Initial Design Report

December 4, 2005

SmartGuardian

SmartGuardian™ Software

SmartSoft Network Solutions, Inc.

Tayfun Şen - Özden Meren - Sedat Zelyüt - Hakan Özadam - Akın Öztürk

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to state the design constraints of the computer software called SmartGuardian. It is supposed to be the major guideline for software developers working on this project.

This document contains detailed design description of the software system. Modules of the system and their functional, structural and behavioral aspects are explicitly stated. Corresponding DFD, ER and UML diagrams are also included in the document.

1.2 Problem Definition

There are various issues related with the Internet access of organizations, which can result in big revenue losses. These issues include employees consuming Internet resources contradicting with the company policies and various threats stemming from the Internet itself. That is why employers today want to control the Internet access of their organizations.

Network administrators are incapable of handling problems arising from local area network users since the problems are of high complexity and they should be handled the instant of occurrence.

For these reasons there is an increasing demand for gateway applications which takes care of such issues.

1.3 Goals and Objectives

The Primary goal of SmartGuardian is to control and monitor the internet access of an organization. It is supposed to enforce the rules and policies defined by the network administrator of the organization. It checks the incoming and outgoing traffic for validity and bandwidth usage limits. It displays detailed reports about the internet usage of the organization via web pages.

SmartGuardian is designed to optimize the internet access of the organization by implementing bandwidth rules and policies. It is also supposed to improve the efficiency of the employees by implementing content filtering and by restricting the internet access according to the rules defined by the administrator.

1.4 System Overview

SmartGuardian is an application level gateway. It is going to be developed on Fedora Core 4 Linux operating system. It will be implemented using C++ and PHP. It has a web interface which runs on Apache web server. MySQL is the database of the system, though the system will use several external text files to store some system settings and most of the activity logs.

SmartGuardian is composed of 3 major components: SmartGuardian Wall, SmartGuardian Crawler and SmartGuardian Controller. SmartGuardian Wall and SmartGuardian Crawler will be implemented in C++ and SmartGuardian Controller will be implemented using PHP.

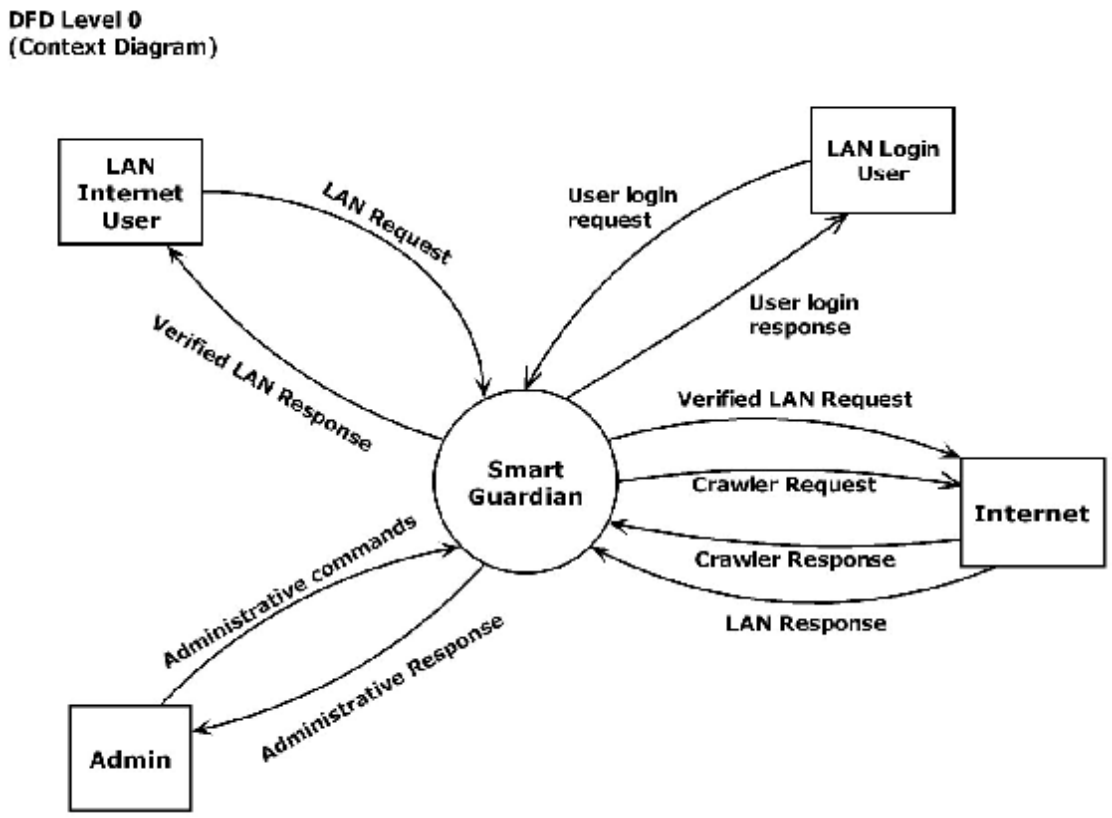

Figure 1.1: DFD Level 0 of the system

1.5 Definitions and Acronyms

DFD: Data Flow Diagram

ER: Entity Relationship

HTTP: Hyper Text Transfer Protocol

IP: Internet Protocol

SG: SmartGuardian software system

UML: Unified Modeling Language

URL: Uniform Resource Locator

2 DESIGN CONSIDERATIONS

2.1 Module Decomposition

The SmartGuardian software system is composed of four main modules: Authenticator, SmartGuardian Wall, SmartGuardian Crawler and SmartGuardian Controller.

System structure is presented in a tree diagram in the next page.

2.1.1 System Structure

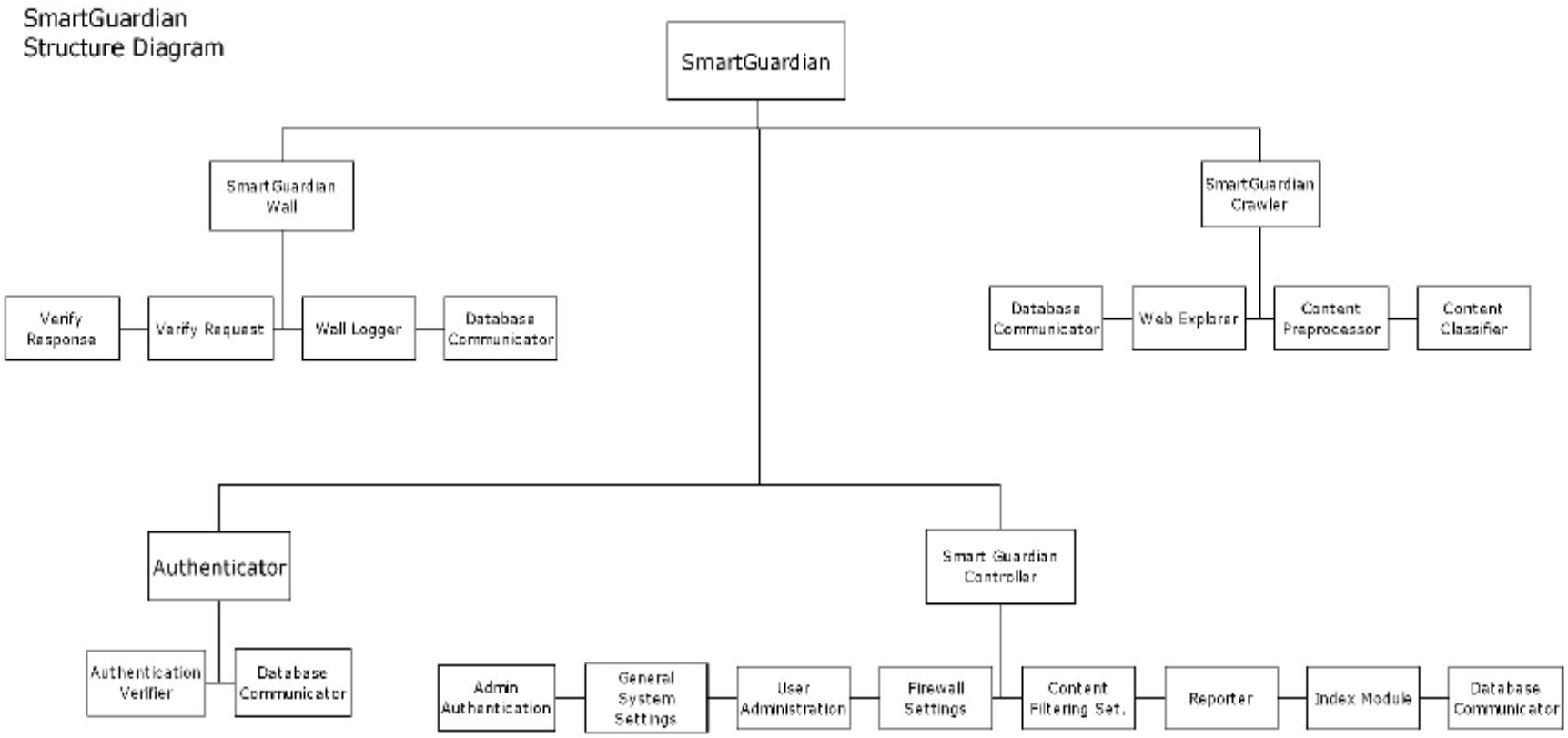

Figure 2.1: Structure Diagram

Level 1 Data flow diagram showing the main modules and their dependencies in the system is shown in the following figure.

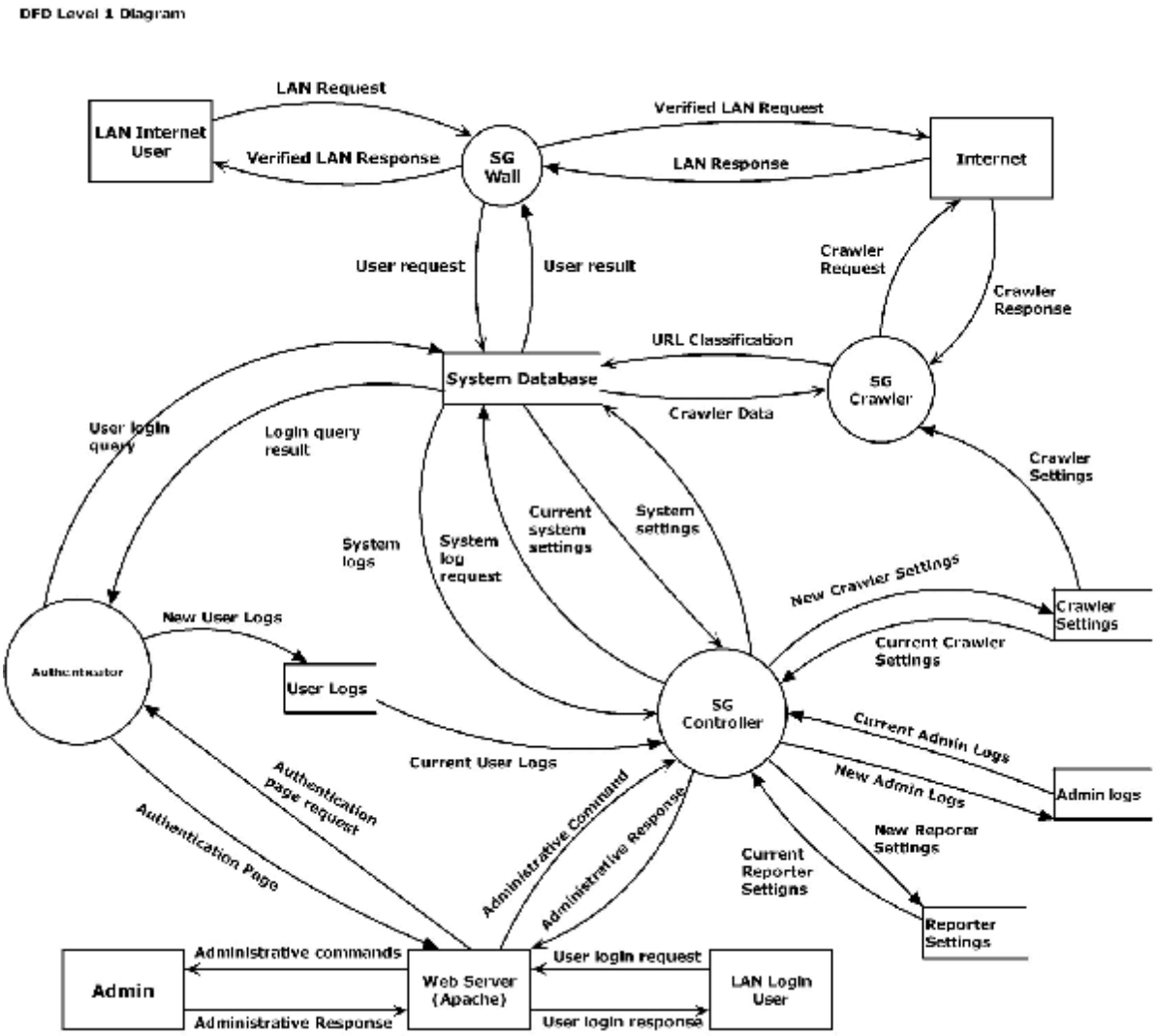

Figure 2.2: DFD Level 1

2.1.2 SmartGuardian Wall

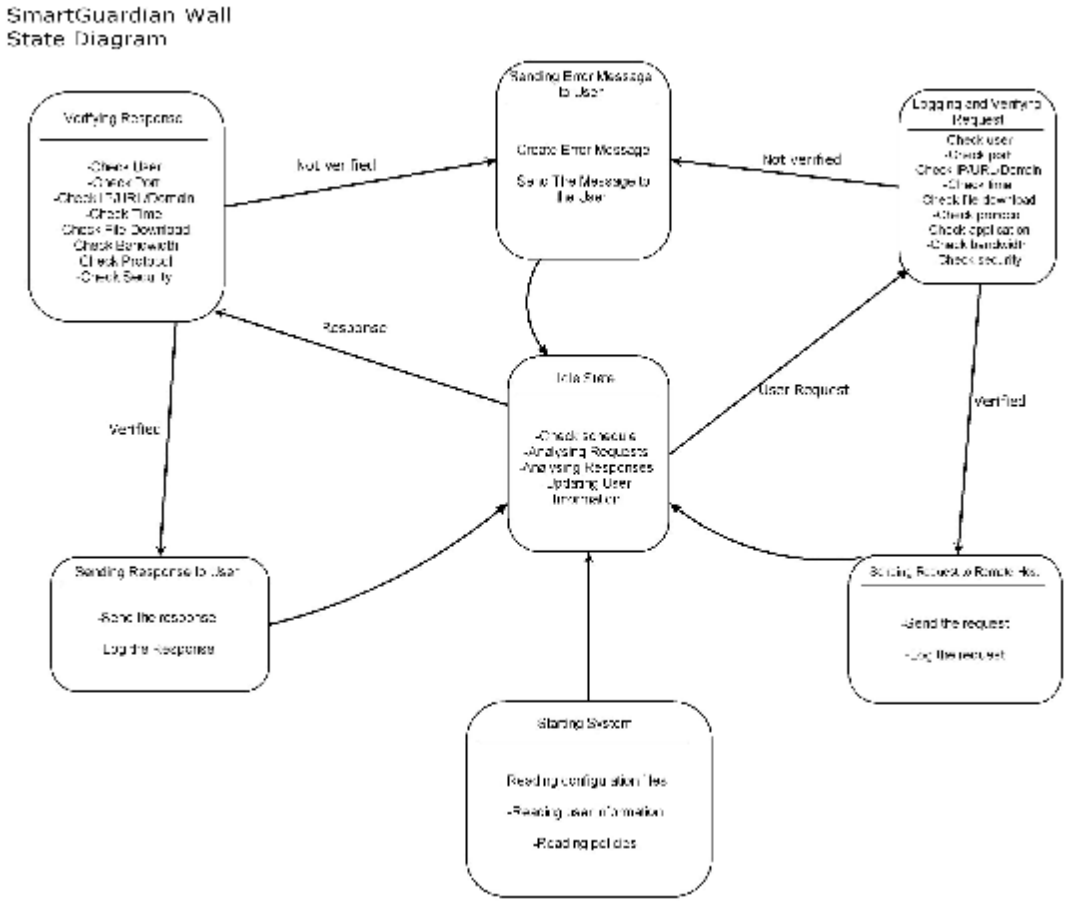

Figure 2.3: State Diagram of SmartGuardian Wall

The state diagram of SmartGuardian Wall is sown in the figure above. There are seven states that the system can be in. When there are no requests or responses the system is in ***idle state***. In this state the system checks the schedule for Web Crawler. If the admin has configured the crawler to start automatically and it is time for the Crawler to start, then the system initiates the crawler. In this state the system updates the user information such as the login status, bandwidth usage and the download limits of the users.

When a request comes the system goes to the ***verifying request*** state. In this state the system checks the status (logged in/ not logged in) of the user, and according to the policy it checks the port to be used and protocol to be used, the domain to be accessed, the time of the request, the file to be downloaded (if this is a file download request), the application that made the request, current bandwidth of the user and finally the security. Depending on the policy and the request, it either verifies the request and serves it or rejects it. If the request is verified the system goes to ***sending request*** state. In this state it sends the request to the remote host and then logs the request. Then it goes to the ***idle state.*** If the request is not verified it goes to ***sending error*** state. In that state it firstly creates an error report stating the reason of the error and then sends the report to the user. Then it logs the request and the error and then goes to the idle state.

Similarly when the system is in ***idle state*** and a response to a request arrives, the system goes to ***Verifying Response*** state. In this state the system checks whether there is a logged in user at the destination IP, and according to the policy it checks the port and protocol of the response, the domain from which the response comes, the time of response, the file to be downloaded (if this is a file download request), current bandwidth of the user and finally the security. Depending on the policy and the response, it either verifies the response and sends it to the user or rejects it. If the response is verified the system goes to ***sending response*** state. In this state it sends the response to the user and then logs the response. Then it goes to the ***idle state.*** If the response is not verified it goes to ***sending error*** state. In that state it firstly creates an error report stating the reason of the error and then sends the report to the user. Then it logs the response and the error and then goes to the idle state.

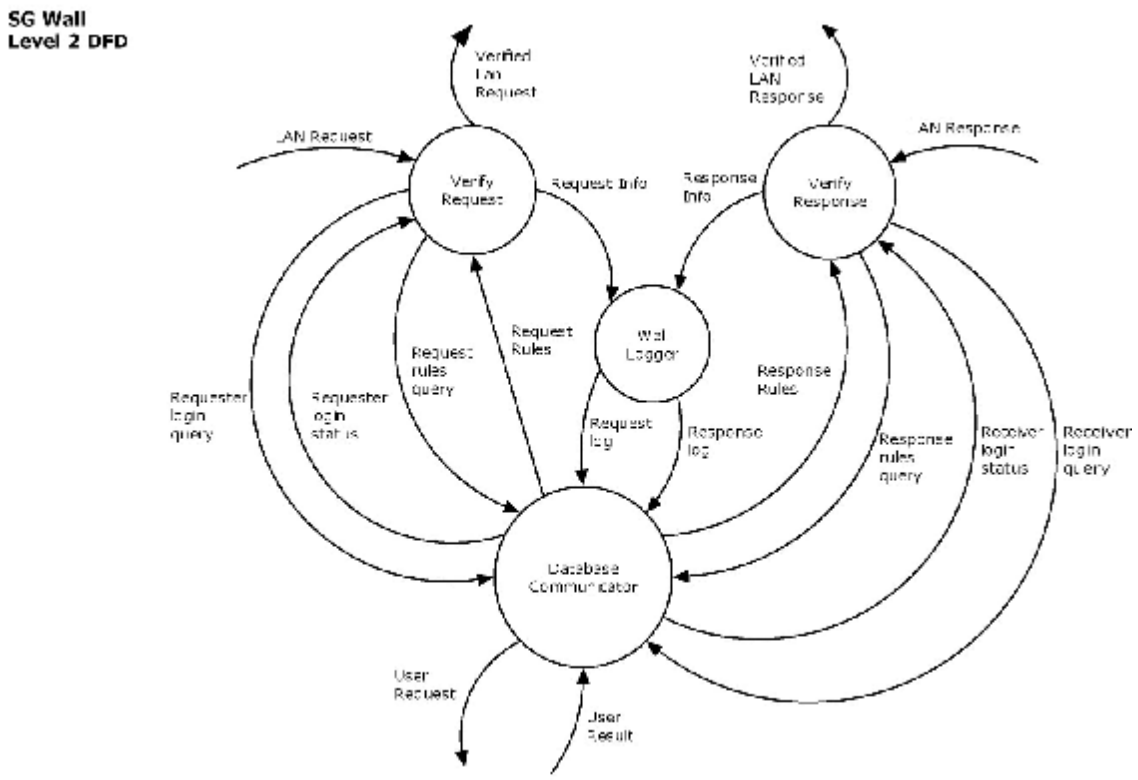

Figure 2.4: Level 2 DFD of SmartGuardian Wall

The main function of SmartGuardian Wall is to control incoming and outgoing traffic. It checks LAN requests (outgoing traffic) & LAN responses (incoming traffic) for validity.

SmartGuardian Wall controls the incoming and outgoing traffic according to the predefined rules and policies. SmartGuardian Wall works in the application layer; it inspects the packets flowing through the network for validity. It has the ability to block selected ports, IPs, URLs and filenames and also it is capable of detecting threats caused by incoming or outgoing traffic and is capable of withstanding these threats.

This module is composed of four sub modules: Verify Request, Verify Response, Wall Logger and Database Communicator.

LAN requests (outgoing traffic) are handled by the sub module Verify Request, LAN responses are handled by the sub module Verify Response, these two sub modules report the result of their actions to Wall Logger which records them for monitoring purposes. All the data flow to the system database is via Database Communicator.

2.1.2.1 Verify Request

Verify Request takes the requests coming from a logged in user, examines the request, in its sub modules layer by layer and decides whether it is a valid request or not. Verification result (it can be passed or failed) is sent to Wall Logger module. Verify Request needs rules in order to make decisions about request. It gets those rules from the system database.

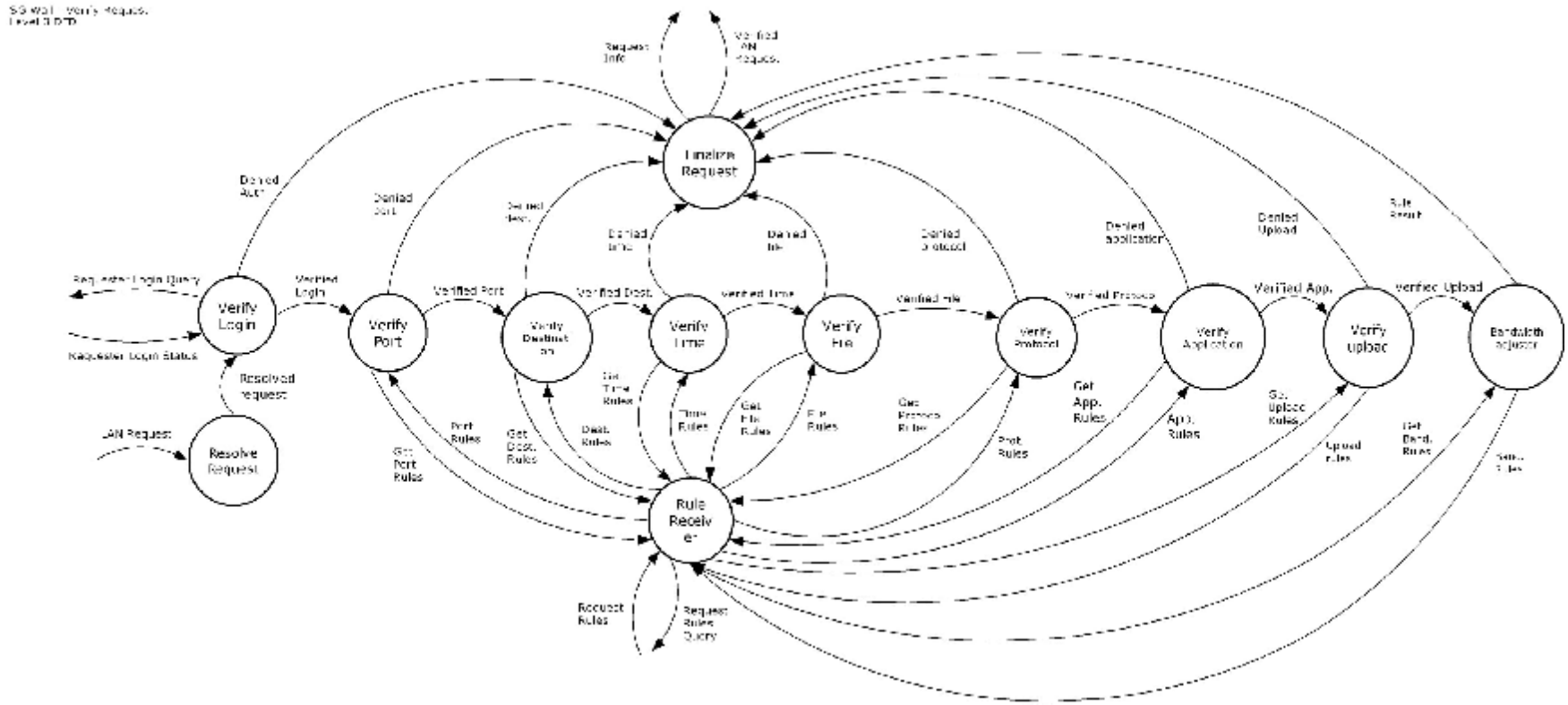

Figure 2.5: Level 3 DFD of Verify Request

2.1.2.1.1 Resolve Request

Incoming requests are resolved before any further processing. This process extracts the following information from the IP datagram which belongs to the request and prepares necessary data structures for the use of other processes verify response. Resolve Request extracts: IHL (internet header length), Total Length, Protocol, Source Address and Destination Address.

2.1.2.1.2 Verify Login

This process checks the source IP of the LAN request and determines if this IP is among the logged in users (users are identified by their IPs). It gets the logged in users information from system database through Database Communicator module. If the IP does not belong to any logged in user, request is ignored. If the IP is idle longer than a max idle time which is determined by the system administrator, corresponding user is marked as logged out and request is ignored. If IP belongs to a logged in user (who has had network activities recently), then this process certifies that request comes from a logged in user and request is further analyzed by Verify Port.

2.1.2.1.3 Verify Port

Verify Port, obtains the allowed and / or blocked ports information from the system through the process Rule Receiver. If user is allowed to send request using that port then port verification is complete, and request is analyzed by next process Verify Destination. If the user is not allowed to use the port then Verify Port reports the situation to Finalize Request and the verification process terminates.

2.1.2.1.4 Verify Destination

Verify Destination, queries if the destination is among the allowed and or blocked address list, if it is, Verify Destination reports the situation to Finalize Request and verification process terminates if not, request analysis continues at Verify Time. The destination address rules are received from Rule Receiver.

2.1.2.1.5 Verify Time

Verify Time gets the rules related to time limitations from Rule Receiver, determines if the user can make such a request taking into account the current server time. If time is verified, request is analyzed further at Verify File.

2.1.2.1.6 Verify File

If the request is a file download request, Verify File checks the request otherwise; Verify File transmits the request to the next process for further inspection. Verify File gets the file download rules from rule receiver. If the file request is allowed, request analysis continues at the next process, if not the situation is reported to Finalize Request and verification terminates.

2.1.2.1.7 Verify Protocol

Verify Port gets the protocol rules from Rule Receiver. It checks if the current request is valid in terms of protocol being used. If the request is valid it is further analyzed at Verify Application, if not it reports the denial of protocol to Finalize Request and verification process terminates.

2.1.2.1.8 Verify Application

Verify Application tries to determine which application the current request belongs. If the application can be determined, it checks the request validity if not request is checked in the next process. Application rules are received from Rule Receiver, if the current application is an allowed one verification continues otherwise it is denied and application is reported to Finalize Request.

2.1.2.1.9 Verify Bandwidth

Verify Bandwidth gets the bandwidth information of the user from Rule Receiver, if the user has exhausted his bandwidth limit, request is denied and denial is reported to finalize request and verification terminates. If bandwidth usage is verified then verified upload is sent to bandwidth adjuster.

2.1.2.1.10 Bandwidth Adjuster

Verified uploads are checked to comply with the real-time bandwidth restrictions. The upload speed is adjusted in order to provide fair bandwidth distribution among the LAN users. Bandwidth adjuster sends the response to the Finalize Request process.

2.1.2.1.11 Finalize Request

Finalize Request is the last process of the verification chain. If there are no denials until this process then request is verified and sent to the internet and is recorded. If Finalize Request receives any denial of rules then request is ignored and result is recorded.

2.1.2.1.12 Rule Receiver

All the process beginning from Verify Port to Verify Bandwidth needs rules in order to determine whether the request is a valid one or not. They receive these rules from Rule Receiver which gets them from system database via Database Communicator.

2.1.2.2 Verify Response

Verify Response is an analog of Verify Request which has minor differences both in verification process and in decision mechanism.

Verify Response examines the incoming traffic for validity. If the response is valid it is sent to corresponding destination if not the response is ignored. In both cases the result is logged.

The data flow diagram of Verify Response is shown on the next page.

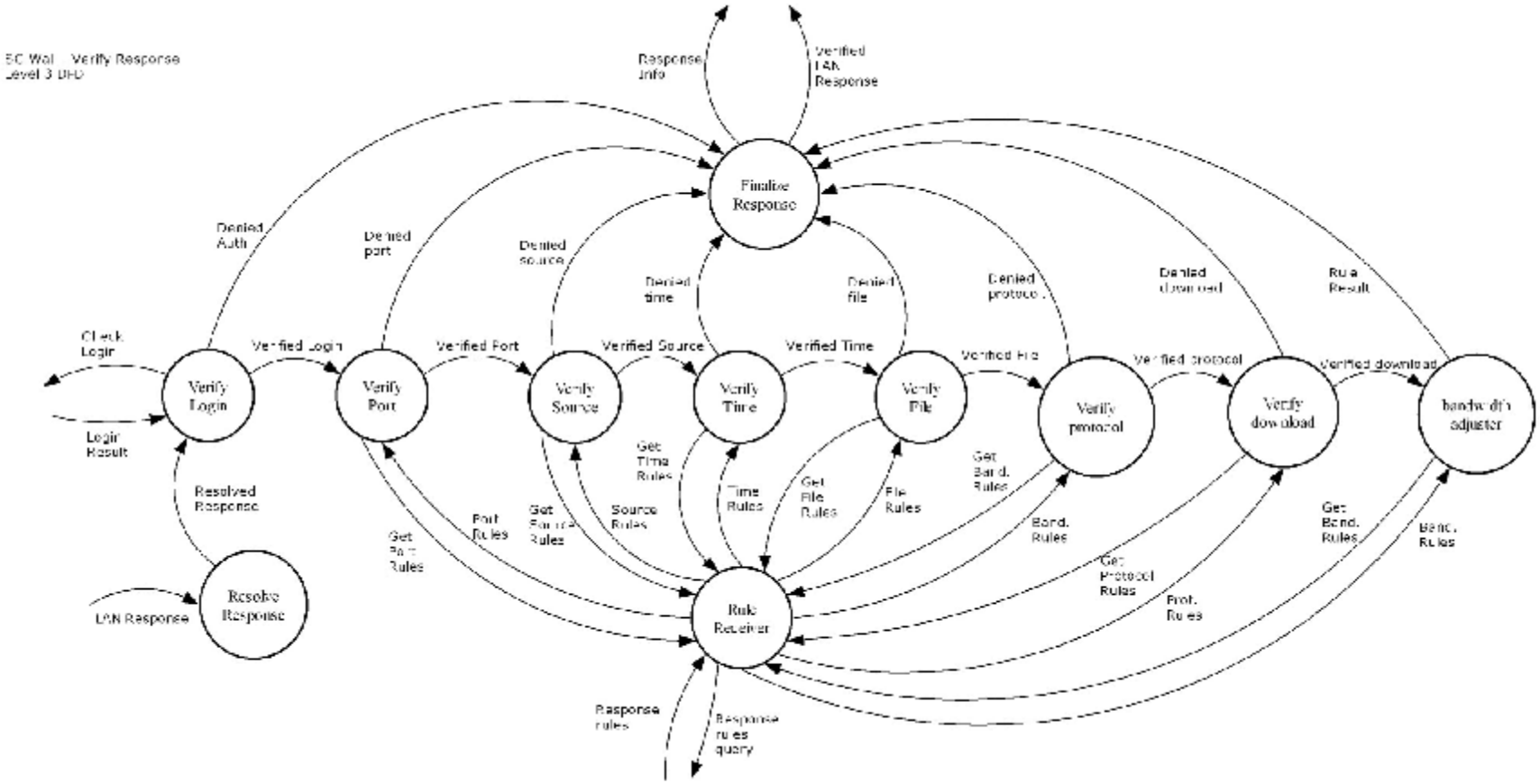

Figure 2.6: Level 3 DFD of Verify Response

Verify Response has the following sub modules:

2.1.2.2.1 Resolve Response

Responses coming from the internet are first resolved; destination address, source address, port, internet header length, total length information are extracted for the use of other processes of Verify Response.

2.1.2.2.2 Verify Login

Verify Login checks whether the destination IP is among the logged in IPs. If it is the response is further analyzed by verify port, if not response verification terminates and situation is reported to Finalize Response.

2.1.2.2.3 Verify Port

Requests whose login is verified are next examined by Verified Port. Verify Port gets the allowed and or blocked port lists from Rule Receiver. It checks whether the port can be used or not. If it can be used, Verify Port hands out the control to verify time if not, verification terminates and situation is reported to Finalize Response.

2.1.2.2.4 Verify Source

Verify Source gets the allowed and or blocked address lists from Rule Receiver. It checks whether the destination address is allowed to receive responses from the source address or not. If it is, response is further analyzed by Verify Time. If not denial of source is reported to Finalize Response and verification ends.

2.1.2.2.5 Verify Time

Verify Time, compares the current server time with the rules regulating the time restrictions. It determines if the destination address is allowed to receive responses at that time. In case of denial it reports the case to Finalize Response.

2.1.2.2.6 Verify File

If the incoming traffic is a file request, Verify File gets the file rules from Rule Receiver, checks if it is a valid request. If it is, response is further analyzed by Verify Bandwidth. If it is not a valid request, result is reported to Finalize Report.

2.1.2.2.7 Verify Protocol

Verify Protocol checks if the protocol used by the response is a valid one or not. Invalid responses are ignored whereas valid ones are sent to Finalize Response.

2.1.2.2.8 Verify Download

This module gets the total bandwidth usage of the destination, checks if the user has exceeded his bandwidth usage, response is blocked. If the user has exceeded his/her instant bandwidth limit the response is blocked again. Result of verification is then sent to Bandwidth Adjuster.

2.1.2.2.9 Bandwidth Adjuster

Verified downloads are checked to comply with the real-time bandwidth restrictions. The download speed is adjusted in order to provide fair bandwidth distribution among the LAN users.

Bandwidth adjuster sends the request to the Finalize Request process.

2.1.2.2.10 Rule Receiver

Rule Receiver gets the information concerning the rules and policies of incoming traffic. It processes this information and extracts rules which will be used by the sub modules of Verify Response.

2.1.2.2.11 Finalize Response

Finalize Response gets the verification result, sends the response to LAN or blocks it according to the result and records it.

2.1.2.3 Wall Logger

Wall Logger keeps records of requests for both incoming and outgoing traffic. It records the verification results coming from Verify Response & Verify Request to the according tables in the system database through Database Communicator module.

2.1.2.4 Classes

The class diagram shown in figure 2.7 describes the basic classes of the SG Wall. Below are the descriptions of the most important ones.

2.1.2.4.1 User Class

In this part of the report the clients in the LAN of the organization which will be using SmartGuardian are referred as users. In other words, the users are the people that are connecting to the Internet through the SmartGuardian. The ***User*** class is designed to keep the information of these users and handle their request.

Since SmartGuardian will give different access rights to different user classes and will keep user based statistical information, the identification of the users is very

important. Therefore before connecting to the internet the users will have to login. By this way, the users will be identified with their username and passwords. The ***username*** and ***password*** properties of the User class are necessary for this purpose. In addition to these a user has properties for personal information, such as name, surname and e-mail address. The ***login*** property indicates whether the user is login or not. The ***gid*** property is the id the of the user group of the user.

Users can access the Internet from different computers. Therefore, when a request arrives, to determine the user making the request we need to keep the IPs of the computers that each user is using. The ***currentIP*** property in ***User*** class is used for this purpose. This value will be set when a user logins. The SGWall has rules that limit the bandwidth and total download amount of the users. The properties ***currentBandwidth*** and ***totalDownload*** keep this information respectively. If a user is does not make a request for a determined time his/her session will be timed out and the user will need to login again. So we will keep this information in the property ***lastRequestTime***. The method ***checkLogin()*** will be used to get the login status of a user when a request arrives. ***checkPassword()*** method will be used when a user is trying to login.

2.1.2.4.2 Policy Class

A policy is a set of rules that determines the action (allow/deny) of SmartGuardian Wall to take when a request or a response to a request arrives and the ***policy*** class keeps the information of a policy. A policy includes seven types of rules, namely application rules, port rules, protocol rules, bandwidth rules, download rules, file download rules and domain/URL rules. An application rule determines which applications can access to the internet and which cannot. The ***alowedApplications*** and ***deniedApplications*** keep this information and ***defApplcAct*** keeps the default action for this rule. Similarly a port rule determines the ports through which a user can connect

to a server and the ones that a user cannot use. The ***allowedPorts*** and ***deniedPorts*** properties of the policy class keep this information and ***defPortAct*** is the action to be taken when a port is neither of the alow and deny lists. Protocol rules are implemented similar to the port rules. ***allowedProtocols***, ***deniedProtocols*** and ***defProtAct*** properties of the ***policy*** class are used for keeping the information necessary for a protocol rule. The downloadLim property keeps the maximum amount of download allowed and bandwidthLim keeps the maximum bandwidth a user can consume. The ***policy*** class contains one ***FileRule*** class and zero or more ***DomainRule*** class.

2.1.2.4.3 FileRule Class

A file rule indicates the files are denied to download. A file with a size greater than the ***maxSize*** property of the ***FileRule*** class cannot be downloaded. Moreover, a file whose name includes one of the words in the ***keywords*** list is not allowed to download. Furthermore, the files having one of the extensions in the ***extensions*** list are also denied.

2.1.2.4.4 DomainRule Class

The DomainRule class is composed a ***DomainClass***, a ***TimeRange*** class and an ***action***. ***Action*** determines whether to allow or deny a request to a domain in the ***DomainClass*** at a time in ***TimeRange***.

2.1.2.4.5 DomainClass Class

The ***domains*** list keeps the domains, URLs and IPs that are in this domain class and the ***isMember()*** method is used to learn whether a given domain is in this class or not.

2.1.2.4.6 Request Class

When a request arrives all the information necessary for deciding whether to allow or deny is kept in this class. This information includes the port and protocol used, the time at which the request is done, the application that made the request, source IP, destination IP, size of the request, the domain to which the request is done and the user that made the request. This class also includes the data, i.e. the content, of the request and the information whether the request is allowed or denied. This class has two methods. One of them is the ***verify()*** method. This method checks whether the request is to be allowed of denied. It firstly checks whether the user is logged in or not. If the user is logged in, then it finds the user group of the user that made the request and gets the corresponding policy. Then using the information and policy it decides to allow or deny the request. The second method is the send() method. As its name implies, it sends the request to the destination. This method is called after the request is validated.

2.1.2.4.7 Response Class

The response class is analogous to the request class. It keeps all the information about the responses to the requests. The kept information is same as the requests information. The only difference is that response class does not have the field called application. Also the request's source IP is response's destination IP and vice versa.

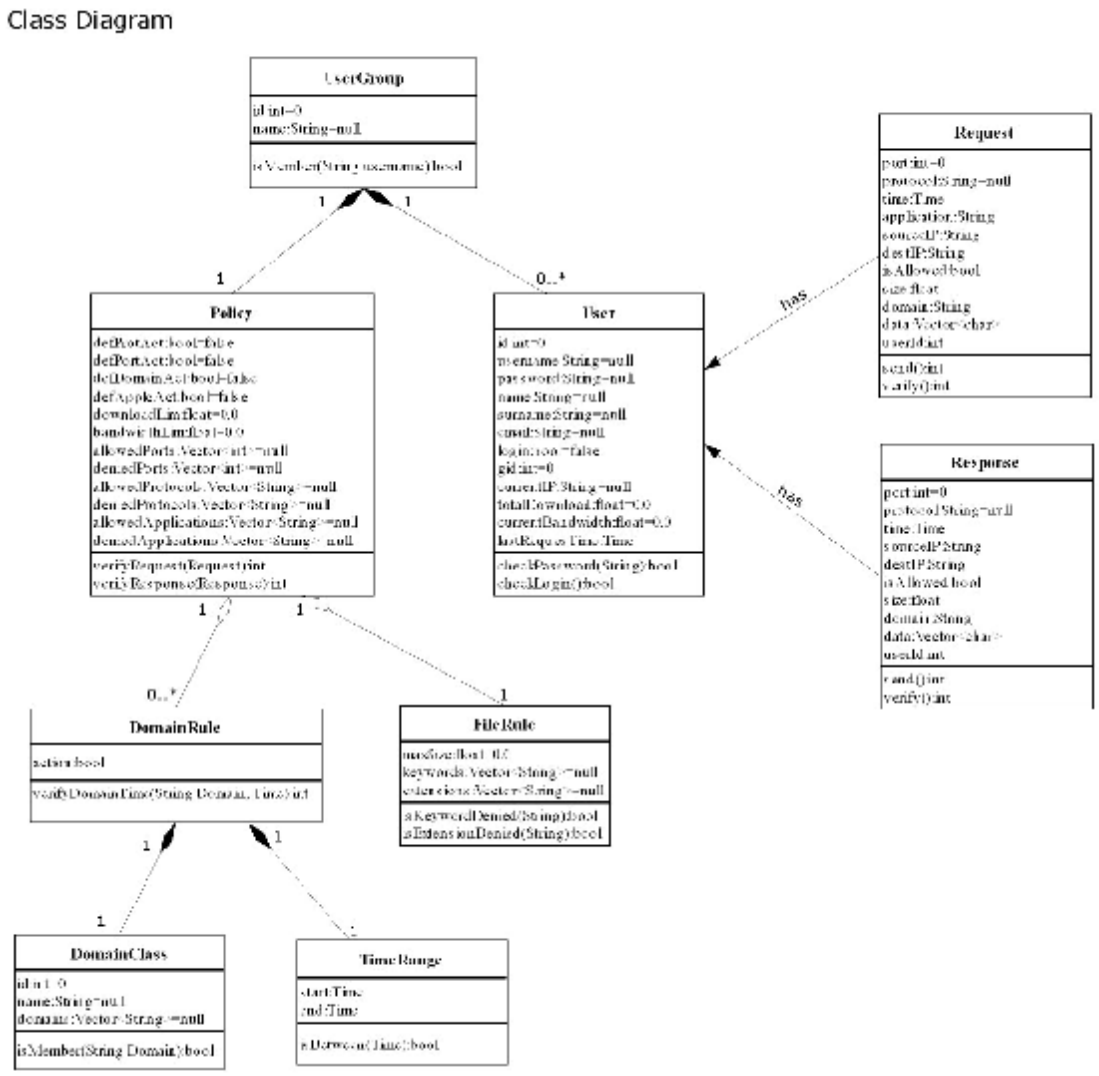

Figure 2.7: Class Diagram

2.1.3 SmartGuardian Controller

SmartGuardian Controller is totally web based, it checks, the incoming requests, (Administrative Command) for authentication. If the user (network admin) is not logged in, it displays the login page. Login requests and responses are handled by its sub module called Admin Authentication.

This module handles all the administrative requests and displays the results of these requests. It can also display system logs via its sub module called Reporter.

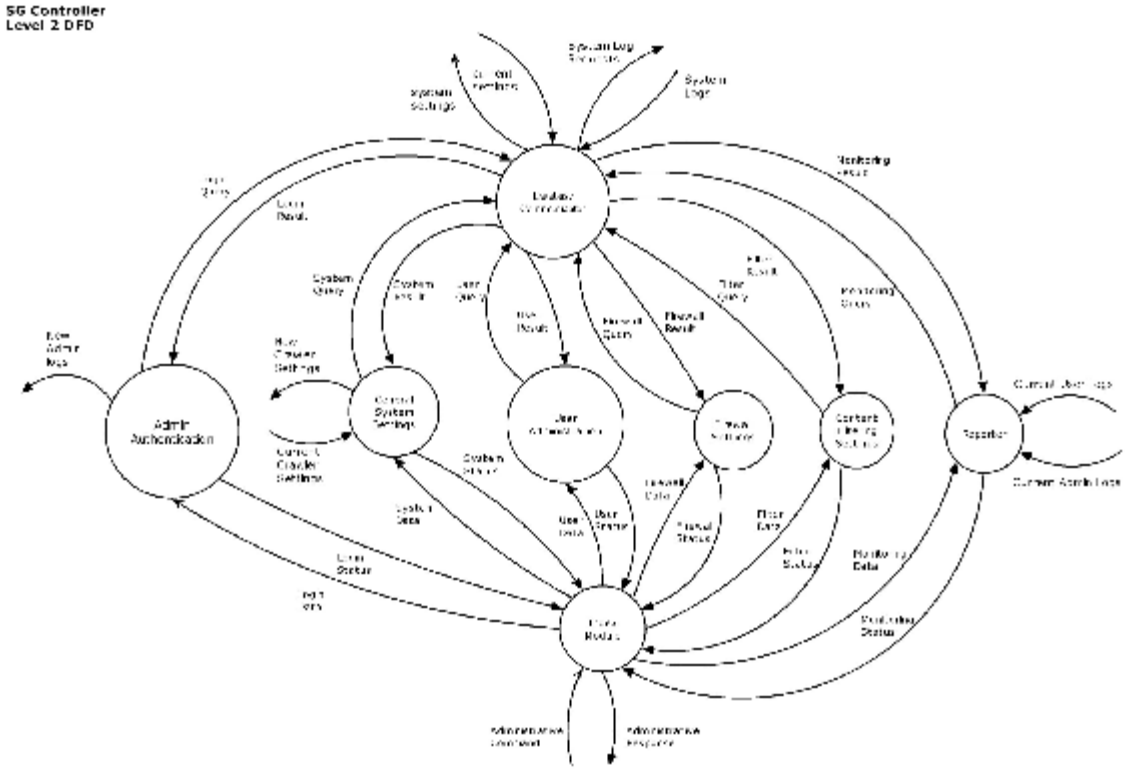

Figure 2.8: Level 2 DFD of SG Controller

When the SmartGuardian is started, the SmartGuardian Controller is in initial state. If a login request arrives, it goes to ***Checking Login*** state. In this state it firstly gets the admin usernames and passwords from the database, and checks the requested username and password. If the login is successful, moves to the ***Idle State***. If the login is unsuccessful, it goes to ***Error Reporting*** state.

When the Controller moves to the ***Error Reporting*** state it generates the error message, and sends it to the admin. Then it logs the unsuccessful login and moves to the ***Initial State***.

When the Controller is in ***Idle State*** and an admin command arrives, the controller moves to the ***Applying Command*** state. In this state it firstly applies the command. These commands may be to create a new user, to move a domain to a domain class, to start the Web Crawler, to create a new policy, to modify a policy, etc. After applying the command the Controller logs the changes. Then it generates a message stating whether the command is successfully applied or not and sends it to the administrator. Then it goes back to the idle state.

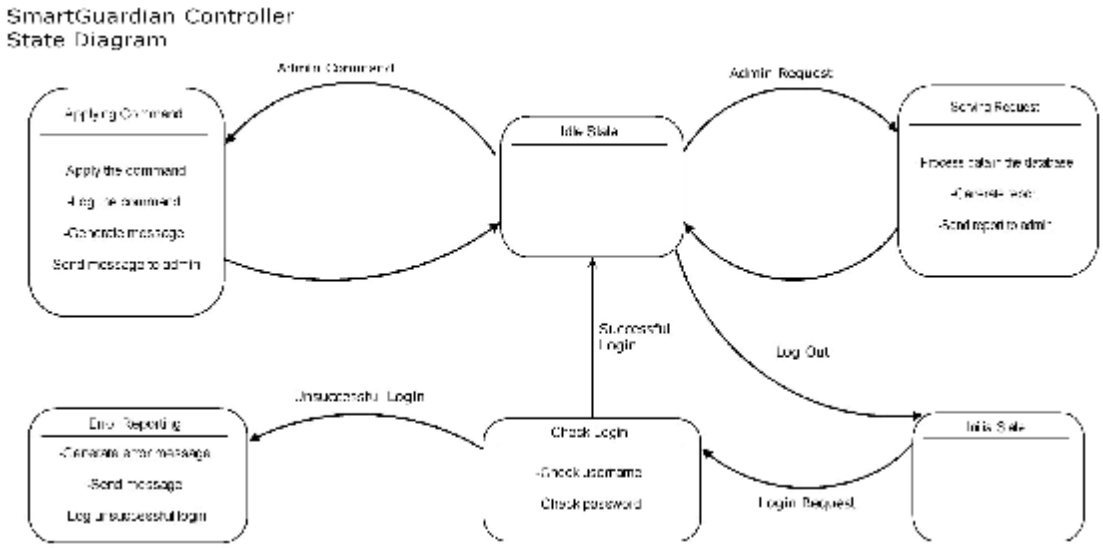

Figure 2.9: State Diagram for SG Controller

When the Controller is in ***Idle State*** and an administrator request arrives, it moves to the ***Serving Request*** state. In this state, it firstly analyzes the request. An admin can request to see statistical information about the users or domains, may request to see the current bandwidth usage, etc. Then the necessary data is obtained from the database and processed according to the request of administrator. After processing data, reports are generated and sent to the admin. Having finished with this state it returns to the ***Idle State***.

2.1.3.1 Admin Authentication

This process determines if requests are coming from a logged in user (possible network administrator) or not. Administrator logs into the system by this process. It checks username and password and records login result.

2.1.3.2 General System Settings

This module displays general system settings, and modifies those settings according to the demand of the user.

2.1.3.3 User Administration

This module displays current users of the system, new users can be added or existing ones can be deleted via this module. The status of each user is also displayed.

2.1.3.4 Firewall Settings

This module displays current setting of the firewall. Firewall settings can be modified by this module. It shows the current status of the firewall in detail, examines the modification requests for any possible inconsistencies.

2.1.3.5 Content Filtering Settings

This module displays current settings of the content filtering and its rules. Administrator can define, modify or delete policies using this module.

2.1.3.6 Reporter

System monitoring is achieved via Reporter. Reporter settings can also be done by this module. It is capable of displaying system status in various details which is determined by the user.

2.1.4 SmartGuardian Crawler

Crawler is responsible for maintaining URL classification tables of the database. Periodically, it gets the addresses from the database which is to be visited. It visits those addresses, analyses and processes their contents and tries to classify these contents then writes the result to the database.

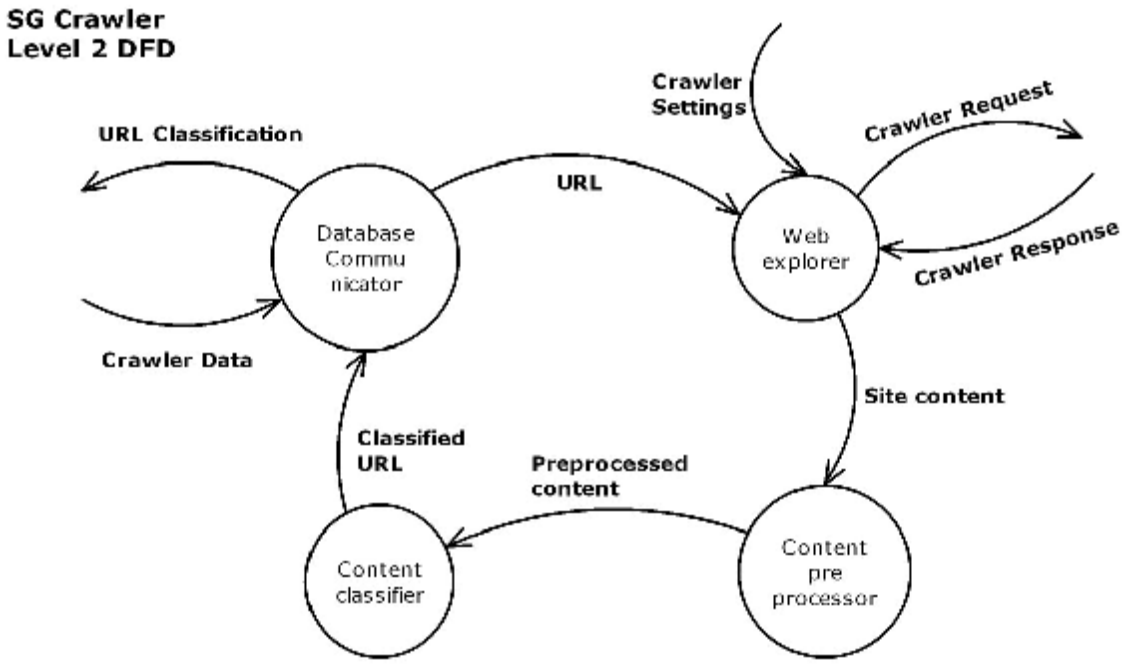

Figure 2.10: Level 2 DFD of SG Crawler

Crawler is composed of following sub modules:

2.1.4.1 Web Explorer

Web Explorer visits the addressees in the given list in appearance order; it also gets the response coming from the sites of these addresses and sends site contents to Content Preprocessor.

2.1.4.2 Content Preprocessor

Content Preprocessor inspects the site content; it processes this content and sends it to Content Classifier.

2.1.4.3 Content Classifier

This sub module takes preprocessed content, tries to classify it, if successful it puts the site to corresponding classification, if not, it marks the site as unclassified and writes the result to the database.

2.1.5 Authenticator

LAN Users are allowed to access the Internet only after logging into system. LAN users log in to the system from a web page generated by Authenticator. This module checks the user name and password for validity. If login is successful, it records the result to the corresponding table in the database.

2.1.6 Database Communicator

Though it has been stated that SmartGuardian is going to use MySQL as the database, it will provide great portability if it is implemented database independent. Database Communicator is the module of the system which will make it easy to port the system to other databases.

Modules of the SmartGuardian are going to be implemented database independent. Generic functions are used in order to access database. Those generic functions belong to the module Database Communicator and it is Database Communicator which takes care of database dependent issues. Thus, in order to port SmartGuardian to another database platform, developers only need to rewrite the Database Communicator module.

Database Communicator is used by all parts of the system which need database access. Those modules call generic database functions of Database Communicator.

2.1.7 Data Description

In this section the entities, data flows and processes are described.

2.1.7.1 Notation

The notation utilized in the data dictionary uses square brackets [ ] to denote optional items, curly brackets { } to denote items which may occur one or more times (repetition), and a vertical bar | to separate alternatives (selection).

2.1.7.2 The Data Dictionary

External Entity: LAN Internet User
Description: A user with the intention of connecting to the Internet.

External Entity: LAN Login User
Description: A user with the intention of connecting to the internet by first logging into the system. The user then will be presented a login page and after a successful login, he/she will be able to connect to the Internet.

External Entity: Internet
Description: The Internet here is not only the web page content but connection to any remote server which has a service associated with it.

External Entity: Admin
Description: Admin is a privileged user who has administration privileges. He/she will be able to manage the system, i.e. adding users, changing module settings, viewing current system status etc.

Process: SmartGuardian
Description: This is the system at its highest abstraction. The SmartGuardian Gateway is essentially a software system which controls the Internet access of organizations, taking security issues and organization policies into account.

Data Flow: LAN Request
Description: This is the request a LAN user makes with the intention of connecting to the Internet.
LAN Request = Source IP, destination IP, protocol, source port no, destination

port no, [additional request info], data.
Source IP = IP of the requesting client. It is the IP of the LAN user in this context.
Destination IP = IP of the destination server. It is the IP of the remote server in this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the requesting client during the communication process. It is the port number of the LAN user in this context.
Destination port no = The port number used by the remote server during the communication process. It is the port number of the remote server in this context.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: Verified LAN Response
Description: This is the data from the Internet a LAN user receives, which is verified by the system.
Verified LAN Response = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.
Source IP = IP of the responding server. It is the IP of the remote server in this context.
Destination IP = IP of the destination server. It is the IP of the LAN user in this context.

Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the requesting client during the communication process. It is the port number of the remote server in this context.
Destination port no = The port number used by the remote server during the communication process. It is the port number of the LAN user in this context.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: User Login Request
Description: This is the login request a LAN user makes before initiating communication with the Internet.
User Login Request = LAN IP, LAN port no, gateway port no, user name, password.
LAN IP = IP of the login requesting client. It is the IP of the LAN user in this context.
LAN port no = The port number used by the requesting client during the communication process. It is the port number of the LAN user in this context.
Gateway port no = The port number used by the gateway server during the communication process. It is the port number of the gateway in this context.
Username = Username of the user who is using the LAN IP. This username must

match with the password in the database.
Password: Password of the user who is using the LAN IP. This password must match with the username in the database.

Data Flow: User Login Response
Description: This is the login response a LAN user gets after his/her login attempt. It is created by the gateway server.
User Login Response = {error number, error explanation} | success message.
Error number: This is the error number returned with the error message if the login attempt is unsuccessful.
Error explanation: This is the error explanation given by the gateway.
Success message: This is a confirmation that the login attempt was successful.

Data Flow: Verified LAN Request
Description: This is the request made by the LAN user which is verified to comply with the policies the user is subject to. It has the same attributes as the User Internet Request since it is verified and forwarded by the gateway.
Verified LAN Request = Source IP, destination IP, protocol, source port no, destination port no, [additional request info], data.
Source IP = IP of the requesting client. It is the IP of the gateway in this context.

Destination IP = IP of the destination server. It is the IP of the remote server in this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the requesting client during the communication process. It is the port number of the gateway in this context.
Destination port no = The port number used by the remote server during the communication process. It is the port number of the remote server in this context.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: LAN Response
Description: This is the response made to the LAN user which is verified to comply with the policies the user is subject to. It has the same attributes as the User Internet Response since it will be verified and sent to the user.
LAN Response = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the responding server. It is the IP of the remote server in this context.
Destination IP = IP of the destination server. It is the IP of the gateway in this context.

Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the responding client during the communication process. It is the port number of the remote server in this context.
Destination port no = The port number used by the gateway during the communication process. It is the port number of the gateway in this context.
Additional response info = Any additional response information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the LAN user.

Data Flow: Crawler Request
Description: This is the request made by the crawler module to the Internet. The crawler downloads web sites and classifies them.
Crawler Request = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the crawler residing server. It is the IP of the gateway server in this context.
Destination IP = IP of the destination server. It is the IP of the remote server in this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the requesting client during the

communication process. It is the port number of the gateway in this context.
Destination port no = The port number used by the remote server during the communication process. It is the port number of the remote server in this context.
Additional request info = Any additional request information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is sent to the Internet.

Data Flow: Crawler Response
Description: This is the response to the crawler from the Internet for its request. It has the same attributes as the “crawler request” data flow since it is only a response to that request.
Crawler Response = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the responding server. It is the IP of the remote server in this context.
Destination IP = IP of the destination server. It is the IP of the gateway in this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the responding client during the communication process. It is the port number of the remote server in this context.
Destination port no = The port number used by the gateway during the

communication process. It is the port number of the gateway in this context.
Additional response info = Any additional response information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is processed by the SmartGuardian Crawler module.

Data Flow: Monitoring commands
Description: This is the response to the crawler from the Internet for its request. It has the same attributes as the “crawler request” data flow since it is only a response to that request.
Crawler Response = Source IP, destination IP, protocol, source port no, destination port no, [additional response info], data.
Source IP = IP of the responding server. It is the IP of the remote server in this context.
Destination IP = IP of the destination server. It is the IP of the gateway in this context.
Protocol no = Protocol used by the client and server while communicating with each other.
Source port no = The port number used by the responding client during the communication process. It is the port number of the remote server in this context.
Destination port no = The port number used by the gateway during the communication process. It is the port number of the gateway in this context.

Additional response info = Any additional response information that is relevant to the used protocol and communication type.
Data: This is the reason for communication. Data is processed by the SmartGuardian Crawler module.

Data Flow: Administrative Command
Description: This is a system command that the administrator issues to control the way system works.

2.2 Database Design

ER diagram is a representation of our database which is the core part of our system. In the ER diagram we have a set of entities that are designed for each module's needs.

Most important entity in this relation is our ***Policy*** entity. It has a relation with ***ports, protocols,user_group & Domain_Classes.*** For the policy entity there is a relation with ports that has information ***deny***. Deny attribute contains 0 or 1 that is the status of ports, whether allowed or not. Likewise the same relation exists with Domain_Classes and protocols entity. Since for a specific policy there must be some ports, protocols and domain classes and also their status. There is also relation between Policy & user_group. For each group there must be one unique policy. Policy entity also has attributes which have some default actions for indefinite ports, protocols or URLs.

For storing ports and protocols we have two entities which are ***ports*** & ***protocols.*** Each ports and protocols is an attribute and primary key of these tables. Besides that we have ***File_Rules*** table that keeps an id and size attribute. This entity has two relations with ***Keywords*** & ***extensions*** entities. Keywords entity has ***word*** attribute that keeps some key words that will be used in filtering process. Extensions entity has extension attribute that contains extensions of file.

Each policy has some defined status for domain classes whether it is allowed or not. It is kept in ***deny*** attribute of ***has_domain_class***. Domain_Classes entity has an id ***did & name*** attribute that defines that domain. In ***Domain_Classes*** has ***domain.*** Domain entity is a set of URLs. In URLs attribute there will be kept IP "10.34.234.13", domain "www.ceng.metu.edu.tr" or URLs "http://www.gospatial.com/html/viewers.html". These are some sample data that can be stored in here. For a specific policy and a specific domain class there are some time intervals that are kept in ***timerange*** entity. It has two attributes ***start*** & ***end.*** Type of these attributes is ***time*** and they are keys of this entity.

The other important entity is ***User***. It keeps whole user in local area network. Name and surname of each user are attributes of this entity. It also keeps username & passwords that are checked in user authentication process. Each user has an e_mail address that is kept in the system in order to communication of system with the user. Each user must be a member of a ***user_group.*** Each user group has an id ***gid*** and must have unique policy.

In this ER diagram we have two relation ***response*** & ***request.*** These relations are going to be a table in the database. Each user has some requests that have a set of attributes such as ***source_ip, destination_ip, time, protocol*** etc. that defined the request. These attributes are used in validation and security consideration of request. Likewise such consideration will be achieved for responses.

Besides the entities in ER, we have an isolated entity ***admin*** that will be stored into database as table. Since system use this table for authentication process apart from other tables, there is no relation between admin table and other tables.

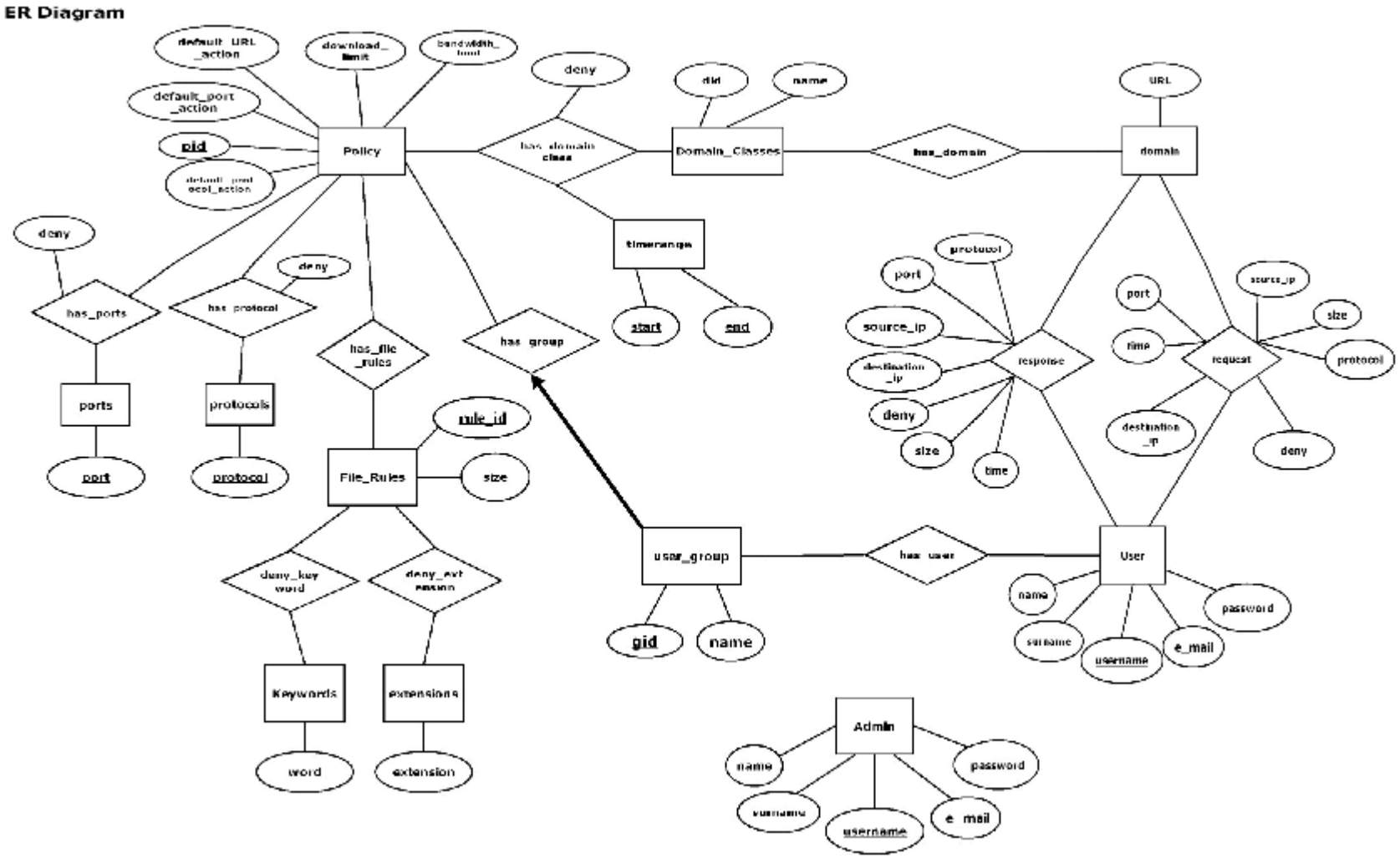

Figure 2.11: The ER Diagram

Below we have the data dictionary of ER. In the data dictionary we have entity descriptions and some relation descriptions of our ER diagram. I the data dictionary explanations of entities and relations are given. There we have also description of attributes of entities and relations.

Entity type name: Policy	
Definition: Policy entity has a set of attributes related which form a policy of a user group which is references by has group relation. Policy has domain classes and an attribute with that domain class and each relation has a status deny means that for that policy which domain classes allowed or denied. Policy has also relation with ***ports & protocols*** which have same deny attribute for that policy. Besides that, Policy table has relation has_file_rules with ***File_Rules*** table that keeps of extension and content keywords that will be used in content filtering. Policy table has also attributes that is described below;	
Data element:	default_URL_action
Description:	Flag for URL to allow/deny [0 | 1]
Where used:	In checking the status of a URL is allowable or not
Data element:	default_port_action
Description:	Flag for Port to allow/deny [0 | 1]
Where used:	In checking the status of a port is allowable or not
Data element: Description: Where used: Data element: Description: Where used: Data element:	default_protocol_action Flag for Protocol to allow/deny [0 | 1] In checking the status of a Protocol is allowable or not bandwidth_limit A defined maximum size of speed for user to download a file. In comparing the current speed with the user speed download_limit

Description:	A defined maximum size of packages allowed for the policy
Where used:	In comparing the total size of usage of the user in current time in order to apply interruption or not.
Data element:	pid
Description:	Primary key for Policy entity which is a integer

Entity type name: ports
Definition: Ports entity has a set of ports which is defined by user. It has a unique attribute Port where contains port like 80. It has a relation has_ports n to n with policy and each have an attribute which are allow/deny. Which ports are open for which policies and which are close.

Data element:	port
Description:	An integer that represents the port number
Where used:	In checking for a given relation of a policy with ***Ports*** table which ports are defined for that policy

Entity type name: Protocols
Definition: Protocols entity has a set of protocols which is defined by user. It has a unique attribute Protocol. It has a relation has_protocol n to n with policy and each have an attribute which is deny. Which protocols are allows or banned for which policies.

Data element:	protocol
Description:	A string like (http, ftp, smtp etc.) that is the name of protocol that are inserted by user
Where used:	In checking for a given relation of a policy with ***Protocols*** table which protocols are defined for that policy

Entity type name: File_Rules

Definition: Is an entity with attribute size; it has a relation ***has_file_rules*** relation with Policy entity. It keeps the file rules in a given policy; maximum downloadable size of file, allowable or deniable files extensions and keyword in the file.	
Data element:	size
Description:	The value that keeps the maximum downloadable size of a file
Where used:	It is used in getting the bandwidth limit of policies for specific user
Entity type name: Keywords	
Definition: Keeps a set of keyword that is used in both file rules of policies and ***Web Crawler*** module. It will be one of the largest tables of the database. It has an attribute Word as a string	
Data element:	Keywords
Description:	Contains a string like "***sex***" that give hints as a keyword for the what the file contains
Where used:	In data mining issues of ***Web Crawler*** and file comparing file names with those keywords - in string matching -
Data element:	rule_id
Description:	Primary key for Keywords entity which is a integer
Entity type name: extensions	
Definition: Keeps a set of extensions of files that will be used in file rules for which is for a policy.	
Data element:	extensions
Description:	Contains a string like "***mp3, jpeg, mpeg, zip***" that defines the type of file
Where used:	It is used in file rules application of a policy whether the file with this extension is allowable or not
Entity type name: Domain_Classes	
Definition: In the database we have 18 different types of domain classes which have ***has_domain*** relation that keeps domains that are inserted by either user or Web Crawler. Each domain has id	

and name.	
Data element:	did
Description:	Primary key for domain classes entity
Data element:	name
Description:	Like did it is also some default name that is an attribute of Domain_Classes
Where used:	It is useful and more recognizable in defining a ***Policy*** & ***Domain_Classes*** relation
Entity type name: timerange	
Definition: timerange keeps some definite time interval in ternary ***has_domain_class*** relation for a specific Policy for a specific Domain_Classes.	
Data element:	start
Description:	Type is ***time*** ; it is the start time of ***timerange*** and a part of key
Data element:	end
Description:	Type is ***time*** ; it is the end time of ***timerange*** and a part of key
Entity type name: domain	
Definition: the entity set is composed of URLs and domains which are unique for each domain	
Data element:	URL
Description:	URL is a string that specify the domain; it can be an IP ***10.345.35.7*** or ***www.metu.edu.tr***
Where used:	It will be used in checking of request or response of a user whether the domain is allowable or not. Besides that Web Crawler will be used for data mining process of related domains web pages
Entity type name: user_group	

Definition: It represent types of user; each user has unique policy	
Data element:	gid
Description:	It represents the user group name as an integer
Where used:	Primary key of the entity

Entity type name: User	
Definition: It keeps the entire user that benefit from Internet in LAN. Besides that it has attributes of description of the user. Each user that has ***request*** & response relation	
Data element:	name
Description:	String ; name of user
Where used:	Identifying user in order to send a message
Data element:	surname
Description:	String ; surname of user
Where used:	Identifying user in order to send a message
Data element: Description: Where used: Data element: Description: Where used: Data element:	username String ; username of the user : Primary of entity Is checked in authentication of user password String ; password of the user Is checked in authentication of user e_mail

Description:	Email address of the user
Where used:	It will be useful for sending system administrator messages
Entity type name: Admin	
Definition: Admin will be also stored in database. But it does not have any relation with other entity sets. It all have same attribute with user. However admin authentication will be achieved from system directly.	

Relation type name: response	
Definition: Response is a relation between ***User*** and ***domain***. For each request that send internet throughout system, related domain returns response. Then some information related with response stored in database. Attributes of response taken from package header are below:	
Data element:	source_ip
Description:	IP address of domain that have been requested
Where used:	Checking for security and policy consideration
Data element:	destination_ip
Description:	IP address of user that requested related domain's pages
Where used:	Describing the computer
Data element: Description: Where used: Data element: Description: Where used:	deny Status of related response is valid or not Checking of validity of request size Size of pages or files that the user requested Comparing with download limit of the policy that the user in

Data element:	time
Description:	Time of the response
Where used:	Checked in updating & deleting response from database
Data element:	port
Description:	Port number of the response
Where used:	For security consideration
Data element:	protocol
Description:	Protocol of the response
Where used:	For security consideration

Relation type name: request
Definition: Request is a relation between ***User*** and ***domain***. Each user has a request during internet application for a specific domain. Then some information related with request stored in database. Attributes of request taken from package header are below:

Data element:	source_ip
Description:	IP address of user that requested related domain's pages
Where used:	Describing the computer
Data element:	destination_ip
Description:	IP address of domain that have been requested
Where used:	Checking for security and policy consideration
Data element:	deny
Description:	Status of related request is valid or not
Where used:	Checking of validity of request
Data element:	size

Description:	Size of pages or files that the user requested
Where used:	Comparing with download limit of the policy that the user in
Data element:	time
Description:	Time of the requested
Where used:	Checked in updating & deleting requests from database
Data element:	port
Description:	Port number of the request
Where used:	For security consideration
Data element:	protocol
Description:	Protocol of the request
Where used:	For security consideration
Relation type name: has_ports	
Definition: has_ports is a relation between Policy and ports. Each policy has some predefined port numbers that have a status whether allowed or denied which is represented in ***deny*** attribute.	
Relation type name: has_protocols	
Definition: has_protocols is a relation between Policy and protocols. Each policy has some predefined protocols that have a status whether allowed or denied which is represented in ***deny*** attribute.	
Relation type name: has_domain_class	
Definition: has_domain_class is a relation between Policy and Domain_Classes. Each policy has a elation with all 18 different domain classes and for each of them has a status which is represented in ***deny***.	
Data element:	deny
Description:	Status of has_ports, has_protocols, has_domain_class relations [0 | 1]

Where used:	Checking in decision of policy rules

2.3 Behavioral Description

You can see the use-case diagram below. It shows actions from a sample space. One can notice what the administrator and/or users are expected to do by interacting the software system.

As can be seen, the administrator can initiate a crawling session, configure the system in a detailed sense and monitor the system status. A user is expected to request a web page using the SmartGuardian software.

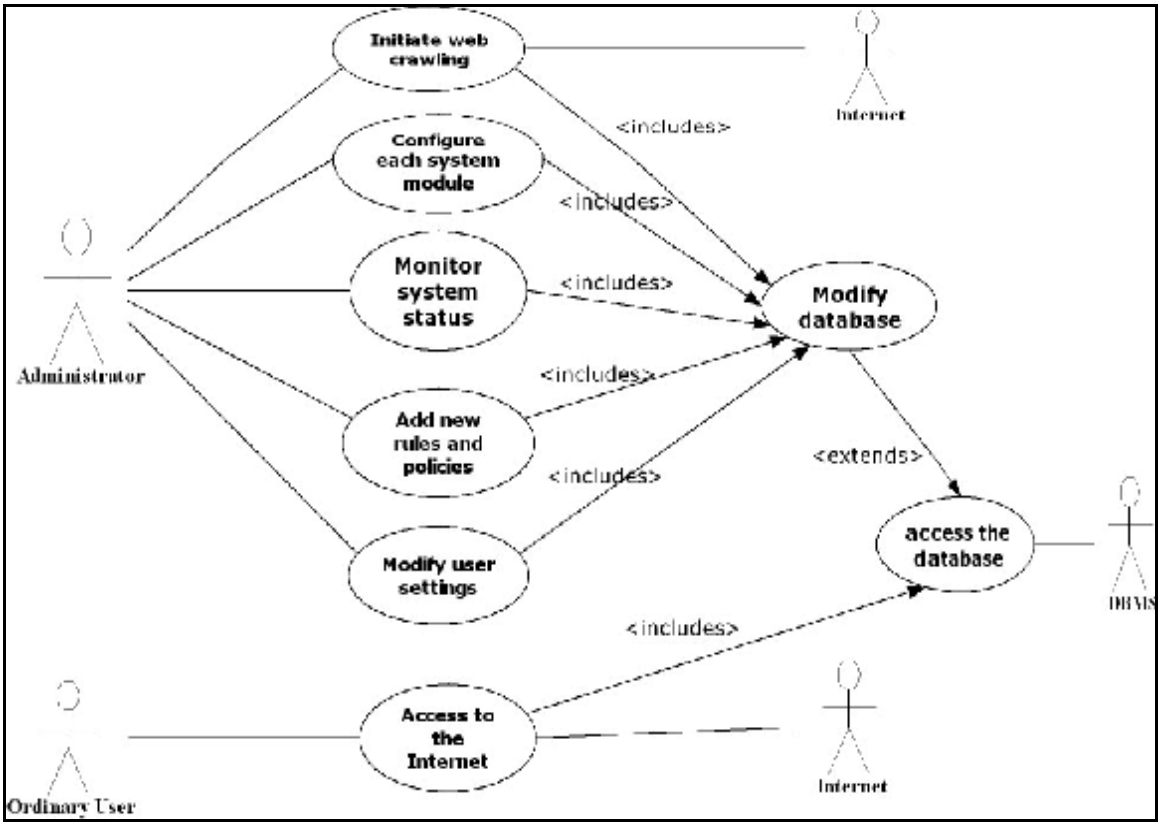

Figure 2.12: The use-case diagram

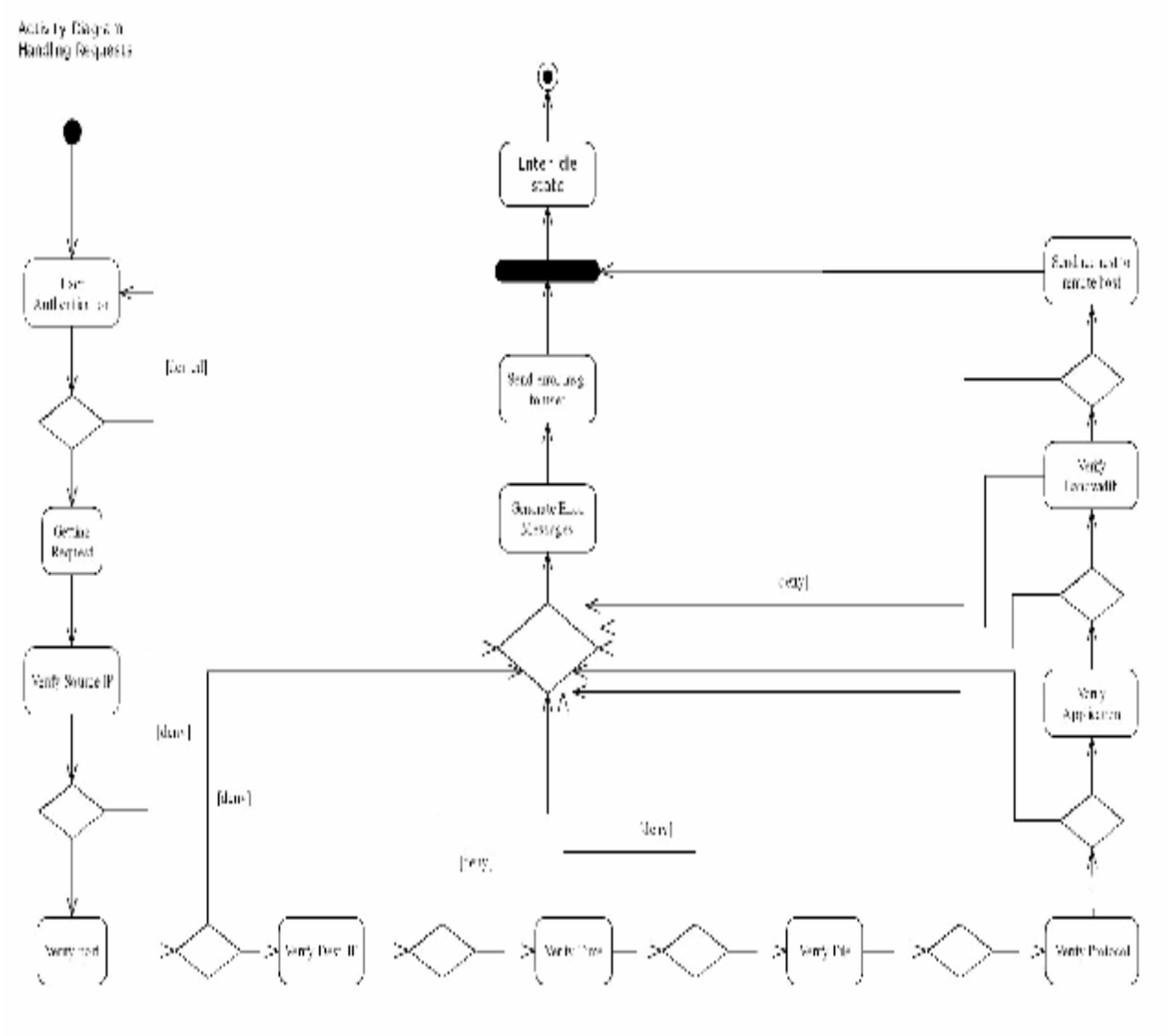

Figure 2.13: Activity Diagram of Handling Request

In figure 2.13 we have a revolution of an activity that is achieved by user represented as an activity diagram. User has an authentication process at the beginning; if he/she is not successful, authentication process will be repeated again and again. After authentication the request of user will be get by system in order to analyze. Firstly Source IP of the system will be verified. If it is not valid; it means that given IP of computer is not in database then it is denied. For the owner of the user verified port, protocol URL or IP, time violates the policy defined for the user. Moreover if the

request is a file, extension and keywords protocols & bandwidth are checked for policy validity. If any of them violates it invokes the system in order to create an error messages. Then the message will be posted to user as an error messages. Then the process life will be over, if the results of all verifies are valid then the request will send through internet.

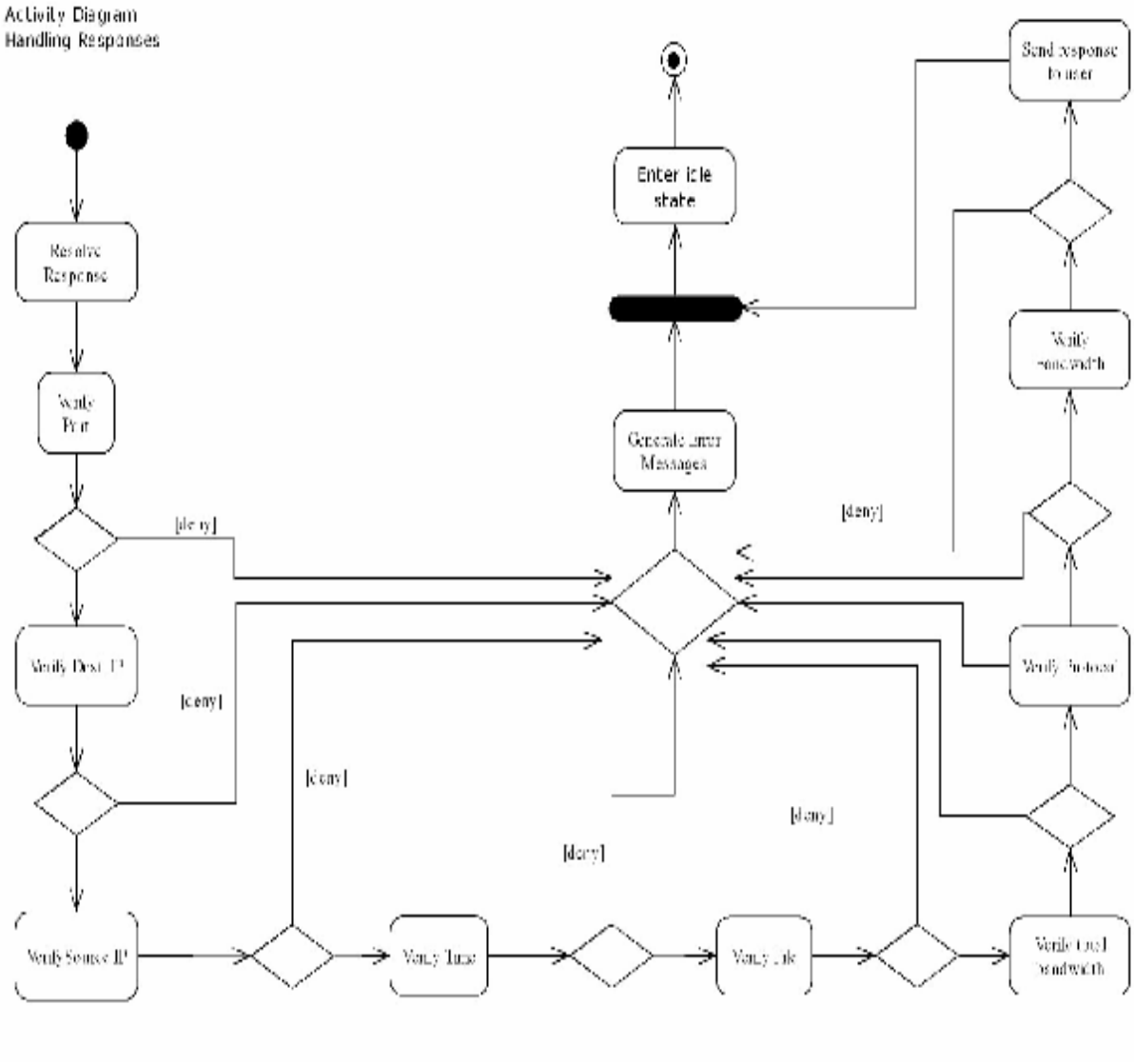

Figure 2.14: Activity Diagram of Handling Response

In figure 2.14 we have a lifecycle of a response activity that is achieved by related domain that is requested before. As it seen in response activity, a response is verified many times in order to check whether the policy is violated or not. Firstly port is verified; this is done for security reasons. Some malicious code use strange ports generally. Secondly destination IP is verified. This means that for which policy we consider the filtering. Then source IP is verified, if it is in URL list then the time, bandwidth and protocol is requested for validity. All the verifying results are negative (denied) then error messages created and will be posted to the user. Otherwise the response is sent through user.

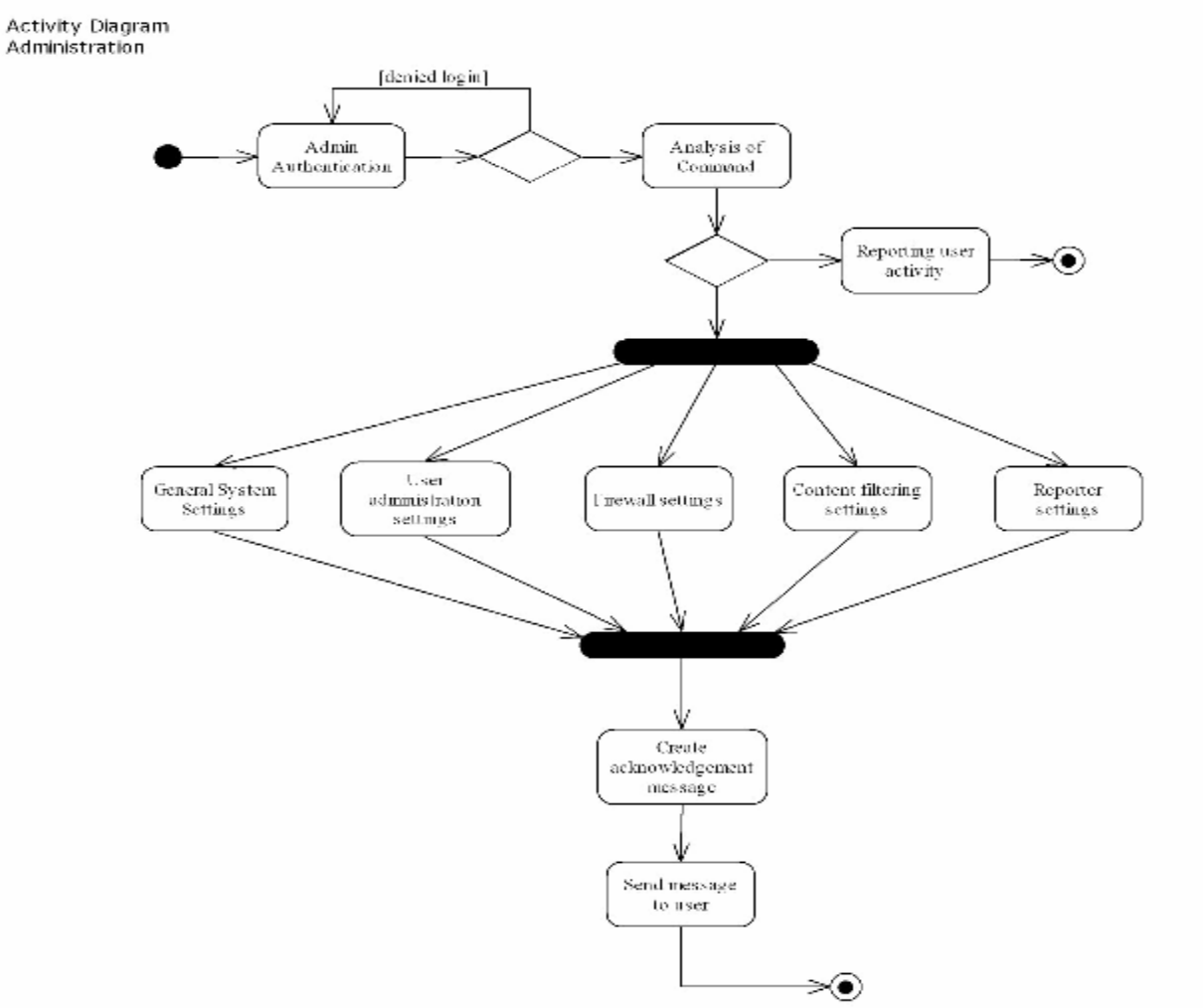

Figure 2.15: Activity Diagram of Handling Administrator Command

In figure 2.15 we have a lifecycle of an administrator command. Firstly, like users the administrator should also authenticate to the system in order to effect to system settings or monitoring action of users. After authentication the user has choices, one is just monitoring user actions history with Reporter sub module. It is done by investigation of reporter to log scripts that are stored in log files. The other choice of administrator is system settings. It would be general system setting, user setting, firewall setting, content filtering setting or reporter setting. Each of these settings is done by PHP interface. Then an acknowledgement will be sent to the administrator whether the settings are successfully done or not. That is the end of the administrator command cycle.

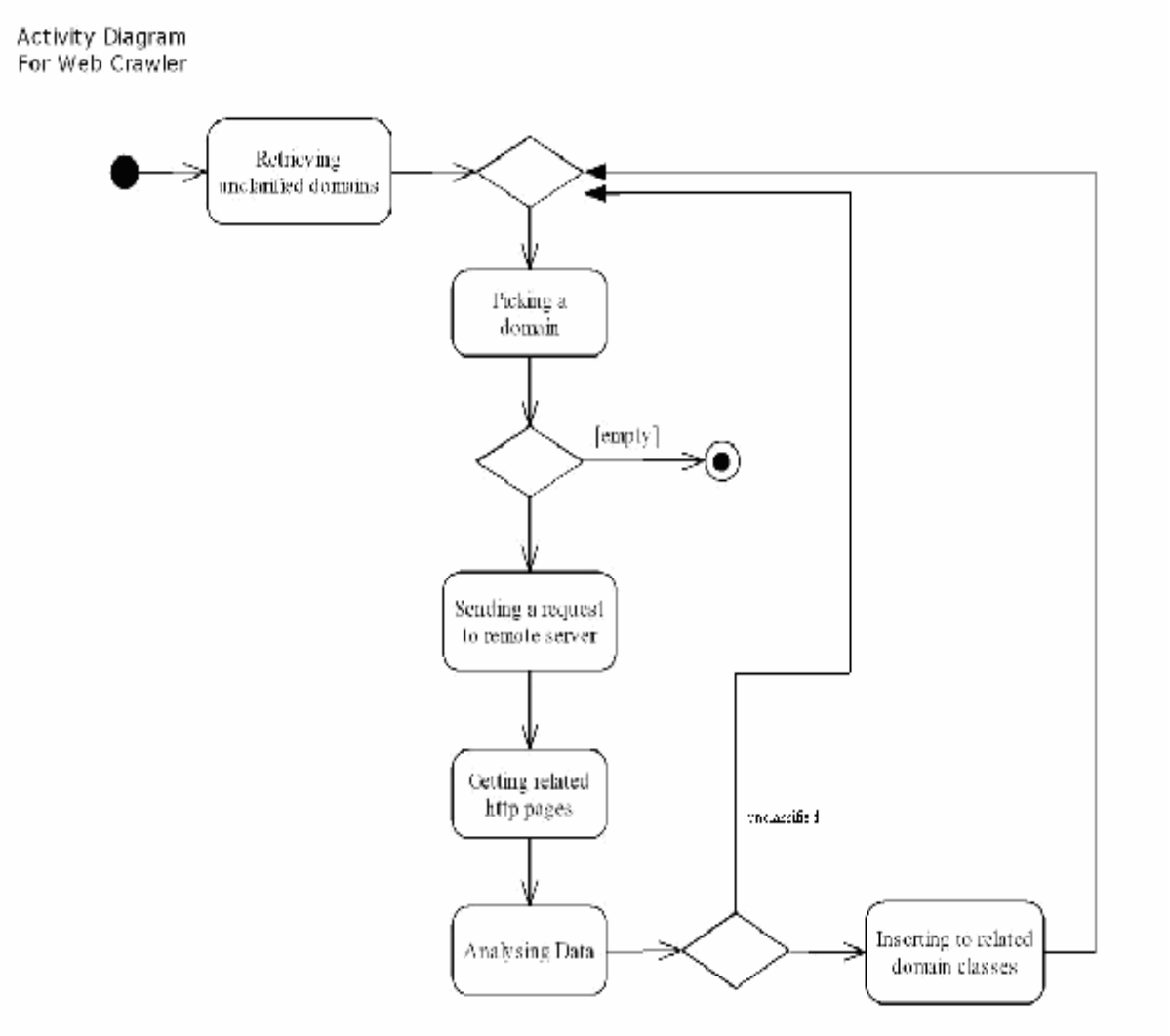

Figure 2.16: Activity Diagram of Web Crawler lifecycle

The aim of the web crawler is domain classification of unclassified domains in the ***Unclassified Domain Class*** table. It gets domain from URL list to a buffer. After investigating each domain and applying data mining process after getting related HTTP pages it try to classify these domains into one of the 18 different domain classes. If it is not successful after applying data mining it skip insertion process and pick another domain.

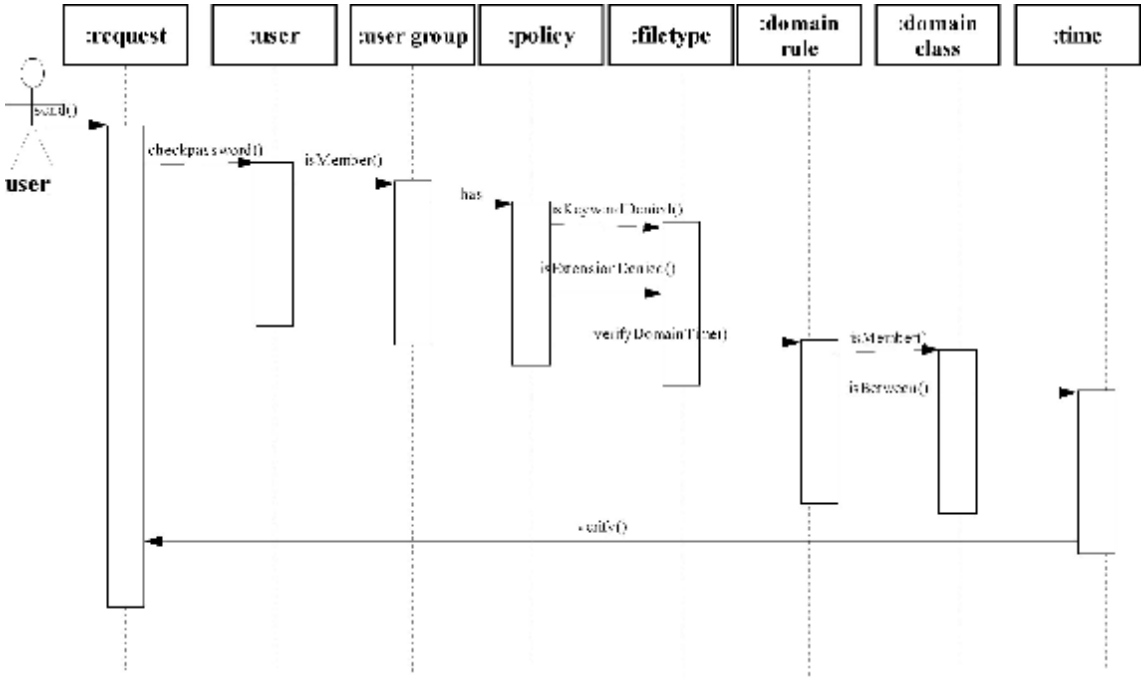

Figure 2.17: A Sequence Diagram

A request by the user is first controlled by the user class. In order to control the password we use the checkpassword() function. After this process the user group is being checked by a function called isMember() and is send to user group class. All the users have a policy defined. Inside the policies we have different classes. A request can be both checked by keyword and file extension with the help of isKeywordDenied() and isExtensionDenied() respectly. The policies have also

domain rules checking both for domain class and time. For checking domain class we basically control if the requested address is a member of the domain classes or not with the help of the function isMember(). For the time restriction we look at the time class if there is any restrictions defined for the time we are connecting by isBetween() function.

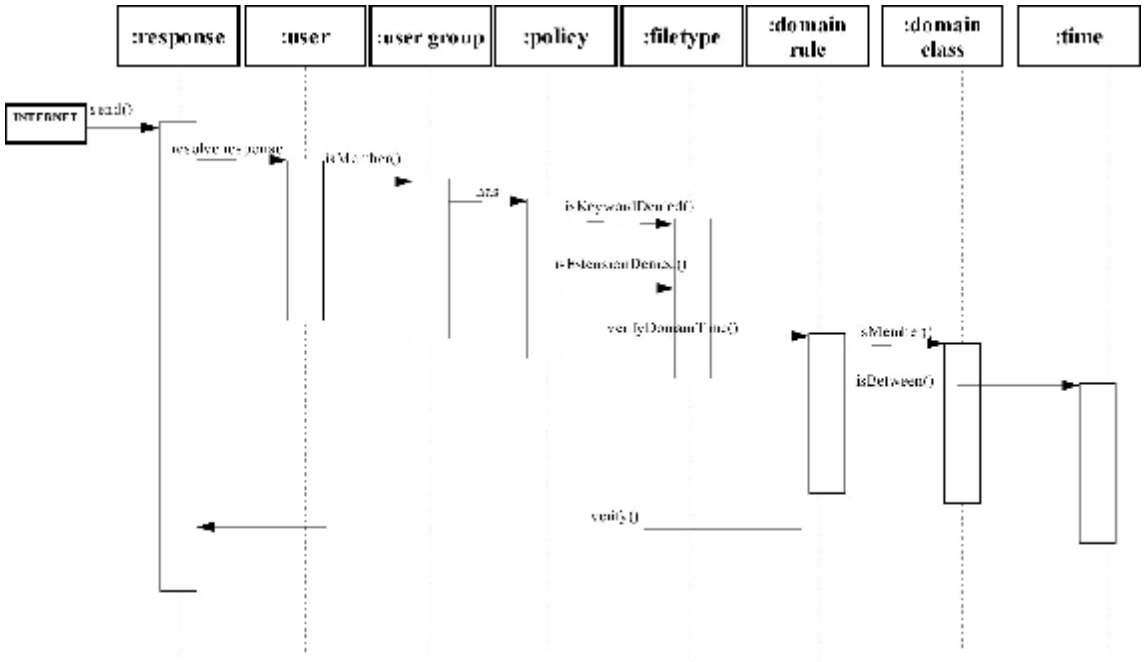

Figure 2.18: A Sequence Diagram

The steps in the response are mostly the same as request. The significant differences are; this time we get the information from Internet and there is no password control that means we directly transfer response to user class. The other steps are same as request class.

2.4 Syntax Specification

User and admin login history syntax.

For ordinary user logs:

<username> <date> <time> <result> <source ip> <\n>

For administrator logs:

<username> <date> <time> <result> <source ip> <action> <\n>

Action is an integer code representing the action taken by the administrator.

Each action will be on a separate line. And each entity is separated by a whitespace.

File Names:

Actions are logged in a daily basis, more explicitly, there will be a separate log file for each day with the day being the file name. ex. User logs on the day 03/12/2005 will be saved in a file u031205.log, also admin files created on the same day will have a031205.log as its name.

Crawler Settings:

<start_time> <end_time>

The two items we have determined which will be included for sure in the settings file is the starting time and the ending time for the crawler module. We will add new items as may be necessary.

Crawler Logs:

<starting time> <number of websites visited> <number of websites classified> <finishing time>

<starting time> : The time crawling started.

<number of websites visited> : This is the number of websites visited during this crawling session.

<number of websites classified> : This is the number of websites classified during this crawling session. This number does not include websites that are not classified, for example when a website cannot be reached.

<finishing time> : This is the time crawling session finished.

Reporter Settings:

<sort by=[table column]> <show=[table]> <use charts=[pie | bar]>

<sort by=[table column]> : This item determines according to which criteria will the information coming from the DB will be sorted.

<show=[table]> : This item determines which tables from the database will be displayed to the administrator.

<use charts=[pie | bar]> : This item determines how the database information will be shown to the administrator. It will enable visual presentation to the administrator.

Controller Settings:

<timeout> <ip range> <max number of denials>

<timeout> : It is the maximum amount of time that administrator can stay idle until the system logs him out.

<ip range> : This is the allowed IP range that can connect to the controller module. It is added for extra security.

<max number of denials> : This is the maximum number of login failures from a particular IP. If the number of denials exceeds this number, the IP will be banned.