites has been included where a detail description (and pictures) can be found.

Term	Definition
AFP	Adjusted function points. Used during feasibility studies to determine the potential size of a project.
Alpha Collision	This is an error that one must be concerned with in the implementation of fail-over clusters. It occurs when two or more computers claim to be the alpha unit. The cluster implementation should either strive to eliminate the possibility of alpha collisions, or require that a quorum vote declare the alpha unit. Alpha collisions can also cause IP collisions with units inside the cluster.
Alpha Unit	The computer that is currently running the fail-over cluster. In many cases and implementations, the alpha unit is the unit that is running all server applications.
Apache	The name of a popular public domain web server.
API	Application Program Interface
Beta Unit	The computer that is checking to make sure the alpha unit is still alive. Theoretically, this can be extended to multiple Beta units.
BNF	Backus-Naur Form
Cluster	A group of computers linked together to simulate one computer system. The group can be either a scientific cluster or a fail-over cluster.
COCOMO	Constructive cost model – a method for estimating the size of a project.
COTS	Commercial off the shelf - software bought from a store.
CPU	Central Processing Unit.
Daemon	A UNIX program that runs without needing constant user input.

Table 2: Glossary

DFD	Data flow diagrams. A way to visualize the flow of data within a project.
DHCP	Dynamic host configuration protocol - assigns IP address as needed.
DIPC	Distributed Inter-Process Communication
DRDB	The program DRDB sends a request over the web to the on-line cluster to execute the DRDB-LIST command.
ERM	Entity relationship model
Fail-over Cluster	Cluster is used to ensure system reliability. If one computer in the cluster goes down for any reason, another unit in the cluster will take over the responsibility of managing the system. Typically, fail-over clusters require that its alpha and beta units be exactly identical in hardware. Gamma units can usually differ. This all depends on the implementation.
Fibre Channel	A high-speed connection between a storage device and a server.
FTP	File transfer protocol.
Gamma Unit	The computer that is used to make an odd number of units in the Quorum. A gamma unit cannot become an alpha unit and is reserved for the purpose of deciding the quorum vote. This unit is not always required; it depends on the implementation of the clustering software.
Gantt	A chart used in project management to visualize a project timeline.
Giga	A billion.
GNU	GNU's Not UNIX
GPL	GNU General Public License
GUI	Graphical user interface.
Heartbeat	The communication between computers in a fail-over cluster that tests to see if any part of the cluster has gone down.
Hyperlink	A way to link objects.
IMAP	Internet messaging access protocol.
IMAPD	Courier-IMAP server that provides IMAP access to maildir mailboxes.
IMP	Interface message process.
Internet	A large network of smaller networks.

IP	Internet protocol.
IP Collision	This is an error that occurs at the TCP/IP level when any two computers on a network (inside or outside a cluster) share the same IP addresses. At best, the consequences can result in having one of the machines go offline. At worst, IP collisions can take down a whole section of the network.
IT	Information technology.
Kernel	The central part of a program.
LAN	Local area network.
LOC	Lines of code. Used in feasibility study to determine the estimated size of the project.
MB	Megabyte - a million.
MHz	Megahertz – measures transmission speed of electronic devices.
MPI	Message Passing Interface.
Network	A system that allows data to be passed from one user to another.
OS	Operating system.
Pentium	A series of CPU chips made by the Intel corporation.
Pert	A chart used in project management to visualize a project timeline.
ProFTPD	Highly configurable GPL-licensed FTP server software.
PVM	Parallel virtual machine.
Qmail	A UNIX based email program.
Quorum	The entire group of units in a fail-over cluster that has an odd number of units. With an odd number of units, the cluster can cast a decisive <i>quorum vote</i> to determine which computer is the alpha unit, with the gamma unit forcing the decision. Not all clusters require a Quorum.
RAM	Random access memory.
Red Hat	A software company specializing in the Linux operating system.
RS232	Recommended standard 232.
Samba	A UNIX based file server program.
Scientific Cluster	A cluster that is used to simulate a very powerful computer – using each computer to aid in the solution of a larger computational problem.
SCSI	Small computer system interface.

SMB	Server message block.
SMTP	Simple mail transfer protocol.
TCP/IP	Transmission control protocol / internet protocol
UNIX	A computer operating system.
VORD	Viewpoint – Oriented Requirements Definition, A method to define requirements.
WBS	Work breakdown schedule – used during the project management phase of a project.

2. Project Management

2.1 Project Team and Roles

- ? Project Manager William G. Beeck
 - Organize the project as a whole. Make sure deliverables are complete and handed in on time.
 - Lead the team in the project document.
 - Work with each member to make sure they have work to perform and consider themselves as a valuable asset to the team.
 - Be the focal point to resolve project related issues.
- ? System Analyst Sriram Polepeddi (Ram)
 - Requirements gathering / modeling – interacting with the sponsor and various stakeholders to determine what the requirements are for the project.
 - Lead the team in creating the user manual
- ? Architectural Designer B. L. Speiser
 - Responsible for the architecture for the software solution.
 - The lead programmer for the middle layer of the solution.
 - Manage the workload and schedule for the programmers and front-end designer keeping the project manager informed of status.
- ? Front-End Designer Adebayo Browne
 - Work with the system analyst and the architectural designer to design the front-end.
 - Implement the front-end allowing for multiple iterations as the requirements are changed.
- ? Programmer Ryan Tolboom
 - Work with the architectural designer to design and then implement the install module.
- ? Programmer Mehul Shah

- Work with the architectural designer to design and then implement the modules used by the front-end.

2.2 Resource Management

2.2.1 Work Breakdown Structure

		Task Name	Duration	Start	Finish	Predecessors	Resource Names
1		<input type="checkbox"/> Project Initiation	16 days	Wed 1/22/03	Wed 2/12/03		William
2		Reserve a project	0 days	Wed 1/29/03	Wed 1/29/03		William
3		Form Team	8 days	Thu 1/30/03	Mon 2/10/03	2	William
4		Project Abstract	1 day	Thu 1/30/03	Thu 1/30/03	2	William
5		Introduction Presentation	2 days	Tue 2/11/03	Wed 2/12/03	3,4	William
6		1st Deliverable	0 days	Wed 2/12/03	Wed 2/12/03	5	William
7		1st Presentation	0 days	Wed 2/12/03	Wed 2/12/03	5	William,Ben,Ram,Rya
8		Source Code Control Serv	7 days	Wed 1/22/03	Thu 1/30/03		Mehul
9		System Architecture	5 days	Wed 1/22/03	Tue 1/28/03		Ben
10		<input type="checkbox"/> Project Plan / Feasibility	28.33 days	Wed 1/22/03	Mon 3/3/03		William
11		Develop Project Plan	5 days	Wed 1/22/03	Tue 1/28/03		William
12		<input type="checkbox"/> Review Introduction Se	4 days	Wed 1/22/03	Mon 1/27/03		William
13		Scope of project	1 day	Wed 1/22/03	Wed 1/22/03		Ram
14		Problem Statement	2 days	Thu 1/23/03	Fri 1/24/03		Ram
15		Methodology	2 days	Fri 1/24/03	Mon 1/27/03		Ram
16		Background	2 days	Wed 1/22/03	Thu 1/23/03		Bayo
17		Literature Review	2 days	Thu 1/23/03	Fri 1/24/03		Bayo
18		Glossary	1 day	Mon 1/27/03	Mon 1/27/03		Bayo
19		<input type="checkbox"/> Review Cost Benefit	6 days	Wed 2/19/03	Wed 2/26/03		William
20		Review Function Point	3 days	Wed 2/19/03	Fri 2/21/03		Ben
21		Review COCOMO	3 days	Mon 2/24/03	Wed 2/26/03	20	William
22		Install Module	28 days	Wed 1/22/03	Fri 2/28/03	8FS-7 days	Ryan
23		Prepare document	0.17 days	Thu 2/27/03	Thu 2/27/03	12,11,19	William,Ram,Ben,Rya
24		Prepare Presentation	0.33 days	Mon 3/3/03	Mon 3/3/03	22,23	William,Ryan,Ram
25		2nd Deliverable	0 days	Thu 2/27/03	Thu 2/27/03	23	William,Ram,Bayo
26		2nd Presentation	0 days	Mon 3/3/03	Mon 3/3/03		
27		<input type="checkbox"/> Requirements Analysis	33 days	Fri 1/31/03	Tue 3/18/03		
28		<input type="checkbox"/> Review Requirements :	15 days	Mon 2/24/03	Fri 3/14/03		
31		Back-end Module	28 days	Fri 1/31/03	Tue 3/11/03	8,9	

Figure 2: Work breakdown structure

2.2.2 Project Milestones













		Task Name	Duration	Start	Finish	Predecessors
1		Project Initiation	16 days	Wed 1/22/03	Wed 2/12/03	
2		Reserve a project	0 days	Wed 1/29/03	Wed 1/29/03	
6		1st Deliverable	0 days	Wed 2/12/03	Wed 2/12/03	5
7		1st Presentation	0 days	Wed 2/12/03	Wed 2/12/03	5
10		Project Plan / Feasibility	28.33 days	Wed 1/22/03	Mon 3/3/03	
25		2nd Deliverable	0 days	Thu 2/27/03	Thu 2/27/03	23
26		2nd Presentation	0 days	Mon 3/3/03	Mon 3/3/03	24
27		Requirements Analysis	33 days	Fri 1/31/03	Tue 3/18/03	
35		3rd Deliverable	0 days	Mon 3/17/03	Mon 3/17/03	33
36		3rd Presentation	0 days	Tue 3/18/03	Tue 3/18/03	34
37		Requirements Spec	0 days	Wed 1/22/03	Wed 1/22/03	
38		4th Deliverable	0 days	Wed 1/22/03	Wed 1/22/03	
39		4th Presentation	0 days	Wed 1/22/03	Wed 1/22/03	
40		Final Package	0 days	Mon 3/17/03	Mon 3/17/03	
41		Final Deliverable	0 days	Mon 3/17/03	Mon 3/17/03	6,25,35,38
42		Final Presentation	0 days	Mon 3/17/03	Mon 3/17/03	41



Figure 3: Project milestones

2.3 Gantt Chart

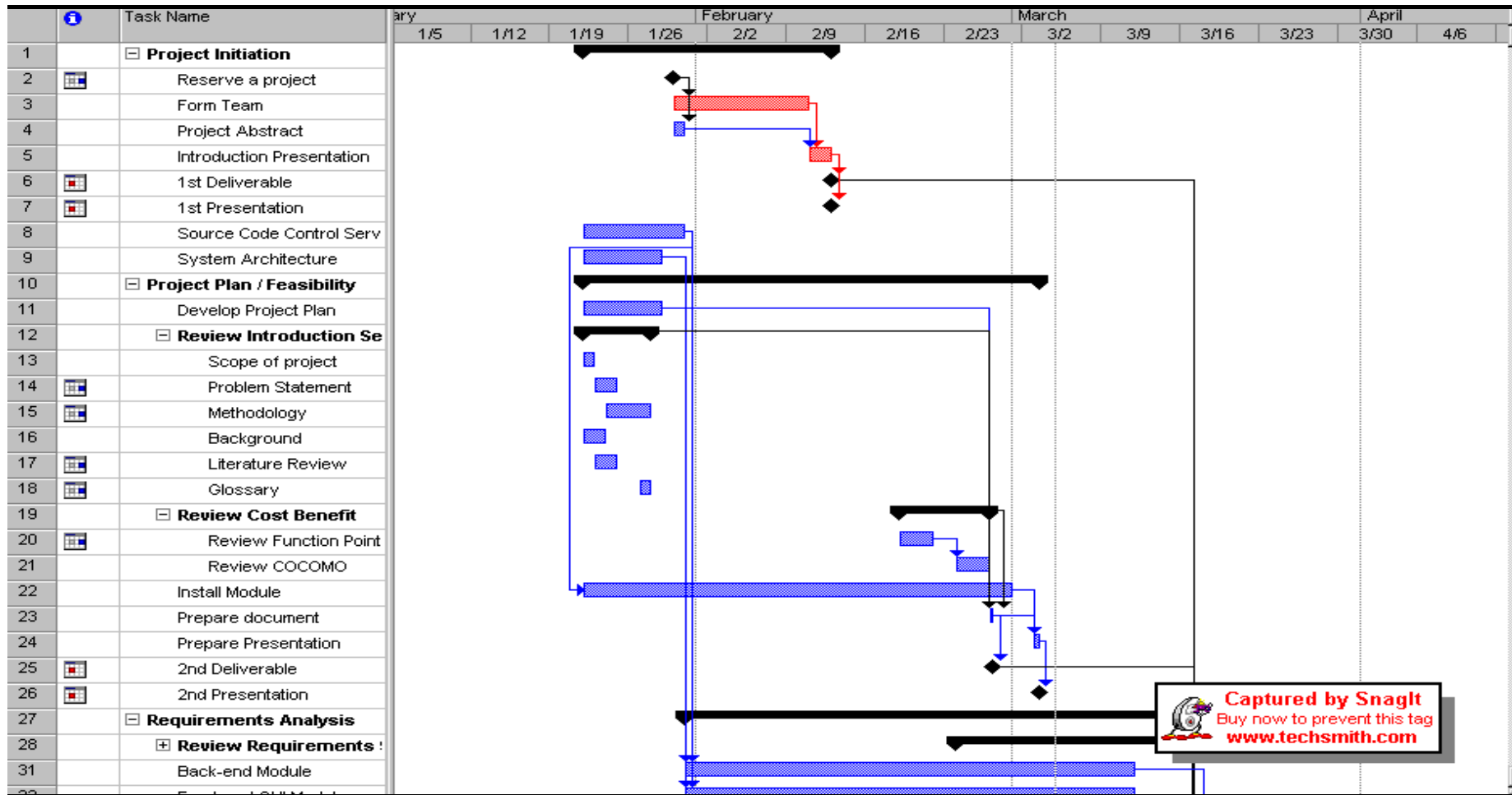


Figure 4: Gantt chart

2.2.4 *Pert Chart*

2.2.4.1 *All Tasks*

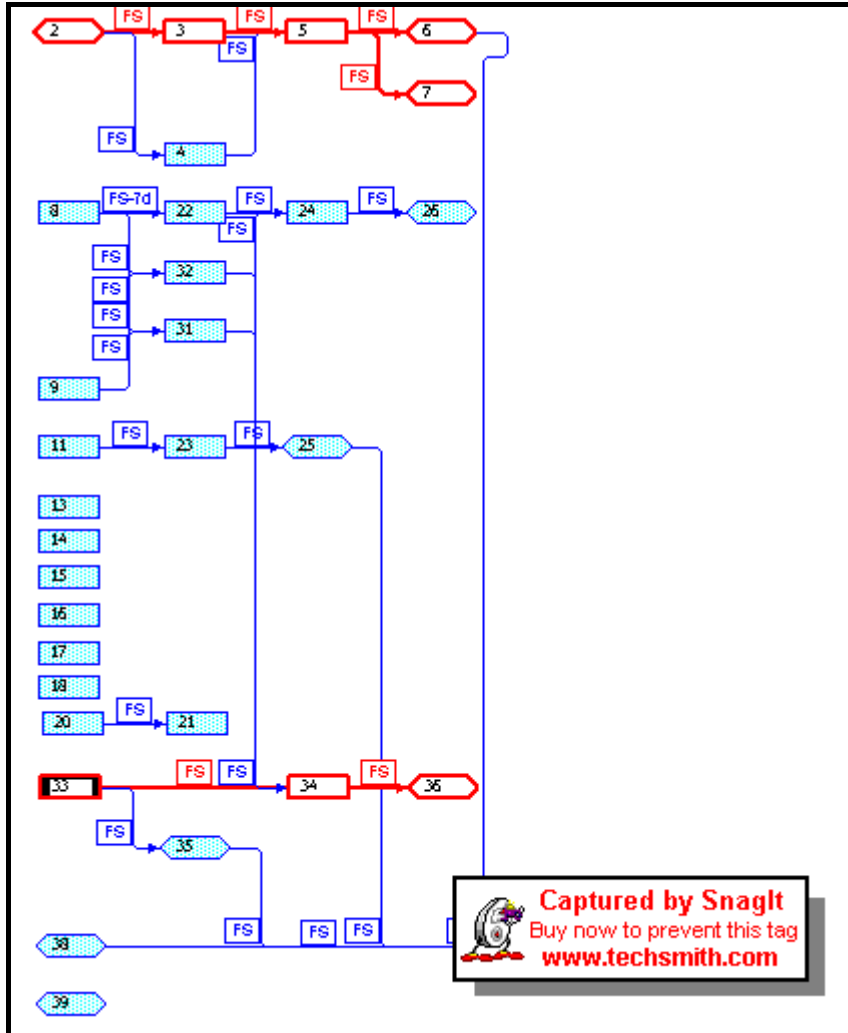


Figure 5: *Pert chart*

2.3 Feasibility Study

2.3.1 *Function Point Analysis*

2.3.1.1 *Inputs*

The inputs for our project come from the graphical user interface and the communicator daemon. In the GUI, we have five tabs and a services tree that we use. The services tree is of high complexity because it dynamically changes to tell the state of your current services. The "Application" tab is of low

complexity as it merely serves as an application manager for the service modules. Internal to that are the application modules, which we will state have a medium level of complexity, as they must take into consideration much more configurations that are individually tailored to specific modules. The "IP" tab is of low complexity because most all you should have to do is input the desired IP address. The "User" tab is of medium complexity as it is used to configure the passwords and options for all user accounts. The "Status" tab gives you log output from various applications and statistics from the operating system, hence it is of medium complexity. As a whole, the Java GUI has two TCP interfaces, one with another computer's GUI interface, and one with the UNIX communicator daemon. These are of low complexity, as it is very easy to communicate through sockets in Java. Finally, the inputs into the UNIX communicator are of medium complexity due to the nature of the development platform. In total, that tallies as nine separate inputs, one of high complexity, four of medium complexity, and four of low complexity.

2.3.1.2 *Outputs*

We have six main outputs that our project uses. We output to our own log, our "Status" tab, our services tree, and our three TCP interfaces. The log should be of low complexity as it would just be a file that we append simple messages to. The "Status" tab will be a medium complexity factor because there will be a little work formatting the messages and displaying them on the screen. The services tree will be of high complexity because it must know the state of all running services and change dynamically as their status changes. Finally, the TCP interfaces that interface in Java will be of low complexity and the

one in the UNIX layer will be of medium complexity, for the same reasons as stated above. In summation, there are six outputs, one of high complexity, two of medium complexity, and three of low complexity.

2.3.1.3 *Queries*

Our project has seven queries that it will use. Basically our queries are anything from which we can derive statistics. All of the queries are very easy to use, due to the ease of UNIX signal calls, TCP communication in Java and UNIX, and the ease of reading files. Therefore all of our queries are of low complexity. We can get the storage space available, CPU utilization, memory utilization, and network usage statistics from the OS layer via the proc file system. We can use Unix signals to determine information about the state of our running application services. We can also use TCP to communicate with the lower level of our application and with the other unit's GUI. Hence, we have a total of seven queries of low complexity.

2.3.1.4 *Files*

There are approximately twenty-four files that our project is going to use. They vary in complexity from low to high. The high complexity files are SMTP configuration, IMAP configuration, Samba configuration, and FTP configuration. In most cases entire open source projects have been built around configuring these files. We realize just how complex editing these files can be, so we intend to only change a few key attributes of each. Even with our simplified modifications we still expect using these files to be of high complexity. Our own configuration file for the cluster will probably be of medium complexity. Considering we will be designing this file ourselves we will be aware of each option and what it does. There will also be a security file of our own design. It

will be of medium complexity because it we will have to use encryption to create the password hashes we store in the file. Finally the low complexity files are the files we read OS information such as CPU, memory, and LAN interface usage from which reside in the proc file system, and our own log file which we append messages to. The remainder of the files are related to the installation package, and these range from pictures to applications. We shall currently assume that there will be eight files of high complexity, eight of medium complexity, and eight of low complexity.

2.3.1.5 *Interfaces*

Our project has five different interfaces. Our project will have to interface with the OS, the communicator daemon, the GUI, the application module framework, and our configuration files. The OS interface and the application interfaces will be handled by UNIX signals and the proc file system, hence they are of low complexity. The LAN interfaces will also be handled through the OS but these interfaces will be used to transfer our heartbeat and network file mirroring information. For these reasons we have determined the LAN interfaces to be of medium complexity. The GUI will be designed by us and should be of medium complexity. Finally the application module framework and configuration files will also be of low complexity, due to the strict object-oriented approach we have taken towards our design architecture. Therefore there are two interfaces of medium complexity and three that are of low complexity.

2.3.1.6 *Function Point Diagram*

	Low	Medium	High	Total
Inputs	1 ↻ 3	4 ↻ 4	4 ↻ 6	43
Outputs	1 ↻ 4	2 ↻ 5	3 ↻ 7	35
Queries	7 ↻ 3	0 ↻ 4	0 ↻ 6	21
Files	8 ↻ 7	8 ↻ 10	8 ↻ 15	256
Interfaces	3 ↻ 5	2 ↻ 7	0 ↻ 10	29
TOTAL	99	120	165	384

Table 3: Function point matrix

2.3.2 AFP and COCOMO Analysis

2.3.2.1 Adjustment Factors

Factor	Description	Influence
Backup and Recovery	The system is a backup and recovery system, this element is critical.	5
Data Communications	The system depends on communications.	5
Distributed Functions	The system is a fail-over cluster. Hence, distributed functions are not critical.	1
Performance	Performance is more of an issue for the services (applications) that are run, but the fail-over system cannot be a performance problem.	2
Operational Environment	The environment should be fairly generic.	1
On-line data entry	There is little data to be entered – and Java is the planned language for the screens – so this is of medium importance.	3
Multiple Screens for Input	One screen with multiple tabs to simplify the user interface.	1
On-line Update	There is no data to be updated – so this is not important.	1
Interface Complexity	On average, this project has interfaces that are of average complexity.	3
Re-usability	The API will be designed for maximum re-usability.	1
Process Complexity	The process is already well defined.	1
Installation Ease	This is one of the key goals for the project.	5

Table 4: Function point adjustment factors

Multiple Sites	The system will be designed so it can be installed at multiple sites for one customer (and at multiple customers' sites).	4
Ease of Use	This is another key goal for the project.	5
TOTAL		38

Adjusted Function Points (AFP) = $384 \times (0.65 + (0.01 \times 38))$

AFP = 395.52

Here we use 34 as the language factor because it is the weighted average of the languages we will be using (25% C, 5% UNIX scripts, 70% Java).

$$\text{Lines of Code (LOC)} = 34 \times 395.52$$

$$\underline{\text{LOC}} = 13,448$$

Finally we use the organic model with basic COCOMO analysis as that model best describes our project.

$$\text{Labor Month (effort)} = 2.4 \times ((\text{LOC} / 1000)^{1.05})$$

$$\text{Labor Month} = 2.4 \times (13.448^{1.05})$$

$$\underline{\text{Labor Month}} = 36.7$$

$$\text{Develop Time (schedule time)} = 2.5 \times ((\text{Labor Month})^{0.38})$$

$$\text{Develop Time} = 2.5 \times (36.7)^{0.38}$$

$$\text{Develop Time} = 2.5 \times (3.9338)$$

$$\underline{\text{Develop Time}} = 9.8$$

$$\text{People Needed} = \text{Labor Month} / \text{Develop Time}$$

$$\text{People Needed} = 36.7 / 9.8$$

$$\underline{\text{People Needed}} = 4$$

2.3.3 Net Present Value, Return On Investment, and Break-even Point Analysis

TANGIBLE BENEFITS WORKSHEET	
Year 1 through 6	
<i>A. Cost reduction or avoidance</i>	\$1,100
<i>B. Increased flexibility</i>	\$500
<i>C. Increased speed of activity</i>	\$200
<i>D. Improvement in management planning</i>	\$200
TOTAL tangible benefits	\$2,000

Table 5: Tangible benefits worksheet

ONE-TIME COSTS WORKSHEET	
Year 0	
<i>A. Development costs</i>	\$0
<i>B. New hardware</i>	\$500
<i>C. New software</i>	\$100
<i>D. User training</i>	\$200
TOTAL one-time cost	\$800

Table 6: One-time costs worksheet

RECURRING COSTS WORKSHEET	
Year 1 through 6	
<i>A. System maintenance costs</i>	\$500
<i>B. Incremental storage</i>	\$200
<i>C. Incremental communications</i>	\$100
<i>D. Software or Hardware leases</i>	\$300
<i>E. Supplies</i>	\$100
TOTAL recurring costs	\$1,200

Table 7: Recurring costs worksheet

Group 2: Snapp 2.3.2 Cost-Benefit Analysis

	Discount Rate							11.00%
	Year Of Project							
	Year 0	year 1	Year 2	Year 3	Year 4	Year 5	Year 6	TOTALS
Net Economic Bene	\$0	\$2,000	\$2,000	\$2,000	\$2,000	\$2,000	\$2,000	
Discount Rate	1.00000	0.90090	0.81162	0.73119	0.65873	0.59345	0.53464	
PV of Benefits	\$0	\$1,802	\$1,623	\$1,462	\$1,317	\$1,187	\$1,069	
NPV of all Benefits	\$0	\$1,802	\$3,425	\$4,887	\$6,205	\$7,392	\$8,461	\$8,461
One-Time COSTS	(800)							
Recurring Costs	\$0	(1,200.00)	(1,200.00)	(1,200.00)	(1,200.00)	(1,200.00)	(1,200.00)	
Discount Rate	1.00000	0.90090	0.81162	0.73119	0.65873	0.59345	0.53464	
PV of Recurring Co	\$0	(1,081.08)	(973.95)	(877.43)	(790.48)	(712.14)	(641.57)	
NPV of all COSTS	(800.00)	(1,881.08)	(2,855.03)	(3,732.46)	(4,522.93)	(5,235.08)	(5,876.65)	(5,876.65)
Overall NPV								\$2,584
Overall ROI - (Overall NPV/NPV of all costs)								0.440
Break-Even Analysis								
Yearly NPV Cash Fl	(800.00)	\$721	\$649	\$585	\$527	\$475	\$428	
Overall Cash Flow	(800.00)	(\$79)	\$570	\$1,155	\$1,682	\$2,157	\$2,584	
Project Break Even Occurs at exactly:			1.1221	years				
BreakEven Fraction = (X-Y)/X=			0.1221					



Figure 6: Cost benefit analysis

SnappCluster Break-even Chart

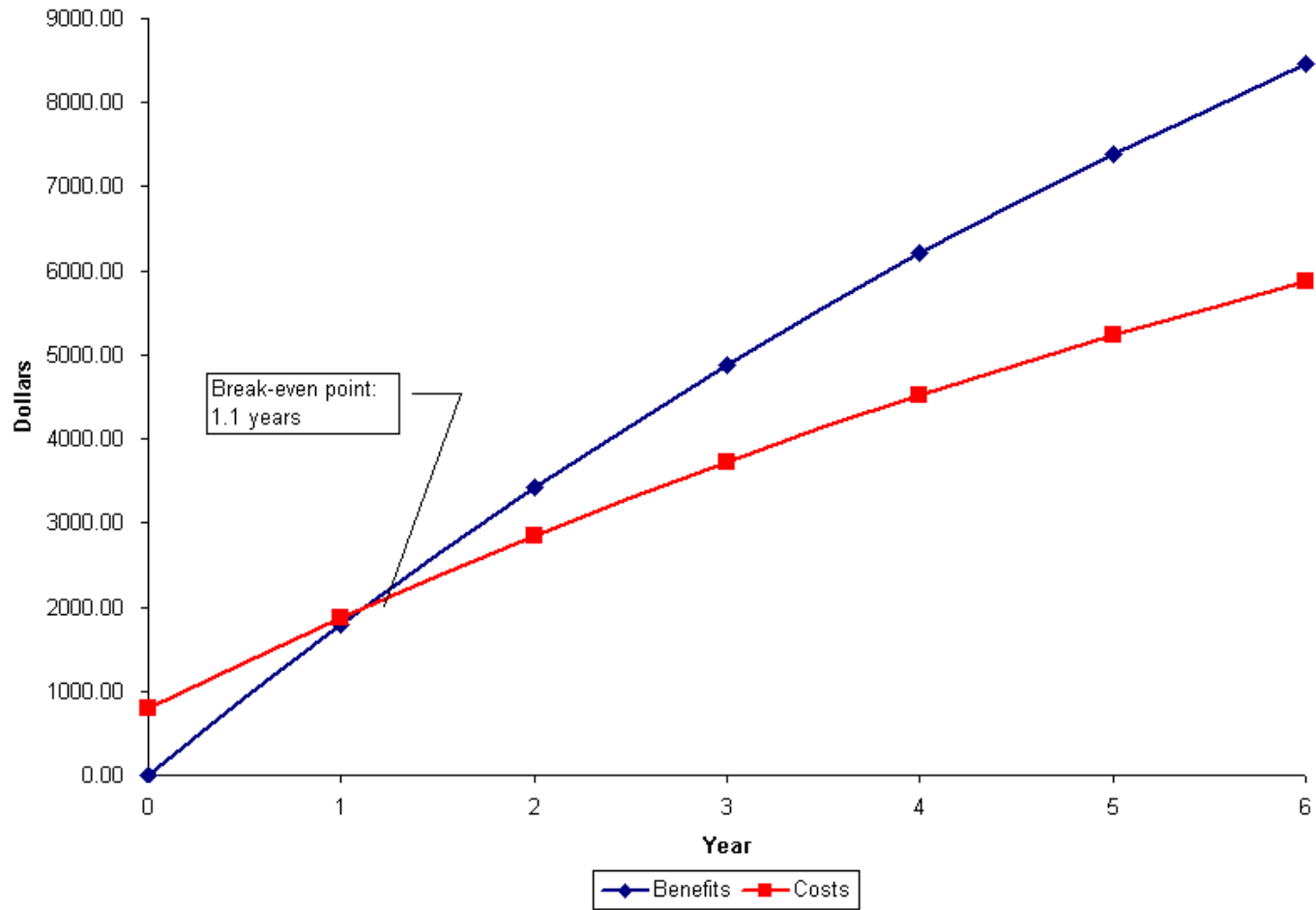


Figure 7: Break-even graph

2.4 Risk Management

Project Size	<p>Due to the significant usage of COTS components and the ability to successfully use CASE tools for most development processes, our project does not entail a significant undertaking.</p> <p>Risk: Low/Medium</p>
Project Structure	<p>The requirements for this project as well as its design for implementation are very structured and provide clear boundaries of what we need to construct and how to do so. The heavy use of COTS also reinforces this structure to insure that separate components work as desired.</p> <p>Risk: Low</p>
Familiarity with Technology/ Application Area	<p>Most of our development team has had experience with Linux and Unix development; some are even extremely knowledgeable of the development platform and how to interface existing systems through the operating system itself. Others in the project, however, have a less involved background in the OS and are not as familiar with Linux. In addition, most have no prior experience in designing middleware applications.</p> <p>Risk: Medium/High</p>
Time Constraint	<p>The sponsor has provided a deadline of 18 months to provide a working prototype. This is a particularly long time to build even a final version of our relatively small project.</p> <p>Risk: Very Low</p>
Systems Interdependence	<p>Our software is built on top of an extremely stable platform with very little room for incompatibility. Nonetheless, due to the fact that it must provide services to other operating systems provides some significant element of risk.</p> <p>Risk: Low/Medium</p>

Table 8: Risk management

3. Analysis

3.1 Stakeholders (who are they and why are they stakeholders)

- ? **Small businesses.** For the purpose of this project, small businesses are defined as those with 1-3 IT members or 15-50 employees. Businesses on this scale cannot afford any of the current clustering packages due to cost and manpower. However, these businesses often strongly rely on technology to service their essential business needs, thereby having great need for server reliability and data protection. In addition, these institutions may want their own email server, which would certainly be an exorbitant expense using any other solutions. These businesses see great value in software that insures them against catastrophe.

- ? **Small or European schools.** Schools share many of the same concerns as small businesses, as the cornerstone of any successful technology is a reliable network. Small schools would also benefit greatly from an email server because it would allow them to give email accounts to students and retain total control over how students use their accounts, as well as monitor for abuse. Schools in Europe are particularly important stakeholders as many of them have already implemented Linux as their primary operating system, allowing easier integration of our clustering software into their existing infrastructure.

- ? **Workgroups in large businesses.** Small workgroups in large businesses have significant reasons to consider investing in our product. Even though our product is not targeted at enterprise data centers, it is an extremely cost-effective way to maintain localized data security for a workgroup server. A more reliable workgroup server reduces downtime and help desk calls, thus saving the company significant time and money for a relatively inexpensive cost. The cost-to-benefit ratio is considerably higher for those companies who also have larger back-end systems that perform main file-sharing services because of data security. With our product, these servers can let local workgroup servers perform the necessary constant backups and therefore relieve stress on back-end systems. These main systems can then perform nightly backups for enterprise quality data security while relieving daytime server load.

- ? **Municipal Governments.** Municipal governments also share many of the same concerns as small businesses, particularly because they must archive a significant amount of local records. Our product focuses on this need by providing inexpensive data reliability and therefore underscores the fundamental need of these customers.

- ? **Academia.** Many individuals in academia who wish to study fail-over clusters spend exorbitant sums of money in order to study its theory and implementation. Our product targets this need by presenting a system

that can run on basic PC hardware and is easy to extensively configure by anyone with a minimal knowledge of systems administration.

3.2 Requirements Gathering

3.2.1 Interview

By conducting an interview with the different types of users at TABC, we gathered the requirements for our system directly from the stakeholder(s) of our project. From our research on how to conduct interviews, we found that people rarely give good answers to subjective questions such as, “What do you feel is the best part of the current system?” or “What would you like in a future system?” (Isaacs)

People tend to give good answers to things that are purely factual, or numerical, or a choice between two, or a maximum of three options. Based on this, we concentrated on asking such questions as: “What do you do when a particular situation arises?” This type of question leads to free-form answers which are not conducive to basic questionnaires, though they are much more demanding on the requirements gatherer.

We conducted an interview of our Sponsor, Mrs. Ceil Olivestone, and Mr. Ben Speiser (TABC Technology Coordinator). We asked Mrs. Olivestone mostly factual questions, but did intersperse some subjective questions in order to get a better ‘feel’ of TABC from her point of view. She directed us to Mr. Ben Speiser to answer some of the more technical questions.

Persons in this interview:**BS:** Mr. Ben Speiser**CO:** Mrs. Ceil Olivestone**SP:** Sriram Polepeddi

This section is for getting the background on the current environment and particular needs of TABC, which we will use in developing our solution.

SP: How often do you use the computing infrastructure at TABC, and for what applications?

CO: *Mostly email, web browsing, most Office Files (Word, Accounting, etc) are stored locally.*

SP: Which of these applications would you describe as mission critical, or necessary for the basic functioning of the organization?

CO: *The Accounting package. I can live without email or web-browsing, or anything network related.*

SP: Which applications at TABC does the general population use most often?

CO: *Email, Office Apps, web-browsing, shared network data drives for faculty and students, and the most complex software we have is the Database.*

SP: Which of these applications would you describe as mission critical, or necessary for the basic functioning of the organization?

CO: *Email is critical as we need it to communicate with the parent body. The web is used heavily by the faculty and students. The Database is used by one School Administrator.*

SP: How would you rate the computing knowledge of: the general population at TABC, the Administration, the Faculty, and the person(s) responsible for maintaining the computing infrastructure?

CO: *Among the School Admins: mostly very little, but there are a few good-to-very knowledgeable persons. Among the Faculty, it is mixed, the younger ones are better than the older ones. The children are very advanced, only a few are not. And I would rate our Tech Department as being very very high on the computing knowledge scale.*

SP: How often do you receive complaints with the Network?

CO: *Not so much about the network, but more about the hardware, esp. the older hardware. But Ben Speiser would know more.*

BS: *It ranges. Sometimes none, sometimes as many as eight complaints per week . It depends mostly on when the administration or faculty have particular deadlines approaching, and things just don't seem to be operating under the strain.*

SP: What is the procedure step-by-step that you follow from the point of receiving the complaint to its final resolution?

CO: *All Ben, Ask him.*

BS: *Well, I first try to determine the severity of the problem. If it's no more than a nuisance, I try to explain how it's low on the priorities and I explain a current workaround. If it is important but not immediate, it goes onto my "to-do" list and gets taken care of during the next round of "fix-it" time. If it is of high priority or can be easily resolved on the spot, it is taken care of immediately if time allows. The problems are from too great a range to elaborate anything beyond this damage assessment phase, though.*

SP: If a disaster struck your current setup that prevented it from providing these services, how much would you lose in tangible terms?

CO: *Cannot put dollar terms on it, but maybe in terms of repair costs.*

SP: And in what other intangible ways would TABC be affected by downtime?

CO: *We can function as a school, as the education, other than the computing department, is not that computing based. But administratively, it would affect us. We would not be able to access the kids' schedules, teachers' contracts, billing info, etc. Ben would be able to provide even more info.*

BS: *Every single thing I should be doing would get put on hold. I would not be able to teach classes, make phone calls, grade tests, call parents, monitor students in the computer lab; deal with students who break school computing policy... the list goes on and on. I remember when our old ISDN router failed because our ISDN line was hacked - our entire DHCP system shut down - I did nothing for two days other than give PCs static IP addresses. Afterwards, I had to switch them back after our DHCP came back online.*

SP: Have you ever had a major issue at TABC, which necessitated an expensive repair solution?

CO: *At least not in the three years that I've been here.*

SP: What issues do you presently have with the current infrastructure even when it is operational, if any?

CO: *Well, no specific issues, other than our low tech knowledge which does cause some issues.*

This section is related to the possible solution we will develop.

SP: Is any solution intended to add to TABC's current infrastructure or to replace existing hardware?

CO: *Ben Speiser would know that*

BS: *Currently, we want to use it to augment our infrastructure. Our current authentication servers do double duty as file servers. In some cases, this makes sense, but it also means that they can be overtaxed. We don't intend to replace any hardware or infrastructure, but we do intend to augment it.*

SP: How important, if at all, is it for our solution to NOT require additional purchases on TABCs part?

CO: *Very important, not much in the budget, but I can perhaps spare about \$100, in addition to all of the parts that are already available in our storage.*

SP: Who do you see as the people who will need to interact with the system on a daily or occasional basis?

CO: *That would be Ben Speiser.*

SP: Where in TABC are you planning on implementing this product?

CO: *Ben Speiser would know that.*

BS: *First off, in the publications room, as a standalone server. Ultimately, I would like to be able to deploy it as a cluster in both the main office and the computer lab.*

SP: Would you prefer a system that gives decent performance on all hardware, such as the old Pentiums in your Hardware Inventory, or a system that can give excellent performance on a smaller subset of your Hardware, such as the Pentium IIIs? We would likely still be able to use whatever hardware you already have, without requiring the purchase of additional equipment.

CO: *Why would I want a slow application if I can have an application that would much faster on better hardware.*

SP: So would you say that you are more concerned about maximizing the productivity of the system than simply maximizing the number of systems that this application could be run on?

CO: Yes

SP: Who at TABC would have the final responsibility for the management of the system? i.e. Is it important for the Administration at TABC to be able to control this application if need be, with their own special privileges, or would the entire administration of this system be left to the person(s) currently maintaining it?

CO: *That would be Mr. Speiser*

SP: Would you or the administration need to be appraised of any and all actions that the system performs or would that again be something left to the administrator of the system?

CO: *Yes, though the administration would not be doing any of the technical activities related with it, we would still need to know what is going on with it.*

SP: How soon would you need this project to be implemented?

CO: *Though, it would be good to implement it this semester, this is not a priority*

SP: What are your expectations of the project once it is completed?

CO: *Well, I do require proper documentation so that anyone who looks at it can administer the system.*

CO: *Other than that, what would I do in case something happens to it 3 years from now?*

SP: *We are building upon open source components that have been widely tested by an entire legion of Linux programmers, so they are very stable. The documentation we give you will allow anyone with even a slight knowledge of Linux Administration to rectify most problems. If this doesn't work, there are plenty of Linux Tech Support Professionals who could take care of any problem.*

CO: *Ok good.*

SP: *Would you have any need for the source code of this project once it is implemented?*

CO: *What can I do with that?*

SP: *Well, it would allow you to make low-level changes in case you wanted to tweak the system even further. But this would assume that you had a Unix programming expert at your school ready to do this.*

CO: *Well in that case I don't see the need for it.*

SP: *Would you want the source code for your computer science classes for your students to work on?*

CO: *I don't know I will ask Mr. Speiser about that and let you know.*

(I later asked Ben, and here is his response.)

BS: *It might be nice to use some of the code in my class, but this is something I don't think I would want students tampering with. I am very security-conscious, and knowing that a student would need to be trusted with the underlying code that makes the server accessible is concerning. In the long run, I would probably prefer (from a security standpoint) that the software source code be unavailable if possible.*

SP: Thank you Ceil. I appreciate the time you have given us.

3.2.2 Observation

Torah Academy of Bergen County (TABC) is a private school based in Teaneck, NJ. The school teaches classes from grade 9 to 12. Students here undertake very challenging projects under guidance from top-notch faculty. There is a science lab at TABC for working on projects involving lasers, jet spectroscopy, monochrometers, telescope positioning systems, etc. Everyone at TABC uses the computing infrastructure in some form or the other. The main users we noticed were the school administration, faculty, and students. Though they all utilized different applications, they all had the same view of the network; i.e. they all saw their applications as files kept somewhere on the network. The only difference between them was the permissions they had to access the network. These permissions were granted to the various users through authentication servers kept in the publications room, the computer lab, and main office.

While students use the latest available PCs for their work, many important administrative responsibilities are handled by older PC architectures. This poses a threat to the TABC's data possibly hurting their operations in case any problem arises with these machines. They are all aging, some are cobbled together using parts harvested from other computers, and none use backup hardware such as tape or CD-ROM. There are at least three servers that are in dire need of a system reliability solution. The first is the office server used by the entire administrative staff to store daily data files, Peachtree Accounting Spreadsheets, etc. The second is a server used to house TABC's newsletter and yearbook work. The third is a server that houses the shared network drives of each faculty,

administrator, and student at TABC. Also, there are servers that perform vital functions for the school, such as the firewall and print servers. These will also use a system reliability solution if it is cost-effective and fully tested.

3.2.4.1 Brainstorming Prioritization

Each user was given 100 points to distribute between these 4 major categories within the project. As the project progresses through the software development life cycle, more efforts will be focused on the categories gathering the highest scores. The reason for this is the user community has voted on what they believe are the more important aspects to a successful project.

User	Security	Reliability	Easy to use	Reporting abilities
Large Corporate User (William)	20	50	20	10
Small School User (Ben)	20	40	25	25
Small Business User (Ram)	20	35	30	15
Academia User (Bayo)	15	30	15	40
Municipal government User (Mehul)	40	35	20	5
System Administrator (Ryan)	10	50	0	40
Totals	125	240	110	135

Table 9: Brainstorming prioritization

3.3 Post Requirements Gathering Problem Statement

The main concept has not changed much from the earlier problem statement as the goal of designing a clustering system that works as a simple and reliable backup solution still exists. All of the features presented here build on the previous problem statement and do not replace its features in any way.

We aim to address the easy installation issue by writing turnkey installation code that automatically installs the required components of our system on each of two machines with little to no user prompting. And are

providing a graphical user interface (GUI) for the person administering the cluster. We look to make the system reliable by taking widely tested open source clustering code and tailoring it for a cluster for a small organization. To maintain the security of the system, we will be designing our own authentication file with a special encryption and hashing algorithm.

We will be building our system on the personal computer (PC) architecture, and after consultation with one of the stakeholders feel that we should optimize the code for a Pentium III Architecture. Also, the administrator at TABC wishes to be alerted by email in case of a problem with the cluster. And in addition to the basic services such ftp, file serving, and email, we have found a need for specially tailoring DNS for the cluster as well.

3.4 Requirements Definition

The following lists summarize the application functional and non-functional requirements.

3.4.1 Functional Requirements

1. There should be a way to maintain the security of the cluster.
2. The system will provide an intuitive GUI to manage the cluster.
3. The system will provide robust fail-over support for our cluster.
4. The system will provide robust network backup services for our cluster data
5. The system should make file sharing and mail services such as FTP, SMB, SMTP, IMAP and DNS available for the users of the cluster.

6. The administrator should have tools to be able to perform administration and configuration of basic system parameters such as user accounts, mail options, and disk usage.
7. The system should have an email alert for the administrator. This is vital for any mission critical errors that may occur while the administrator is away.

3.4.2 *Non-functional Requirements*

1. Our product will be simple to use.
2. The system should be easy to maintain.
3. The system should switch to the BETA server within 30 seconds so that users would not notice a loss of services. (There is no need to mention that users already connected to the Alpha WILL notice as their connection will get re-set, but we should have a time requirement here.)
4. The cluster should be designed to run on PC systems that are at most 3 years old.

3.5 Requirements Specifications

The SnappCluster system is being developed to cater to the needs of organizations needing simple reliable data solutions.

1. Authentication & Security of the System
 - 1.1 Authentication of users permitted to maintain the system will be maintained through a custom-designed password file.
 - 1.2 The various components of the system will communicate securely with each other using the Secure Socket Layer Protocol.
2. Intuitive GUI - Our product will provide an intuitive GUI to manage the cluster. This will be accomplished through the use of a variety of methodologies.
 - 2.1 Framing is the process of focusing the user's attention to groups of components on a GUI through the use of a boundary (i.e. Box that frames common components together).

- 2.2 Color-coded status messages will be used to quickly give the user an idea of what the status of the system is.
 - 2.3 Limiting the total number of components to seven or less will help to keep the user from becoming confused by an overload of information.
 - 2.4 The GUI will have a stoplight-type of alert that lets the administrator immediately see the status of the heartbeat protocol, which server is currently ALPHA, etc.
 - 2.5 Provide a dialog that allows the various log files to be displayed.
 - 2.6 A service 'tree' on the left side of the GUI that will show the status of services running on the cluster,
 - 2.7 These will be done using Java GUI components.
3. Robust fail-over support for our cluster.
 - 3.1 Our system will provide robust fail-over support by switching control over to the BETA computer in the event that ALPHA goes down.
 - 3.2 This will be accomplished through the 'heartbeat' protocol in order to ensure the communication of the proper information between computers in our cluster.
4. Our product will provide robust network backup services for our cluster data.
 - 4.1 Through the use of the DRBD file-system, our cluster will be able to effectively mirror critical data to the beta unit.
 - 4.2 Mirroring will ensure that important data are not lost in the event of a server failure.

5. Our product will make file sharing and mail services such as FTP, SMB, SMTP, IMAP and DNS available for the users of the cluster.
 - 5.1 Our product will build on stable COTS components such as DRBD, Proftpd, SAMBA, qmail, imapd, Apache, and IMP.
 - 5.2 Our product will offer two default services. These services are SMTP and FTP.
 - 5.3 Other services will be offered as add-ons, which will be included within the system if the system administrator desires.
6. The system will allow simple configuration of certain system parameters such as user accounts, mail options, and disk usage.
 - 6.1 The user will be given a GUI based configuration tool, which can be accessed from within our product. This GUI will provide the user with the most used configuration options and leave out configuration options that are either very seldom used, or that are very advanced.
 - 6.2 Our product will limit configurations of the COTS systems we use to certain options. For example we will not allow the cluster administrator to change the position of a user's entry in the *passwd* file, but we will allow them to add/remove users and change their passwords.
7. Ease of Maintenance of the system
 - 7.1 Will be achieved through use of well-known, stable COTS systems such as the Krux distribution of Linux, DRDB, ProFTPD, SAMBA, qmail, IMAPD, Apache, and IMP.

- 7.2 Our product will not use a load-sharing scheme to share processor usage between computers in the cluster. Our product is only meant to be a simple fail-over cluster, not a scientific cluster.
 - 7.3 Our product will not independently handle initial installation of the Linux system. We will use COTS components such as Krux install scripts to handle this.
 - 7.4 Our product will not independently implement any file-sharing or mail protocols. The only protocol we will implement ourselves is the heartbeat protocol.
8. Software Development tools
- 8.1 The front-end interface will be developed using Java.
 - 8.2 The operating system level code will be developed mostly using C and C++.
 - 8.3 These two layers will be able to communicate with each other by Wrapper Classes using Java, C, and C++.
9. Hardware and Software Requirements - The development team has determined that the SnappCluster can run on of the following hardware. It has been split into required and recommended hardware.
- 9.1 Minimum Hardware Specifications (per server):
 - 9.1.1 Video Cards – any PCI or AGP video Card
 - 9.1.2 IDE Hard Drives - any IDE Hard Disk
 - 9.1.3 Monitor – capable of handling 640x480 resolutions.
 - 9.1.4 Network Adapters -- any Ethernet (10base2, 10base5, and 10baseT, 100baseT)

- 9.1.5 CD-Rom common IDE CD-Rom Drives – as far back as 1x
- 9.1.6 CPUs – Pentium 686 or later
- 9.1.7 RAM – minimum: 16 MB
- 9.2 Recommended Hardware Specifications (per server):
 - 9.2.1 For 0-10 users: Will support basic DNS, SAMBA file serving, email, Intranet web server
 - 9.2.1.1 IDE Drives: 1
 - 9.2.1.2 Size: 4 Gig+
 - 9.2.1.3 RAM: 32 Megs
 - 9.2.1.4 CPU: Pentium MMX – 166Mhz or faster
 - 9.2.1.5 Video Card: any supported video card
 - 9.2.2 For 10-40 users: Will support basic DNS, SAMBA file serving, email, Intranet web server
 - 9.2.2.1 IDE Drive: Number of drives: 1 Size: 10-20 Gigs
 - 9.2.2.2 RAM: 128 Megs
 - 9.2.2.3 CPU: Pentium 2 – 266Mhz or faster
 - 9.2.2.4 CD-ROM: any supported IDE CD-ROM
 - 9.2.2.5 Video Card: any supported video card
 - 9.2.3 Optionally Recommended:
 - 9.2.3.1 UPS with at least 10 minutes backup. (We are not making any software to automatically shutdown the system but this is a good thing to have for \$20)
 - 9.2.3.2 For more than 2 systems: a kvm switch would make switching between PCs a breeze.

4. Requirements Modeling

4.1 AS-IS DFD Diagrams

4.1.1 Context Diagram

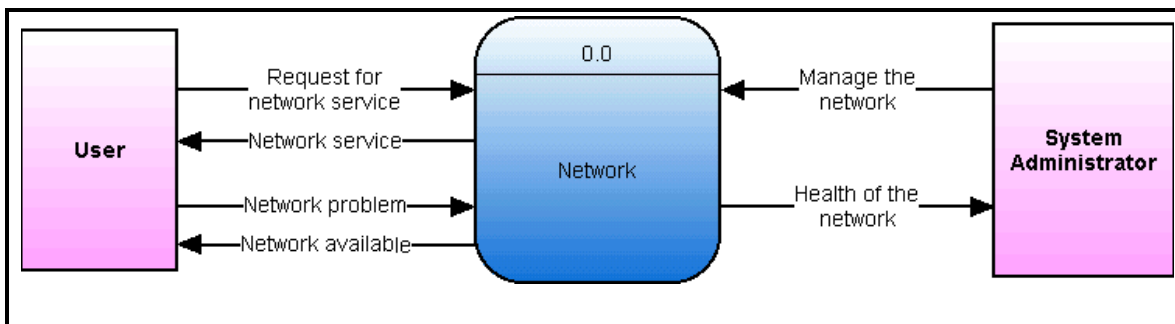


Figure 10: AS-IS DFD - Context diagram

4.1.2 General Diagram

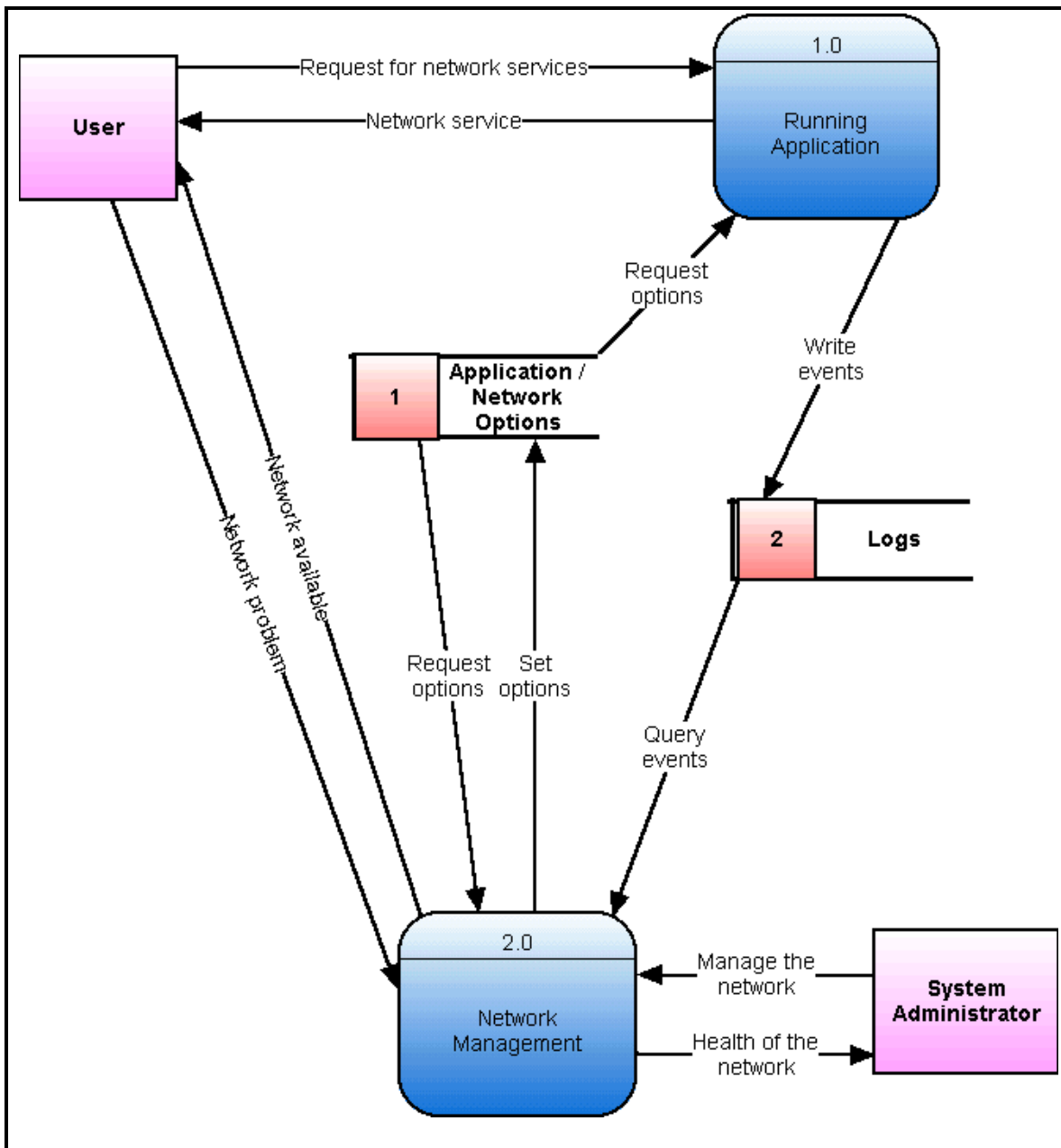


Figure 11: AS-IS DFD - General diagram

4.1.3 Network Management – 2.x

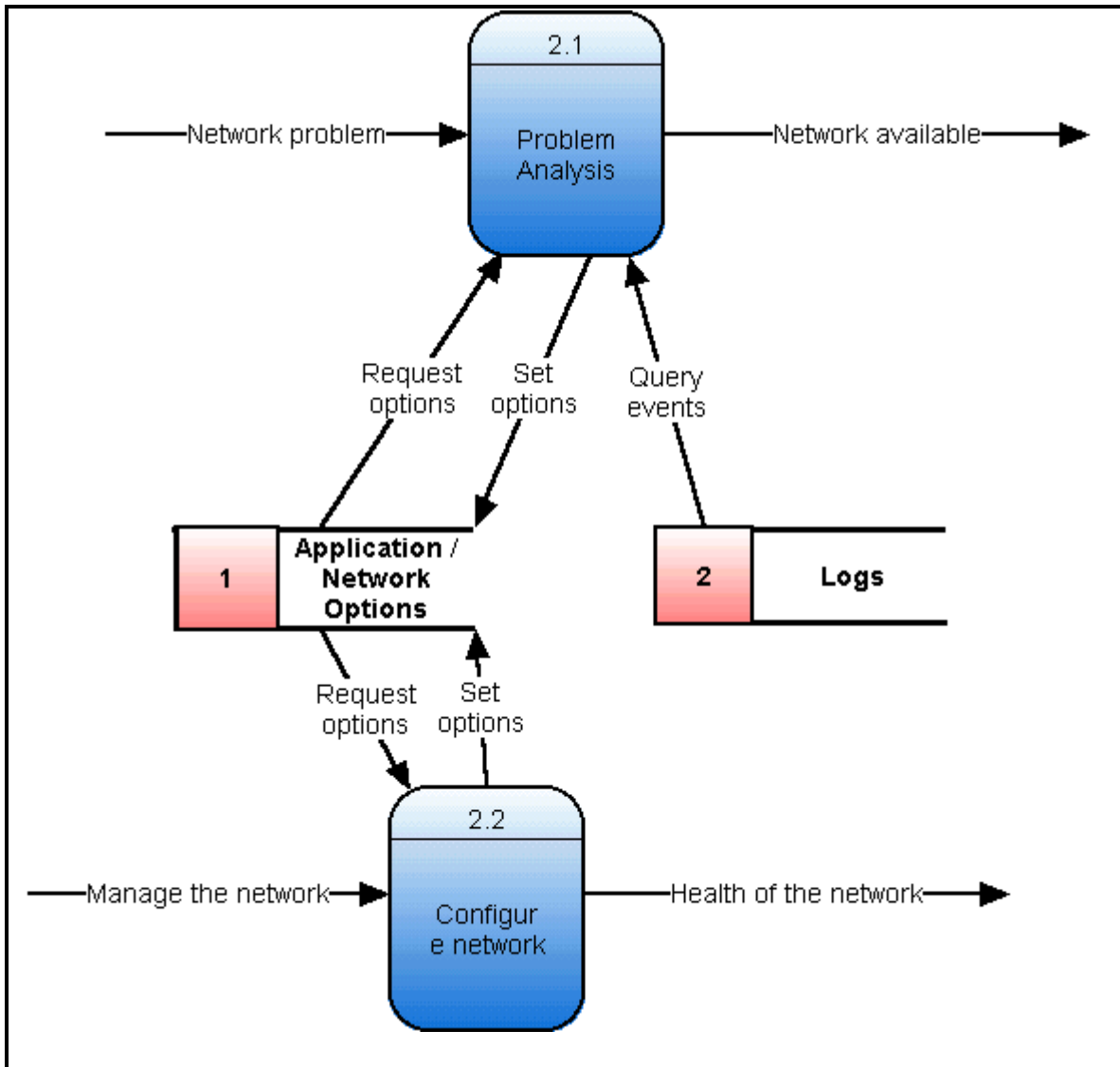


Figure 12: AS-IS DFD - Network management - 2.x

4.1.4 Problem Analysis – 2.1.x

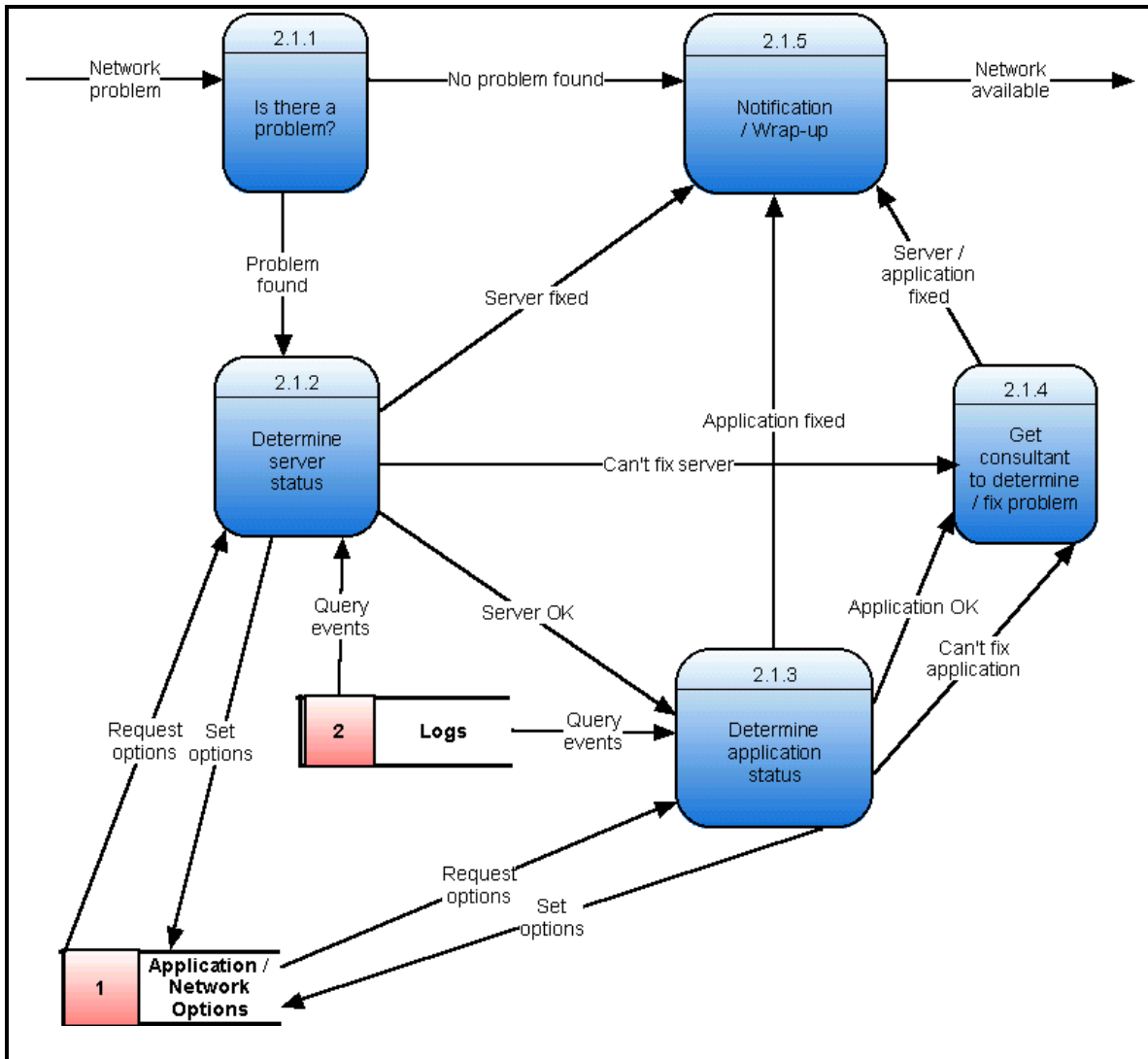


Figure 13: AS-IS DFD - Problem analysis - 2.1.x

4.1.5 Determine Server Status – 2.1.1.x

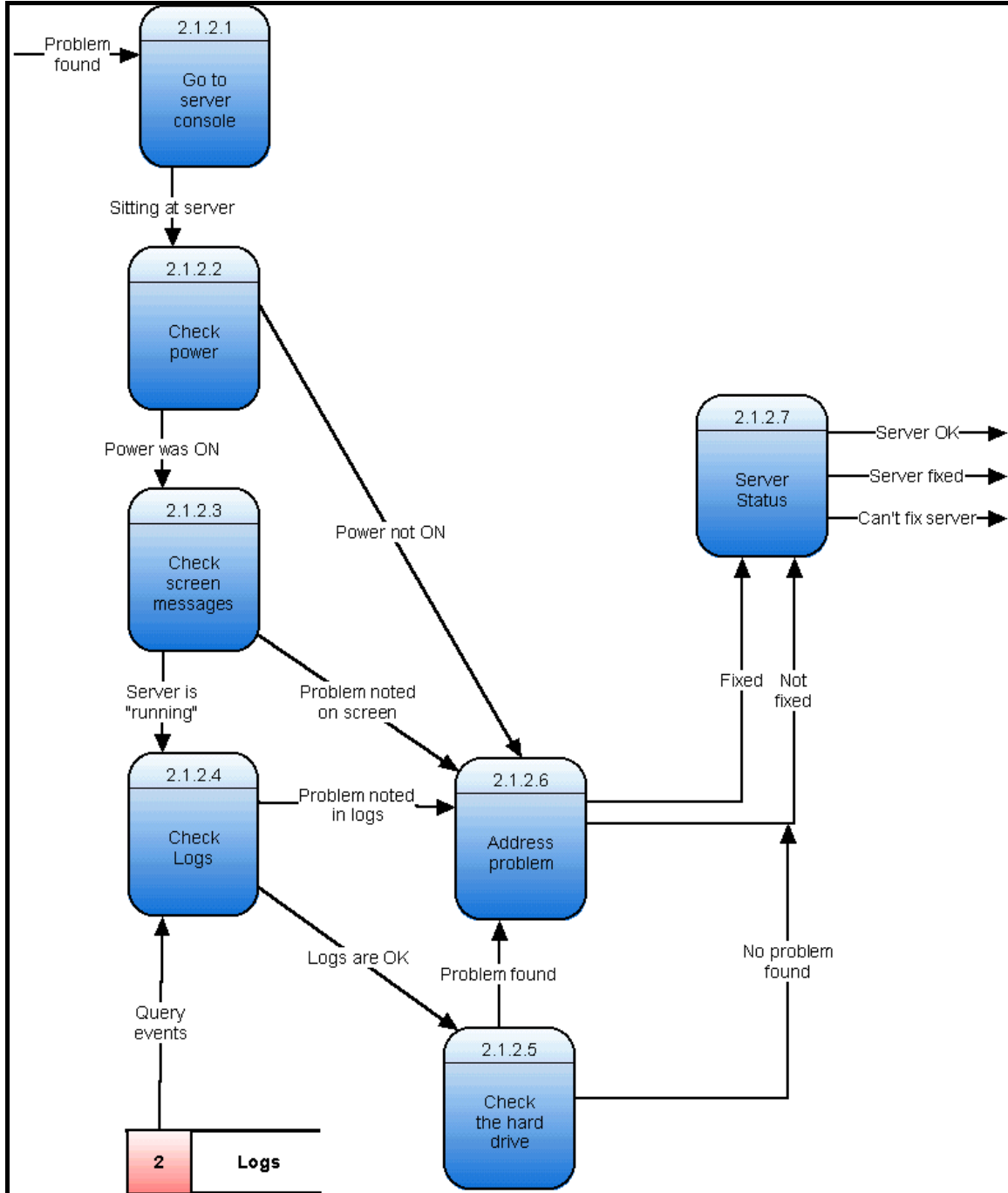


Figure 14: AS-IS DFD – Determine server status – 2.1.1.x

5. Design

5.1 TO-BE DFD Diagrams

5.1.1 Context Diagram

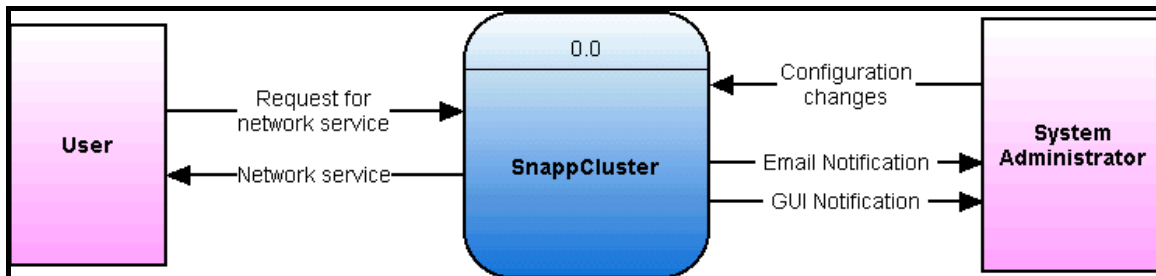


Figure 15: TO-BE DFD - Context diagram

5.1.2 General Diagram

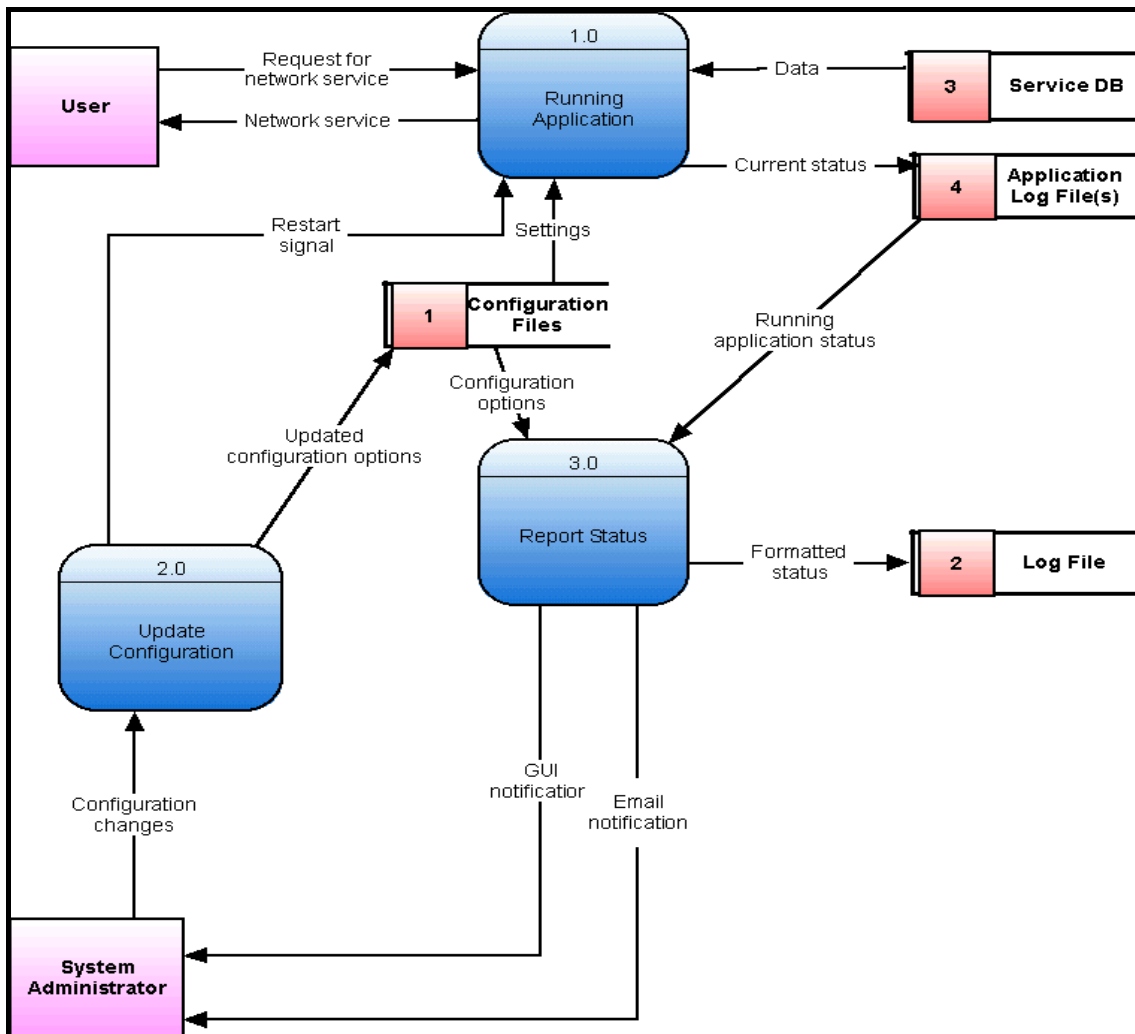


Figure 16: TO-BE DFD – General diagram

5.1.3 Running Application – 1.x

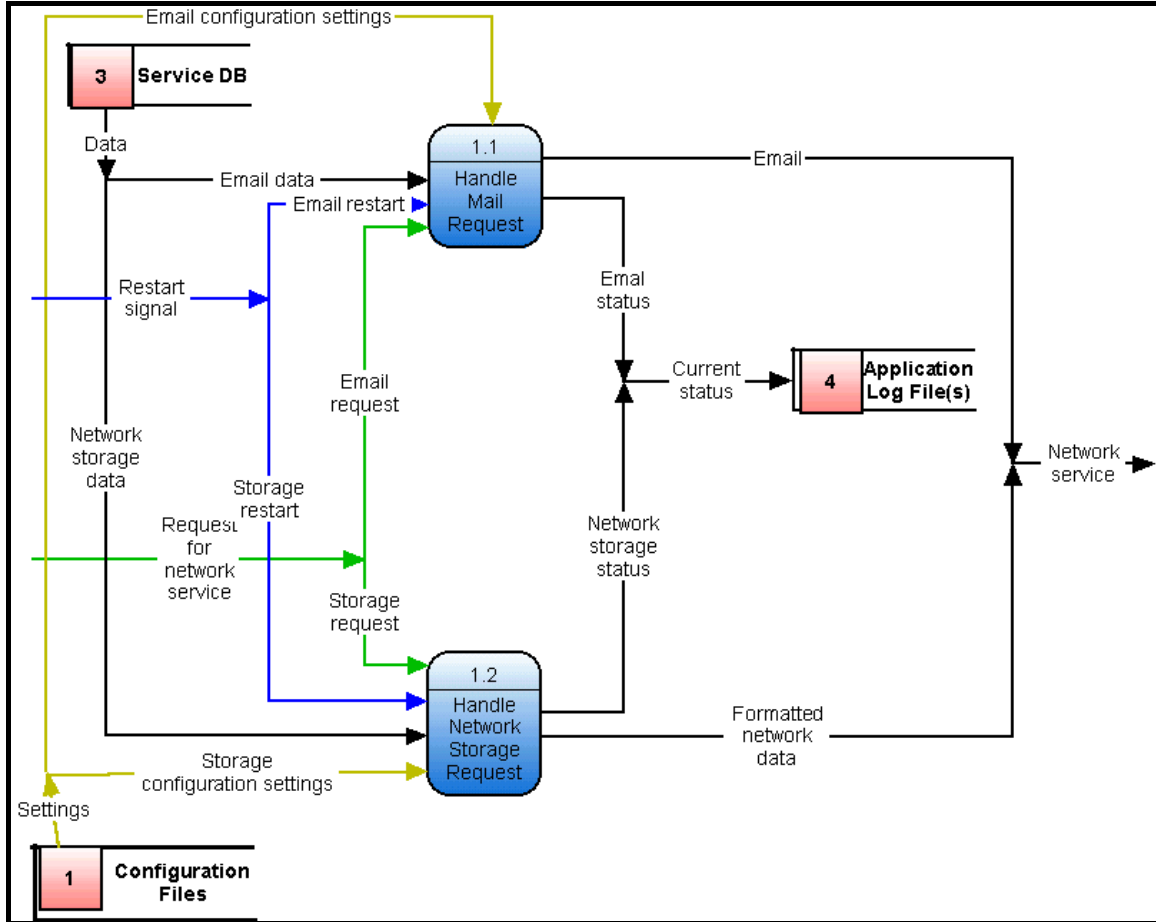


Figure 17: TO-BE DFD - Running application - 1.x

5.1.4 Update Configuration – 2.x

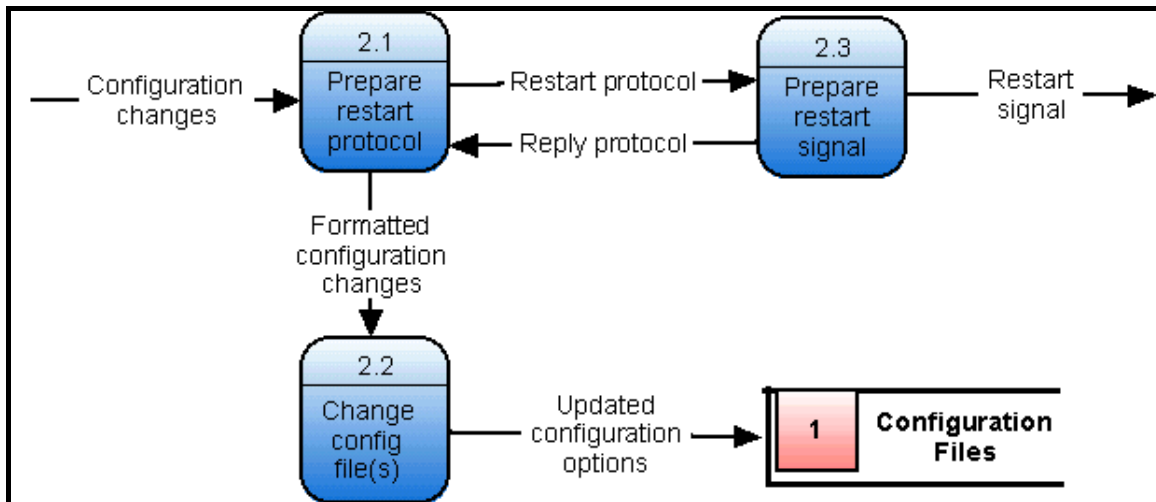


Figure 18: TO-BE DFD - Update configuration - 2.x

5.1.5 Prepare Restart Signal – 2.3.x

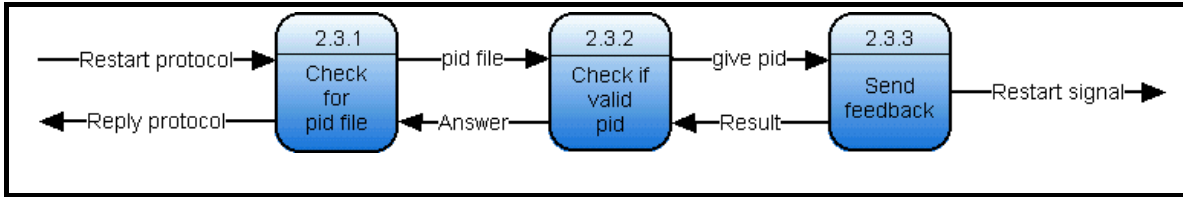


Figure 19: TO-BE DFD - Prepare restart signal - 2.3.x

5.1.6 Report Status – 3.x

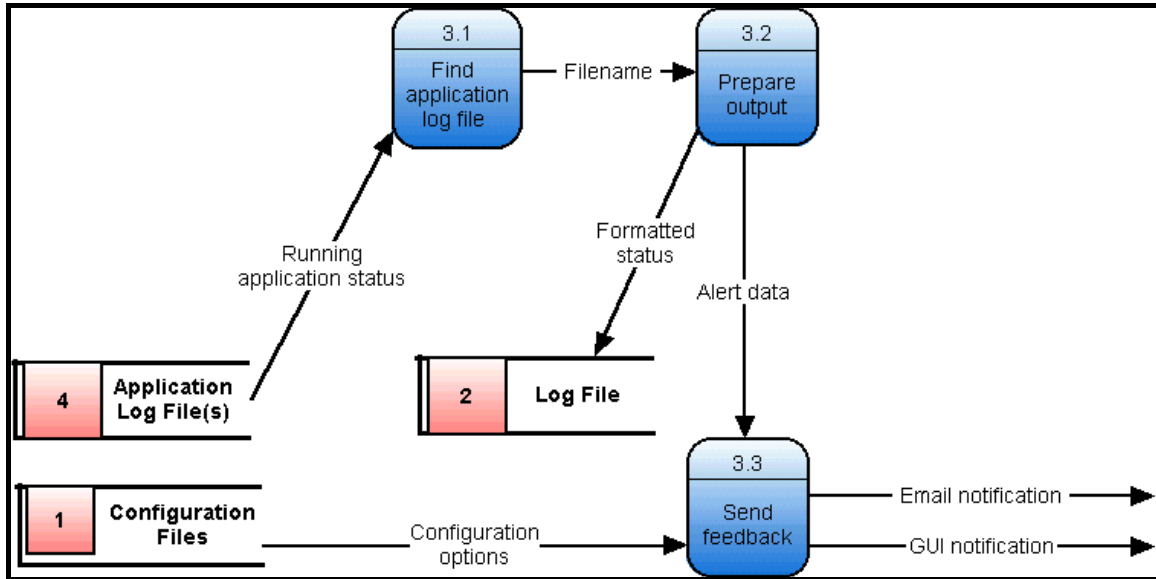


Figure 20: TO-BE DFD - Report status

5.2 Process Specifications

5.2.1 Structured English

5.2.1.1 Process 1: Clustering Software Starting

```
Start Log as Beta unit;
Record start in Log;
Start GUI as Beta unit;
Start Clustering Service on this unit as Beta;

IF (This Unit's Rank > Other Unit's Rank)
  THEN
    Consider Alpha promotion for this unit;

IF ( not alpha unit by now )
  THEN
    This unit should act as Beta unit;
```


5.2.1.2 Process 2: Acting as Beta Unit

```
Record acceptance of Beta status in log;
LOOP Indefinitely:
  Get reports from all clustering applications;
  Update the GUI;
  Ask Heartbeat system if Alpha unit if it is alive;
  Get commands from Heartbeat unit;

  SWITCH (Commands):
    CASE Alpha caused Fail-over:
      Consider Alpha promotion for this unit;
    CASE Request for Info:
      Send appropriate status report to heartbeat;
    CASE Shutdown this machine:
      Reboot the machine;
    CASE Other command:
      Handle the command requested;
    CASE Everything is OK:
  END SWITCH;

  IF ( this Beta unit needs to send a message to the alpha unit )
    THEN
      Tell heartbeat to send a request for desired information
        or commands;

  This Beta unit's Rank is set to 0;
  Mirror Storage as Beta unit;
END LOOP
```

5.2.1.3 Process 3: Consider Alpha Promotion for this Unit

```
Record Alpha contest in Log;
Increase this unit's Rank by 2;

*COMMENT: Check for Alpha Collisions
WHILE ( This unit's Rank == Other unit's Rank ):
  WAIT random number of milliseconds from 0 to 50 to prevent Alpha
  collision;

  IF ( This unit's Rank < Other unit's Rank )
    THEN
      EXIT this while loop;
    ELSE
      Increase this unit's rank by 2;
END WHILE Loop;
IF ( This unit's Rank > Other unit's Rank )
  THEN
    Record Alpha promotion in Log;
    IF ( supposed to send Alert )
      THEN
        Send fail-over alert to Systems Administrator;
        Start Clustering Service on this unit as Alpha;
      ELSE
        Resume acting as Beta;
```

5.2.1.4 Process 4: Act as Alpha Unit

```
LOOP Indefinitely:
  Get reports from all clustering applications;
  Update the GUI;
  Respond to Heartbeat "Are you alive?" request;
  Get commands from GUI;
  Get commands from Heartbeat;

  SWITCH ( Commands ):
    CASE Deliberate Fail-over:
    CASE Shutdown:
      Reboot this machine;
    CASE Request for Info:
      Send appropriate status report to heartbeat;
    CASE Other Command:
      Fork a new process that handles the command;
    CASE Beta unit had error:
      Record Beta unit error in Log;

      IF ( supposed to send Alert )
        THEN
          Send alert to Systems Administrator;
    CASE Everything is OK:
  END SWITCH;
  Mirror storage as Alpha unit;

  If( Other unit's Rank is stable at 0)
    This unit's Rank is set to 1;
  Else
    Clarify ranking through Alpha contest;
  End LOOP;
```

5.2.1.5 Process 5: Start Clustering Service

```
Inputs: what type of unit to start as - Alpha or Beta

Open configuration data store;
Read in Alpha unit's IP settings;
Read in Beta unit's IP settings;
Read in all application settings;
Close configuration data store;

IF ( Starting service as Alpha)
  THEN
    Set unit's IP address to proper static IP address for Alpha
      unit;
  ELSE
    IF ( Beta unit is supposed to receive a dynamic address )
      THEN
        Retrieve a dynamic IP address;
        Set unit's IP address to received dynamic IP address;
      ELSE
        Set unit's IP address to proper static IP address for Beta
          unit;
```

```
Fork a new process:
    Start Heartbeat as appropriate unit type;
Fork a new process:
    Start Storage as appropriate unit type;
Fork a new process:
    Start Log as appropriate unit type;

IF (Starting service as Alpha )
    THEN
        Fork a new process:
            Start desired cluster applications;

Update the GUI;
Record clustering service started as appropriate unit type in
    Log;
```

5.2.2 Decision Tree

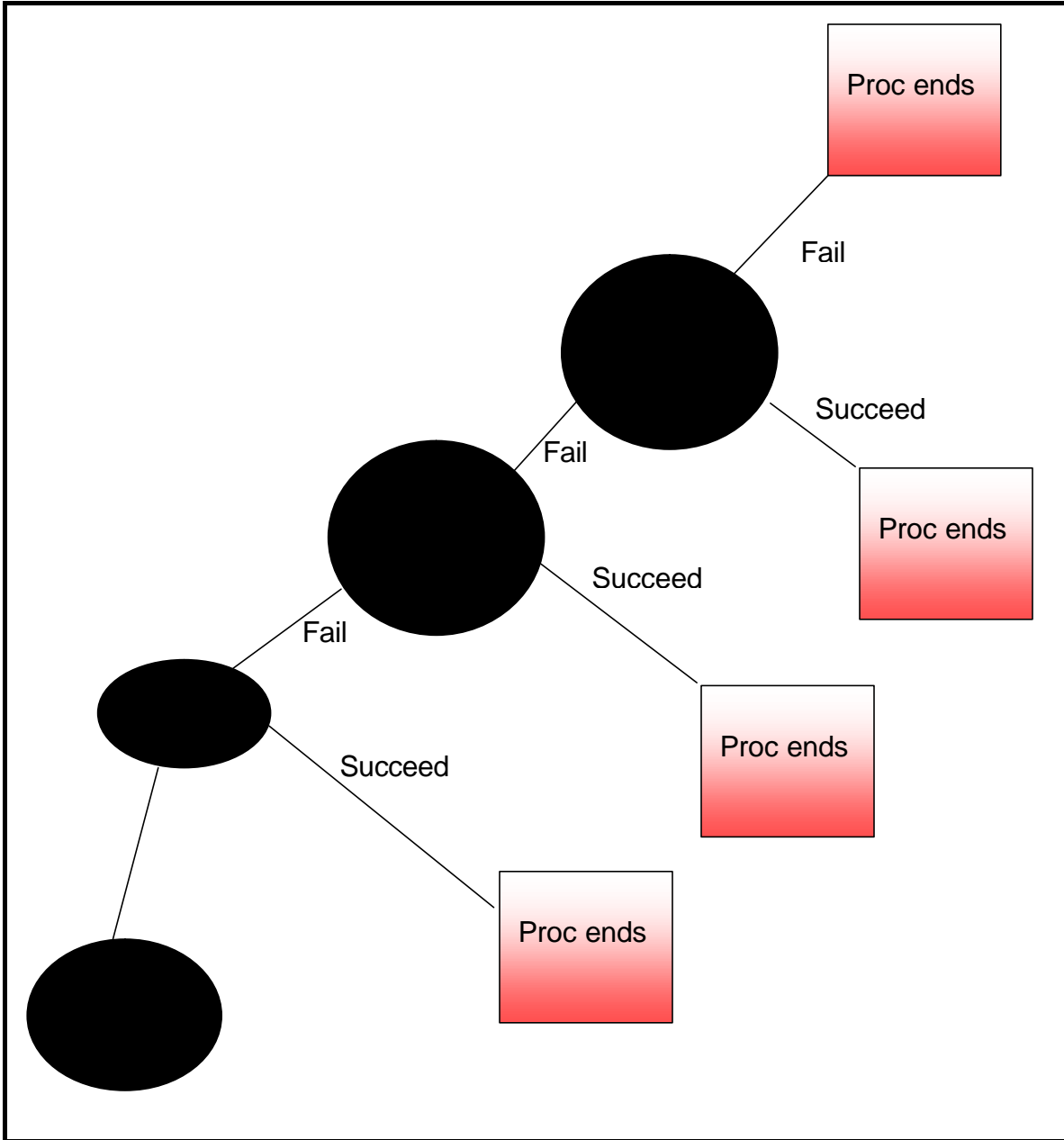


Figure 21: Decision tree

5.2.3 Decision Table

Conditions	R U L E S						
	1	2	3	4	5	6	7
SysAdmin wants email update	Y	Y	Y	N	N	N	N
Alpha unit failed in natural operation	Y	N	N	Y	N	N	N
Beta unit failed in natural operation	N	Y	N	N	Y	N	N
Configuration changes made	N	N	Y	N	N	Y	N
ACTIONS							
Send Email Alert	☞	☞					
Send GUI update	☞	☞	☞	☞	☞	☞	
Record in Log file	☞	☞		☞	☞		
Deliberate reboot of Beta unit		☞			☞		
Alpha promotion on Beta unit	☞			☞			
Restart Applications	☞		☞	☞		☞	

Table 10: Decision table

The reason why there are not 2^4 or 16 total rules is due to the fact that many of the conditions are mutually exclusive. For example, units cannot be crashing and have configuration changes implemented simultaneously. In addition, many of the scenarios presented are simply a reflection of whether or not the Systems Administrator wants an email update. Hence, through simplification and elimination of impossible outcomes, the decision table displays the above scenarios only.

5.2.4 Data Dictionary

Name	BNF
AppName	[Qmail Samba imapd ftpd]
ClusterOptions	{UserDiskQuota} + MainIPAddress + ServicesProvided + NotificationType
ConfigChanges	ConfigType + Settings
ConfigOptions	NotificationType
ConfigType	[SMTP IMAP SMB FTP Cluster]
CurrentStatus	AppName + Message
Data	[A B C ... a b c ... 0 1 2 ...]
DiskQuota	[0 1 2 ... MB]
DNSOptions	NameResType + NameResIPAddress
EmailNotification	SysAdminEmailAddress + AlertMessage
FormattedMessage	[A B C ... a b c ... 0 1 2 ...]
FormattedStatus	FormattedMessage
FTPOptions	StorageOptions
GUINotification	Message
IMAPOptions	Username + Password
LocalHomeDir	[A B C ... a b c ... 0 1 2 ...]
LocalMailbox	[A B C ... a b c ... 0 1 2 ...]
MainIPAddress	[0 1 2 ... A B C ...]
Message	[A B C ... a b c ... 0 1 2 ...]
NameResIPAddress	[0 1 2 ... A B C ...]

Table 11: Data dictionary

NameResType	[Primary Secondary Relay hosts]
NetworkData	[A B C ... a b c ... 0 1 2 ...]
NetworkService	PortNumber + ResponseData
NotificationType	{SysAdminEmailAddress}
Password	[A B C ... a b c ... 0 1 2 ...]
PID	[0 1 2 ...]
PortNumber	[0 1 2 ...]
RecieveFromSubnet	[A B C ... a b c ... 0 1 2 ...]
RequestForNetworkService	PortNumber + NetworkData
ResponseData	[A B C ... a b c ... 0 1 2 ...]
RestartSignal	PID + Signal
RunningAppStatus	AppName + Message
ServicesProvided	{[SMTP IMAP SMB FTP DNS]}
Settings	[SMTPOptions IMAPOptions SMBOptions FTPOptions DNS Options ClusterOptions]
Signal	[0 1 2 ...]
SMBOptions	StorageOptions
SMTPOptions	{RecieveFromSubnet} + SMTPUserData
SMTPUserData	Username + LocalMailbox
StorageOptions	Username + Password + LocalHomeDir
SysAdminEmailAddress	[A B C ... a b c ... 0 1 2 ...]
UserDiskQuota	Username + DiskQuota
Username	[A B C ... a b c ... 0 1 2 ...]

5.3 ERM Model

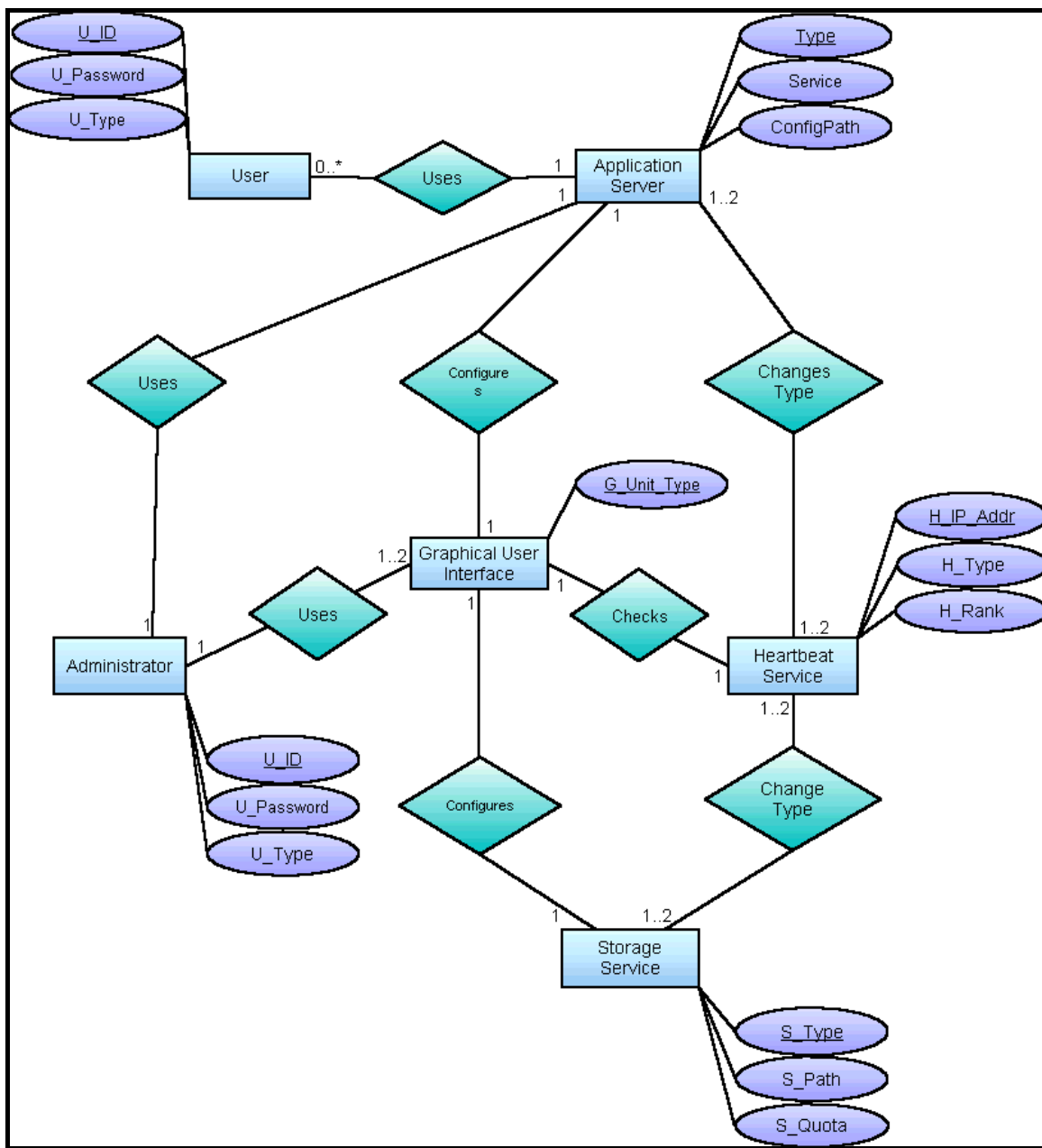


Figure 22: ERM diagram

1.4 Structured Chart

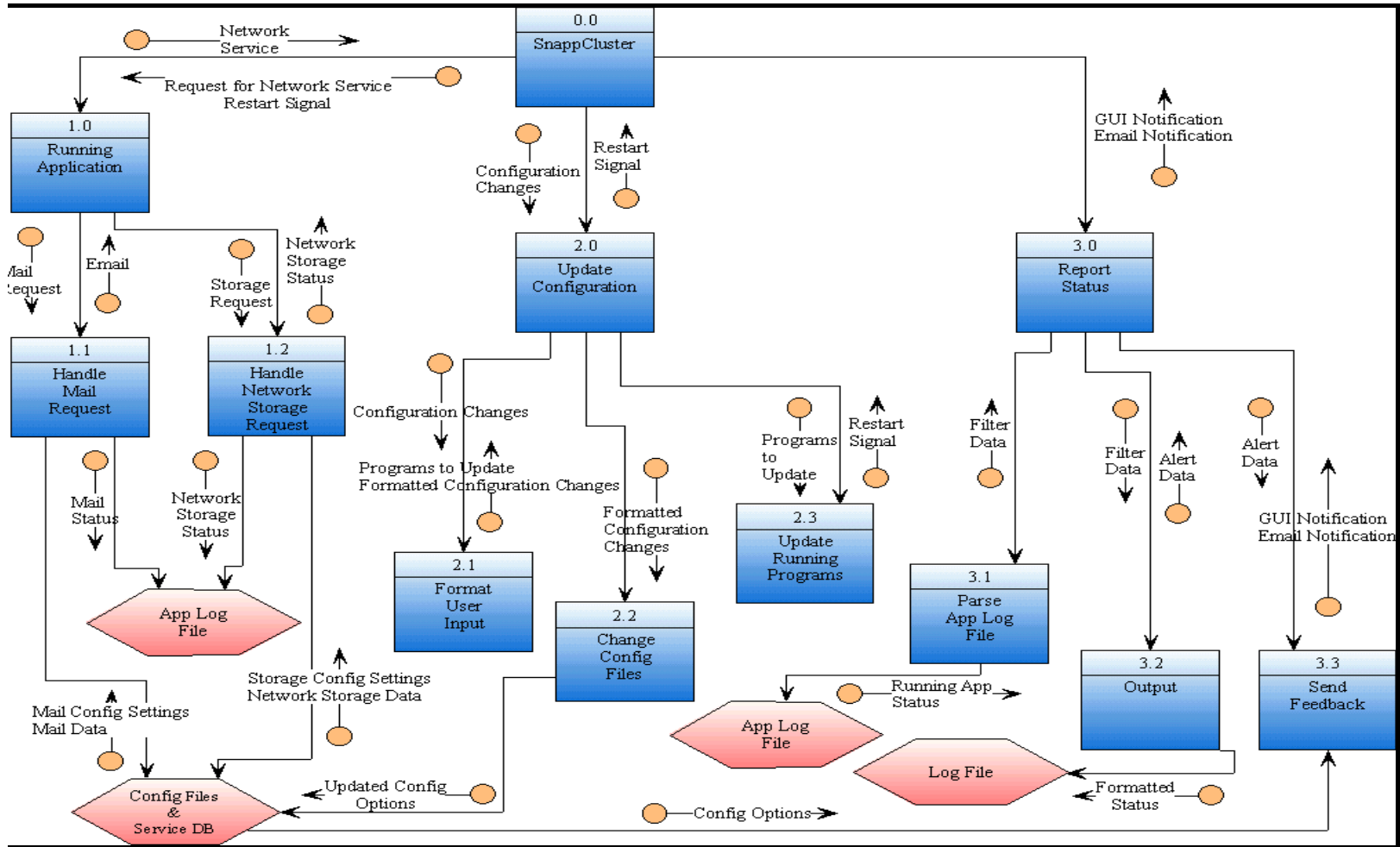


Figure 23: Structured chart

5.5 State Transition Diagram

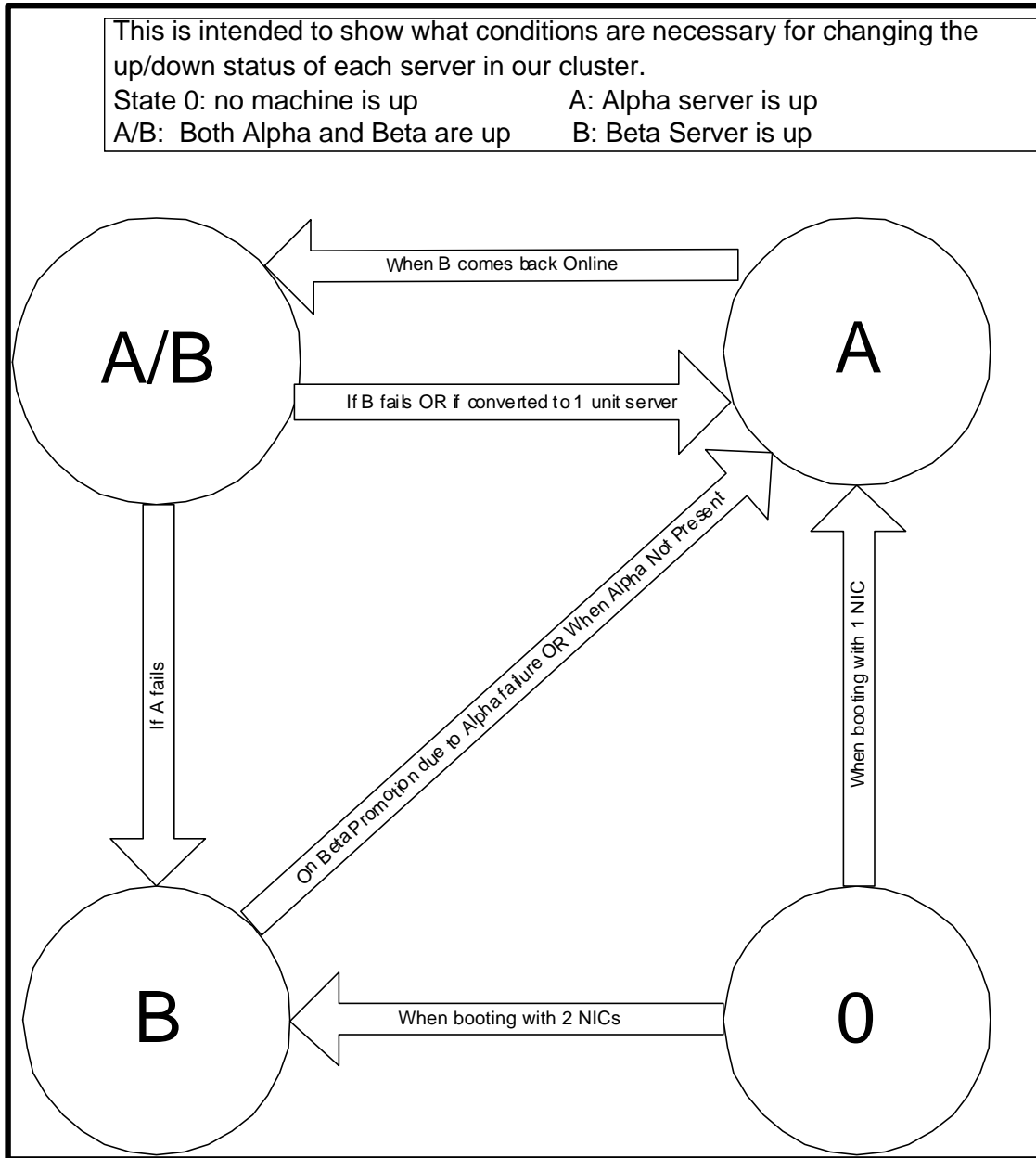


Figure 24: State transition diagram

Description of State Changes From

Starting State	Ending State	Description
0	A	
0	B	
A	A/B	
A	B	This should never happen as the system is built in such a way that this is impossible.
B	A	
B	A/B	This will never happen because B will promote to A before the other unit comes up, and when there are 2 units, the servers always start at B; so the "correct" sequence for B -> A/B is actually B -> A -> A/B.
A/B	A	
A/B	B	

Table 12: State transition table

5.6 Network Diagram

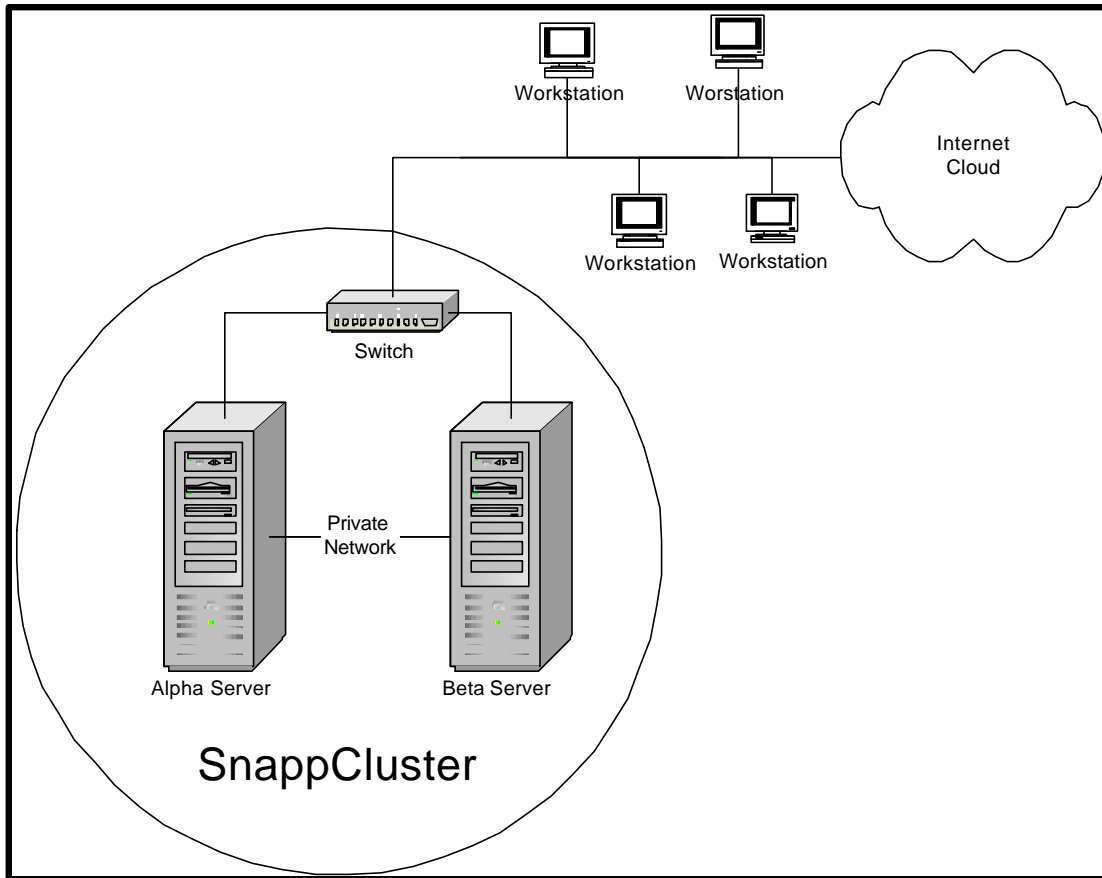


Figure 25: Network diagram

5.7 Repository Model

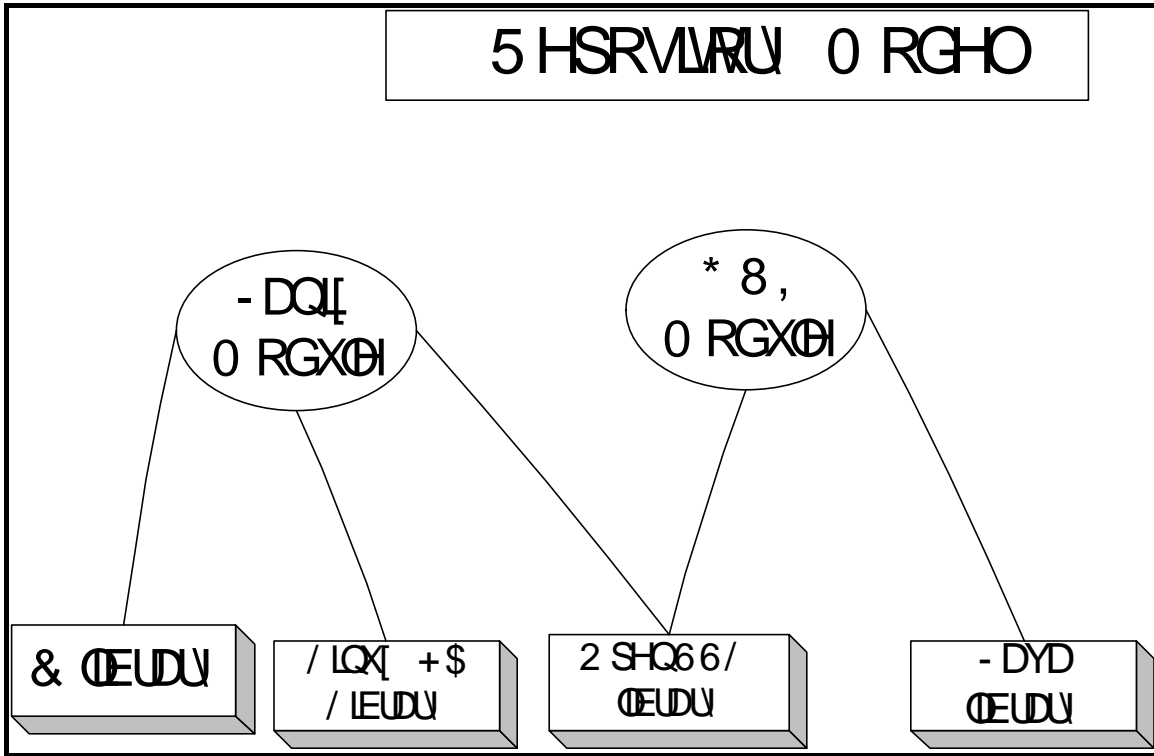
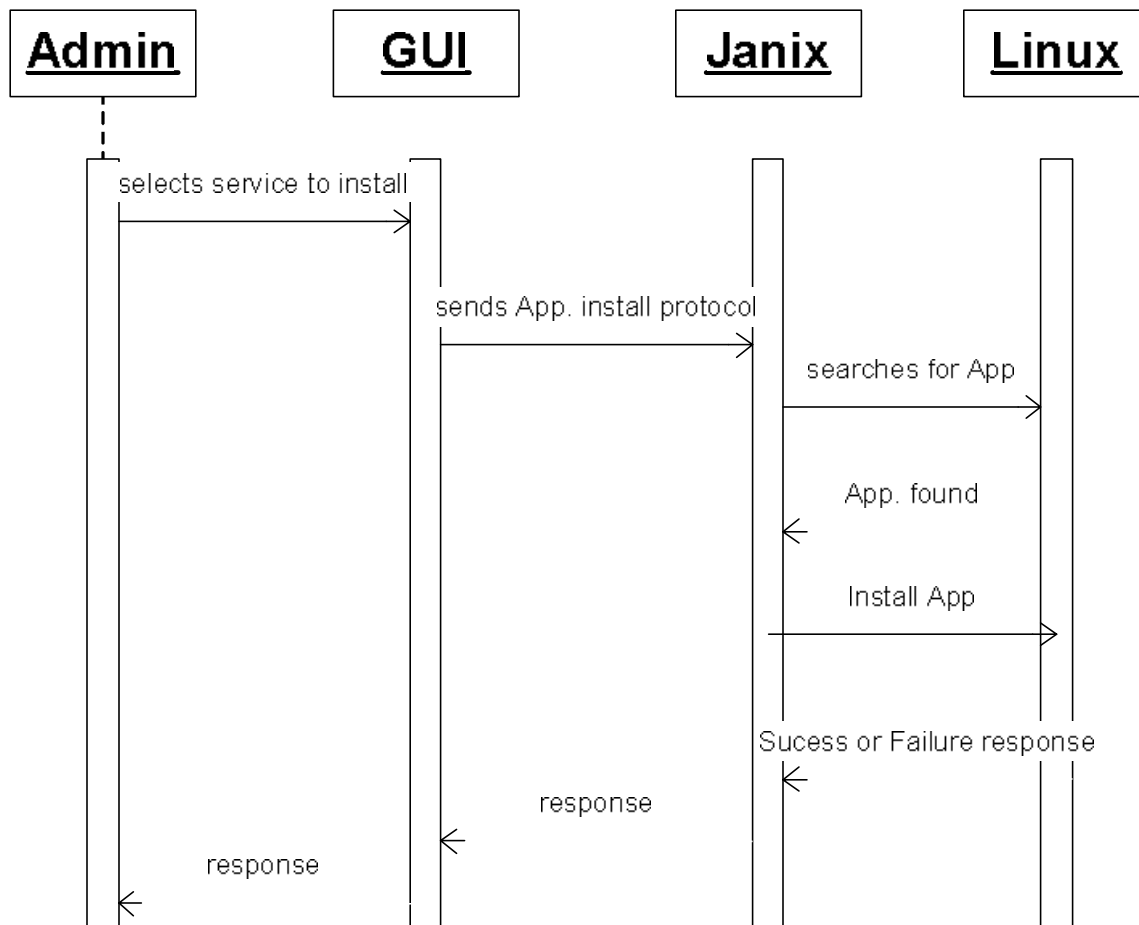


Figure 26: Repository model

5.8 Sequence Diagram – Installing a Service



5.9 Abstract Machine Diagram

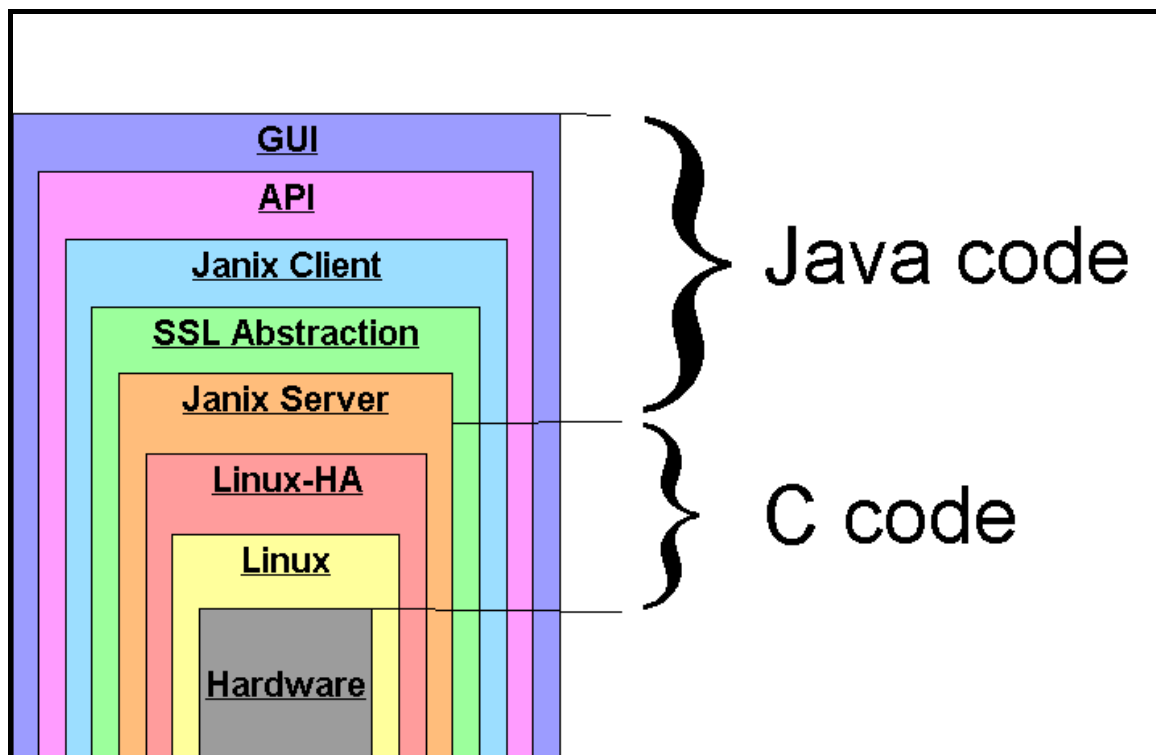


Figure 28: Abstract machine diagram

6. Testing

6.1 Function testing

Function testing has been performed and all requirements specified in section 3.5 have been met. The following table documents requirements that have not been met and the current plan to meet them.

Requirement Number (per section 3.5)	Testing Status	Requirement Disposition
1.2	Not able to get SSL (secure socket layer) to work.	A valid requirement, keep it.
5.2	FTP services were not completed.	A valid requirement, keep it.
6.2	The application tab of the GUI was not completed.	A valid requirement, keep it.

Table 13: Requirements not met

6.2 Performance testing

Under this heading, we performed compatibility, configuration, security, timing, and recovery tests.

6.2.1 Compatibility tests

Since the application is very dependent on the hardware installed on the server, numerous tests were run with as many types of hardware that the sponsor and the project team member could find. A list of the compatible hardware is provided as part of the application install CD.

6.2.2 Configuration tests

A number of tests were run with various configuration settings at the operating system level. The various applications running on the network were

also tested using various configuration settings. There are no known issues at this time.

6.2.3 Security tests

Tests were run to make sure that known holes that allow hackers to take control of a server running the same operating system as the SnappCluster application are in fact plugged. Tests were also run to make sure that once the cluster is locked, that people can't bypass the SnappCluster security. There are no known issues at this time.

6.2.4 Timing tests

Various tests were run to validate that the Beta server takes over in a reasonable amount of time when the Alpha server comes down. There are no known issues at this time.

6.2.5 Recovery tests

Various tests were run to make sure that once a fail-over condition is detected, that the cluster could recover correctly with minimal amount of lost data. There are no known issues at this time.

6.3 Acceptance testing

For this project, the plan is to have alpha, and beta testing. Since there is no computer-based system to parallel, parallel testing does not apply.

6.3.1 Alpha testing

Because the sponsor was able to provide servers, the team has built and been testing on these servers "in-house". The install module has the first module of the SnappCluster application to be tested followed by the actual cluster and GUI modules.

6.3.2 Beta testing

The SnappCluster application was installed at the sponsor site on a not very important server. Both the sponsor and this team consider this beta testing. There are no known issues outstanding from this testing at this time. The sponsor has made suggestions for improvements to the application and they will be considered as future releases of the application are developed.

6.4 Installation testing

With agreement from the sponsor, installing the SnappCluster application on their mission critical server will not occur until at least 4Q03.

7. Works Cited

¹ CarePaq(tm) for installation & start-up. 15 Jan. 2003

http://www.compaq.com/services/carepaq/us/install/cp_win2000.html

² Microsoft(r) Windows NT(r) Server Complete Overview. 13 Aug. 2001. 03 Feb. 2003.

<http://www.microsoft.com/ntserver/productinfo/enterprise/clustering/competeoverview.asp>.

³ Microsoft Cluster Server General Questions. 13 July 2001. 05 Feb. 2003

http://www.microsoft.com/ntserver/productinfo/enterprise/clustering/clustering_faqs.asp.

⁴ Microsoft® Windows NT® Server, Enterprise Edition . 7 Jan. 2003. 10 Feb. 2003.

⁵ Beowulf mailing list FAQ, version 2. 1995. 15 Feb. 2003

<<http://www.canonical.org/~kragen/beowulf-faq.txt>>.

⁶ Beowulf Org. 01 Jan. 2000. 11 Feb. 2003 <<http://beowulf.org>>.

⁷ The Beowulf Project at CACR. 28 Aug. 2001. California Institute of Technology.

15 Jan. 2003 <<http://www.cacr.caltech.edu/beowulf/index.html>>.

⁸ Red Hat Linux Advanced Server. 15 Jan. 2003

<http://www.in.redhat.com/products/linux/rhl_advanced.php3>.

- ? Abts, Chris, et al. Software Cost Estimation with COCOMO II. Upper Saddle River: Prentice Hall PTR, 2000.
- ? CBL FAQ. 15 Jan. 2003 <<http://www.cbltech.com/faq.html>>.
- ? DRDB. 02 Jan. 2003 <<http://www.complang.tuwien.ac.at/reisner/drdb/>>.
- ? Function Point Counting Practices Manual. 4.1.1st ed. Princeton Junction: International Function Point Users Group, 2000.
- ? <<http://www.microsoft.com/catalog/display.asp?site=458&subid=22&pg=8>>.
- ? Sams Teach Yourself Microsoft Project 2000 in 24 Hours. Indianapolis: Sams, 2000. 1-566.
- ? TechEncyclopedia. 02 Feb. 2003 <<http://www.techweb.com/encyclopedia/>>.
- ? The UNIXÆ Operating System: A Robust, Standardized Foundation for Cluster Architectures. 12 Feb. 2003 <<http://fsmllabs.com/developers/docs/html/susv2/whitepapers/cluster.html>>
- ? Isaacs, Ellen. Interviewing Customers: Discovering What They Can't Tell You
21 Oct 02 <http://www.izix.com/pro/interviewing/>)
- ? Microsoft® Project 2000
- ? Microsoft® Excel 2000
- ? Microsoft® VISIO 2000
- ? Smartdraw - <http://www.smartdraw.com>

8. List of Tables and Figures

Table 1: Methodology matrix	24
Table 2: Glossary	25
Table 3: Function point matrix.....	37
Table 4: Function point adjustment factors	38
Table 5: Tangible benefits worksheet.....	41
Table 6: One-time costs worksheet.....	41
Table 7: Recurring costs worksheet	42
Table 8: Risk management.....	46
Table 9: Brainstorming prioritization.....	63
Table 10: Decision table.....	87
Table 11: Data dictionary	88
Table 12: State transition table	94
Table 13: Requirements not met.....	99
Figure 1: Scrum diagram.....	20
Figure 2: Work breakdown structure	30
Figure 3: Project milestones	31
Figure 4: Gantt chart.....	32
Figure 5: Pert chart.....	33
Figure 6: Cost benefit analysis.....	44
Figure 7: Break-even graph.....	45
Figure 8: Use case scenario	61
Figure 9: Brainstorming	62
Figure 10: AS-IS DFD - Context diagram.....	71
Figure 11: AS-IS DFD - General diagram	72
Figure 12: AS-IS DFD - Network management - 2.x.....	73
Figure 13: AS-IS DFD - Problem analysis - 2.1.x	74
Figure 14: AS-IS DFD – Determine server status – 2.1.1.x	75
Figure 15: TO-BE DFD - Context diagram.....	76
Figure 16: TO-BE DFD – General diagram	77
Figure 17: TO-BE DFD - Running application - 1.x.....	78
Figure 18: TO-BE DFD - Update configuration - 2.x	78
Figure 19: TO-BE DFD - Prepare restart signal - 2.3.x.....	79
Figure 20: TO-BE DFD - Report status.....	79
Figure 21: Decision tree	86
Figure 22: ERM diagram	90
Figure 23: Structured chart.....	92
Figure 24: State transition diagram	93
Figure 25: Network diagram	95
Figure 26: Repository model.....	96
Figure 27: Sequence diagram.....	97
Figure 28: Abstract machine diagram.....	98

9. Extra Work

Data Dictionary – Provided 42 entries (+32 Above the minimum	88
Structure English – provided five diagrams (+? above the minimum)	85
TO-BE DFD – provided six diagrams (+1 above the minimum)	76