SMART ANALYZER

Major Project

Submitted towards partial fulfillment of the Degree of **Bachelor of Engineering**

Year 2010

Department Of Information Technology

Guided By:
Mr. Anand S. Rajawat
Mr. Anurag Golwelkar

Submitted By:

Apoorv Shrivastava	(0802IT071020)
Minakshi Gupta	(0802IT071037)
Pulkit Verma	(0802IT071047)
Udayan Gupta	(0802IT071058)

Shri Vaishnav Institute of Technology and Science, Indore

Shri Vaishnav Institute of Technology and Science, Indore

<u>Certificate</u>

This is to Certify that **Mr. Apoorv Shrivastava, Ms.Minakshi Gupta, Mr.Pulkit Verma** and **Mr. Udayan Gupta** working in a Group have satisfactorily completed the major project titled “Smart Analyzer” towards the partial fulfillment of the degree in Bachelor of Engineering (Information Technology) Awarded by Rajiv Gandhi Technical University, Bhopal for the academic year 2010.

Head of Department

Date:

Internal Examiner

Date:

External Examiner

Date:

ACKNOWLEDGEMENT

“We follow your foot-steps, we move on the path shown by you, we acknowledge you, and we are proud to have guides like you.”

We feel it our proud privilege to express our deep sense of gratitude and indebtedness to **Dr. Ashish Bansal** (Head of Department), **Mr. Anand Rajavat** (Project Incharge)**, Mr. Anand S. Rajawat** and **Mr. Anurag Golwelkar** (Project Guides), **Mr. Rajesh Chakrawati, Ms. Shweta Pandey** and **Mr. Vijay Prakash** (Project Coordinators) for providing their painstaking and untiring supervision. We own our deep sense of gratitude and thanks for their constructive criticism, valuable suggestions and constant encouragement at all stages of development of this project.

We wish to express our sincere thanks to all the faculty of Information Technology Department, for providing a conductive environment for proper development of project and the necessary facilities for completion of project.

We also express our sincere thanks and gratitude to all of them without whose constant support and guidance, this project would not have been a success.
Last but not the least we owe sincere thanks to Dr. Rajeev G. Vishwakarma for mentoring projects at **Centre of Excellence at SVITS**.

Apoorv Shrivastava

Minakshi Gupta

Pulkit Verma

Udayan Gupta

Table of Contents

Smart Analyzer

Problem Definition

Version 1.0

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Problem Definition of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Problem Definition

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the proxy server like SQUID on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

The major problems with the existing system are:

1. The systems do not provide the graphical analysis tools for easy understandability.

2. The systems initiate processes that consume large number of resources hence put an additional load on the server.

3. Various formats of information in log files are difficult to analyze and understand.

There is not any simplified server monitor tool used on many proxy servers. Other solutions include manual analysis in which a user has to analyze the log file manually which is tedious and cumbersome job. Access Log Analysis in manual way is highly complex and doesn't provide the fast analysis. Smarter Log Analyzer provides a simple and reliable way of analyzing the log files.

The existing systems are very hard to handle as they use manual analysis in which a user has to analyze the log file manually which is tedious and cumbersome job. Access Log Analysis in manual way is highly complex and doesn't provide the fast analysis. The Access Log Analyzer provides a simple and reliable way of analyzing the log files.

In our proposed system we overcome the problem of readability of log files by providing the various commonly used formats of respective data items. Also the graphical analysis facility for various dynamically changing parameters like the URL being accessed presently and the number of times a page or a website is accessed.

The problem of unreadability of log file is solved by designing a graphical user interface which is easy to handle and operate. As the system is fully automated only one user that is proxy server administrator can complete the task of analysing the logs on proxy server. The time spent for each activity is constant for all users.

Smarter Log Analyzer for Proxy Server can be implemented successfully in the commercial market. We intend to overcome the problems of intractability by producing a spreadsheet type system that the user can guide in an informed and useful way. The new project will raise the level of managing computer and will develop a revolution in the same field.

As the system is fully automated only one user that is proxy server administrator can complete the task of analysing the logs on proxy server. The time spent for each activity is constant for all users. The system is implemented using Java which is platform independent, so application can run on all platforms.

The system is implemented using Java which is platform independent, so application can run on all platforms. As the application can be accessed directly by proxy server administrator, no hardware budget is required. The only cost will be the cost of implementing and maintaining the system.

The system is to be made so as to support minimal resources for the GUI and still provide basic functionalities to monitor the server.

Smart Analyzer

Vision

Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Vision of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Vision of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present an access log analyzer for proxy server. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the mentors and the developers of the system.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 References

Applicable references are:

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.
- Web object life measurement using Squid Log File - *Khunkitti A,Intraha W,* ISBN 0-7695-1187-4.
- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)
- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

2. Positioning

2.1 Business Opportunity

Access Analyzer for Proxy Server can be implemented successfully in the commercial market. We intend to overcome the problems of intractability by producing a spreadsheet type system that the user can guide in an informed and useful way. The new project will raise the level of managing computer and will develop a revolution in the same field.

2.2 Problem Statement

The problem of	Analyzing a access file on a proxy server is very difficult task due to complexity of the format of log file
Affects	Server Administrator
the impact of which is	It results in the waste of time.
a successful solution would be	This problem can be overcome through a system which provides a simple view of the data in the log file.

2.3 Product Position Statement

For	Systems where proxy server is used as a gateway to access other networks (like Internet).
Who	Effective way of analyzing logs on proxy server
The (product name)	is a Software
That	Enables server administrator to easily analyze the traffic on proxy server.
Unlike	The existing system for reading log files manually.

3. Stakeholder and User Descriptions

There is only one user to this system known as Server Administrator.

3.1 Market Demographics

The Software can be easily installed onto the system and user can very easily handle it. The initial release of this system will be limited to Administrator on a single server.

3.2 Stakeholder Summary

Name	Description	Responsibilities
Project Guide	CSE Department and SVITS college as a whole	Responsible for project funding approval. Monitors project progress
Customer	Proxy Server Administrator	Ensures that the system will meet the needs of the server administrator.
System Designers	Other stakeholders include system designers who design & look the entire system	The responsibility of the system designers is to develop a system which is easily maintainable, secure and error free to increase the market demand of the product.

3.3 User Summary

Name	Description	Responsibilities
Proxy Server Administrator	Acts as administrator and is main user of the system	Responsible for analyzing the traffic on the server.
System Designers	Other stakeholders include system designers who design & look the entire system	The responsibility of the system designers is to develop a system which is easily maintainable, secure and error free to increase the market demand of the product.

3.4 User Environment

As the system is fully automated only one user that is proxy server administrator can complete the task of analysing the logs on proxy server. The time spent for each activity is constant for all users. The system is implemented using Java which is platform independent, so application can run on all platforms.

3.5 Key Stakeholder or User Needs

There is not any simplified server monitor tool used on many proxy servers. Other solutions include manual analysis in which a user has to analyze the log file manually which is tedious and cumbersome job. Access Log Analysis in manual way is highly complex and doesn’t provide the fast analysis. The Access Log Analyzer provides a simple and reliable way of analyzing the log files.

3.6 Alternatives and Competition

The system is developed in its generalized form for any Proxy Server. Other alternatives available provide manual way of analyzing log file which is useful on proxy server where the traffic is very low and the number of nodes handled by it is less.

4. Product Overview

This section provides a high level view of the structure of exam scheduling system.

4.1 Product Perspective

The access log analyzer will act as an interface for analyzing the logs and generate graphs for various parameters.

4.2 Assumptions and Dependencies

The following assumptions and dependencies relate to the capabilities of the exam scheduling system as outlined in this Vision Document:

- The user must be fluent with English.
- He must know the working of the system.
- The system is using existing college database.

4.3 Summary of capabilities:

Customer Benefit	Supporting Features
Reduced total time	Proxy Server Administrator can analyze the logs in many effective ways cutting the effective time.
Customer satisfaction is improved	Graphical Analysis is implemented successfully.

4.4 Cost and Pricing

As the application can be accessed directly by proxy server administrator, no hardware budget is required. The only cost will be the cost of implementing and maintaining the system.

4.5 Licensing and Installation

There are no licensing requirements for system, as it will be available only to any University.

Installation of the client component must be available via diskette, CD, or downloadable from the Internet.

5. Product Features

5.1 Hardware requirements:

- Pentium IV, 750 MHz, 20GB HDD
- RAM: 64MB minimum
- 400MB Minimum Free Space on Drive
- One Server machine with Tomcat container
- Other Server with MySql
- A Color Monitor, a Keyboard and a Mouse (optionally).

5.2 Software requirements:

- Operating System: Linux OS.
- Apache Tomcat container
- MySql
- JDK 1.4 or above

6. Constraints

- It must be designed specifically for single user.
- Efficiency of the system depends on the algorithm used for analysis.

7. Quality Ranges

This section defines the quality ranges for performance, robustness, fault tolerance, usability, and similar characteristics for the exam scheduling system

Availability: The System shall be available 24 hours a day, 7 days a week.

Usability: The System shall be easy-to-use. The System shall include online help for the user. Users should not require the use of a hardcopy Manual to use the System.

Maintainability: The System shall be designed for ease of maintenance

8. Precedence and Priority

This section provides some direction on the relative importance of the proposed system features. The features defined in this Vision Document should be included in the first 2 releases of the system. As development progresses on this system, the feature attributes will be used to weight the relative importance of the features and to plan the release content. The benefit, effort, and risk attributes are used to determine priority of a feature and target release.

9. Other Product Requirements

9.1 Applicable Standards

The desktop user-interface shall be GNOME compliant.

9.2 System Requirements

- The client component of the system shall operate on any personal computer with a 486 Microprocessor or better.
- The client component of the system shall not require more than 32 MB RAM and 20 MB Disk Space.
- The client component of the system shall run on any flavor of Linux Operating System with SQUID Server running on it.

9.3 Performance Requirements

The system shall initiate the execution of 85% of the desired tasks specified within 10 seconds.

9.4 Environmental Requirements

None

10. Documentation Requirements

10.1 User Manual

The User Manual shall describe use of the System from the operator's view point. The User Manual shall include:

- Minimum System Requirements
- Installation of the system
- Logging On
- Logging Off
- All System Features
- Customer Support Information

10.2 Online Help

Online Help shall be available to the user for each system function. Each topic covered in the User Manual shall also be available through the online help.

10.3 Installation Guides, Configuration, and Read Me File

The installation for exam scheduling includes the following things-

- Minimum Hardware Requirements
- Installation Instructions
- User Specific Parameters
- Customer Support Information
- How to Order Upgrades

The Read Me File shall be available for display following installation. The Read Me File will also reside on disk and be available for viewing at any time by the user. The Read Me File shall include:

- New release features
- Known bugs and workarounds.

10.4 Labeling and Packaging

The SVITS College logo shall be prominent on the user documentation and splash screens.

Smart Analyzer

Glossary

Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Glossary of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Glossary of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present glossary for an access log analyzer for proxy server. The glossary contains the working definitions for terms and classes in the access analyzer for proxy server. This glossary will be expanded throughout the life of the project.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.
- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.
- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)
- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.4 Overview

This document will contain the following sections:

- ***Definitions***

2. Definitions

Term	**Description**	**Additional Information**
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.	Access logs in Squid server are stored in a file named “access.log” in /var/log/squid directory.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.	
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.	A coherent set of roles that users of use cases play when interacting with these use cases. An actor has one role for each use case with which it communicates.
Actor (instance)	Someone or something, outside the system that interacts with the system.	
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.	Analysis focuses on what to do;design focuses on how to do it.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.	
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.	
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.	Examples of Artifacts include models, source files, scripts, and binary executable files.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.	Cache Logs are stored in a file named “cache.log” in /var/log/squid directory.
Change	The activity of controlling and tracking	

Management	changes to artifacts.	
Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.	
Class	A description of a set of objects that share the same attributes, operations, methods, relations hips, and semantics.	A class may use a set of interfaces to specify collections of operations it provides to its environment.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.	
Class Hierarchy	The relationships among classes that share a single inheritance.	
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.	It is a class diagram that contains classifier roles and association roles rather than just classifiers and associations.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.	
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.	Included are artifacts such as training materials and installation procedures.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.	
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.	
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.	

Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.	
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.	
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.	
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.	
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.	
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.	
IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.	IP Address is unique for any device on a network.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.	Linux OS is available as various distros like Ubuntu, Fedora, Red Hat, Suse, etc.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.	

Prototype	A release that is not necessarily subject to change management and configuration control.	
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.	This allows a number of systems to connect to the internet via a single IP address.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.	
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.	
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.	A server program is simply a program that operates as a socket listener.
Stakeholder	An individual who is who is materially affected by the outcome of the system.	
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.	
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.	Store Logs are stored in a file named “store.log” in /var/log/squid directory.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.	Squid includes limited support for several protocols includin g HTTP, FTP, TLS, SSL, Internet Gopher and HTTPS.
Unicode	A character coding system designed to support the interchange, processing, and display of the written texts of the diverse languages of the modern world.	Unicode characters are typically encoded using 16-bit integral unsigned numbers.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system	

URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.	
Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.	A use case defines a set of use-case instances or scenarios.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.	
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.	
Use Case Model	A model that describes a system's functional requirements in terms of use cases.	
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.	
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.	
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.	
Workspace	The work area that contains all the code you are currently working on; that is, current editions.	

Smart Analyzer

Requirements Management Plan

Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Requirements Management Plan of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Requirements Management Plan of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Overview

1.1 Purpose

The purpose of this document is to establish a common understanding of technical and non-technical requirements of the system that will be addressed throughout the lifecycle of this project. The goals of requirements management are to ensure that requirements are controlled to establish a baseline for development, acquisition, or management; and to ensure plans, work products, and activities are consistent with the requirements.

1.2 Scope

- Simplified view of the access logs on Squid server is provided.
- Dynamic view of various URLs being accessed from various IPs connected via proxy server.
- Graphical view of URLs being accessed versus following parameters is provided:
  - Time window during which URL is accessed.
  - Number of times URL is accessed.
- Break-up of the analysis on the basis of IP from which URL was accessed.
- Bandwidth used by each system can be viewed.
- Break-up analysis on the basis of domain of website.
- Analyze the processing time taken by proxy server to service a particular system.
- Analyze the complete user traffic of a system.

Detailed analysis of scope:

What shall be done?	**Simplified view of the access logs on Squid server is provided.**
How shall it be done?	By providing a Graphical User Interface using simplified values for values like access time, etc.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

What shall be done?	**Dynamic view of various URLs being accessed from various IPs connected via proxy server.**
How shall it be done?	By accessing the log file and updating the database simultaneously in real time, from which analysis is done.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on back-end development (database management).

What shall be done?	**Graphical view of URLs being accessed versus various given parameters.**
How shall it be done?	By using APIs for providing dynamic image in Java Servlets.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on back-end development.

What shall be done?	**Break-up of the analysis on the basis of IP from which URL was accessed.**
How shall it be done?	By providing a Graphical User Interface to read entries for host (domain name or IP Address).
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

What shall be done?	**Bandwidth used by each system can be viewed.**
How shall it be done?	By providing a Graphical User Interface to show bandwidth usage by each user.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

What shall be done?	**Break-up analysis on the basis of domain of website.**
How shall it be done?	By providing a Graphical User Interface to show the domain-wise analysis.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

What shall be done?	**Analyze the processing time taken by proxy server to service a particular system.**
How shall it be done?	By providing a Graphical User Interface to show processing time of each system.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

What shall be done?	**Analyze the complete user traffic of a system.**
How shall it be done?	By providing a Graphical User Interface to show the user-traffic analysis.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

1.3 Definitions, Acronyms and Abbreviations

Term	**Description**
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.
Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.

Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.
IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Prototype	A release that is not necessarily subject to change management and configuration control.

Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support the interchange, processing, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.
Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.

Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.
- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.
- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)
- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

Requirements Management Plan will contain 3 sections:

- Requirements Management.
- Levels of Requirements Management.
- Requirements Documentation and Organization.

2. Requirements Management

2.1 Organization Overview

ROLE	NAME	ORGANIZATION
Project Incharge	Anand Rajavat	CSE Department, SVITS
Project Guide	Anand S. Rajawat	CSE Department, SVITS
Project Co-guide	Anurag Golwelkar	CSE Department, SVITS
Project Developer	Apoorv Shrivastava	Student, IT 4th Year, SVITS
Project Developer	Minakshi Gupta	Student, IT 4th Year, SVITS
Project Developer	Pulkit Verma	Student, IT 4th Year, SVITS
Project Developer	Udayan Gupta	Student, IT 4th Year, SVITS

2.2 Tools

Software tools to be used in fulfilling the Requirements Management functions throughout the project.

- Rational Rose.
- Rational Software Architect.
- Rational Requirements Composer.

3. Levels of Requirement Management

The five levels of maturity for our RMM are:

1) Written
2) Organized
3) Structured
4) Traced, and
5) Integrated.

Actually, there is one other level on the requirements maturity scale: **Level Zero - no requirements.**

At Level Zero, they make broad assumptions that they know what to build; and the time in which the software can be build. Sometimes this gamble works, but more often than not, a product is released that is missing functionality, has functions that are not needed, or is of poor quality.

3.1 Level One - Written

The first step up is simply to write the requirements. Once you write requirements, several benefits become obvious. First, you have a real basis for a contract with your customer. If you write them well, the requirements can clearly state your understanding of what the customer wants you to build, and they can read the requirements and agree (or disagree). Second, everyone on your development team has a basis for doing his or her work. Third, as you staff up the project, new members, too, will have a source for figuring out what the application or system is supposed to do. Our project meets this level as we have written all the requirements in a well-organized manner which states all the objectives clearly. The customer is agreed with all requirements. Also, it helped us a lot while assigning the tasks to the members of the team based on the different objectives classified. Everyone on our development team has a basis for doing his or her work.

3.2 Level Two - Organized

At this level an organization deals with things like the quality of the requirements, their format, their security, where they are stored, and how to keep track of different versions. The goal of a requirement is to clearly communicate the proposed behavior of the software to a diverse set of constituents: users, customers and other stakeholders, as well as the development team. A good set of requirements does this job well; a bad one does not. Good requirements are understandable by stakeholders who specify them.

- **Formatting**
  Requirements must also be formatted effectively. Consistent numbering schemes, headers, fonts, and a good table of contents make your documents easier to read, understand, and use. We have used better fonts, headers and classified all the req. uniquely & effectively.

- **Accessibility, Security, and Version Control**
  At Level Two, you need a central, well known location for the requirements, one that is accessible by the server administrator. Limiting the ability to modify requirements to authorized persons only can help ensure the requirements' integrity. Costs associated with getting to Level Two relate mostly to training and review.

In our project we have given authentication levels to the administrator. When your requirements are more readable and easier to understand (and more trustworthy), you will have a better basis for a contract with the customer, the development team will find the requirements easier to use, and new staff will be able to come up to speed more quickly. Writing quality requirements is not a simple task. Getting to, and staying at, a given maturity level will require consistent review of requirements documents and some level of "process enforcement".

3.3 Level Three - Structured

Getting to Level Three involves being more specific about the "types" of requirements you gather. Are they functional or nonfunctional? Making these distinctions helps you get a better understanding of the requirements and manage them better.

- **Getting Your Types Straight**
  The first issue arises if you do not distinguish among different types of requirements. If your current requirements specification simply contains a big list of requirements with no indication of their type, it is likely that the list contains a mix of different types.

- **Attributes**
  All requirements are not created equal: Some are more important than others; some are more stable than others. These are important things to keep track of, and adding attributes for requirements can help you do so. Attributes include information that supplements the requirement text and helps you better understand and manage your requirements. Benefits of getting to Level Three revolve around better understanding and easier management.

In our project we have used well-structured requirements clearly identify different requirement types, and attributes provide the ability to query and filter groups of requirements. Better typing of requirements also provides greater assurance that the team has identified all important requirements. The main cost of getting to Level Three is in planning and maintenance. Determining appropriate requirement types and attributes is not a trivial task.

Usually this information is captured in a Requirements Management Plan (RMP). Then, requirements attributes are of little use if they are not kept up-to-date, so there is a maintenance burden that goes beyond the one for Level Two. There is an increased cost too, because determining the correct attribute values takes time.

3.4 Level Four - Traced

Most systems of any significant complexity will have a hierarchy of requirements. In a systems engineering environment, this hierarchy starts with system requirements and moves down to subsystem requirements, program requirements, and element requirements. In the Rational Unified Process, the hierarchy starts with user needs, proceeds to features, and then goes on to use cases. The ability to keep track of these relationships is commonly called traceability, and it entails identifying and documenting the derivation path (upward) and allocation/ flow down path (downward) of requirements in the hierarchy. Traceability provides the ability to understand how changes to one requirement might affect others (impact analysis) and can also help determine if your requirements are complete (coverage analysis). Usually, an organization at Level Four will develop an explicit traceability strategy prior to starting a project and then document it in the requirements management plan. The strategy will define the requirements levels and how they fit in the hierarchy. In addition, it will lay down some "rules" for requirements relationships.

3.5 Level Five - Integrated

It is often the case that requirements are used up front to get agreement from the customer on what the software is supposed to do, but then those requirements are not really tied in to the way the software is developed. This results in stale requirements and software that doesn't meet its objectives. Reaching Level Five means integrating the requirements management process with the rest of your software development environment. Software that does what the customer expects is built to comply with the requirements -- that is, the team's software development process uses the requirements as a key input. Integrating requirements into your change management process helps ensure that no changes are made to requirements without review and approval. A comprehensive, requirements-based software development process as described here takes significant planning, training, and process enforcement.

- **Requirements Management Tool Support**

  Until Level Five, it is theoretically possible to do everything that we have talked about either "manually" or with general-purpose tools like a word processor and spreadsheet. However, starting at Level Two, an RM tool can help you be far more efficient and consistent. Table 1 shows how the important features of Rational RequisitePro support key characteristics of the five RM maturity levels.

4. Requirements Documentation and Organization

4.1 Applicability

The access analyzer is designed and developed for the use by the administrator of the proxy server. It will make the analysis of the proxy server easier and hence will make the management tasks easier for the administrator done on the basis of this analysis.

4.2 Document Organization

The document will aim at developing the management plan for managing the resources needed for fulfilling the requirements of the project, that is, Access Analyzer for Proxy Server.
The document will contain the structure of the team and who will perform which task during the complete lifecycle of the project.

4.3 Applicable Documents

The documents that will control the requirement management plan contents are:

- Software Requirement Specification.
- Storyboard.
- Stakeholders Request Document.
- Use Case Requirements.

4.4 Changes and Revisions

Since the project is developed by a team formed as per the Democratic Decentralized Model so the whole team is responsible for controlling the changes in the requirement management plan and related information.

4.5 Issues

The issues that that affect implementation of the requirements management plan are as follows:
Training:

- Java will be used for developing the project as all team members are trained to use it.

Tools:

- Netbeans IDE will be used as the development tool due to its familiarity with all the team members.
- Since all the team members are trained to use MySql as the database server, so it will be implemented in the back-end development.

Smart Analyzer

Software Requirements

Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Software Requirements of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Software Requirements of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present software requirement for an access log analyzer for proxy server. The glossary contains the working definitions for terms and classes in the access analyzer for proxy server. This glossary will be expanded throughout the life of the project.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 Definitions, Acronyms and Abbreviations

Term	Description
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.

Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.

IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.
Prototype	A release that is not necessarily subject to change management and configuration control.
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support interchange, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.

Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.

- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.

- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)

- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

This document will contain the following sections:

- ***Software Requirements***

2. Software Requirements

- **Web Server:** Apache Tomcat Web Server, Linux based Operating System
- **Data Base Server:** MySQL, Linux based Operating System
- **Development End:** MySQL, Linux based Operating System (Ubuntu 10.10), Apache Tomcat Web Server.
- **Design Tool:** Rational Software Architect, Rational Requirement Composer, Rational Rose.

Smart Analyzer

Software Requirements Specification

Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Software Requirements Specification of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Software Requirements Specification of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present software requirement specification for an access log analyzer for proxy server. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the mentors and the developers of the system.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 Definitions, Acronyms and Abbreviations

Term	Description
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.

Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.

IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.
Prototype	A release that is not necessarily subject to change management and configuration control.
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support interchange, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system

URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.
Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.
- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.
- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)
- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

SRS will contain two sections:

- ***Overall Description*** will describe major components of the system, interconnection and external interfaces.
- ***Specific Requirements*** will describe the functions of actors, their role in the system and constraints.

2. Overall Description

2.1 Product Perspective

- The product will contain a GUI that will access the log file through a set of algorithms and will analyze the log file as per those algorithms.

2.2 Product Function

- **Access Logs :** The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption (eg. memory, disk space).
- **Cache Logs :** The Cache Log file contains the debug and error messages that Squid generates.
- **Store Logs :** The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
- **Analysis Data :** The analysis data will be stored in database in the form of tables, and will be updated dynamically.

2.3 User Characteristics

Every user should be comfortable of working with He must have basic knowledge of English too.

2.4 Constraints

- GUI is only in English.
- Login and password is used for authentication of the server administrator, biometric authentication is not used.

2.5 Assumptions and Dependencies

- Administrator account is created in system already.
- The roles and tasks of administrator are predefined.

3. Specific Requirements

3.1 Functionality

3.1.1. Simplified view of the access logs on Squid server is provided.

What shall be done?	**Simplified view of the access logs on Squid server is provided.**
How shall it be done?	By providing a Graphical User Interface using simplified values for values like access time, etc.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

3.1.2. Dynamic view of various URLs being accessed from various IPs connected via proxy server.

What shall be done?	**Dynamic view of various URLs being accessed from various IPs connected via proxy server.**
How shall it be done?	By accessing the log file and updating the database simultaneously in real time, from which analysis is done.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on back-end development (database management).

3.1.3. Graphical view of URLs being accessed versus following parameters is provided:

- Time window during which URL is accessed.
- Number of times URL is accessed.

What shall be done?	**Graphical view of URLs being accessed versus various given parameters.**
How shall it be done?	By using APIs for providing dynamic image in Java Servlets.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on back-end development.

3.1.4. Break-up of the analysis on the basis of IP from which URL was accessed**.**

What shall be done?	**Break-up of the analysis on the basis of IP from which URL was accessed.**
How shall it be done?	By providing a Graphical User Interface to read entries for host (domain name or IP Address).
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

3.1.5. Bandwidth used by each system can be viewed.

What shall be done?	**Bandwidth used by each system can be viewed.**
How shall it be done?	By providing a Graphical User Interface to show bandwidth usage by each user.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

3.1.6. Break-up analysis on the basis of domain of website.

What shall be done?	**Break-up analysis on the basis of domain of website.**
How shall it be done?	By providing a Graphical User Interface to show the domain-wise analysis.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

3.1.7. Break-up analysis on the basis of domain of website.

What shall be done?	**Analyze the processing time taken by proxy server to service a particular system.**
How shall it be done?	By providing a Graphical User Interface to show processing time of each system.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

3.1.8. Analyze the complete user traffic of a system.

What shall be done?	**Analyze the complete user traffic of a system.**
How shall it be done?	By providing a Graphical User Interface to show the user-traffic analysis.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on front-end development.

3.2 Usability

Our main criteria for making the system usable is the difficulty of performing each high-frequency use case. Difficulty depends on the number of steps, the knowledge that the user must have at each step, the decisions that the user must make at each step, and the mechanics of each step (e.g., typing a book title exactly is hard, clicking on a title in a list is easy).

The user interface should be as familiar as possible to users who have used other desktop applications. E.g., we will follow the UI guidelines for naming menus, buttons, and dialog boxes whenever possible.

3.3 Design Constraints

- Java will be used for developing the project as all team members are trained to use it.
- Netbeans IDE will be used as the development tool due to its familiarity with all the team members.
- Since all the team members are trained to use MySql as the database server, so it will be implemented in the back-end development.

3.4 On-line User Documentation and Help System Requirements

Since there will only be one user of the system and the project is not being implemented on a large scale so the documentation will be provided offline.

3.5 Purchased Components

No purchased components are used for development. The development is completely done on freeware development tools.

3.6 Interfaces

3.6.1 ***Software Interface****:*

- **Web Server:** Apache Tomcat Web Server, Linux based Operating System
- **Data Base Server:** MySQL, Linux based Operating System
- **Development End:** MySQL, Linux based Operating System (Ubuntu 10.04), Apache Tomcat Web Server.
- **Design Tool:** Rational Software Architect, Rational Requirement Composer.

3.6.2 ***Hardware Interface****:*

Minimum Requirements:

Technology	Processor	RAM	Disk Space
Firefox 2.0	Pentium II at 500MHz	64 MB	20 MB
Java SDK 1.6	Pentium III at 1GHz	512 MB	132 MB
Apache Tomcat Application Server V6.0	Pentium III at 1GHz	512 MB	1 GB
MySql	Pentium III at 1 GHz	512 MB	1GB (Excluding data size)

3.7 Licensing Requirements

No licensing requirements are needed.

Smart Analyzer

Stakeholder Requests

Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Stakeholder Requests of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Stakeholder Requests of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present the requests of stakeholders for an access log analyzer for proxy server. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the mentors and the developers of the system.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 Definitions, Acronyms and Abbreviations

Term	Description
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.

Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.

IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.
Prototype	A release that is not necessarily subject to change management and configuration control.
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support interchange, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.

Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.

- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.

- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)

- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

This document will contain four sections:

- ***Establish Stakeholder or User Profile.***
- ***Accessing the problem.***
- ***Understanding the User environment.***
- ***Accessing the opportunities***

2. Establish Stakeholder or User Profile

There is only one user to this system known as Server Administrator.

Name	**Description**	**Responsibilities**
Project Guide	CSE Department and SVITS college as a whole	Responsible for project funding approval. Monitors project progress
Customer	Proxy Server Administrator	Ensures that the system will meet the needs of the server administrator.
System Designers	Other stakeholders include system designers who design & look the entire system	The responsibility of the system designers is to develop a system which is easily maintainable, secure and error free to increase the market demand of the product.
Proxy Server Administrator	Acts as administrator and is main user of the system	Responsible for analyzing the traffic on the server.
System Designers	Other stakeholders include system designers who design & look the entire system	The responsibility of the system designers is to develop a system which is easily maintainable, secure and error free to increase the market demand of the product.

Establishing the User Profile for the stakeholder:

- Company / Industry: Shri Vaishnav Institute of Technology and Science, Indore
- Job Title: Server Administrator
- What are your key responsibilities?

  To manage the server and analyse the data patterns.
- What deliverables do you produce?

  No deliverables as such are produced as the main task is to ensure the proper working of the server.
- How is success measured?

  The server should run smoothly at all times without any technical glitches.
- Which problems interfere with your success?

  The power cut results in restart of the server, thereby hindering the continuous running of the server.
- Which, if any, trends make your job easier or harder?

  The job is simple as such, but sometimes problem with internet connectivity and various network related problems make the job tough.

3. Accessing the problem

- For which analysis problems do you lack good solutions?

  The analysis of SQUID server file for linux is quite tough as some entries in it are impossible to read.

- What are they?

  Some of the entries in the log file like that of access time are in non-readable format. So using it to analyse the access patterns of the webpages is quite difficult.

Ask for each problem:

- Why does this problem exist?

  The standard format of the log file has the problem.

- How do you solve it now?

  As such there is no proper solution. Hit and Trial is used.

- How would you like to solve it?

  A system to convert the non-readable entries into readable format.

4. Understanding the user environment

- Who are the users?

  Only the server administrator is the user.

- What is their educational background?

  Should have atleast the graduation level degree, degree related to computer field is preferred.

- What is their computer background?

  Should be proficient with the management of server, and should have a good knowledge of handling various types of servers.

- Are users experienced with this type of application?

  Yes experience is very important in this type of work.

- Which platforms are in use?

  Windows Server edition and Linux are mostly used.

- What are your plans for future platforms?

  Using everything based on linux is preferred due to improved security, but the lack of a good User interface is the reason for existence of Windows.

- Which additional applications do you use that we need to interface with?

  Also with analysis, if server management applications are included it is a huge boost to the Linux usage for server.

- What are your expectations for usability of the product?

  Nothing as such. It should have a good User interface and should be easily operatable.

- What are your expectations for training time?

  The person recruited for this kind of job is sufficiently trained. So no need for training is needed.
- What kinds of hard copy and on-line documentation do you need?

  A user manual would be highly useful.

5. Analyst's Inputs on Stakeholder's Problem

- Is this a real problem?

  Yes managing a server is really tough job.
- What are the reasons for this problem?

  The improved security features of linux also results in poor user friendly interface.
- How would you like to solve the problem?

  Creating a good user interface using which analysis becomes easy.

6. Accessing the solution

6.1 User Manual

The User Manual shall describe use of the System from the operator's view point. The User Manual shall include:

- Minimum System Requirements
- Installation of the system
- Logging On
- Logging Off
- All System Features
- Customer Support Information

6.2 Offline Help

Offline Help shall be available to the user for each system function. Each topic will be covered in the User Manual.

7. Accessing the Opportunity

The server administrator requires an effective way to monitor the traffic and to analyze the obtained results.

The Software can be easily installed onto the system and user can very easily handle it. It saves the time to analyze the results.

Smart Analyzer

Story Board

Version 1.1

Revision History

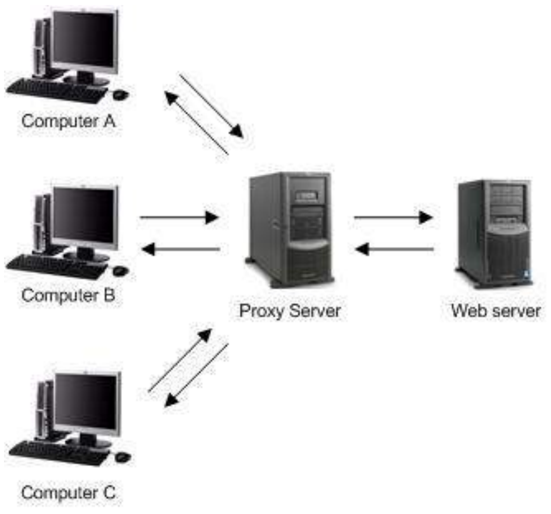
Date	Version	Description	Author
04/10/2010	1.0	Story Board of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Story Board of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

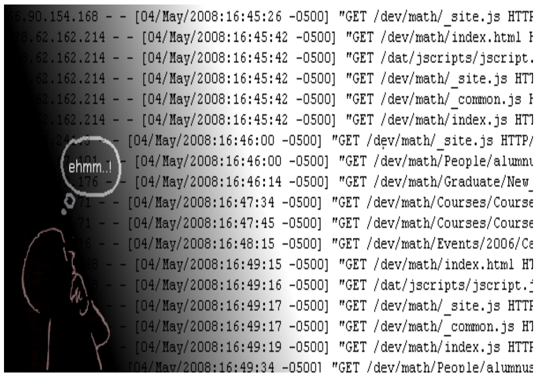
Table of Contents

1. Introduction

Storyboards are graphic organizers such as a series of illustrations or images displayed in sequence for the purpose of pre-visualizing a motion picture, animation, motion graphic or interactive media sequence, including website interactivity.

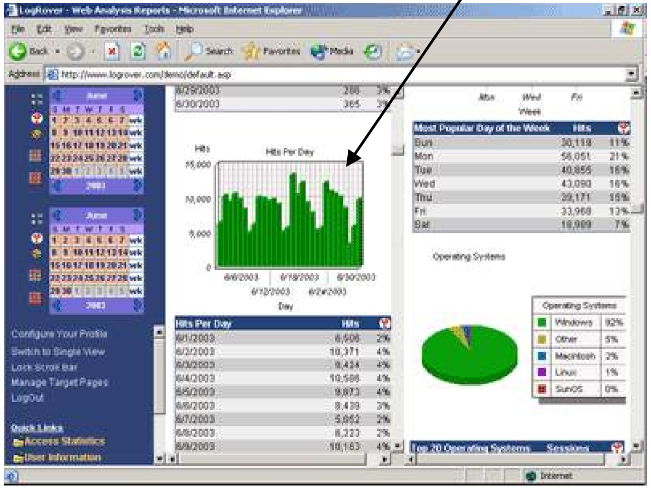
2. Story Board





```
66.90.154.168 - - [04/May/2008:16:45:26 -0500] "GET /dev/math/_site.js HTTP
28.62.162.214 - - [04/May/2008:16:45:42 -0500] "GET /dev/math/index.html H
 8.62.162.214 - - [04/May/2008:16:45:42 -0500] "GET /dat/jscripts/jscript.
  .62.162.214 - - [04/May/2008:16:45:42 -0500] "GET /dev/math/_site.js HTT
   62.162.214 - - [04/May/2008:16:45:42 -0500] "GET /dev/math/_common.js H
    .162.214 - - [04/May/2008:16:45:42 -0500] "GET /dev/math/index.js HTT
             - [04/May/2008:16:46:00 -0500] "GET /dev/math/_site.js HTTP/
              - [04/May/2008:16:46:00 -0500] "GET /dev/math/People/alumnu
      .176 - - [04/May/2008:16:46:14 -0500] "GET /dev/math/Graduate/New_
        71 - - [04/May/2008:16:47:34 -0500] "GET /dev/math/Courses/Course
        71 - - [04/May/2008:16:47:45 -0500] "GET /dev/math/Courses/Course
        36 - - [04/May/2008:16:48:15 -0500] "GET /dev/math/Events/2006/Ca
        48 - - [04/May/2008:16:49:15 -0500] "GET /dev/math/index.html HT
         8 - - [04/May/2008:16:49:16 -0500] "GET /dat/jscripts/jscript.j
           - - [04/May/2008:16:49:17 -0500] "GET /dev/math/_site.js HTTP
             - [04/May/2008:16:49:17 -0500] "GET /dev/math/_common.js HT
             - [04/May/2008:16:49:19 -0500] "GET /dev/math/index.js HTTP
               [04/May/2008:16:49:34 -0500] "GET /dev/math/People/alumnus
```

Client are communicating with Proxy server ⟶ Server Admin finding difficulty in analyzing log files.



6/29/2003	288	3%
6/30/2003	365	3%

Hits Per Day	Hits	
6/1/2003	6,506	2%
6/2/2003	10,371	4%
6/3/2003	9,424	4%
6/4/2003	10,588	4%
6/5/2003	9,873	4%
6/6/2003	8,439	3%
6/7/2003	5,052	2%
6/8/2003	6,223	2%
6/9/2003	10,163	4%

Most Popular Day of the Week	Hits	
Sun	30,118	11%
Mon	56,051	21%
Tue	40,855	16%
Wed	43,090	16%
Thu	39,171	15%
Fri	33,968	13%
Sat	18,909	7%

Operating Systems	
🟩 Windows	92%
🟨 Other	5%
🟦 Macintosh	2%
🟪 Linux	1%
🟥 SunOS	0%

Top 20 Operating Systems	Sessions

Smart analyzer generates a easily understandable log file report.

Smart Analyzer

Supplementary Specification

Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Supplementary Specification of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Supplementary Specification of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present supplementary requirements specification for an access log analyzer for proxy server. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the mentors and the developers of the system.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 Definitions, Acronyms and Abbreviations

Term	Description
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.

Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.

IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.
Prototype	A release that is not necessarily subject to change management and configuration control.
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support interchange, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.

Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.

- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.

- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)

- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

Supplementary Specifications will contain following sections:

- *Functionality*
- *Usability*
- *Design Constraints*
- *On-line User Documentation and Help System Requirements*
- *Purchased Components*
- *Interfaces*
- *Licensing Requirements*

2. Functionality

- Having only one username and password, that is, of administrator so the security issues are properly addressed.

What shall be done?	**Having only one username and password, that is, of administrator so the security issues are properly addressed.**
How shall it be done?	By providing a single user field in the database.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on back-end development.

- Having hours of operations that are 24X7, since system is an automated process.

What shall be done?	**Having hours of operations that are 24X7.**
How shall it be done?	By keeping the resource utilization low for the system.
When it must be performed?	During the implementation phase.
Who will perform it?	Team members working on back-end.

3. Usability

Our main criteria for making the system usable is the difficulty of performing each high-frequency use case. Difficulty depends on the number of steps, the knowledge that the user must have at each step, the decisions that the user must make at each step, and the mechanics of each step (e.g., typing a book title exactly is hard, clicking on a title in a list is easy).The user interface should be as familiar as possible to users who have used other desktop applications. E.g., we will follow the UI guidelines for naming menus, buttons, and dialog boxes whenever possible.

4. Design Constraints

- Java will be used for developing the project as all team members are trained to use it.
- Netbeans IDE will be used as the development tool due to its familiarity with all the team members.
- Since all the team members are trained to use MySql as the database server, so it will be implemented in the back-end development.

5. On-line User Documentation and Help System Requirements

Since there will only be one user of the system and the project is not being implemented on a large scale so the documentation will be provided offline.

6. Purchased Components

No purchased components are used for development. The development is completely done on freeware development tools.

7. Interfaces

7.1 Software Interface:

- **Web Server:** Apache Tomcat Web Server, Linux based Operating System
- **Data Base Server:** MySQL, Linux based Operating System
- **Development End:** MySQL, Linux based Operating System (Ubuntu 10.04), Apache Tomcat Web Server.
- **Design Tool:** Rational Software Architect, Rational Requirement Composer.

7.2 Hardware Interface:

Minimum Requirements:

Technology	Processor	RAM	Disk Space
Firefox 2.0	Pentium II at 500MHz	64 MB	20 MB
Java SDK 1.6	Pentium III at 1GHz	512 MB	132 MB
Apache Tomcat Application Server V6.0	Pentium III at 1GHz	512 MB	1 GB
MySql	Pentium III at 1 GHz	512 MB	1GB (Excluding data size)

8. Licensing Requirements

No licensing requirements are needed.

Smart Analyzer

Use Case Model

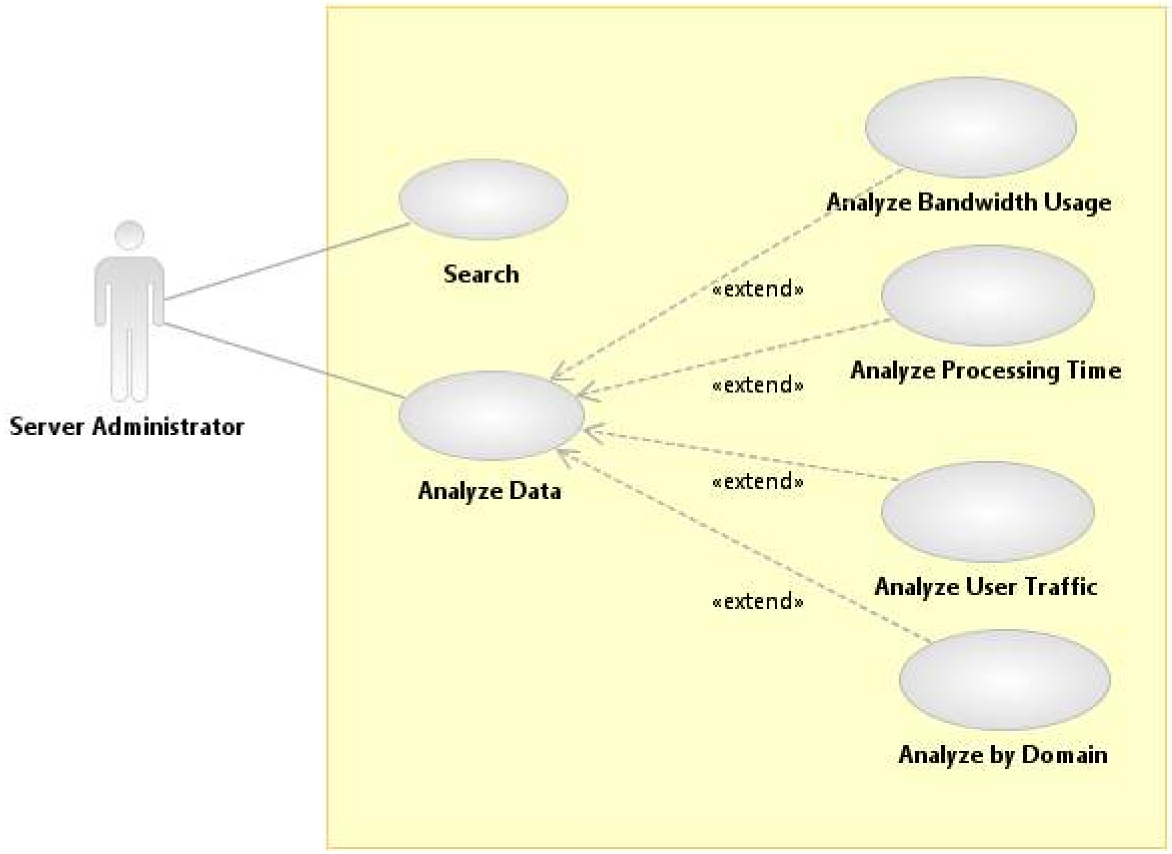
Version 1.1

Revision History

Date	Version	Description	Author
04/10/2010	1.0	Use Case Model of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta
30/11/2010	1.1	Use Case Model of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Use Case Model Survey



Server Administrator: The user of the system responsible for analyzing the data on the proxy server.

- **Search:** The server administrator can search any string in the logs. The administrator will enter the search string and the system will match the string in the available logs. If a match is found it is reported to the user.

- **Analyze Data:** The server administrator will analyze the access logs on the proxy server. He can find various parameters with respect to the IP address.

2. Use Case Report

2.1 Use Case: Search

- **Brief Introduction:** The server administrator can search any string in the logs. The administrator will enter the search string and the system will match the string in the available logs. If a match is found it is reported to the user.

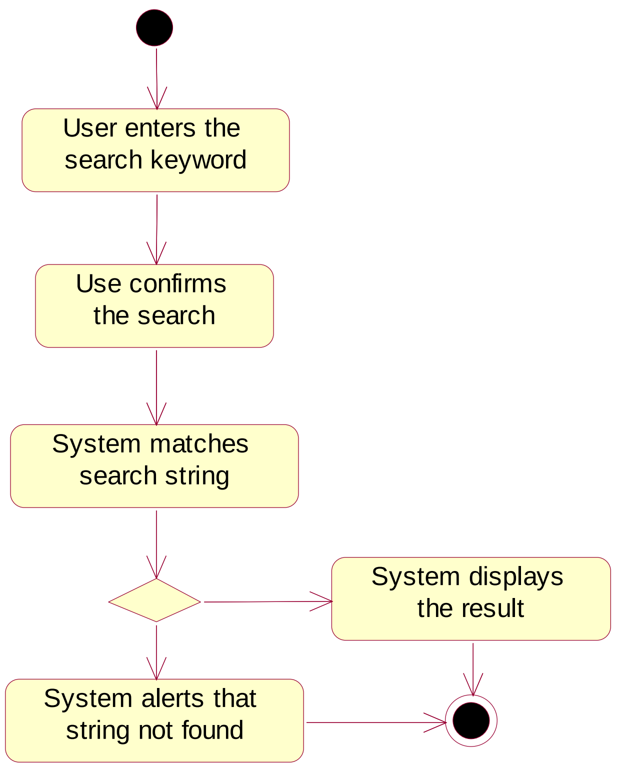
- **Flow Of Events:**

BASIC FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	System generate search page which request server administrator to enter search string.	----------	-----------
2.	Server Administrator	The server administrator enters the search string.	----------	-----------
3.	Server Administrator	System generates required page listing all the available matching results.	Search string not found	Alternate flow
4.	Server Administrator	Use Case Ends.	-----------	------------

ALTERNATE FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	The System displays an alert message telling that the search string does not match any search	---------	-----------
2.	Server Administrator	The actor can choose to either return to the main flow or cancel the searching, at which point the use case ends.	-----------	----------- -


User enters the search keyword
Use confirms the search
System matches search string
System displays the result
System alerts that string not found

- **Precondition:** None.
- **Post condition:** If the use case was successful, the valid list of search results is shown.
- **Actor:** Server Administrator.
- **Special Requirements:** None.

2.2 Use Case: Analyze Data

- **Brief Introduction:** The server administrator will analyze the access logs on the proxy server. He can find various parameters with respect to the IP address.

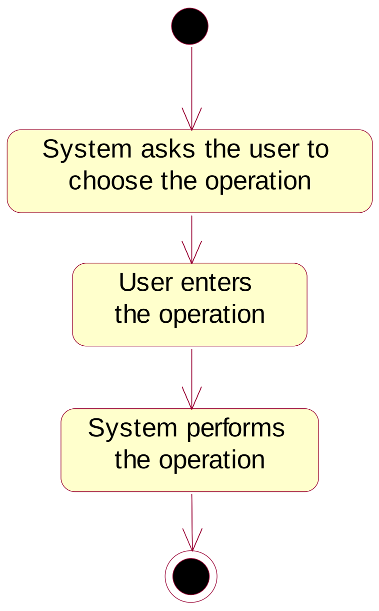
- **Flow Of Events:**

BASIC FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	System generate data analysis which request server administrator to choose the operation.	----------	-----------
2.	Server Administrator	The server administrator chooses the operation.	----------	-----------
3.	Server Administrator	The server administrator chooses the mode view of the data.	----------	----------
4.	Server Administrator	System generates required result of the query it searched for in the log file.	Cannot process data	Alternate Flow
5.	Server Administrator	Use Case Ends.	-----------	------------

ALTERNATE FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	The System displays an error message that the data queried for cannot be processed.	---------	-----------
2.	Server Administrator	The actor can choose to either try again or cancel the operation, at which point the use case ends.	-----------	----------- -


System asks the user to choose the operation
User enters the operation
System performs the operation

- **Precondition:** None**.**
- **Post condition:** None.
- **Actor:** Server Administrator.
- **Special Requirements:** None.

2.3 Use Case: Analyze Bandwidth Usage

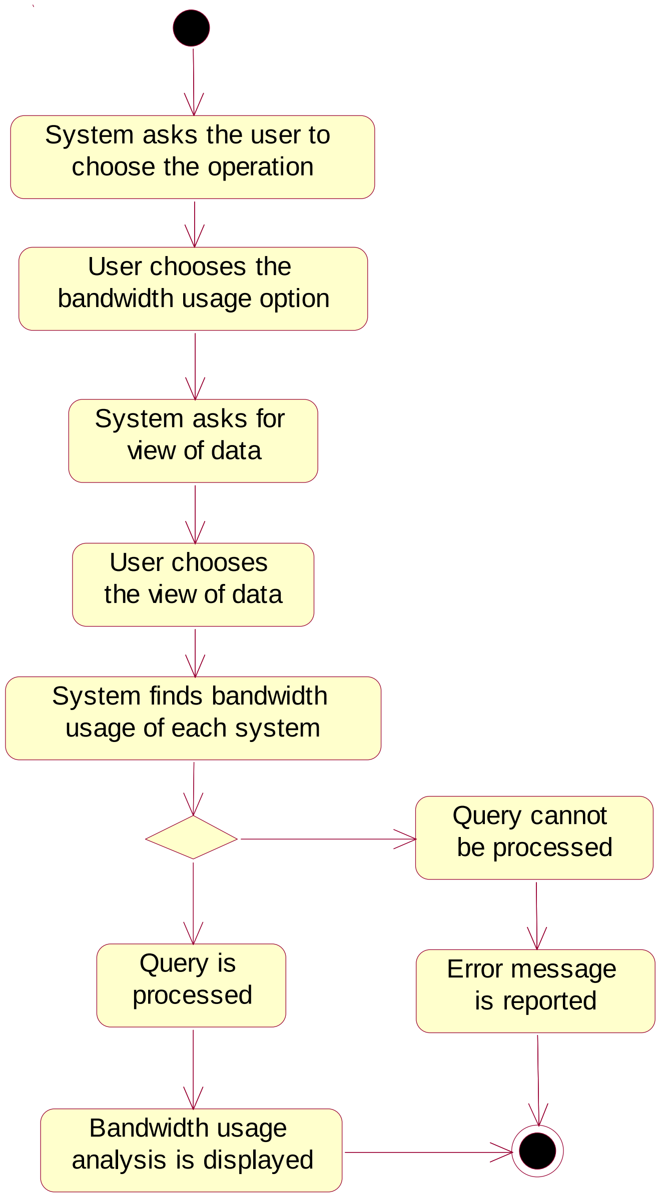
- **Brief Introduction:** The server administrator will analyze the bandwidth usage by each system. He can find bandwidth usage with respect to the IP address.
- **Flow Of Events:**

BASIC FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	System generate data analysis which request server administrator to choose the operation.	----------	-----------
2.	Server Administrator	The server administrator chooses the operation of finding bandwidth usage of each system.	----------	-----------
3.	Server Administrator	The server administrator chooses the mode view of the data.	----------	----------
4.	Server Administrator	System generates required result of the query it searched for in the log file.	Cannot process data	Alternate Flow
5.	Server Administrator	Use Case Ends.	-----------	------------

ALTERNATE FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	The System displays an error message that the data queried for cannot be processed.	---------	-----------
2.	Server Administrator	The actor can choose to either try again or cancel the operation, at which point the use case ends.	-----------	----------- -



System asks the user to choose the operation
User chooses the bandwidth usage option
System asks for view of data
User chooses the view of data
System finds bandwidth usage of each system
Query cannot be processed
Query is processed
Error message is reported
Bandwidth usage analysis is displayed

- **Precondition:** None**.**
- **Post condition:** If the use case was successful, the bandwidth usage of each system is shown.
- **Actor:** Server Administrator.
- **Special Requirements:** None.

2.4 Use Case: Analyze Processing Time

- **Brief Introduction:** The server administrator will analyze the processing time of data packets of each system.

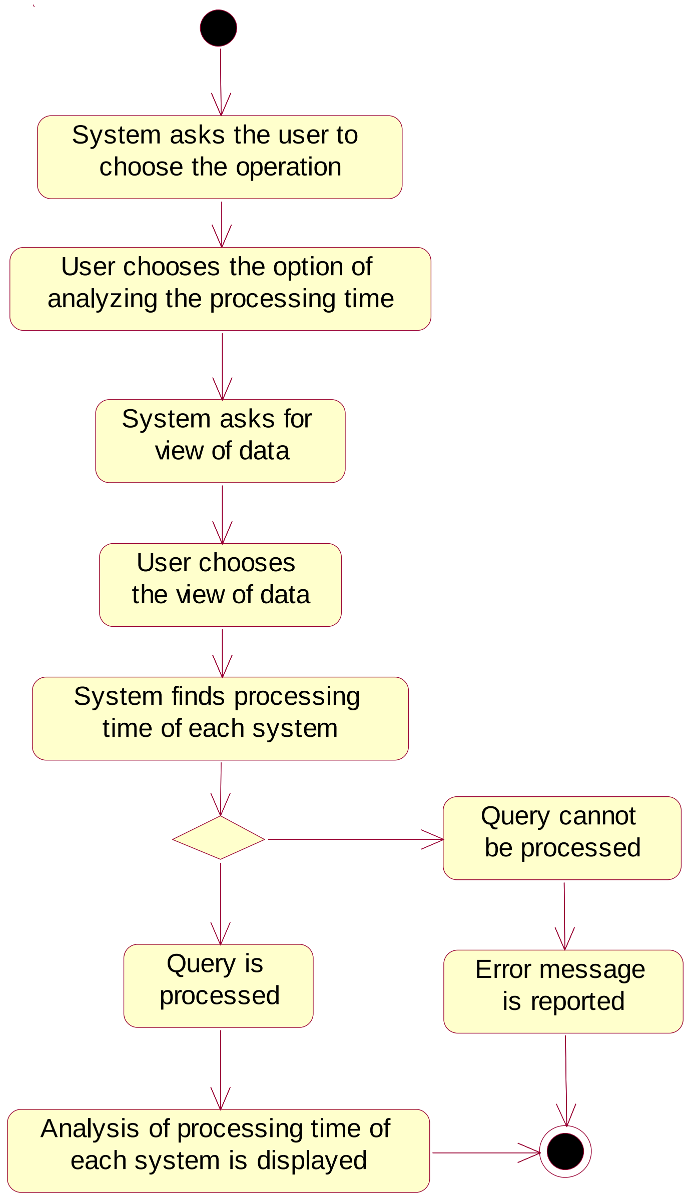
- **Flow Of Events:**

BASIC FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	System generate data analysis which request server administrator to choose the operation.	----------	-----------
2.	Server Administrator	The server administrator chooses the operation of finding processing time of each system.	----------	-----------
3.	Server Administrator	The server administrator chooses the mode view of the data.	----------	----------
4.	Server Administrator	System generates required result of the query it searched for in the log file.	Cannot process data	Alternate Flow
5.	Server Administrator	Use Case Ends.	-----------	------------

ALTERNATE FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	The System displays an error message that the data queried for cannot be processed.	---------	-----------
2.	Server Administrator	The actor can choose to either try again or cancel the operation, at which point the use case ends.	-----------	----------- -


System asks the user to choose the operation
User chooses the option of analyzing the processing time
System asks for view of data
User chooses the view of data
System finds processing time of each system
Query cannot be processed
Query is processed
Error message is reported
Analysis of processing time of each system is displayed

- **Precondition:** None**.**
- **Post condition:** If the use case was successful, the processing time of data packets for each system is shown.
- **Actor:** Server Administrator.
- **Special Requirements:** None.

2.5 Use Case: Analyze User Traffic

- **Brief Introduction:** The server administrator will analyze the user traffic for each system.

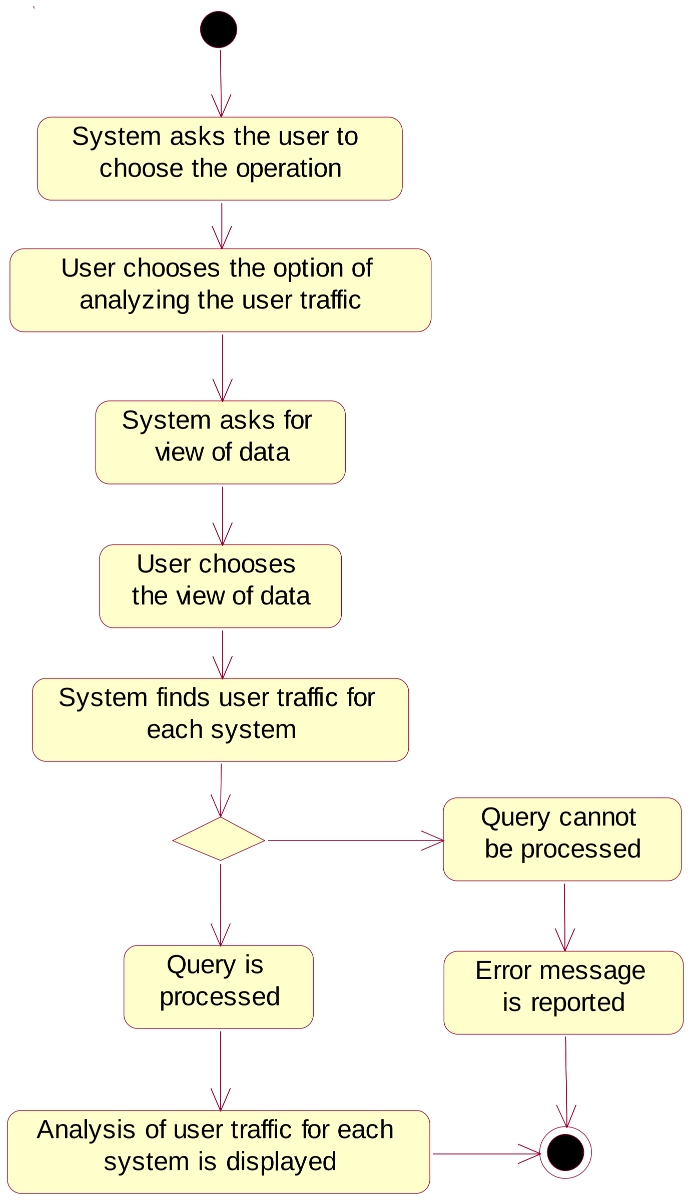
- **Flow Of Events:**

BASIC FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	System generate data analysis which request server administrator to choose the operation.	----------	-----------
2.	Server Administrator	The server administrator chooses the operation of finding user traffic of each system.	----------	-----------
3.	Server Administrator	The server administrator chooses the mode view of the data.	----------	----------
4.	Server Administrator	System generates required result of the query it searched for in the log file.	Cannot process data	Alternate Flow
5.	Server Administrator	Use Case Ends.	-----------	------------

ALTERNATE FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	The System displays an error message that the data queried for cannot be processed.	---------	-----------
2.	Server Administrator	The actor can choose to either try again or cancel the operation, at which point the use case ends.	-----------	----------- -


System asks the user to choose the operation
User chooses the option of analyzing the user traffic
System asks for view of data
User chooses the view of data
System finds user traffic for each system
Query cannot be processed
Query is processed
Error message is reported
Analysis of user traffic for each system is displayed

- **Precondition:** None**.**
- **Post condition:** If the use case was successful, the user traffic for each system is shown.
- **Actor:** Server Administrator.
- **Special Requirements:** None.

2.6 Use Case: Analyze by domain

- **Brief Introduction:** The server administrator will do the domain-wise analysis of each system.

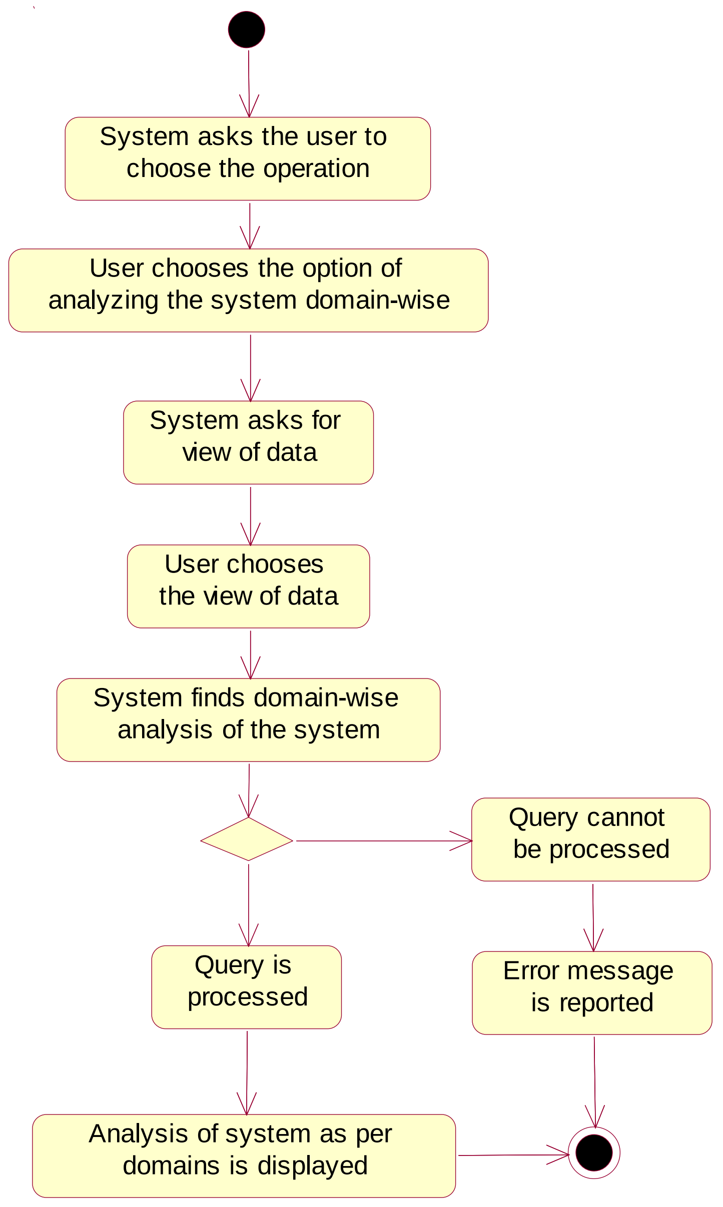
- **Flow Of Events:**

BASIC FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	System generate data analysis which request server administrator to choose the operation.	----------	-----------
2.	Server Administrator	The server administrator chooses the operation of domain-wise analysis of each system.	----------	-----------
3.	Server Administrator	The server administrator chooses the mode view of the data.	----------	----------
4.	Server Administrator	System generates required result of the query it searched for in the log file.	Cannot process data	Alternate Flow
5.	Server Administrator	Use Case Ends.	-----------	------------

ALTERNATE FLOW:

Step	Actor	Description	Condition	Location
1.	Server Administrator	The System displays an error message that the data queried for cannot be processed.	---------	-----------
2.	Server Administrator	The actor can choose to either try again or cancel the operation, at which point the use case ends.	-----------	----------- -

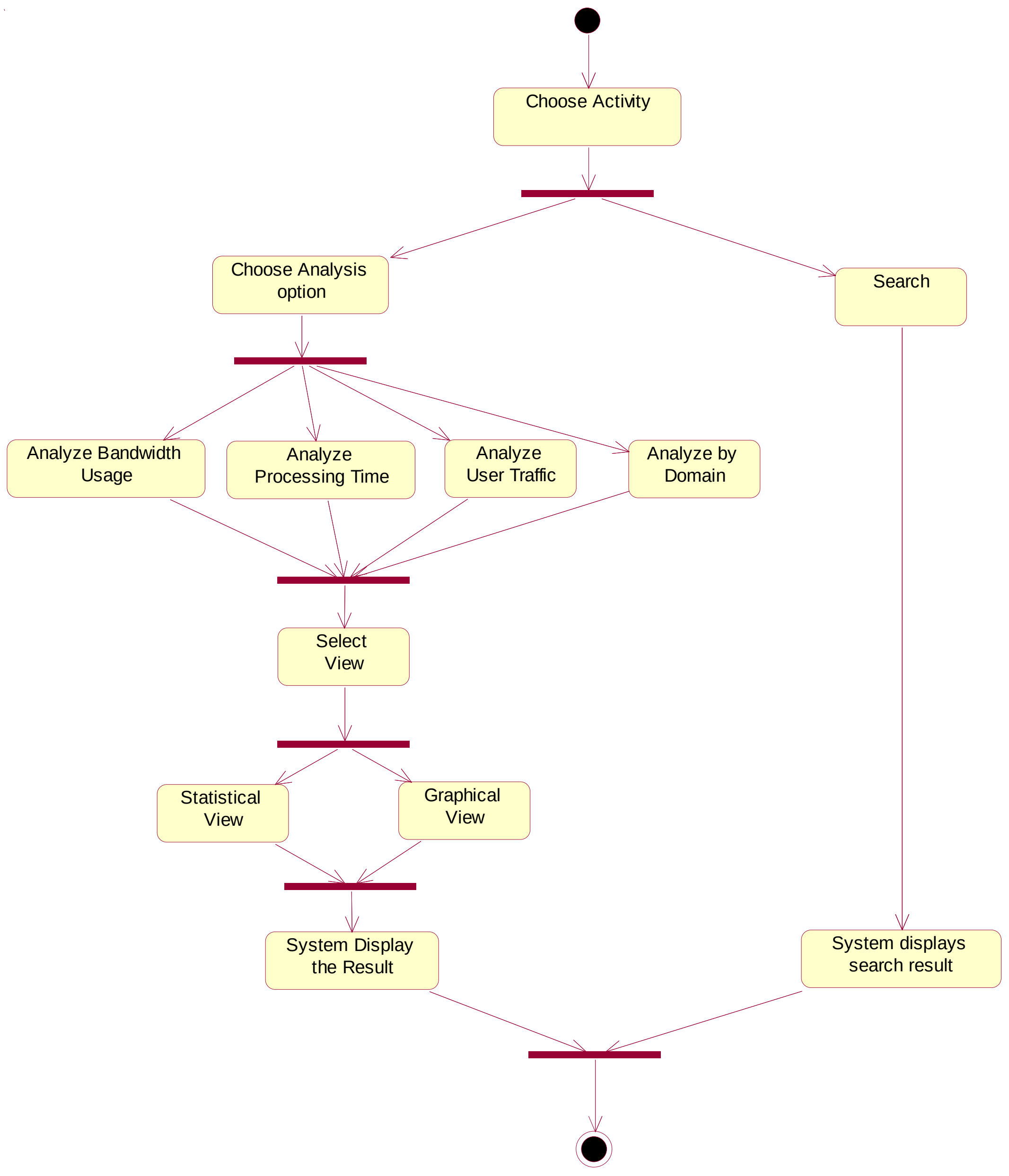

System asks the user to choose the operation
User chooses the option of analyzing the system domain-wise
System asks for view of data
User chooses the view of data
System finds domain-wise analysis of the system
Query cannot be processed
Query is processed
Error message is reported
Analysis of system as per domains is displayed

- **Precondition:** None**.**
- **Post condition:** If the use case was successful, the domain-wise analysis for each system is shown.
- **Actor:** Server Administrator.
- **Special Requirements:** None.

3. Activity Diagram

An Activity Diagram is essentially a flow chart showing flow of control from activity to activity. They are used to model the dynamic aspects of as system. They can also be used to model the flow of an object as it moves from state to state at different points in the flow of control.

Activity diagrams commonly contain Fork Start & End Symbol.



4. Sequence Diagram

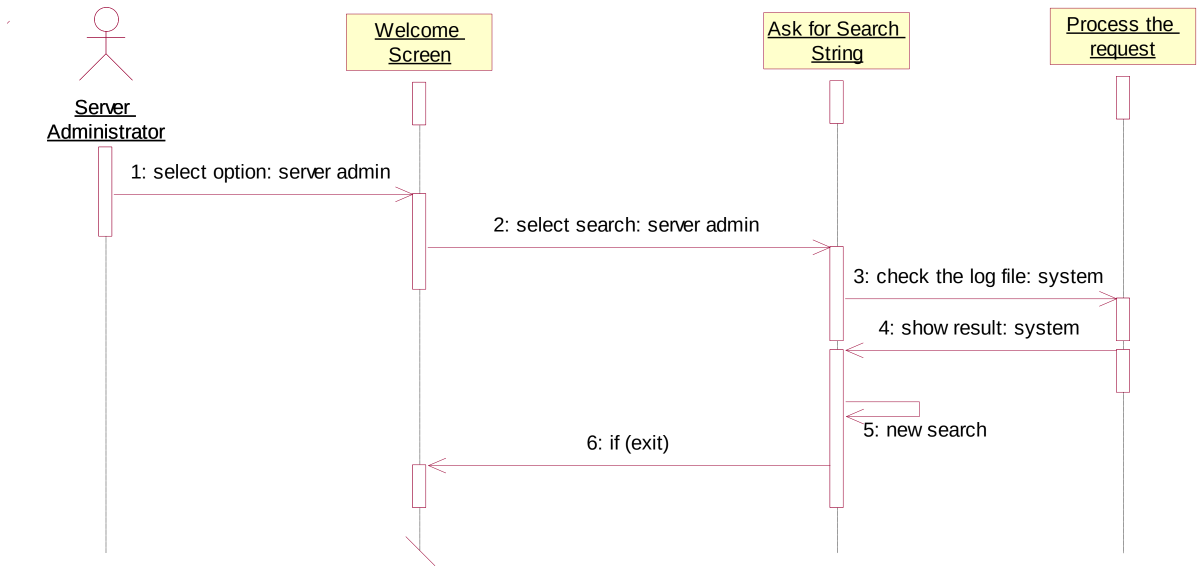
A sequence diagram is an interaction diagram that emphasizes the time ordering of the messages. Graphically, a sequence diagram is a table that shows objects arranged along the X-axis and messages, ordered in increasing time, along the Y axis.

A sequence diagram is a 2-dimensional in nature that depicts the sequence of actions that occur in a system .The invocation of methods in each object and the order in which the invocation occurs is ciphered in a sequence diagram .Thus it easily represents the dynamic behavior of a system .Elements of sequence diagram are:

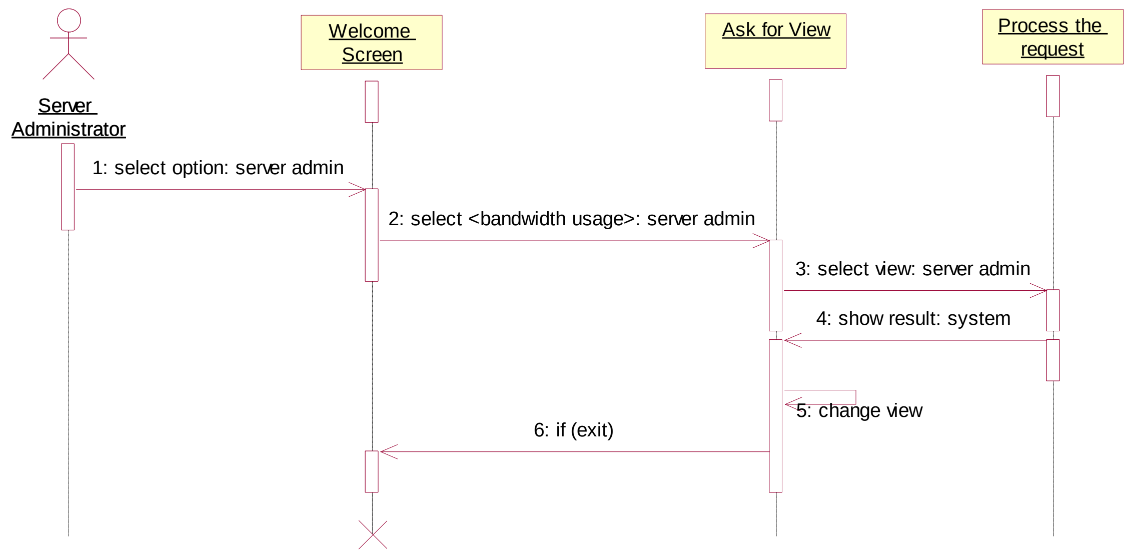
- Object: This is the primary element involved the instance of a class. A sequence diagram consists of sequence of interaction among different objects over a period of time.

- Message: The interaction between different objects in a sequence diagram is represented as message.

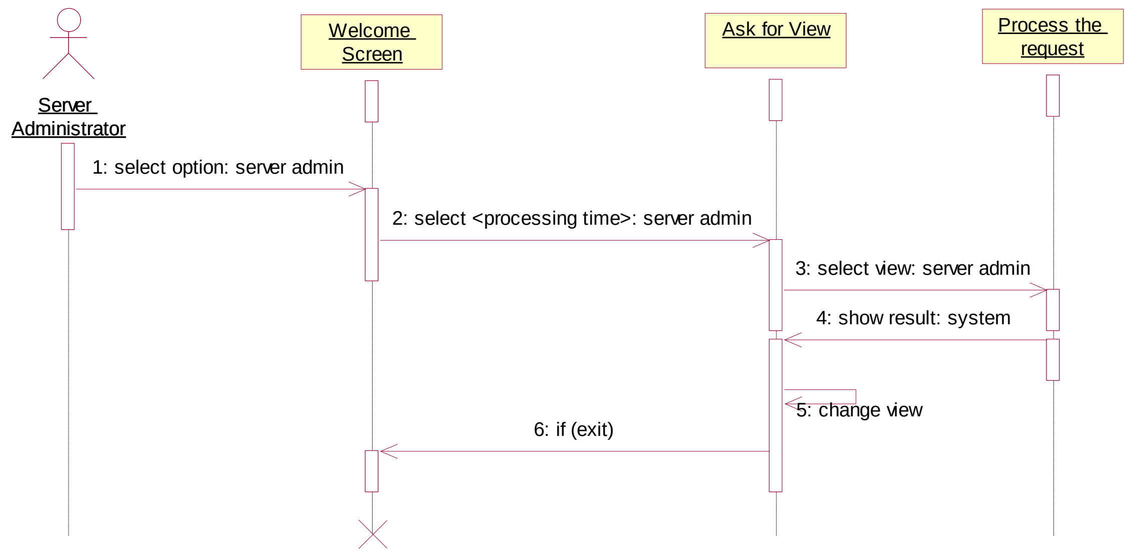
Sequence Diagram for Search:



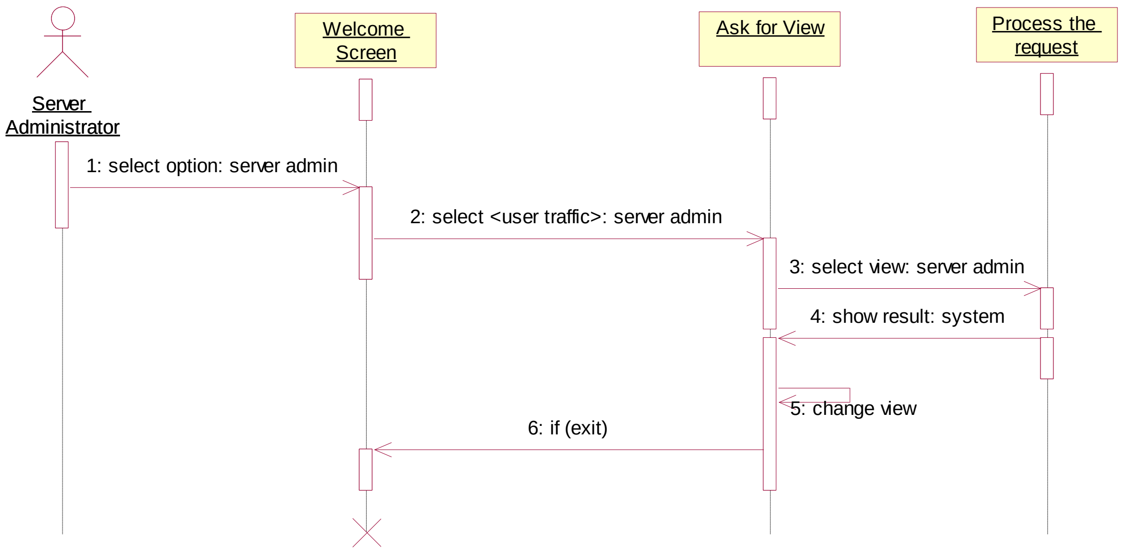
Sequence Diagram for Analysing Bandwidth Usage:



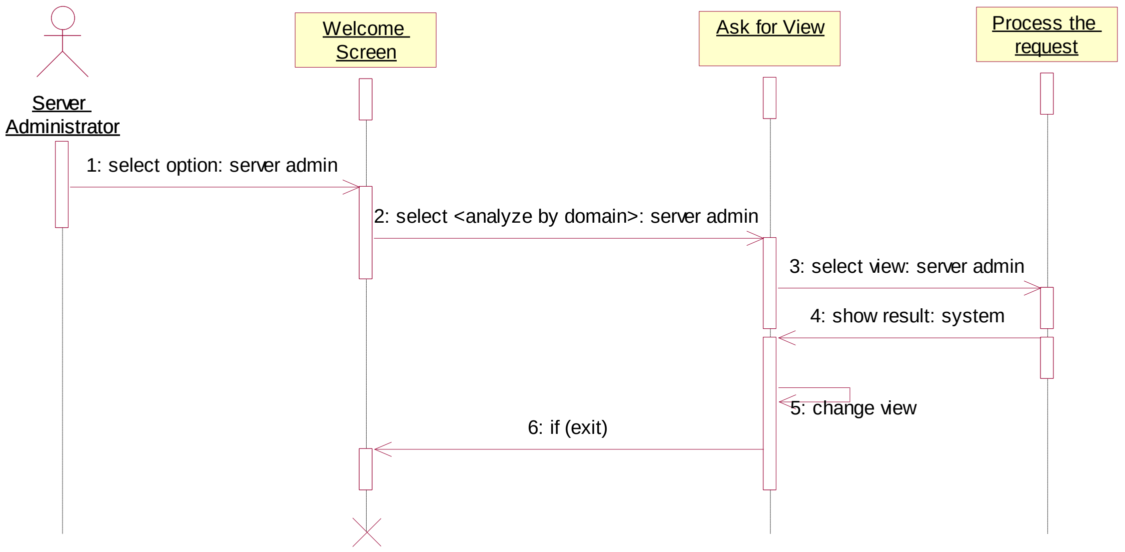
Sequence Diagram for Analysing Processing Time:



Sequence Diagram for Analysing User Traffic:



Sequence Diagram for Analysis by Domain:

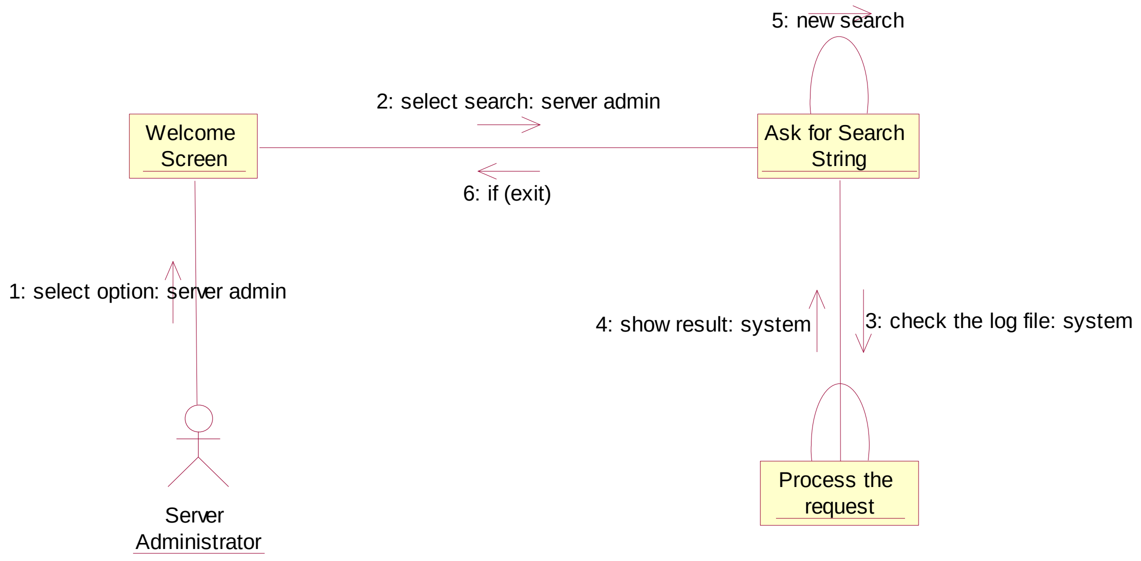


5. Collaboration Diagram

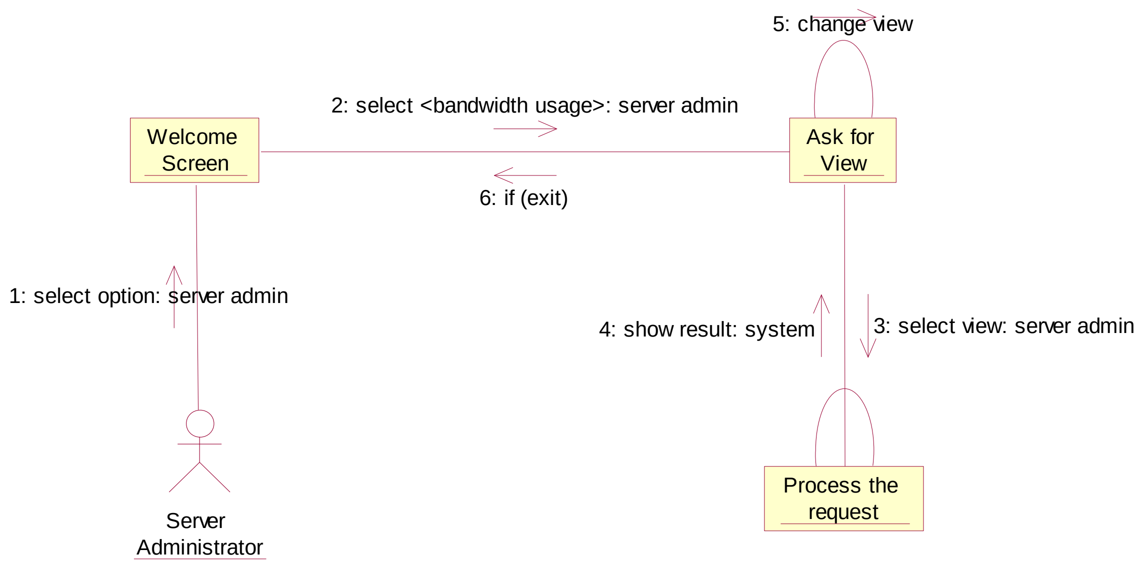
Collaboration Diagrams show the interactions occurring between the objects participating in a specific situation. This is more or less the same information shown by Sequence Diagrams but there the emphasis is put on how the interactions occur in time while the Collaboration Diagrams put the relationships between the objects and their topology in the foreground.

In Collaboration Diagrams messages sent from one object to another are represented by arrows, showing the message name, parameters, and the sequence of the message. Collaboration Diagrams are especially well suited to showing a specific program flow or situation and are one of the best diagram types to quickly demonstrate or explain one process in the program logic.

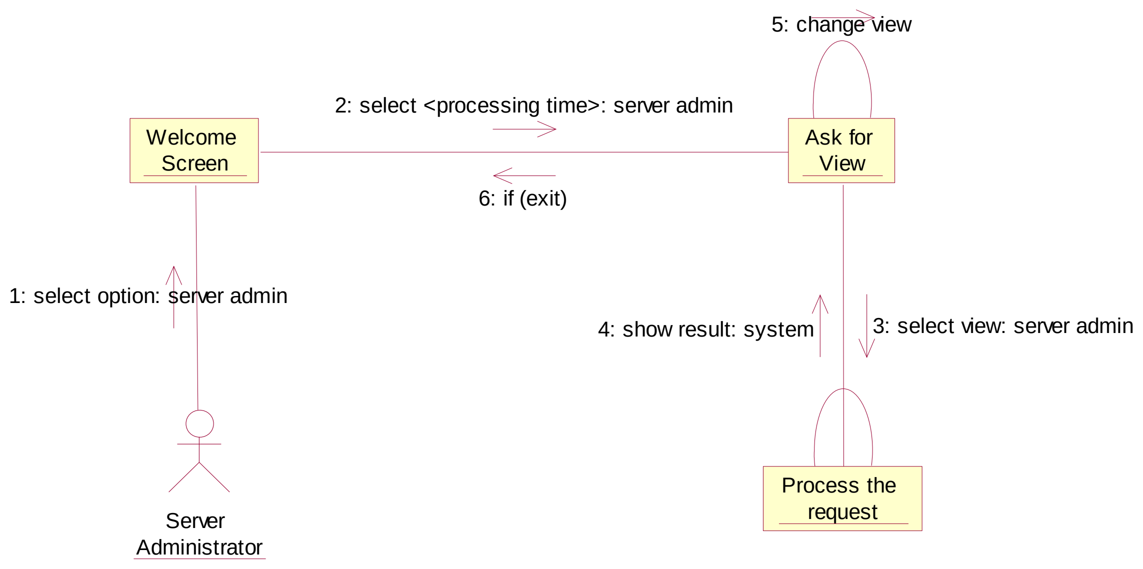
<u>Collaboration Diagram for Search</u>:



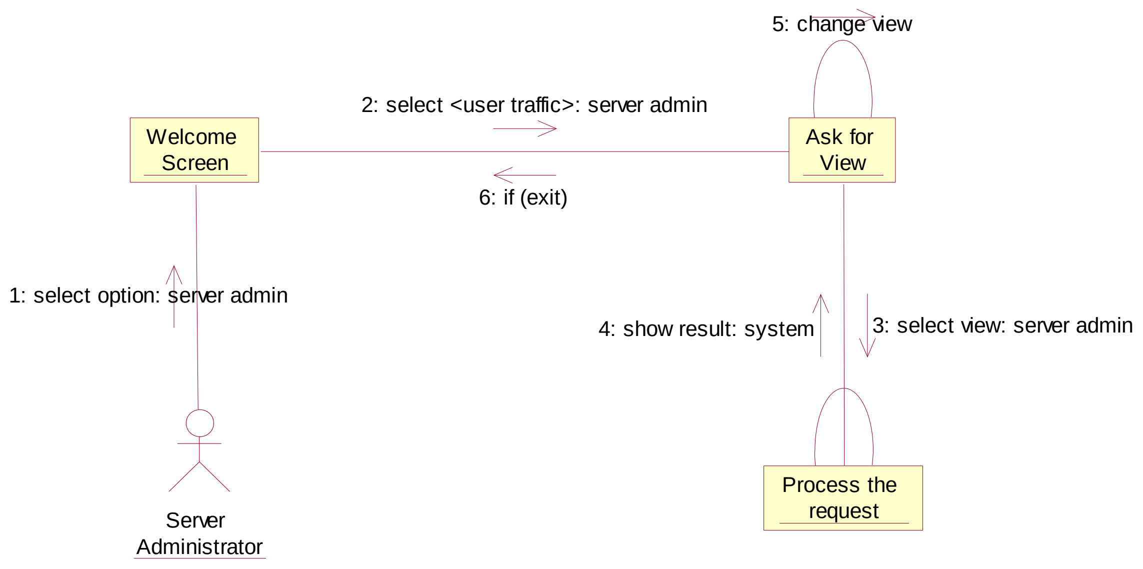
Collaboration Diagram for Analysing Bandwidth Usage:



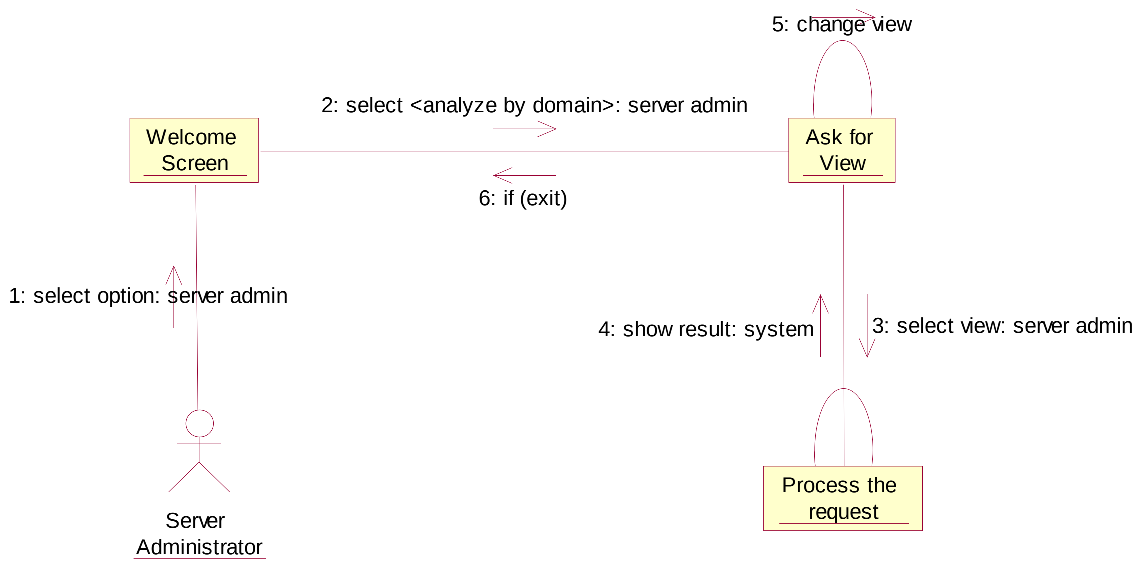
Collaboration Diagram for Analysing Processing Time:



Collaboration Diagram for Analysing User Traffic:



Collaboration Diagram for Analysis by Domain:



6. Statechart Diagram

State diagrams are used to describe the behavior of a system. State diagrams show the dynamic behavior of a system. The diagram shows the various states that an object can get into and the transitions that occur between the states. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system.

It shows the life of an object from birth to death. In this type of diagram, you see the behavior specifying the sequence of states that the object goes through in response to events over its lifetime, and you see the object's responses to those events.

It consists of:

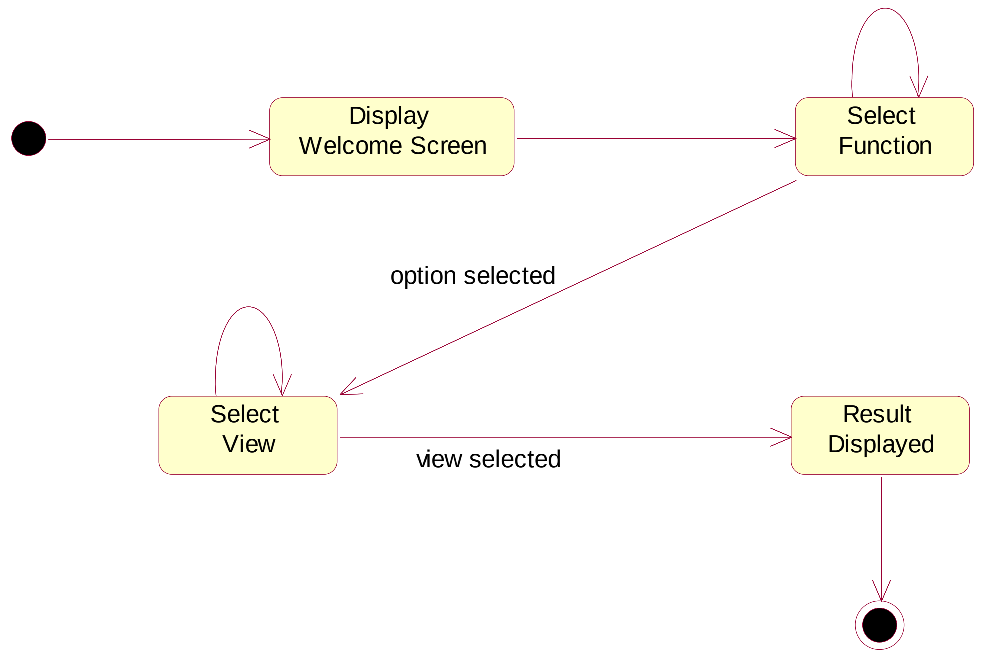
- **State**: The state object is a snapshot of an object at a particular point in its life. A state may have an activity describing the function being performed.

- **Initial State**: The initial state is the starting state of the object with reference to the behavior that the diagram explains. Each state diagram should have only one initial state.

- **Final State**: Each final state is the ending state of the object with reference to the behavior that the diagram explains. There may be multiple final states for an object.

- **Transition**: The transition link represents the relationship between different states of an object. The transition guard is a condition which limits the cases in which a transition can occur. The transition action is performed during the transition and cannot be interrupted.

<u>Statechart Diagram for Server Administrator</u>:



Smart Analyzer

Iteration Plan

Version 1.0

Revision History

Date	Version	Description	Author
30/11/2010	1.0	Iteration Plan of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present the iteration plan for an access log analyzer for proxy server. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the mentors and the developers of the system.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 Definitions, Acronyms and Abbreviations

Term	Description
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.

Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.

IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.
Prototype	A release that is not necessarily subject to change management and configuration control.
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support interchange, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.

Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.

- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.

- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)

- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

This document will contain four sections:

- ***Plan***
- ***Resources***
- ***Use Cases***
- ***Evaluation Criteria***

2. Plan

Software project planning and scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks. It is important to know however that the schedule evolves overtime. During early stage of project planning, a macroscopic schedule is developed.

A number of basic principles guide software project scheduling :

1. **Compartmentalization** - The project must be compartmentalized into a number of manageable activities and tasks. To accomplish compartmentalization, both the product and the process are decomposed.

2. **Interdependency** - The interdependency of each compartmentalized activity or task must be determined. Some tasks must occur in sequence while other can occur in parallel. Some activities can't commence until the work product produced by another is available. Other activity can occur independently.

3. **Time Allocation** - Each task to be scheduled must be allocated some no. of work units. In addition, each task must be assigned a start date and a completion date that are function of interdependency and whether the work will be conducted by the full time or the part time basis.

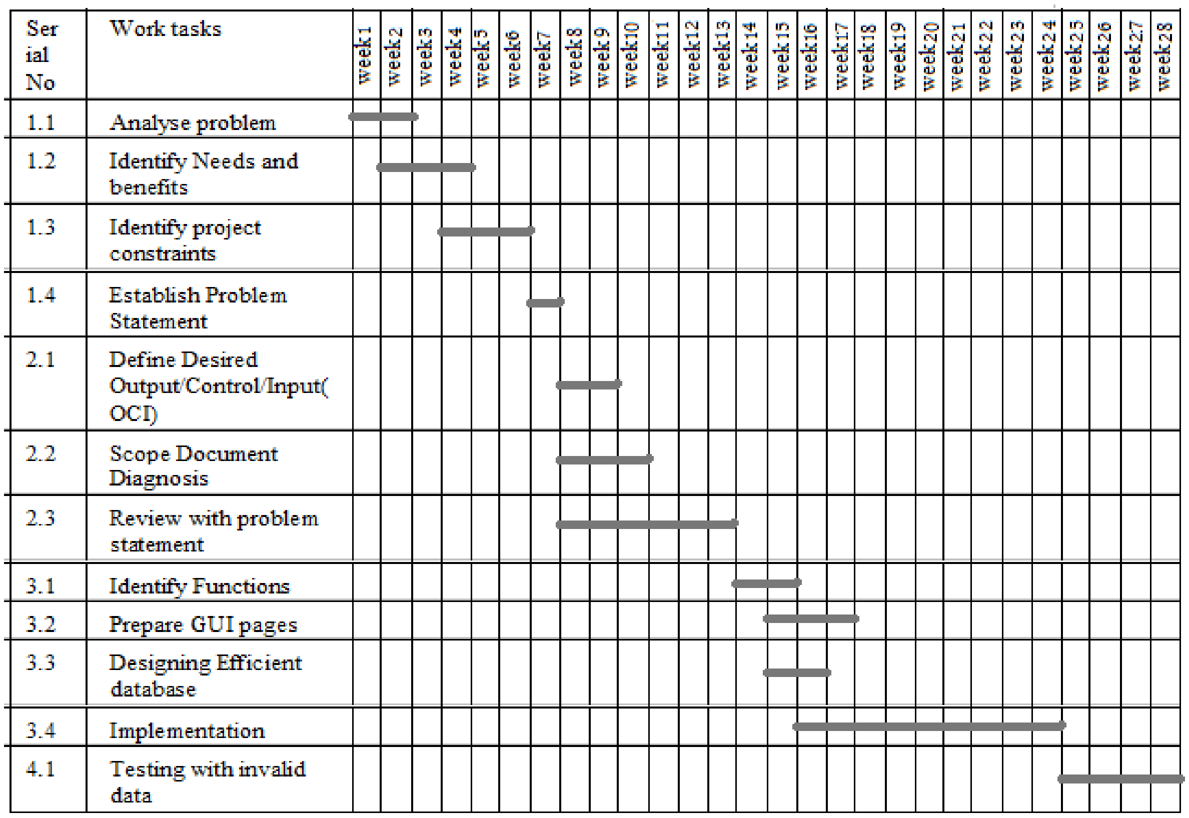
4. **Effort validation** - Each project has a define no. of staff members. As time allocation occurs, a project manager must assure that no more than the allocated no. of people have been schedule at a given time.

5. **Defined responsibility** - Every task that is scheduled should be assigned to a specific team member.

6. **Defined outcomes** - Every task that is schedule should have defined outcomes. For software products the outcome is normally a work product or a part of a work product.

7. **Defined milestones** - Every task or group of task should be associated with a project milestone. A milestone is accomplished when one or more work product has been reviewed for quality and has been approved.

The estimated project plan and schedule keeping the above points in mind is outlined as follows:



Serial No	Work tasks	week1	week2	week3	week4	week5	week6	week7	week8	week9	week10	week11	week12	week13	week14	week15	week16	week17	week18	week19	week20	week21	week22	week23	week24	week25	week26	week27	week28
1.1	Analyse problem	■	■																										
1.2	Identify Needs and benefits		■	■	■																								
1.3	Identify project constraints				■	■	■																						
1.4	Establish Problem Statement							■																					
2.1	Define Desired Output/Control/Input(OCI)								■	■																			
2.2	Scope Document Diagnosis								■	■	■																		
2.3	Review with problem statement								■	■	■	■	■	■															
3.1	Identify Functions														■	■													
3.2	Prepare GUI pages															■	■	■											
3.3	Designing Efficient database															■	■												
3.4	Implementation																■	■	■	■	■	■	■	■	■				
4.1	Testing with invalid data																									■	■	■	■

S. No.	Milestone Date	Milestone	Deliverables
1.	06-Sept,2010 to 09-Oct,2010	Inception: Requirements Signoff	Analysis and requirements specification, Iteration Plan.
2.	11-Oct,2010 to 30-Oct,2010	Elaboration: Iteration-1	Sequence Diagrams, Class Diagrams, Plan for next cycle
3.	01-Nov,2010 to 27-Nov,2010	Elaboration: Iteration-2	Supplementary Specification, Sequence Diagrams, Class Diagrams, Architecture Document, Iteration plan for next cycle.
4.	31-Jan,2011 to 26-Feb,2011	Construction: Iteration-1	Source Code, Review Reports, Test Reports, Iteration for next cycle
5.	28-Feb,2011 to 19-Mar,2011	Construction: Iteration-2	Source Code, Review Reports, Test Reports, Iteration for next cycle
6.	21-Mar,2011 to 16-Apr,2011	Construction: Iteration-3	Source Code, Review Reports, Test Reports, Iteration for next cycle, Deployment Plan for the project.
7.	18-Apr,2011 to 14-May,2011	Integration Testing Phase	Test Reports
8.	16-May,2011 to 28-May,2011	Roll out and Support	Project Sign Off

Other Commitments: This project will follow the Rational Unified Methodology (RUP)

Assumption made while planning: Changes in functional and technical requirements during the life cycle of the project may have an impact on the schedule. Any change in cost or schedule will be intimidated though new version of the document.

3. Resources

3.1 People by Role

Role	Requirement	Phase
Project In-charge	1	Full Time
Project Guide	2	Full Time
Project Developers	4	Full Time

3.2 People by Skill

Area of Skill	Requirement	Phase
Rational Software Architect	1	Elaboration
Java	3	Construction
MySql	2	Construction
Netbeans	1	Construction

3.3 Hardware Resource

Role	Requirement	Phase
PC with 256MB RAM	2	Inception/ Elaboration
1GB space on server	1	Construction
PC with 512MB RAM	5	Construction

3.4 Software Resource

Role	Requirement	Phase
Rational Software Architect	1	Elaboration
Rational Requirement Composer	1	Elaboration
Ubuntu 10.10	4	Construction
MySql	2	Construction

4. Use Cases

Use Case Number	Description	Complexity
Use Case 1	Searches the string entered by user	Simple
Use Case 2	Analyses the data queried by the user	Complex
Use Case 3	Analyses the bandwidth used by each system	Complex
Use Case 4	Analyses the processing time for each system	Medium
Use Case 5	Analyses the User traffic from each IP	Complex
Use Case 6	Analyses the data according to the domain	Complex

Estimation Criteria:

Program/Function (Use Case)	Criteria
Simple Use Case	3 or fewer transactions
Medium Use Case	4 to 7 transactions
Complex Use Case	Greater than 7 transactions

5. Evaluation Criteria

Strategy for meeting Quality Goals:

Strategy	Expected Benefits
Do defect prevention using standard defect prevention guidelines and process: use standards developed in synergy for coding.	10-20% reduction in defect injection rate and about 2% improvement in quality
Group review of program specifications for first few/logically complex use cases.	Improvement in quality as well as defect removal efficiency will improve
Introduction of RUP methodology and implementing the project in iterations. Milestone analysis and defect prevention exercise will be done after each iteration.	Approximately 5% reduction in defect injection rate and about 1% improvement in quality

Smart Analyzer

Software Architecture Document

Version 1.0

Revision History

Date	Version	Description	Author
30/11/2010	1.0	Software Architecture Plan of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present the Software Architecture Document for an access log analyzer for proxy server. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the mentors and the developers of the system.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 Definitions, Acronyms and Abbreviations

Term	Description
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.

Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.

IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.
Prototype	A release that is not necessarily subject to change management and configuration control.
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support interchange, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.

Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.

- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.

- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)

- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

This document will contain four sections:

- ***Architectural Representation***
- ***Architectural Goals and Constraints***
- ***Logical View***
- ***Use Case View***
- ***Deployment View***
- ***Size and Performance***
- ***Quality***

2. Architectural Representation

This document details the architecture using the views defined in the "4+1" model [KRU41], but using the RUP naming convention. The views used to document the Smart Analyzer are:

- **Logical view**

  Audience: Designers.
  Area: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.
  Related Artifacts: Design model

- **Process view**

  Audience: Integrators.
  Area: Non-functional requirements: describes the design's concurrency and synchronization aspects.
  Related Artifacts: (no specific artifact).

- **Implementation view**

  Audience: Programmers.
  Area: Software components: describes the layers and subsystems of the application.
  Related Artifacts: Implementation model, components

- **Deployment view**

  Audience: Deployment managers.
  Area: Topology describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.
  Related Artifacts: Deployment model.

- **Use Case view**

  Audience: all the stakeholders of the system, including the end-users.
  Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail
  Related Artifacts : Use-Case Model, Use-Case documents

- **Data view (optional)**

  Audience: Data specialists, Database administrators
  Area: Persistence describes the architecturally significant persistent elements in the data model
  Related Artifacts: Data model.

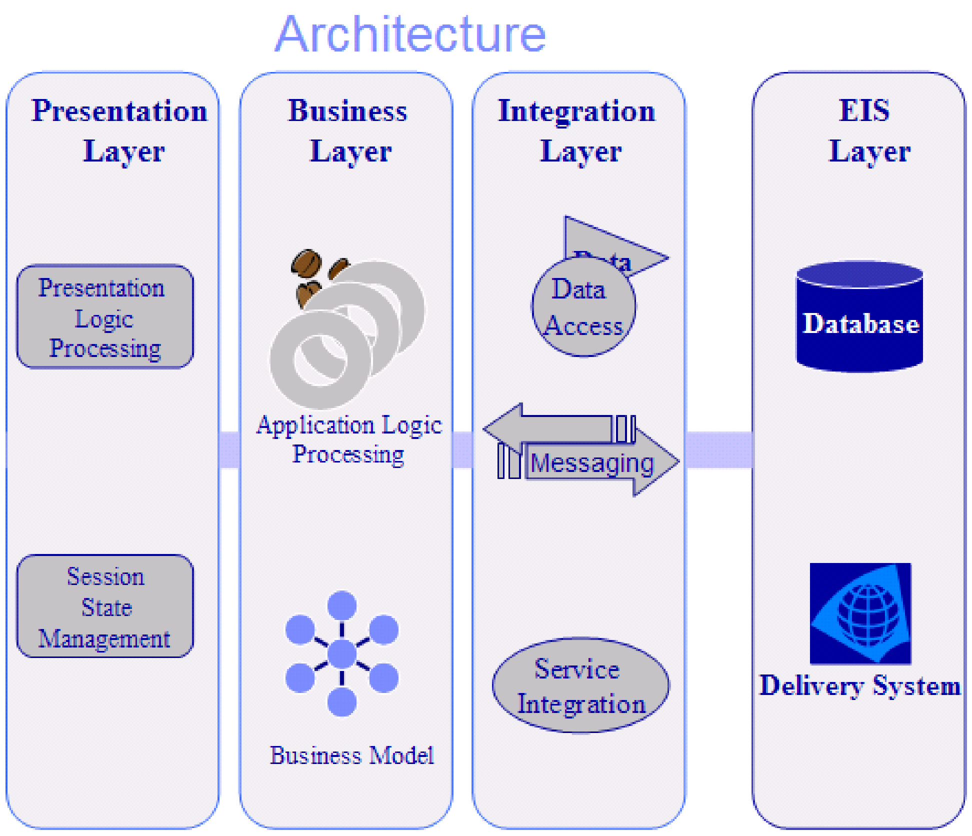
3. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

- The goal of this project is to analyze the request patterns of different web sites on the proxy server.
- It must be designed specifically for single user. The system would be designed for easy to use, providing help instructions, and appropriate error messages for invalid user inputs. It is designed to be used by end users with computer background and would be designed in a user-friendly manner.

4. Logical View

A description of the logical view of the architecture describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.



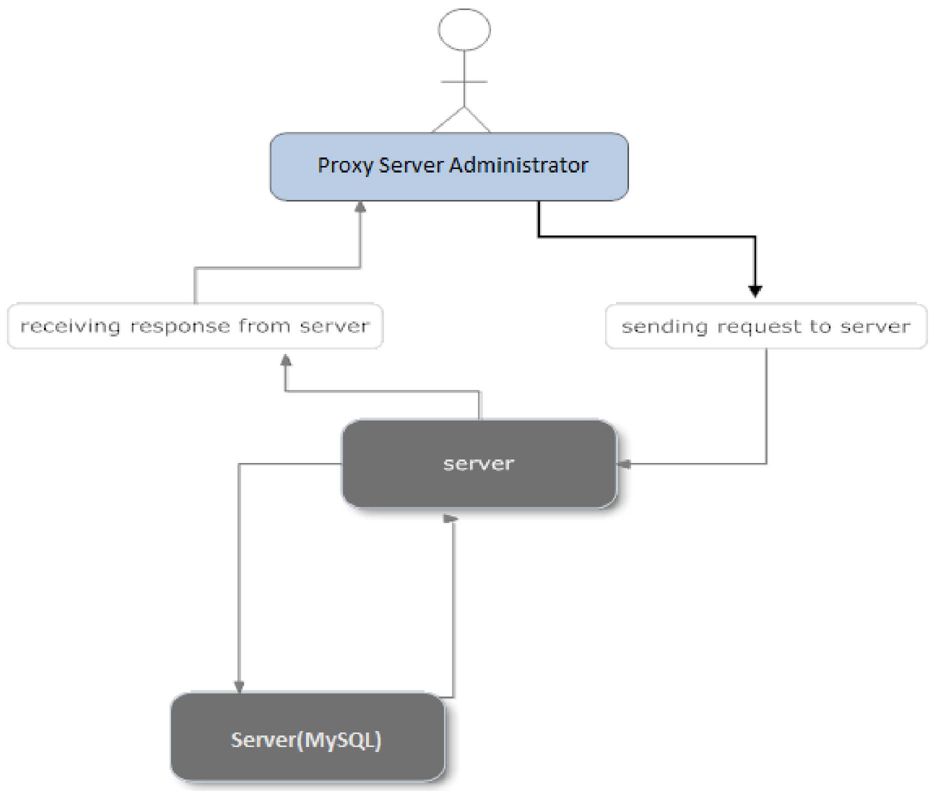
5. Process View

A description of the process view of the architecture describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.

There's only one process to take into account. The program will automatically handle threads which are instances of this process. The diagram below describes the process circles. There are two process circles:

- Proxy server administrator –Server circle
- Server-database circle.

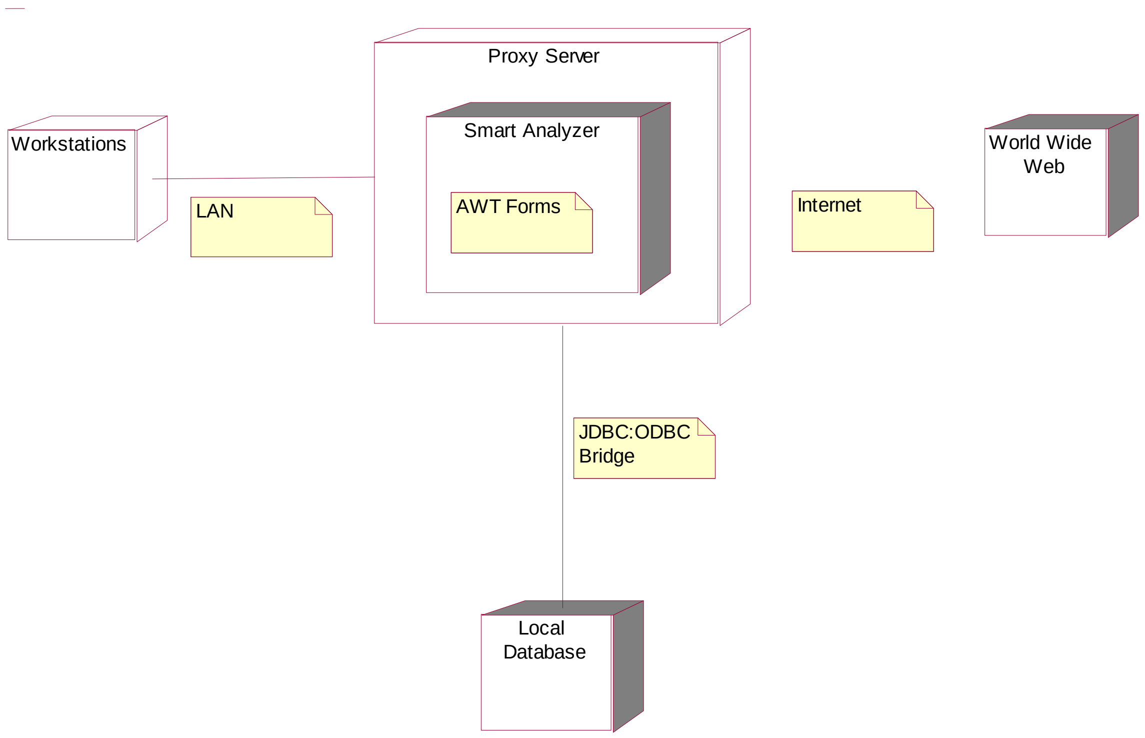
Request message from a Proxy server administrator first will travel to a server. Server first evaluated a request according to the business rules/requirements and determines if a connection to a database needs to be established. If connection is necessary, that is completed first and only then Proxy server administrator is returned with response from a server.


Proxy Server Administrator
receiving response from server
sending request to server
server
Server(MySQL)

6. Deployment View

A description of the deployment view of the architecture describes the various physical nodes for the most typical platform configurations. Also describes the allocation of tasks (from the Process View) to the physical nodes.

The Diagram below shows the proposed deployment view of the smart analyzer with its main components and relation between them

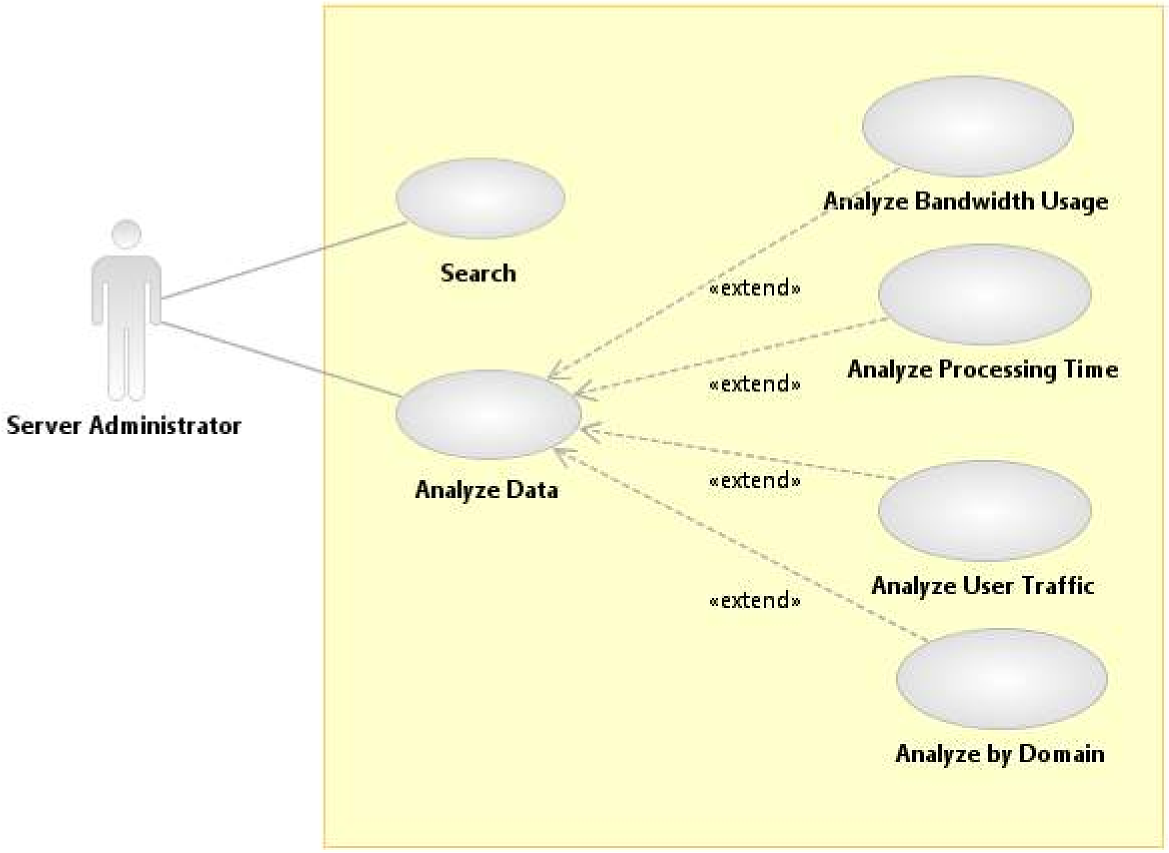


7. Use Case View

Server Administrator: The user of the system responsible for analyzing the data on the proxy server.

- **Search:** The server administrator can search any string in the logs. The administrator will enter the search string and the system will match the string in the available logs. If a match is found it is reported to the user.

- **Analyze Data:** The server administrator will analyze the access logs on the proxy server. He can find various parameters with respect to the IP address.



8. Size and Performance

The software with less loading time and high performance is required for maximum customer satisfaction.

9. Quality

The software architecture supports the following quality requirements:

- The desktop user-interface shall be Linux compliant.
- The user interface of the Smart Analyzer shall be designed for ease-of-use and shall be appropriate for a computer-literate user community with little additional training on the System.
- Each feature of the application shall have built-in help for the user. Help shall include step by step instructions on using the System. Help shall include definitions for terms and acronyms.

Smart Analyzer

Risk List

Version 1.0

Revision History

Date	Version	Description	Author
30/11/2010	1.0	Risk List of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present list of risks involved in developing an access log analyzer for proxy server. The risk list is designed to capture the perceived risks to the success of the project. It identifies, in decreasing order of priority, the events that could lead to a significant negative outcome. It serves as a focal point for project activities and is the basis around which iterations are organized.

1.2 Scope

The problem with for construction of an efficient user interface for analyzing data flow on a proxy server is well known. We are currently developing a system for managing the SQUID proxy server on Linux based operating systems. Till now the analysis of web logs on a server is not easy as most of the entries on the SQUID server log are very hard to be read.

1.3 Definitions, Acronyms and Abbreviations

Term	**Description**
Access Logs	The access logs are a valuable source of information about Squid workloads and performance. The logs record not only access information, but also system configuration errors and resource consumption.
Activity Graph	A special case of a state machine that is used to model processes involving one or more classifiers.
Actor (class)	Defines a set of actor instances, in which each actor instance plays the same role in relation to the system.
Actor (instance)	Someone or something, outside the system that interacts with the system.
Analysis	The part of the software development process whose primary purpose is to formulate a model of the problem domain.
Analyzer	An analyzer is a person or device that analyses given data. It examines in detail the structure of the given data and tries to find patterns and relationships between parts of the data.
API	Application Programming Interface. A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.
Artifact	A physical piece of information that is used or produced by a software development process. An artifact may constitute the implementation of a deployable component.
Cache Log	The Cache Log file contains the debug and error messages that Squid generates.
Change Management	The activity of controlling and tracking changes to artifacts.

Change Request (CR)	A general term for any request from a stakeholder to change an artifact or process. Documented in the Change Request is information on the origin and impact of the current problem, the proposed solution, and its cost.
Class	A description of a set of objects that share the same attributes, operations, methods, relationships, and semantics.
Class Diagram	A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships.
Class Hierarchy	The relationships among classes that share a single inheritance.
Collaboration Diagram	A collaboration diagram describes a pattern of interaction among objects; it shows the objects participating in the interaction by their links to each other and the messagesthey send to each other.
Component	A non-trivial, nearly independent, and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture.
Deployment	A discipline in the software-engineering process, whose purpose is to ensure a successful transition of the developed system to its users.
Deployment Diagram	A diagram that shows the configuration of run-time processing nodes and the components, processes , and objects that live on them. Components represent run-time manifestations of code units.
Deployment Environment	A specific instance of a configuration of hardware and software established for the purpose of installing and running the developed software for its intended use.
Deployment Unit	A set of objects or components that are allocated to a process or a processor as a group. A distribution unit can be represented by a run-time composite or an aggregate.
Deployment View	An architectural view that describes one or several system configurations; the mapping of software components (tasks, modules) to the computing nodes in these configurations.
Design	The part of the software development process whose primary purpose is to decide how the system will be implemented. During design, strategic and tactical decisions are made to meet the required functional and quality requirements of a system.
Design Model	An object model describing the realization of use cases ; serves as an abstraction of the implementation model and its source code.
Design Pattern	A design pattern provides a scheme for refining the subsystems or components of a software system, or the relationships between them. It describes a commonly-recurring structure of communicating components that solves a general design problem within a particular context.
Fault-based Testing	A technique for testing computer software using a test method and test data to demonstrate the absence or existence of a set of pre-defined faults.
Graphical User Interface (GUI)	A type of interface that enables users to communicate with a program by manipulating graphical features, rather than by entering commands.

IP Address	An Internet Protocol address (IP address) is a numerical label that is assigned to any device participating in a computer network that uses the Internet Protocol for communication between its nodes.
Linux	Linux refers to the family of Unix-like computer operating systems using the Linux kernel.
Object Oriented Programming	A programming approach based on the concepts of data abstraction and inheritance. Unlike procedural programming techniques, object-oriented programming concentrates on those data objects that constitute the problem and how they are manipulated, not on how something is accomplished.
Prototype	A release that is not necessarily subject to change management and configuration control.
Proxy Server	A proxy server is a server that acts as an intermediary for requests from clients seeking resources from other servers.
Quality Assurance	All those planned and systematic actions necessary to provide adequate confidence that a product or service will satisfy given requirements for quality.
Scenario	A specific sequence of actions that illustrates behaviours. A scenario may be used to illustrate an interaction or the execution of one or more use-case instances.
Server	A server is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet.
Stakeholder	An individual who is who is materially affected by the outcome of the system.
State Machine	A state machine specifies the behavior of a model element, defining its response to events and the life cycle of the object.
Store Log	The Store Log file covers the objects currently kept on disk or removed ones. As a kind of transaction log it is usually used for debugging purposes.
Squid	Squid is a proxy server and web cache daemon. It has a wide variety of uses, from speeding up a web server by caching repeated requests; to caching web, DNS and other computer network lookups for a group of people sharing network resources; to aiding security by filtering traffic.
Template	A predefined structure for an artifact.
Unicode	A character coding system designed to support interchange, and display of the written texts of the diverse languages of the modern world.
UML	Abbreviation of Unified Modeling Language, a language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
URL	Abbreviation of Uniform Resource Locator, the global address of documents and other resources on the World Wide Web.

Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value to an actor.
Use Case Diagram	A diagram that shows the relationships among actors and use cases within a system.
Use Case Instance	The performance of a sequence of actions being specified in a use case. An instance of a use case.
Use Case Model	A model that describes a system's functional requirements in terms of use cases.
Use Case Package	A use-case package is a collection of use cases, actors, relationships, diagrams, and other packages; it is used to structure the use-case model by dividing it into smaller parts.
Use Case Realization	A use-case realization describes how a particular use case is realized within the design model, in terms of collaborating objects.
Version	A variant of some artifact; later versions of an artifact typically expand on earlier versions.
Workspace	The work area that contains all the code you are currently working on; that is, current editions.

1.4 References

Applicable references are:-

- Design of the Visualized Assistant for the Management of Proxy Server - *Shaowei Feng Jing Zhang Bin Zeng*, ISBN 978-1-4244-8231-3.

- Web object life measurement using Squid Log File - *Khunkitti A, Intraha W.*, ISBN 0-7695-1187-4.

- Mastering UML with Rational Rose by Wendy Boggs Michael Boggs (Sybex Inc.)

- UML Distilled by Martin Fowler and Kendall Scott (Addison Wesley)

1.5 Overview

Supplementary Specifications will contain following sections:

- ***Risks***

2. Risks

2.1 Risk of server overloading

2.1.1 Risk Magnitude/ranking

This risk has magnitude of 6

2.1.2 Description

Load on server may cause decrease in performance or system failure.

2.1.3 Mitigation Strategy

Continuously monitoring the network traffic so that no complications occur.

2.2 Risk of using proxy websites

2.2.1 Risk Magnitude/ranking

This risk has magnitude of 4

2.2.2 Description

If a proxy site is used to open a restricted site, then log of that file cannot be maintained.

2.2.3 Mitigation Strategy

This can be prevented by using deep packet inspection which requires a lot of efforts as it inspects each packet passing through it.

2.3 Risk of using graphs

2.3.1 Risk Magnitude/ranking

This risk has magnitude of 5

2.3.2 Description

The calculations used to generate graphs for log file analysis are complex. Thus the graph generation may be slow. Thus the view generated will not be completely dynamic.

2.3.3 Mitigation Strategy

This can be improved by increasing the server memory.

2.4 Risk of having inexperience of RUP

2.4.1 Risk Magnitude/ranking

This risk has magnitude of 3

2.4.2 Description

The development team is relatively inexperienced with the Rational Unified Process (RUP) and Object Oriented Techniques. This could lead to lower efficiency and poorer product quality.

2.4.3 Mitigation Strategy

This can be minimized by attending training sessions for Object Oriented Development and the Rational Unified Process.

Smart Analyzer

Analysis Classes

Version 1.0

Revision History

Date	Version	Description	Author
30/11/2010	1.0	Analysis Classes of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

There are three kinds of classes in the analysis model (see Figure below):

- Boundary class
- Control class
- Entity class

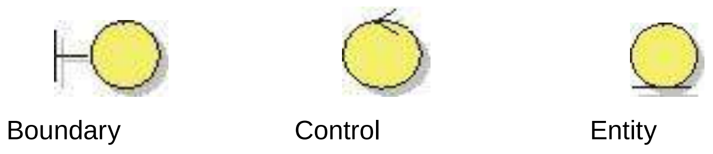


Figure: Analysis model classes

- **Boundary class**

A boundary class is a class used to model interaction between the system's surroundings and its inner workings. Such interaction involves transforming and translating events and noting changes in the system presentation (such as the interface). Common boundary classes include windows, communication protocols, printer interfaces, sensors, and terminals.

Boundary classes model the parts of the system that depend on its surroundings. Entity classes and control classes model the parts that are independent of the system's surroundings.

- **Control class**

A control class is a class used to model control behavior specific to one or a few use cases. It coordinates between the boundary class and an entity class or other components. It contains the logic to invoke appropriate components to complete the path. Control classes can contribute to understanding the system because they represent the dynamics of the system, handling the main tasks and control flows

- **Entity class**

An entity class is a class used to model information and associated behavior that must be stored. Entity objects (instances of entity classes) are used to hold and update information about some phenomenon, such as an event, a person, or some real-life object. They are usually persistent, having attributes and relationships needed for a long period, sometimes for the life of the system.

2. Boundary Classes

- **Proxy Server**
  - Generate data analysis which request server administrator to choose the operation.
  - Generates required result of the query it searched for in the log file.
  - Displays an error message that the data queried for cannot be processed.
  - Asks server administrator to choose the mode view of the data.

3. Control Classes

- **Analyze bandwidth usage:** it is selected for viewing the log as per the bandwidth usage by the users.
- **Analyze processing time:** it is selected for viewing the log as per the processing time by the users.
- **Analyze by domain:** it is selected for viewing the log as per the domain by the users.
- **Analyze user traffic:** it is selected for viewing the log as per the traffic.

4. Entity Classes

- **Change view:** it helps the proxy server administrator to change the view (either textual or graphical) of the log the proxy server would show to him.

Smart Analyzer

Analysis Model

Version 1.0

Revision History

Date	Version	Description	Author
30/11/2010	1.0	Analysis Model of the project	**IT-4th Year, Group-06** Apoorv Shrivastava Minakshi Gupta Pulkit Verma Udayan Gupta

Table of Contents

1. Introduction

An object model describing the realization of use cases, and which serves as an abstraction of the Artifact: Design Model. The Analysis Model contains the results of use case analysis, instances of the Artifact: Analysis Class

The analysis model contains the analysis classes and any associated artifacts. The analysis model may be a temporary artifact, as it is in the case where it evolves into a design model, or it may continue to live on through some or all of the project, and perhaps beyond, serving as a conceptual overview of the system.

The Analysis Model is created in the Elaboration phase, and is updated in the Construction Phase as the structure of the model is updated.

There are three kinds of classes in the analysis model (see Figure below):

- Boundary class
- Control class
- Entity class

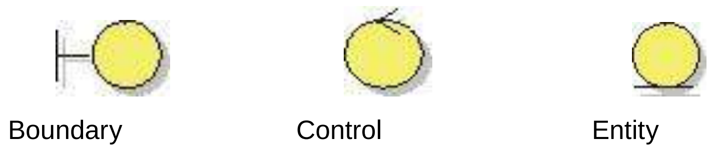


Figure: Analysis model classes

- **Boundary class**

A boundary class is a class used to model interaction between the system's surroundings and its inner workings. Such interaction involves transforming and translating events and noting changes in the system presentation (such as the interface). Common boundary classes include windows, communication protocols, printer interfaces, sensors, and terminals.

Boundary classes model the parts of the system that depend on its surroundings. Entity classes and control classes model the parts that are independent of the system's surroundings.

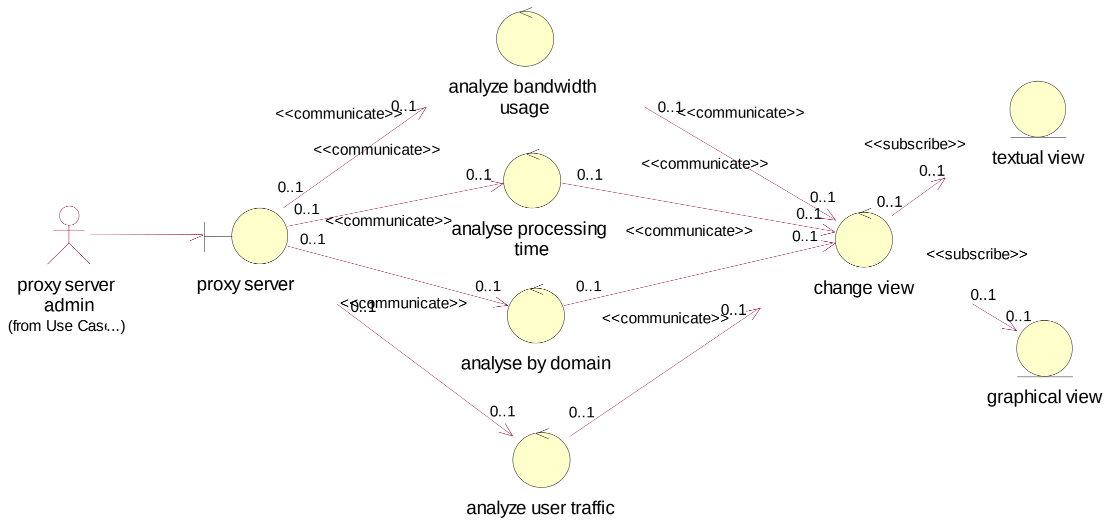
- **Control class**

A control class is a class used to model control behavior specific to one or a few use cases. It coordinates between the boundary class and an entity class or other components. It contains the logic to invoke appropriate components to complete the path. Control classes can contribute to understanding the system because they represent the dynamics of the system, handling the main tasks and control flows

- **Entity class**

An entity class is a class used to model information and associated behavior that must be stored. Entity objects (instances of entity classes) are used to hold and update information about some phenomenon, such as an event, a person, or some real-life object. They are usually persistent, having attributes and relationships needed for a long period, sometimes for the life of the system.

2. Analysis Model


analyze bandwidth usage
analyse processing time
analyse by domain
analyze user traffic
change view
textual view
graphical view
proxy server admin
(from Use Cas...)
proxy server
<<communicate>>
<<subscribe>>
0..1