2009
CEng 491
DSK4BRMS

INITIAL DESIGN REPORT

Yetkin KARIŞ
Metin BARIŞ
Erkan AKYOL
Ghassan ALSHANA

MOCKWARE

TABLE OF CONTENTS

1. INTRODUCTION
- 1.1. PROJECT TITLE
- 1.2. MOTIVATION
- 1.3. PROJECT DEFINITION
- 1.4. PROJECT GOALS

2. REQUIREMENT ANALYSIS
- 2.1. SYSTEM REQUIREMENTS
- 2.2. FUNCTIONAL REQUIREMENT

3. ARCHITECTURAL DESIGN
- 3.1. MODULES
  - 3.1.1. MANAGEMENT MODULE
  - 3.1.2. EXECUTION MODULE
  - 3.1.3. INTERACTION MODULE
- 3.2. DATA FLOW DIAGRAMS
  - 3.2.1. LEVEL 0 DFD
  - 3.2.2. LEVEL 1 DFD
  - 3.2.3. LEVEL 2 DFD
- 3.3. STATE TRANSITION DIAGRAMS
- 3.4. ENTITY RELATIONSHIP DIAGRAMS

4. OO DIAGRAMS
- 4.1. USE CASE DIAGRAM
- 4.2. ACTIVITY DIAGRAMS
- 4.3. CLASS DIAGRAMS
- 4.4. SEQUENCE DIAGRAMS

5. LANGUAGE PATTERNS

6. USER INTERFACE DESIGN

7. PROCESS
- 7.1. TEAM STRUCTURE
- 7.2. PROCESS MODEL
- 7.3. GANTT CHART

8. CONCLUSION

9. REFERENCES

1. INTRODUCTION

1.1. PROJECT TITLE

We decided to name our project as BRules. ‘B’ is stand for business which is our main concern in this project.

1.2. MOTIVATION

Business rules touch our lives in many interesting ways. They can dictate your credit worthiness, what type of loan or insurance rate you qualify for or even why you are overlooked for the last business class upgrade at the airport.

Driving these decisions is a new generation of business rules management systems (BRMS) designed to automate decision making in enterprise IT applications. These systems differ radically from the old 'expert systems' of yesteryear that failed to catch corporate IT attention because they were too complex, expensive to run and maintain and not business-user friendly.

Organisations are now starting to realise that a more hands-on approach is needed. They are looking to a new breed of BRMS technology to empower workers to write their own business policies as the competitive climate demands.

Today rules impact a huge number of target business applications ranging from insurance adjudication, loan approval, claims processing, credit scoring, product/service recommendations, order configuration and fraud detection. A typical application implements between 100 and 1,000 rules. Complex rules are not only difficult to code into applications, they are also a nightmare to maintain using traditional coding.

There are a number of advantages gained by expressing business logic in business rules and using the processing and management facilities included in BRMS to work with them. One is cost and time savings - between 25% to 75% - over the application lifecycle simply through a reduction in modification cycle times. BRMS are simply more flexible. Changes can be made immediately to rules by business whenever the need arises. In a traditional procedural coding implementation they

would have to understand the sequencing of rules programmed in the code, which is incredibly time-consuming and expensive.

Another is consistency across all touch-points that use rules. In most cases a corporation's business rules are defined in manuals or other documents, or possibly not defined formally at all. Many organisations rely on business managers to interpret company policies as business rules, which can be inefficient and lead to inconsistent application of rules. This imperative becomes more important as organisations decide to offer the same services and access to information over multiple channels of communication: to customers, employees, or partners.

BRMS technology continues to mature, with attention now turning to the areas of rules simulation and testing, as well as analytics. Since BRMS is a decision-enabling technology it makes sense to use analytic techniques to optimise them using a scorecard-like approach. For example, routing a customer with a glaring propensity to churn to an experienced customer service agent, or determine if he or she is likely to take up an offer if presented with a rebate incentive.

Rules are also starting to pique the interest of larger IT vendors such as Microsoft, IBM, Oracle and SAP. When these companies start to invest more heavily in rules-based business logic within their applications, industry pundits expect to see a wave of sell-offs.

1.3. PROJECT DEFINITION

BRules is a domain specific toolset for business rule management. It consists of three main parts namely Language, Engine and User Interface. We will use existing rule languages to create our structure.

Currently there are many tools to manage business rules such as ILOG, JDREW and Mandarax. They have the ability to execute rules which are internal i.e. are not have to interact with external databases, web services or other applications. Our aim is to make a generic language and required engine which will also be able to send a query to external sources and request response from external sources.

Users can define, classify and manage rules on the other hand they can send queries to the engine and engine will respond according to defined rules. But the main issue that we will working on is external servers that BRules interacts with.

1.4. PROJECT GOALS

When our project is done, we will like to have created a domain specific language(DSL), a domain specific engine(DSE) and a domain specific toolkit(DST).

- User will be provided a graphical interface with a toolset.

The opening screen in our program will be flexible. According to type of the user opening screen will be changed. All type of users will be able to manage or execute rules without any coding. The toolset of our program will assist user.

- This program will provide a module for testing results.

The regular user will behave like a tester, so rule manager will. This user can test result set of queries.

- The programmer will be able to extend rules.

New rules can be created, or deleted by rule manager. The program will be flexible about managing rules.

- A powerful domain specific language will be created.

We will create a markup language that will be more generic than most present markup languages. This language will have extra keywords and properties to connect more sources.

- The program will be able to interact with external sources.

If the user needs to connect an external source, our program will supply this property in a wide area of sources. Our engine will be able to be in touch with web services, databases and some application programming interfaces. This is the most powerful part of our program since most present markup languages do not support this opportunity.

- The program that we will create will be able to be developed.

Developers can create new interactions with other sources by developing our program. Since we want our program to be generic, it will be adaptable to most development tools.

2. REQUIREMENT ANALYSIS

Interface Requirements

The business rule system in our program will interact with user,get information from user and according to user's requests the business rules will be executed or managed including creating basic rules and composite rules and updating rules. So our program will include external and internal interfaces. The external interface will be available since we want user not to deal with exhaustive codes, so the user will have a graphical user interface for managing business rules easily.

External Interface

The first part of our program is external user interface which is a graphical user interface directly in touch with user. Our external interface will have log in part according to authentication. This interface will have 2 different screens according to authorisation of user. In this part there will be 2 type of users respectively regular user and rule manager. When logged in as a regular user, the user will be redirected to execution module. In this module the user will be able to send queries and get response. When logged in as a rule manager, the user will be redirected to management module. This user will be generaly the business analyst. Rule manager will be capable of creating new basic rules, creating new composite rules from existing ones, deleting existing rules, modifying, inserting and updating rules.

Internal Interface

In internal interface there will be three modules respectively management module, execution module and interaction module. According to user's requests the business rules will be managed in management module which only the business analyst will have permission to manage rules, or the rules will be executed in execution module, or interact with fact set when necessary.

The business system in internal interface will have an architecture. This system has a rule editor that provides a user interface, an inference engine that applies the rules using its algorithms, a rule repository that saves information related

to the rule, a rule object model handler which controls over the rule object model, the rule object model that express the rule class data, rule adapter that supply an interface between rule editor and rule object model handler,rule repository handler that control rule repository, and so on.

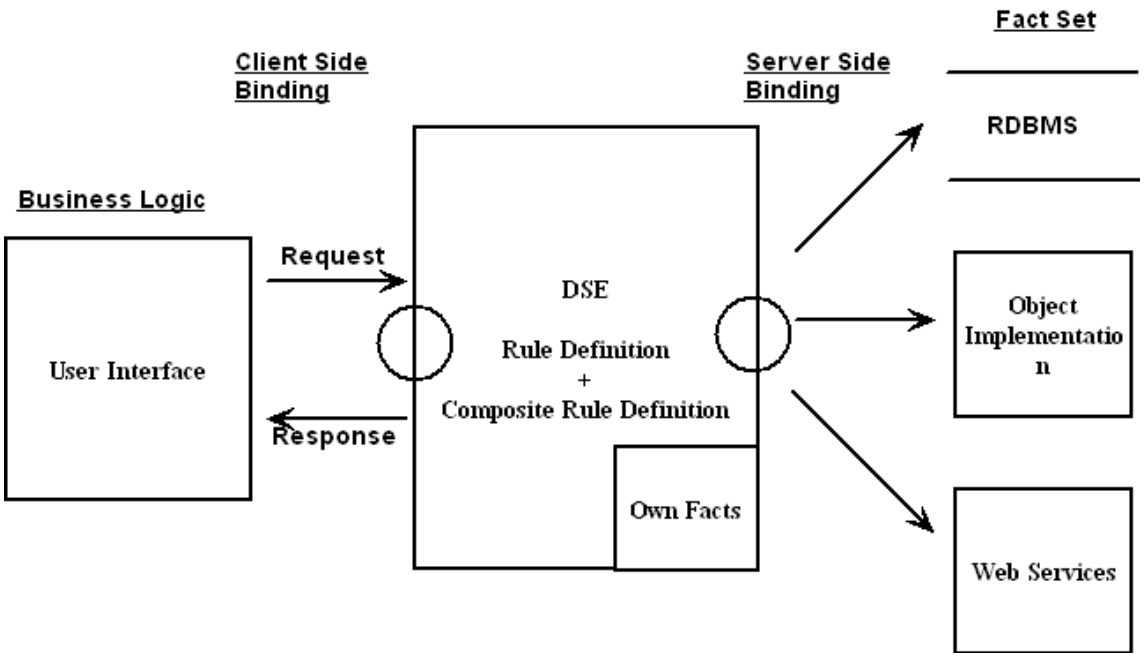


Figure 1. General Architecture of BRules

Functional Requirements

Our program will have three main parts namely user interface, domain specific engine and fact set.

User Interface:

This is the first part of our program which interacts with user. There will be a graphical user interface which will have two types of users. According to authorisation of user, this interface will be in touch with domain specific engine.

Domain Specific Engine:

All rule related operations will be done in this part. When user wants to do operations using graphical interface, this request will be granted in this part and make operations such as executing or managing rules according to information coming from user. This part will also have rulebase and its own facts. While making rule related operations, program will interact with its own fact set or external fact set such as web services, some application programming ınterfaces if necessary.

Fact Set:

This part is the last part of our program. This fact set will include web services, some application programming interfaces, relational database management system(RDBMS), and domain specific engine's own facts. Our engine will interact with this part if necessary according to granted information from user.

Our program will get information from user, make C++ binding to make operations in our domain specific engine for business rules management. If the user is a regular user:

- There will be a log in screen.
- When user log in as a regular user, there will be an execution screen for the user.
- Regular user will send queries using a graphical interface.
- The query will be got with C++ binding and verified in domain specific engine.
- By using query, our program will get initial rule.
- By forward chaining, all related rules will be evaluated and according to corresponding facts.
- If our domain specific engine's own facts are sufficient for evaluating, our program will use C++ binding to use facts inside domain specific engine.
- If our own fact set is not sufficient to evaluate rules and extra fact set is needed, our program will interact with some external fact set among web services, application programming interfaces or relational

database management systems where the corresponding facts for the rule in.

If the user is a rule manager:

- Also called as business analyst.
- This user will already have authorisation as much as regular user have and will be able to send queries to program.
- After logging in as a rule manager, the user will see a management screen.
- Rule manager will use a toolset in management screen.
- This toolset will enable rule manager to create basic rules, create composite rules from existing rules, delete existing rules, modify and update rules.
- Domain specific engine will gather information with C++ binding from the user.
- Then the request will be verified such that the user can create a rule that does not exist or can delete a rule that exists, and so on.
- To make rule related operations, domain specific engine(DSE) will interact with only rule repository.
- If creation or update needed, domain specific engine will do corresponding operations and update rule repository.
- If deletion needed, the corresponding rule will be found in rule repository and will be deleted by our domain specific engine. So on rule repository will be updated.

3. ARCHITECTURAL DESIGN

3.1. MODULES

BRules engine is composed of Manager, Executor and Connector Module.

3.1.1. MANAGER MODULE

Management module has 3 classes. These classes are:

- Parser
- Analyzer
- Organizer

Parser gets the rule or fact input in xml format and sends it Analyzer module right after checking validity and changing format of input.

Analyzer gets FormattedRule from Parser and by combining it with data from rulebase it gives a RuleSet(family tree of rule).

Organizer should take the RuleSet and insert it into right place and in right form to Rule Repository and if needed it makes changes in other rules.

3.1.2. EXECUTOR MODULE

This module has 4 classes. These classes are:

- Query_Listener
- Recognizer
- Trigger
- Engine

Query_Listener is responsible for getting queries from the user. It checks the query and send int to Recognizer.

Recognizer convert the Verified_Query into Logic which has a tree-like data structure.

Trigger is for getting corresponding rules from rulebase, ask for related data from Connector module, and fire the Engine.

Engine is the heart of the structure. When it is fired it finds the result of query. We will use Rete algorithm or forward chaining to achieve this process.

3.1.3. CONNECTOR MODULE

This module is responsible for connection and data transfer from external sources and has 2 classes which should be extended.

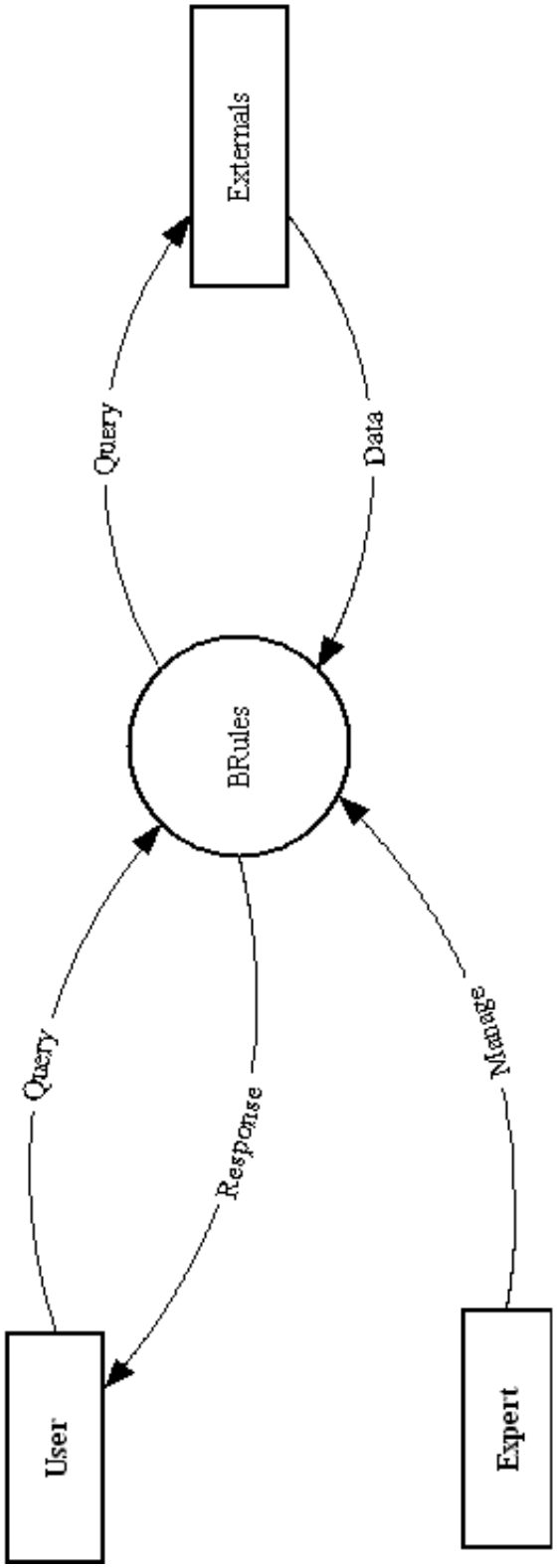
- Solver
- Adapter

Solver is to define the needed data from external sources and determine the server or source to be connected.

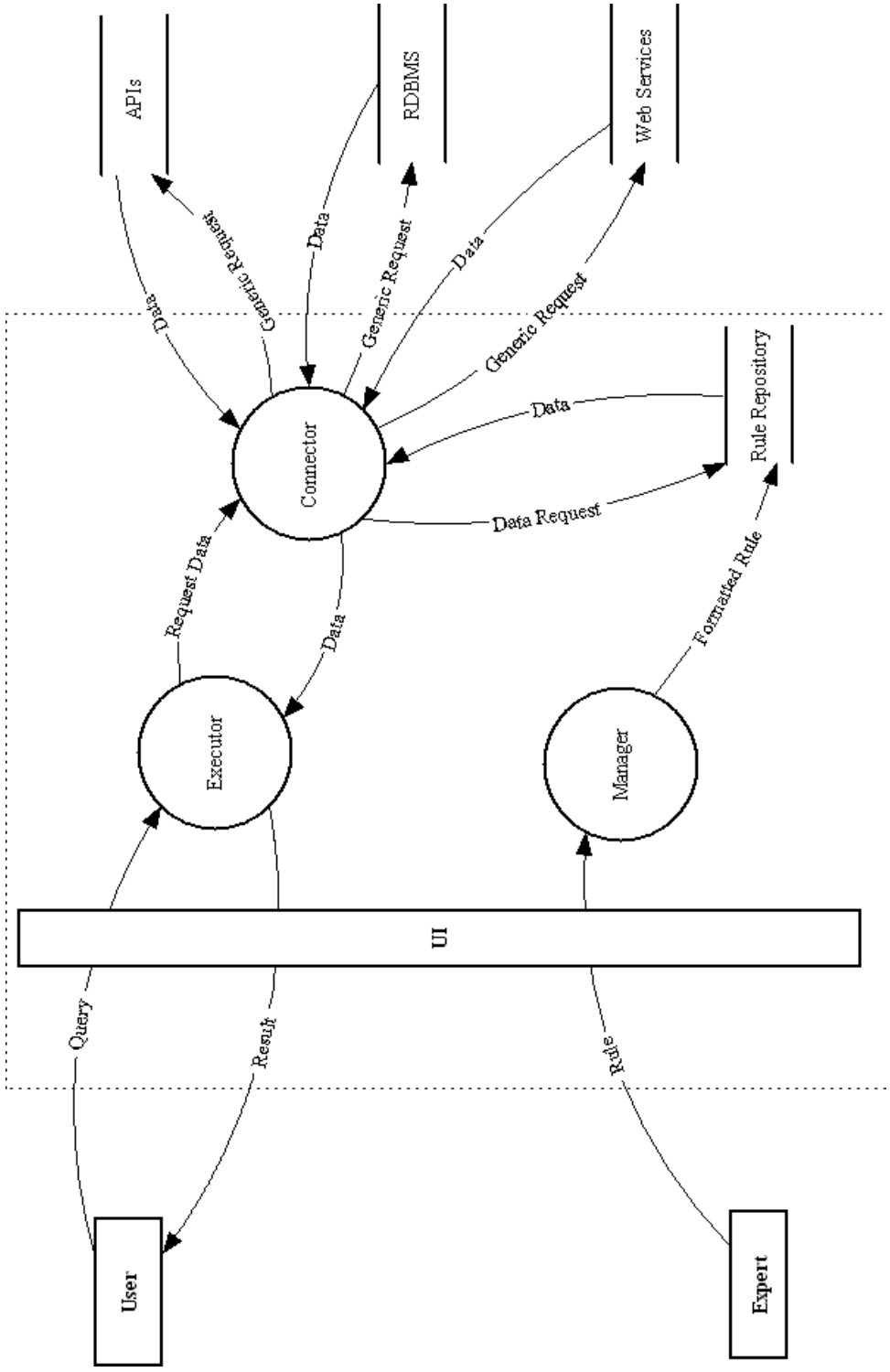
Adapter makes a connection between our Engine and external source and gets required data.

3.2. DATA FLOW DIAGRAMS

3.2.1. LEVEL 0 DFD

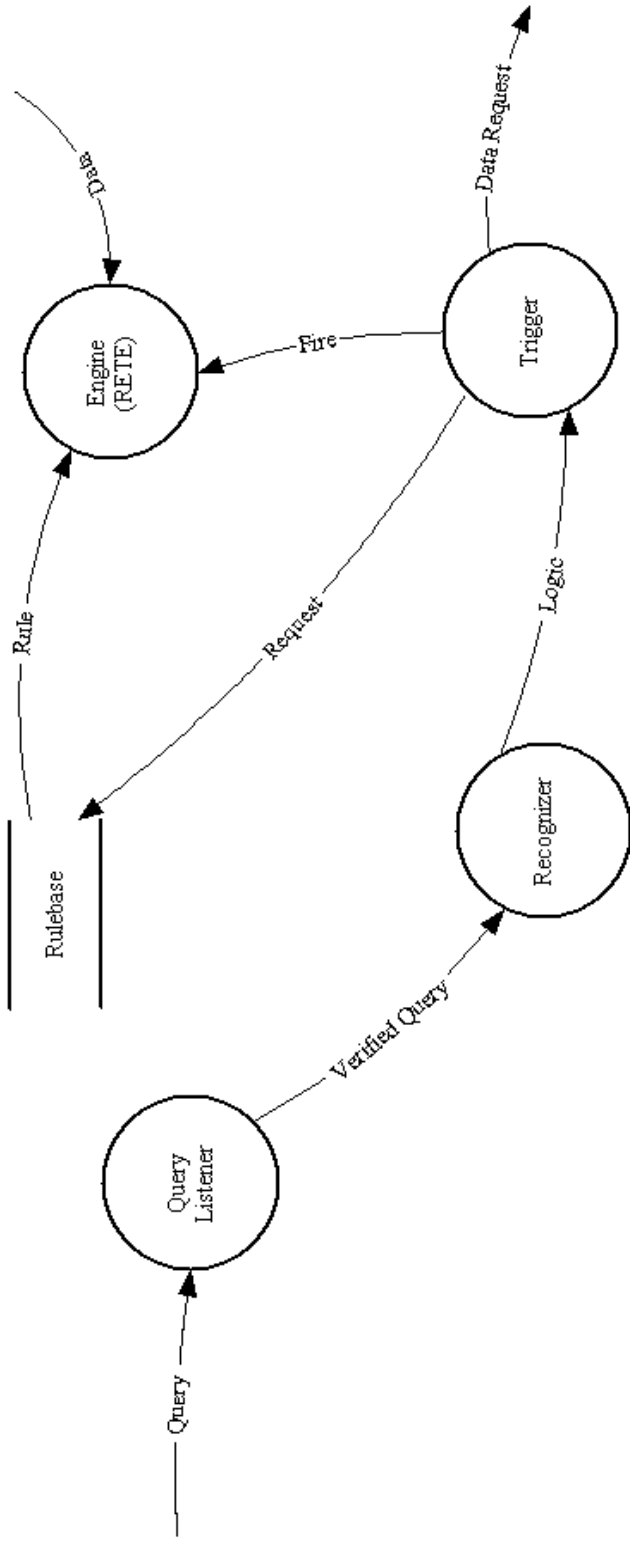


3.2.2. LEVEL 1 DFD



3.2.3. LEVEL 2 DFD

Executor Module



Name	Query
From	User
To	Query_Listener
Description	Data that is sended as input to our program. It is sended by User

Name	Verified_Query
From	Query_Listener
To	Recognizer
Description	This data is qualified to be query for input of our program.

Name	Logic
From	Recogniser
To	Trigger
Description	This data is an instance of a class that will be our logical item like conditions and rule products(under maintenance).

Name	Request
From	Trigger
To	Rulebase(Repository)
Description	This data asks a rule from rulebase

Name	Rule
From	Rulebase
To	Engine
Description	This data is our rule object that asked by trigger for engine to iterate

Name	Data
From	Connector
To	Engine
Description	Data is a needed data from connector to find from to complete engine stage

Name	Data_Request
From	Trigger
To	Connector
Description	It is a query-like data that will be recognized by connector

Manager Module

Analyzer
Ruleset
Organizer
Formatted Rule
Rulebase
Organize
Parser
Rulebase Request
Rule Repository
Rule

Name	Rule
From	Expert
To	Parser
Description	String-like data with keywords and it is nested

Name	Rulebase_Request
From	Parser
To	Rulebase(repository)
Description	Data that asks for rulebase and it is string type

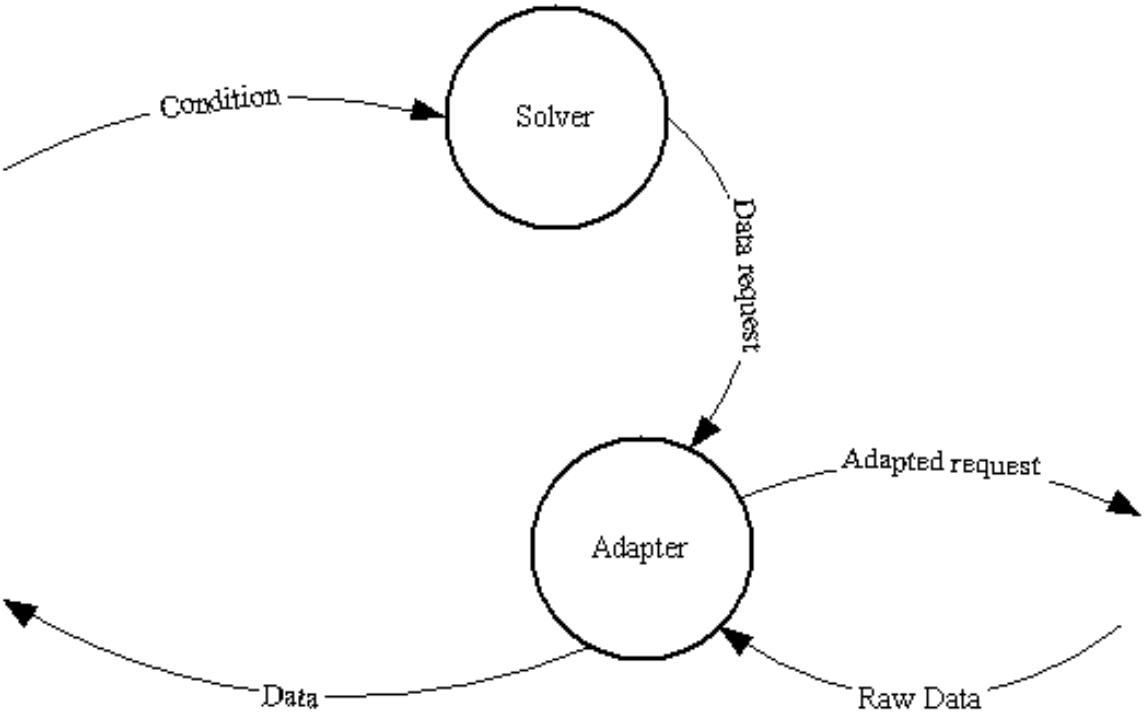
Name	Formatted_Rule
From	Parser
To	Analyzer
Description	Rule class data created by parser from input rulestring,it is verified.

Name	Rulebase
From	Rulebase
To	Analyzer
Description	Root of a family rule tree can be used for reaching rule nodes.

Name	RuleSet
From	Analyzer
To	Organizer
Description	A family rule tree for inserting and organizing rulebase

Name	Organize
From	Organizer
To	Rulebase
Description	Organized ruleset to be inserted to rulebase and extra data to change organization(will be seperated in next phase)

Connector Module



Name	Condition
From	Executor
To	Solver
Description	This data is to create server and Data information

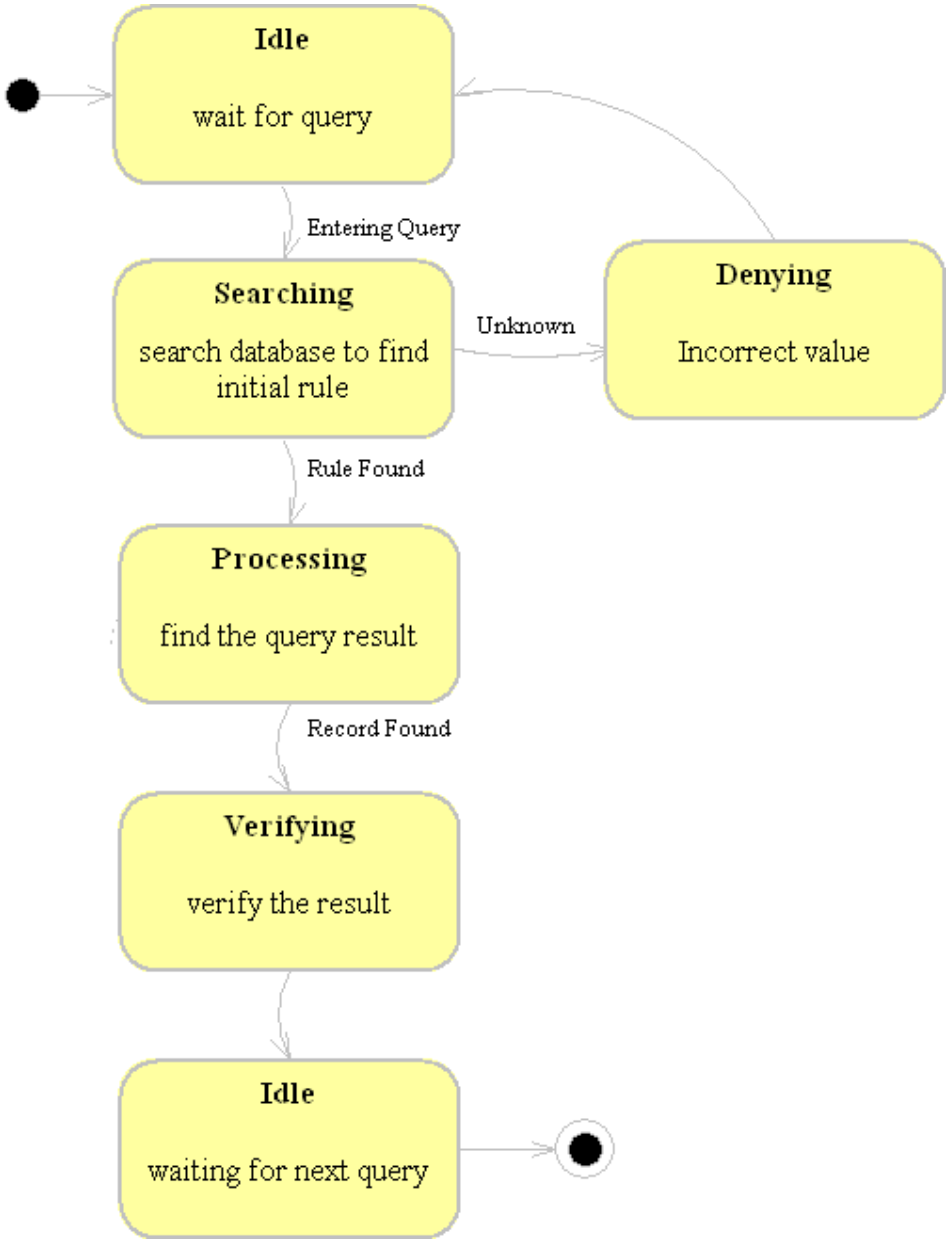
Name	Data_Request
From	Solver
To	Adapter
Description	Data that carries server and data information

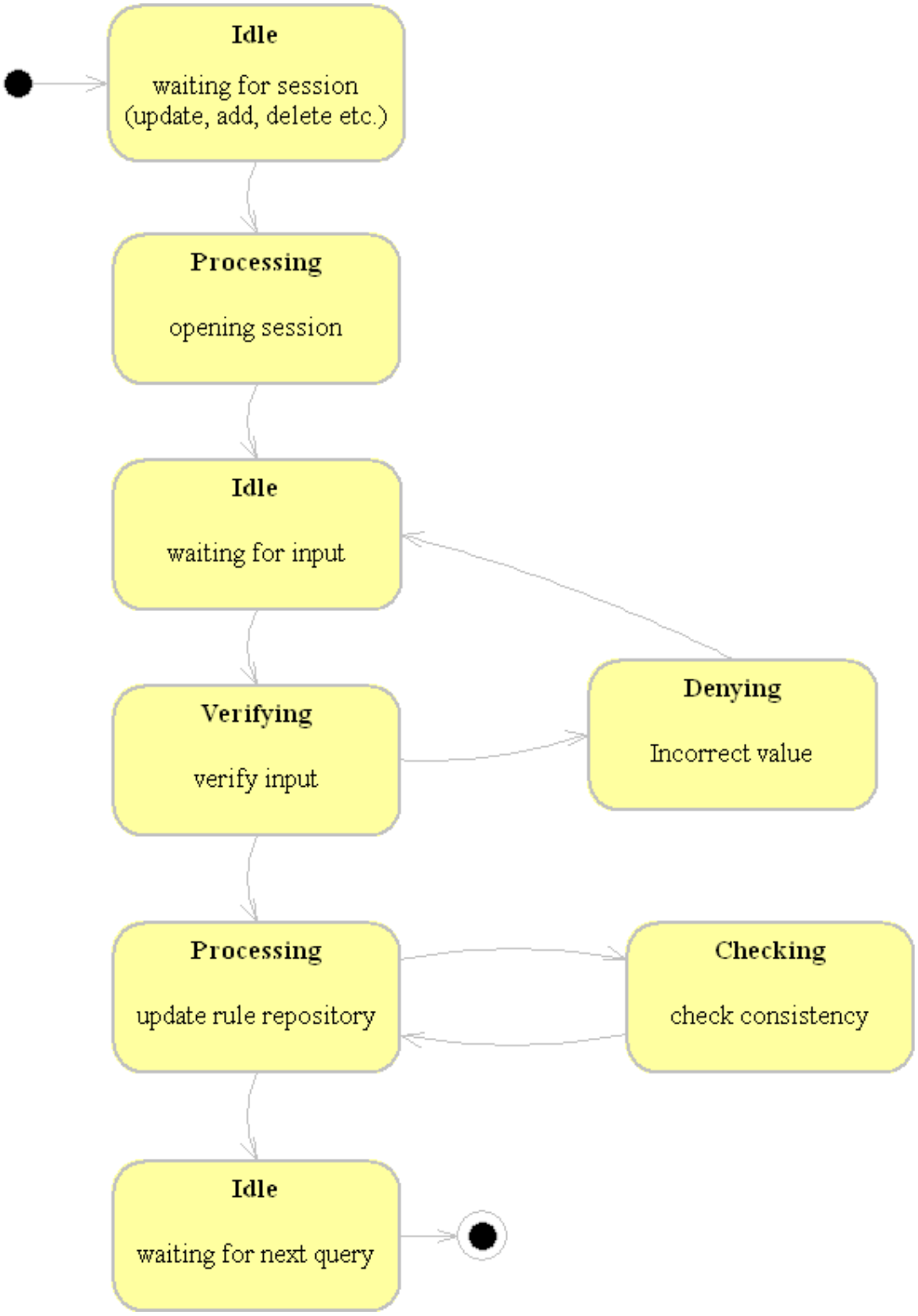
Name	Adapted_Request
From	Adapter
To	External
Description	The data request that can be recognized by externals

Name	Raw_Data
From	External
To	Adapter
Description	Incoming raw data from external

Name	Data
From	Adapter
To	Executor
Description	Needed data in known format to complete execution

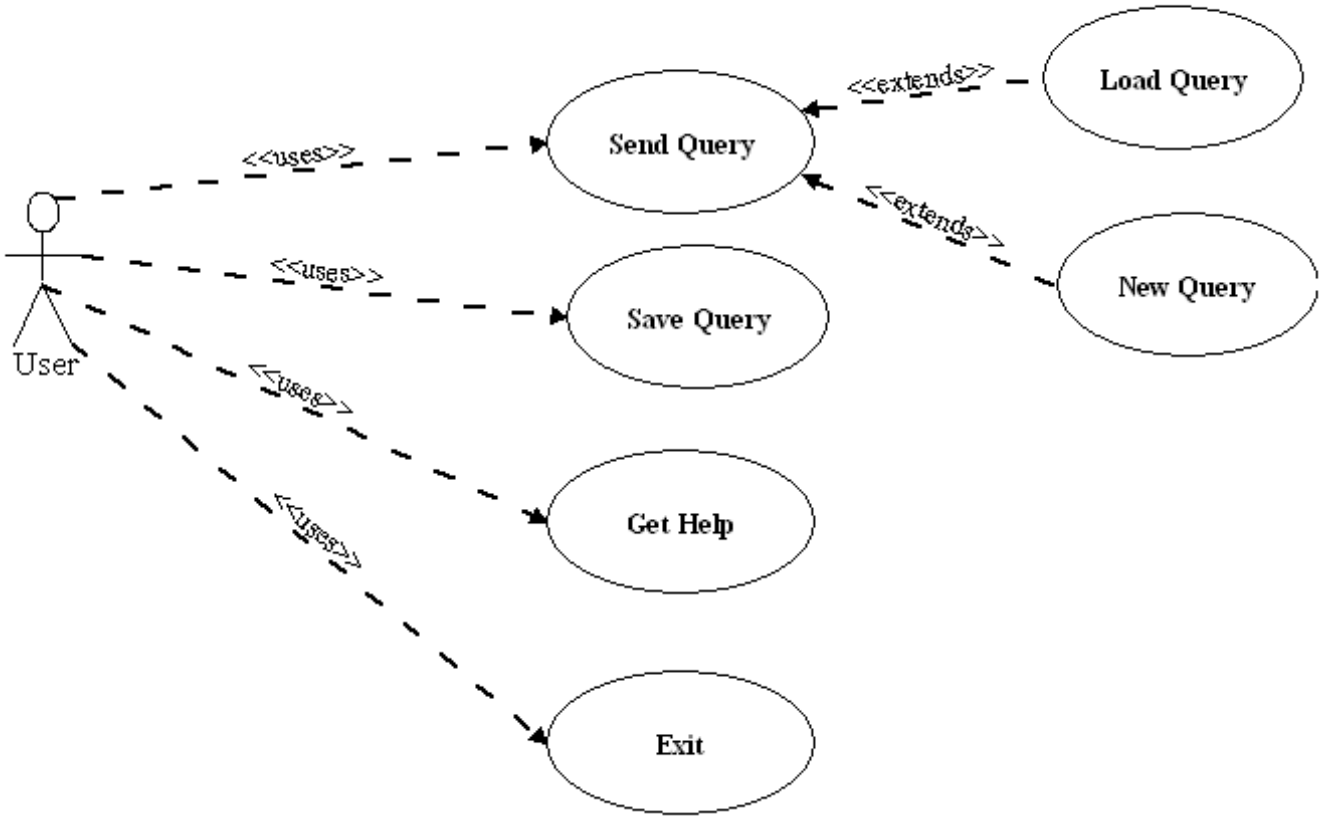
3.3. STATE TRANSITION DIAGRAMS

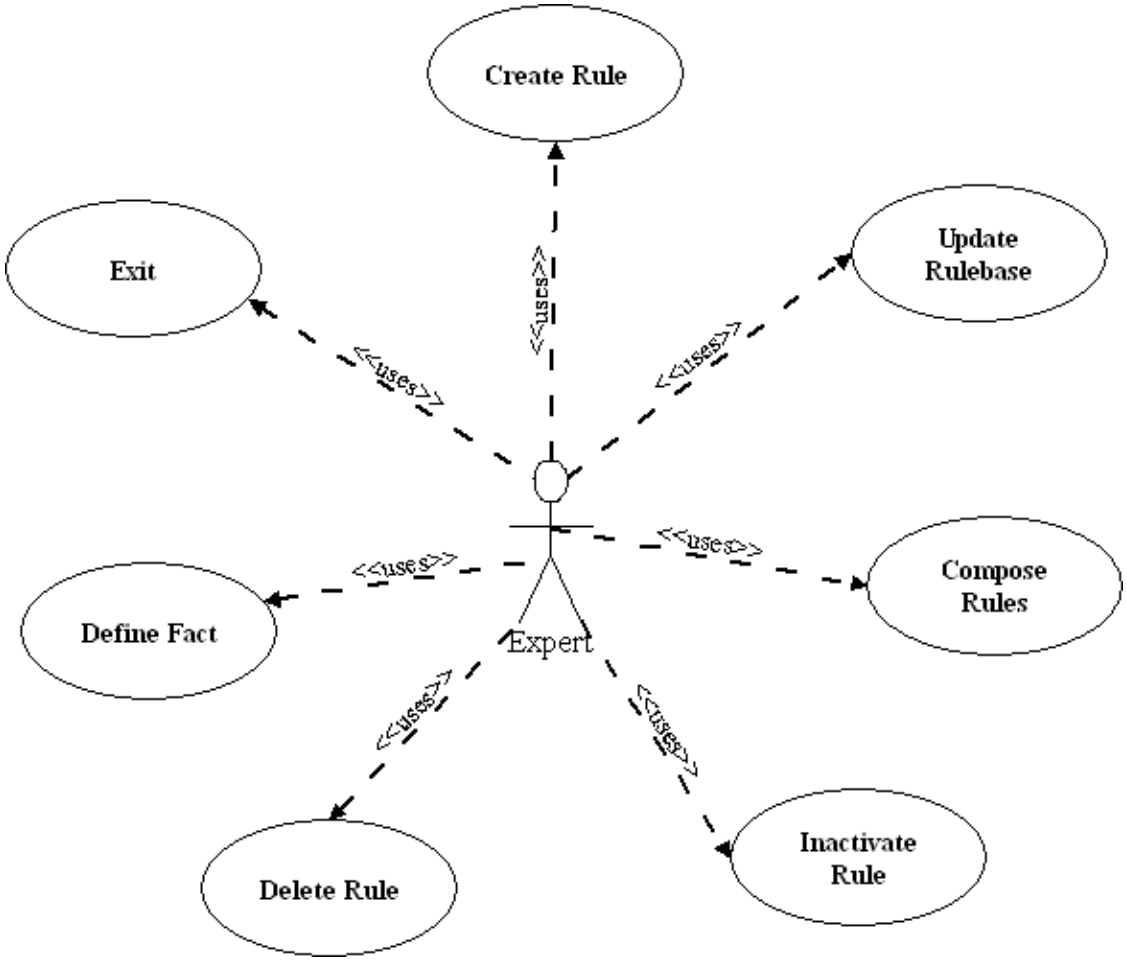



Idle
waiting for session
(update, add, delete etc.)
Processing
opening session
Idle
waiting for input
Verifying
verify input
Denying
Incorrect value
Processing
update rule repository
Checking
check consistency
Idle
waiting for next query

4. OO DIAGRAMS

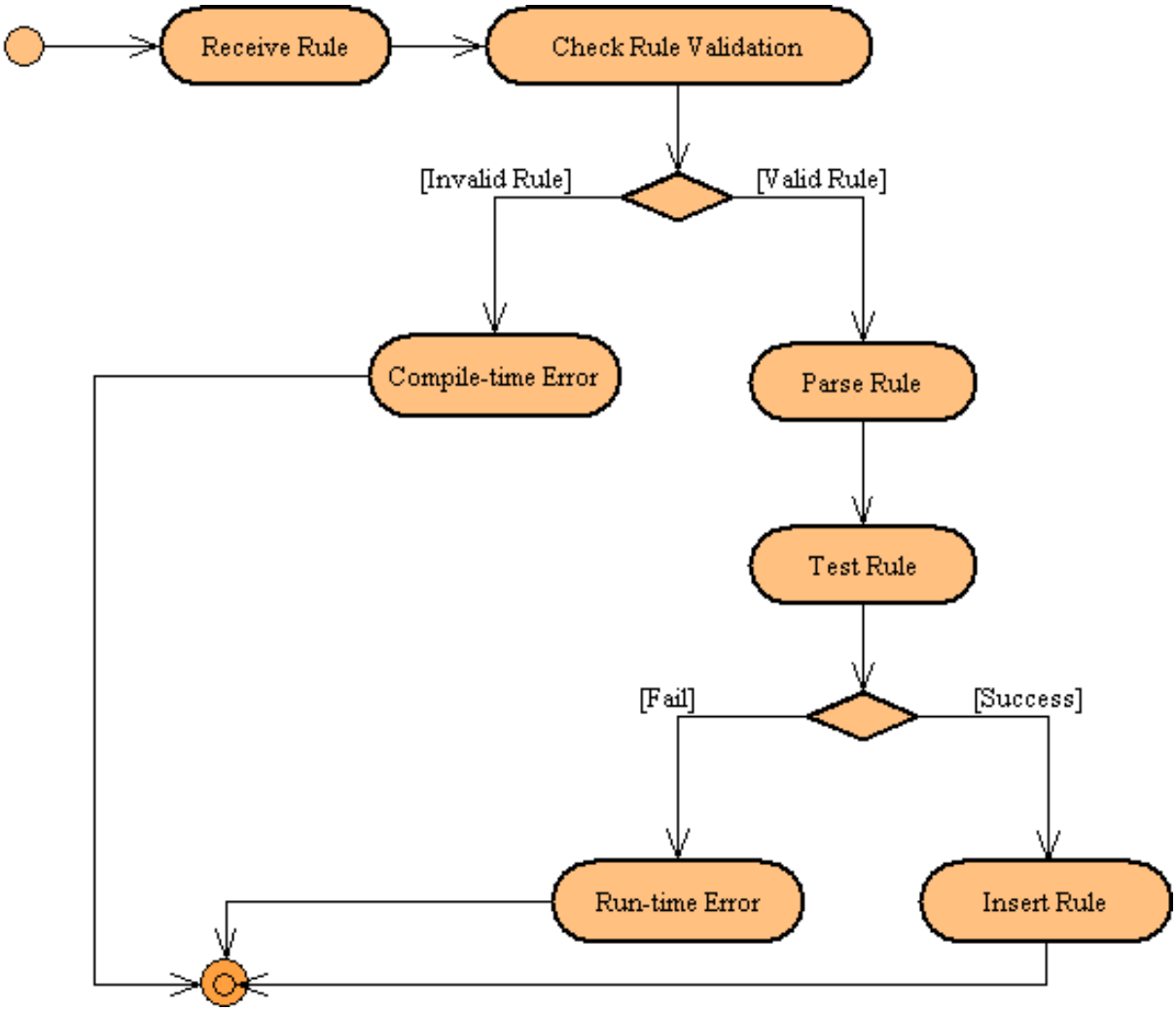
4.1. USE CASE DIAGRAM




Create Rule
Exit
Update Rulebase
<<uses>>
<<uses>>
<<uses>>
<<uses>>
<<uses>>
<<uses>>
<<uses>>
Define Fact
Compose Rules
Expert
Delete Rule
Inactivate Rule

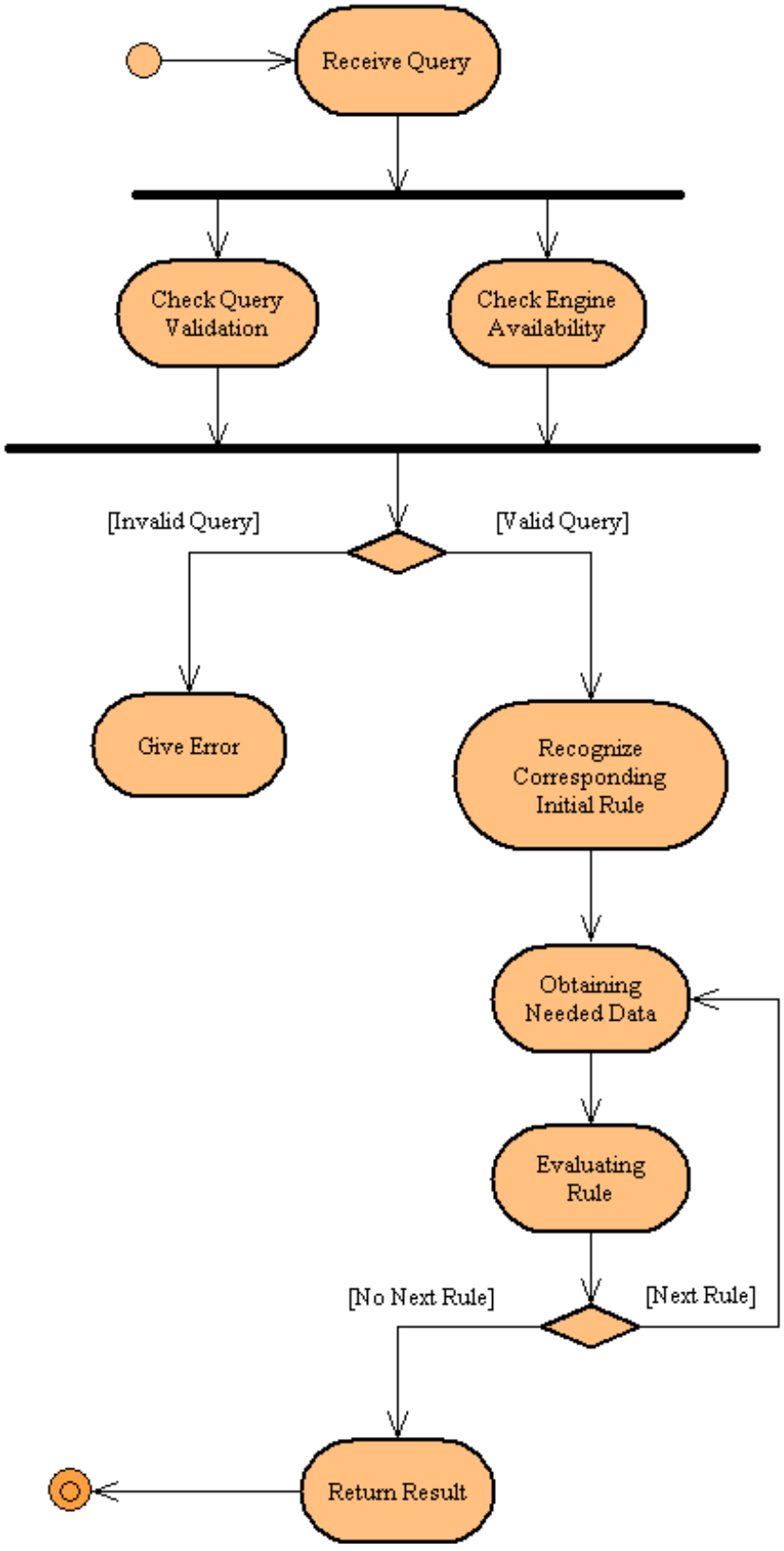
4.2. ACTIVITY DIAGRAMS

Expert Session



- Expert enters a rule or fact to the program. When rule is received it is checked for validation and if it is valid to be a rule program continue running, terminating (compile time error) otherwise.
- Program will parse the valid rule and test it in next step.
- If test is failed then program gives run time error if not continue.
- Then rule will be inserted and rulebase is organized accordingly.

User Session

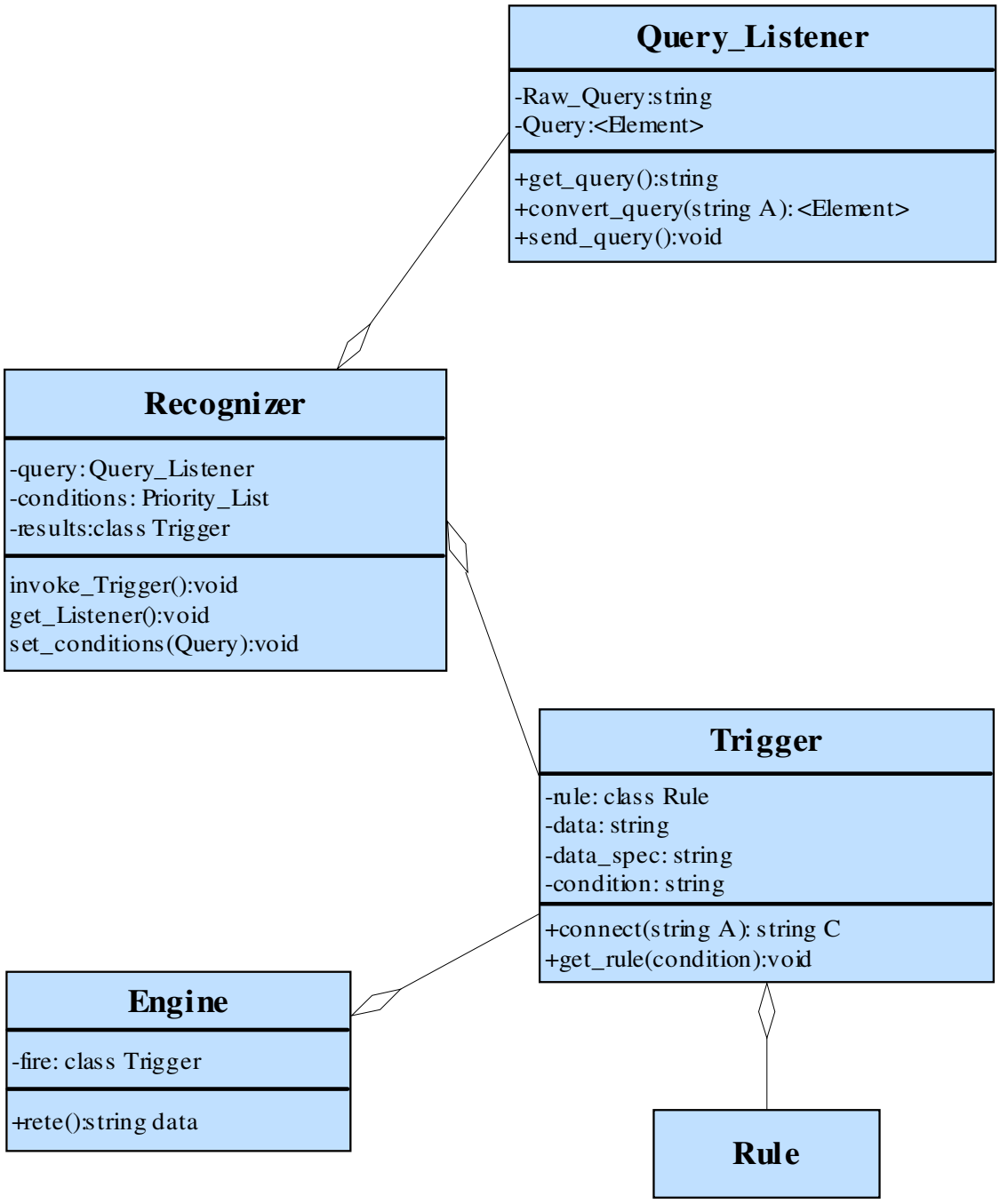


Users (or testers) may write a query (with our specifications) or load a saved query in our interface and get the result of the query visually as a table like SQL tables. Below is the description of the path that our program will follow.

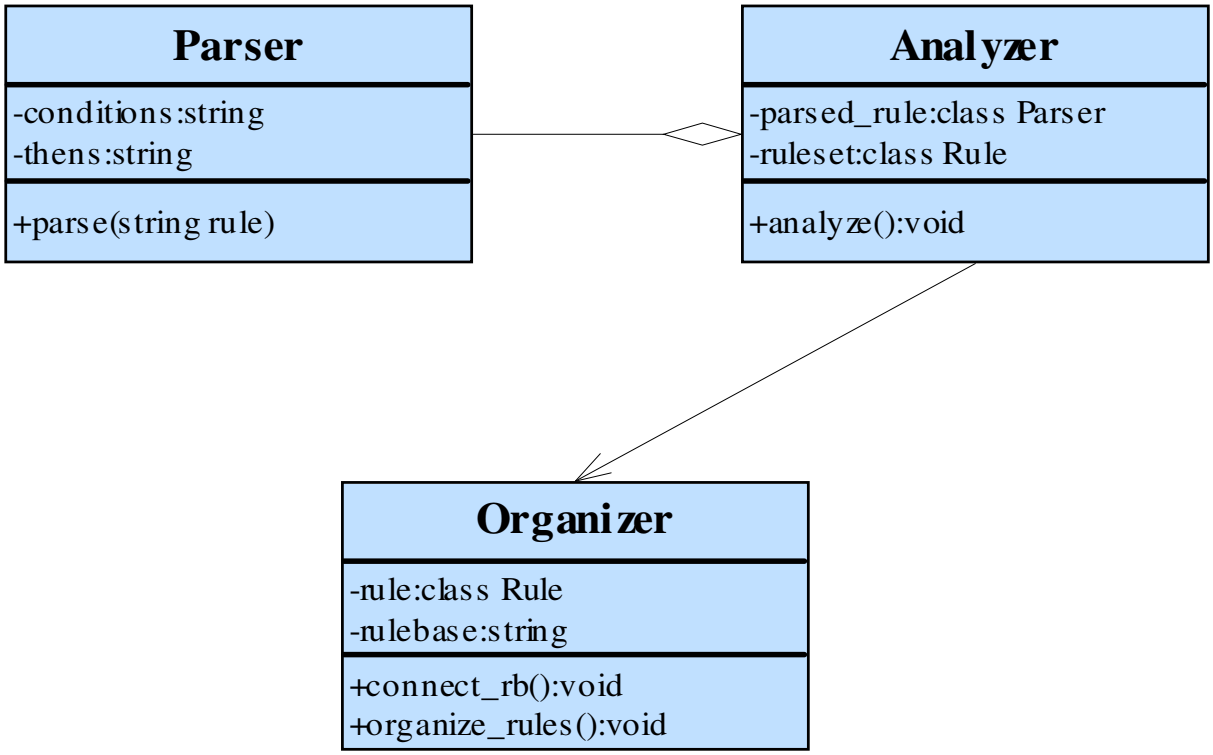
- When query entered the program will check the validity of the query and the engine. If it is a valid query and engine is available then program will continue to execute query otherwise it gives an error and terminate.
- Valid query will be recognized by the program and a rule or a ruleset will be obtained from rulebase to initiate the engine.
- After engine was initiated it starts to iterate until there are no rule will be executed.
- The output will be the result of the query.

4.3. CLASS DIAGRAMS

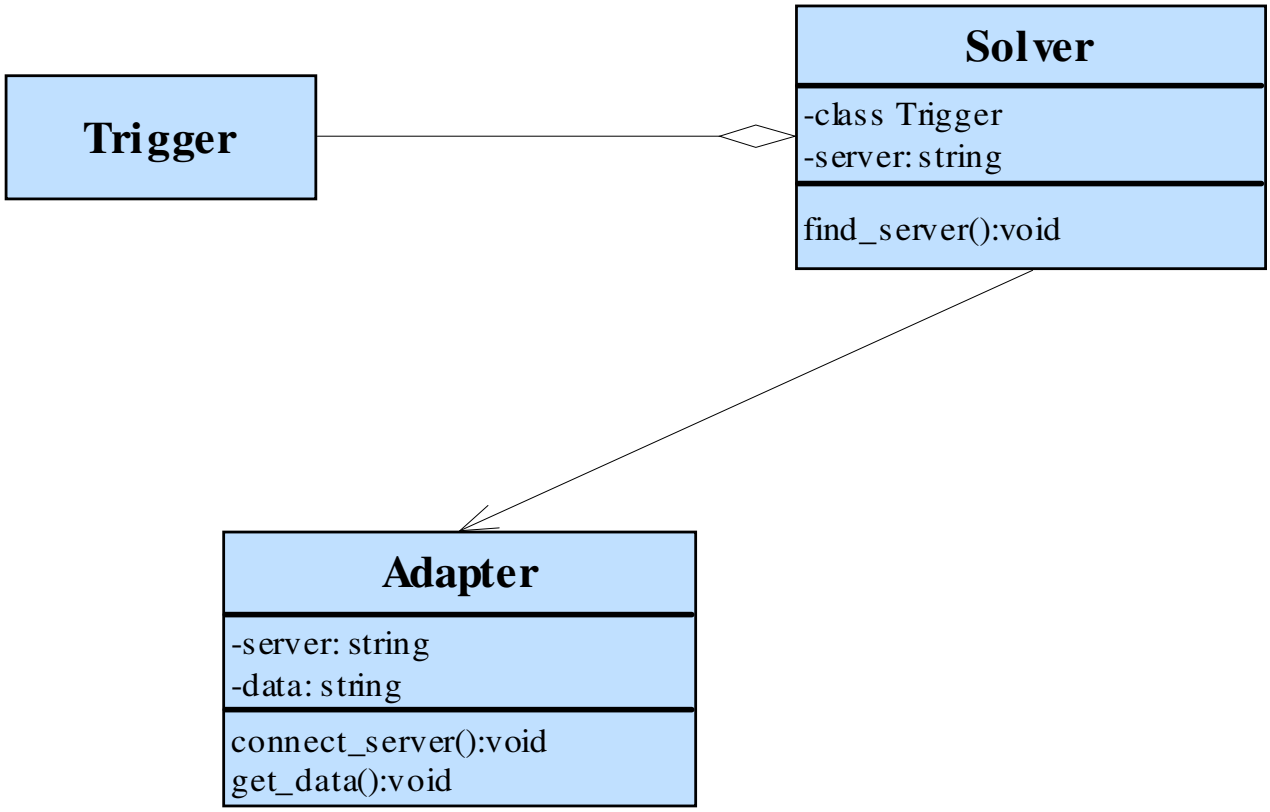
Execution Module



Management Module



Connector Module



5. GRAPHICAL USER INTERFACE DESIGN

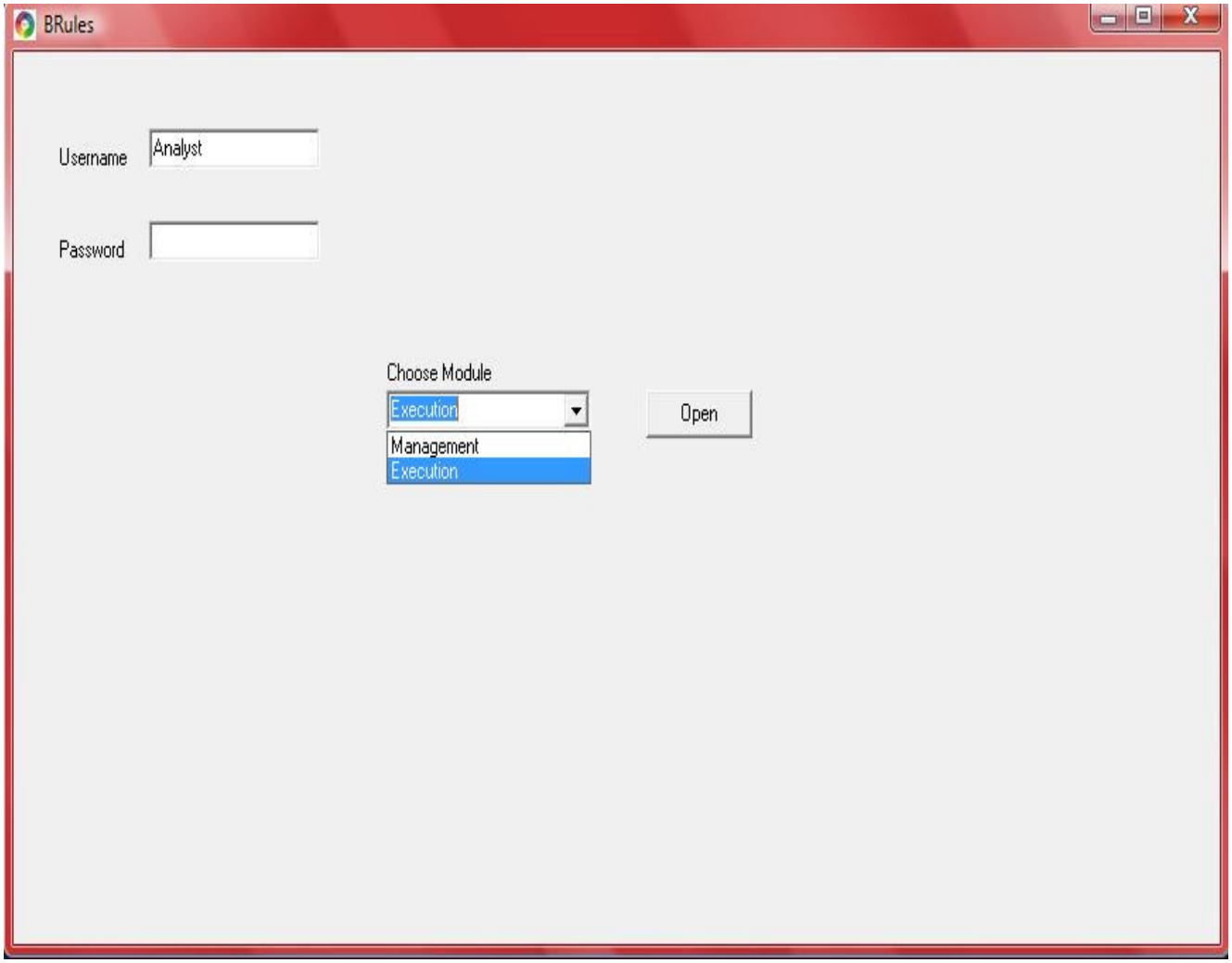


Figure 6.1- Login to Brules

According to authentication of user, the user can login to one of BRules's modules.

There are two modules (screens) can be opened according to the user's choose.

In general,in these two screens are Management Module which user can insert,delete,update,and create new rules and Execution Module which the user can request and be respondend a rule set as a table by the engine .

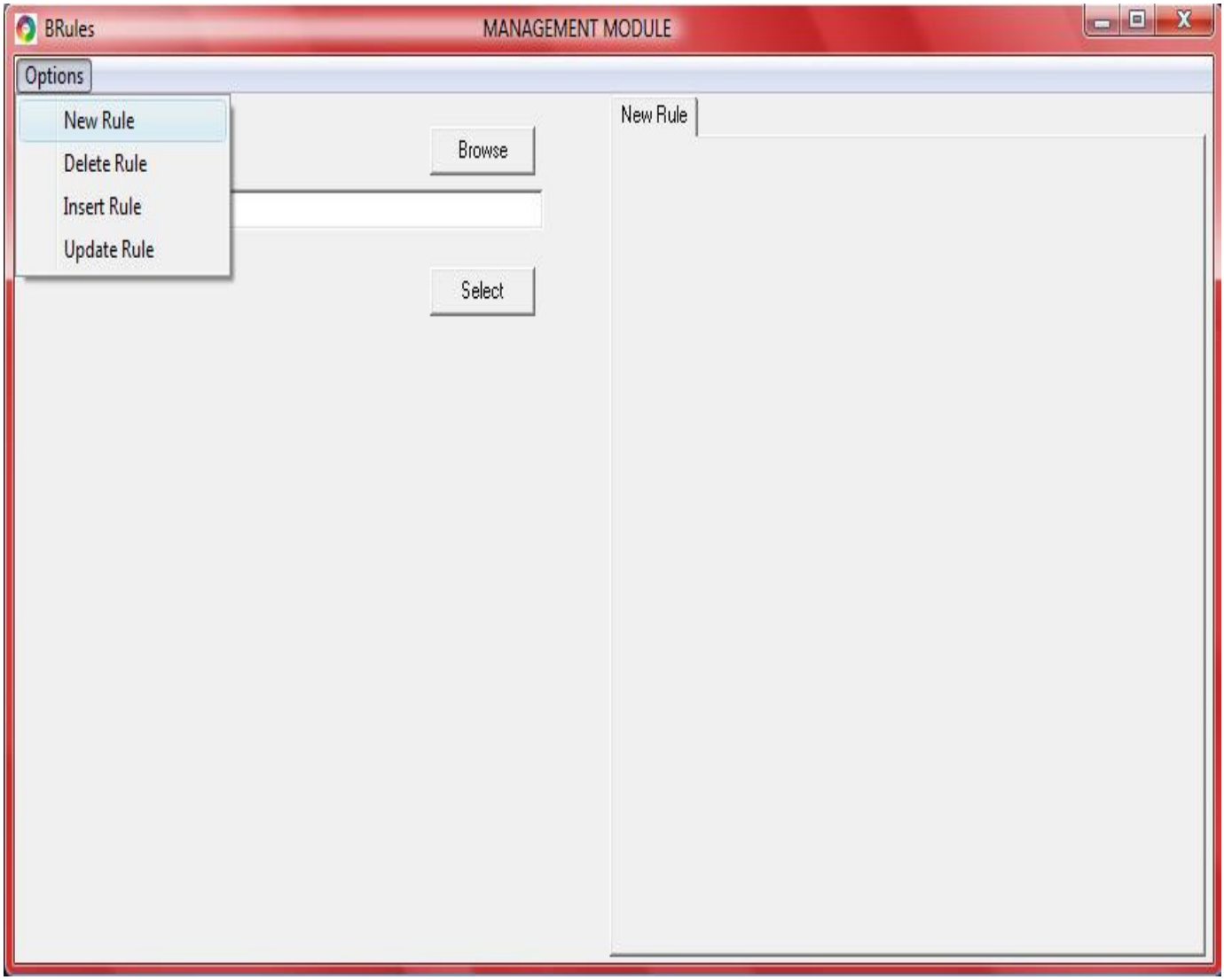


Figure 6.2- Create New Rule in Management Module of Brules

The program have four options ,namely , New Rule, Delete Rule, Insert Rule, and Update Rule.

At the above picture, when clicked on New Rule option , corresponding menu and a toolset menu will be opened. In the corresponding menu , users can browse rules and select them to compose them or users also can directly create a rule by using toolset.

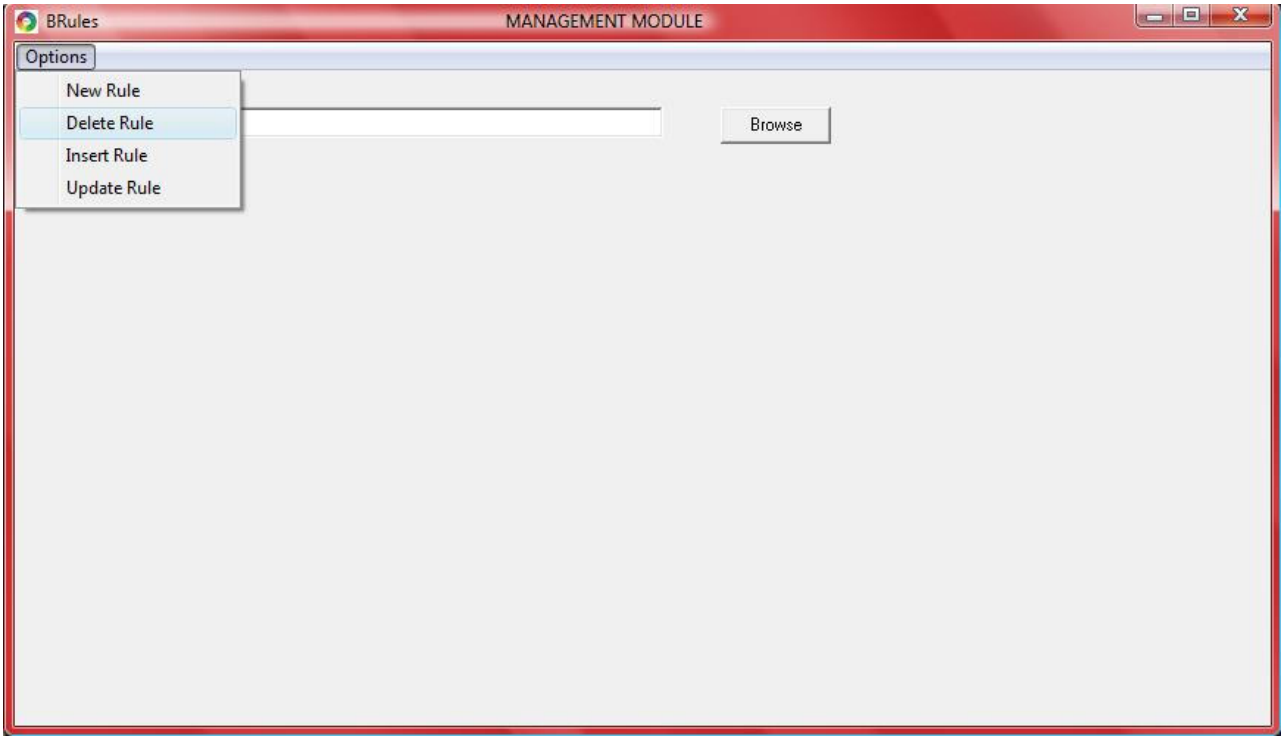


Figure 6.3- Delete Rule in Management Module of Brules

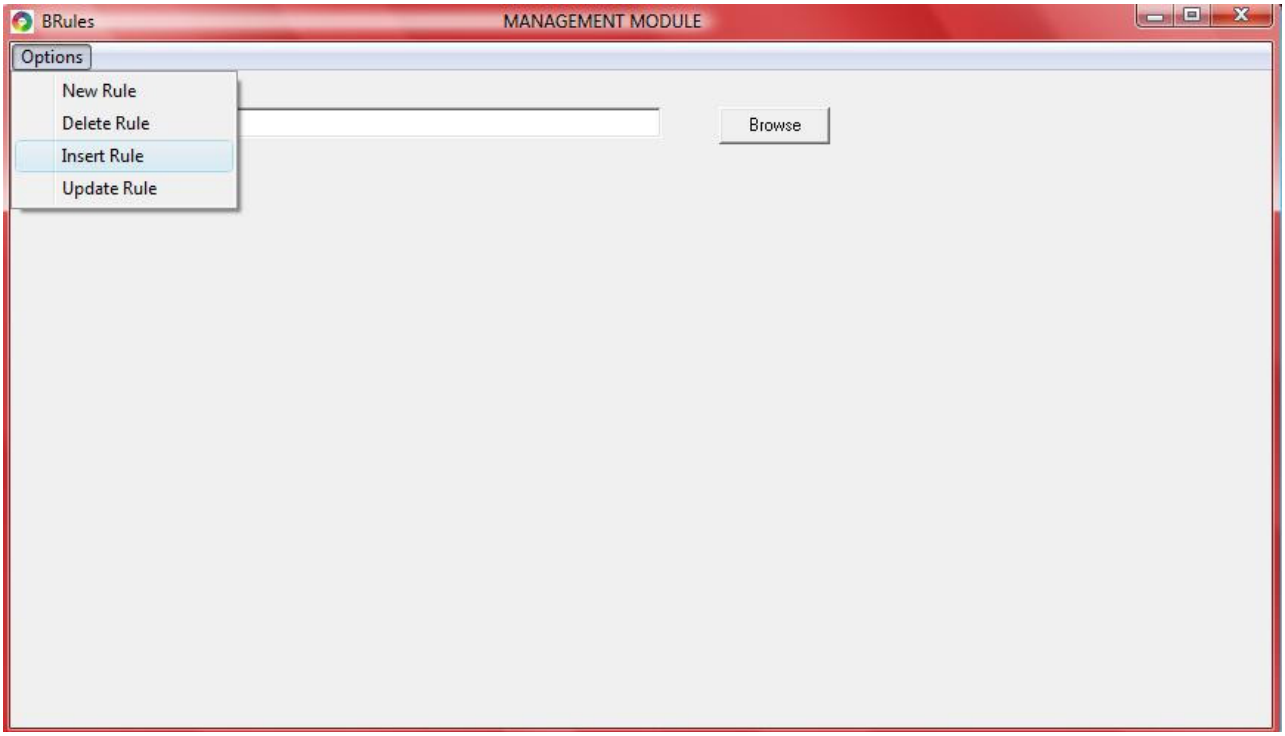


Figure 6.4- Insert Rule in Management Module of Brules

The above two pictures illustrate deleting a rule from Rule Base and inserting an existing rule to the Rule Base.

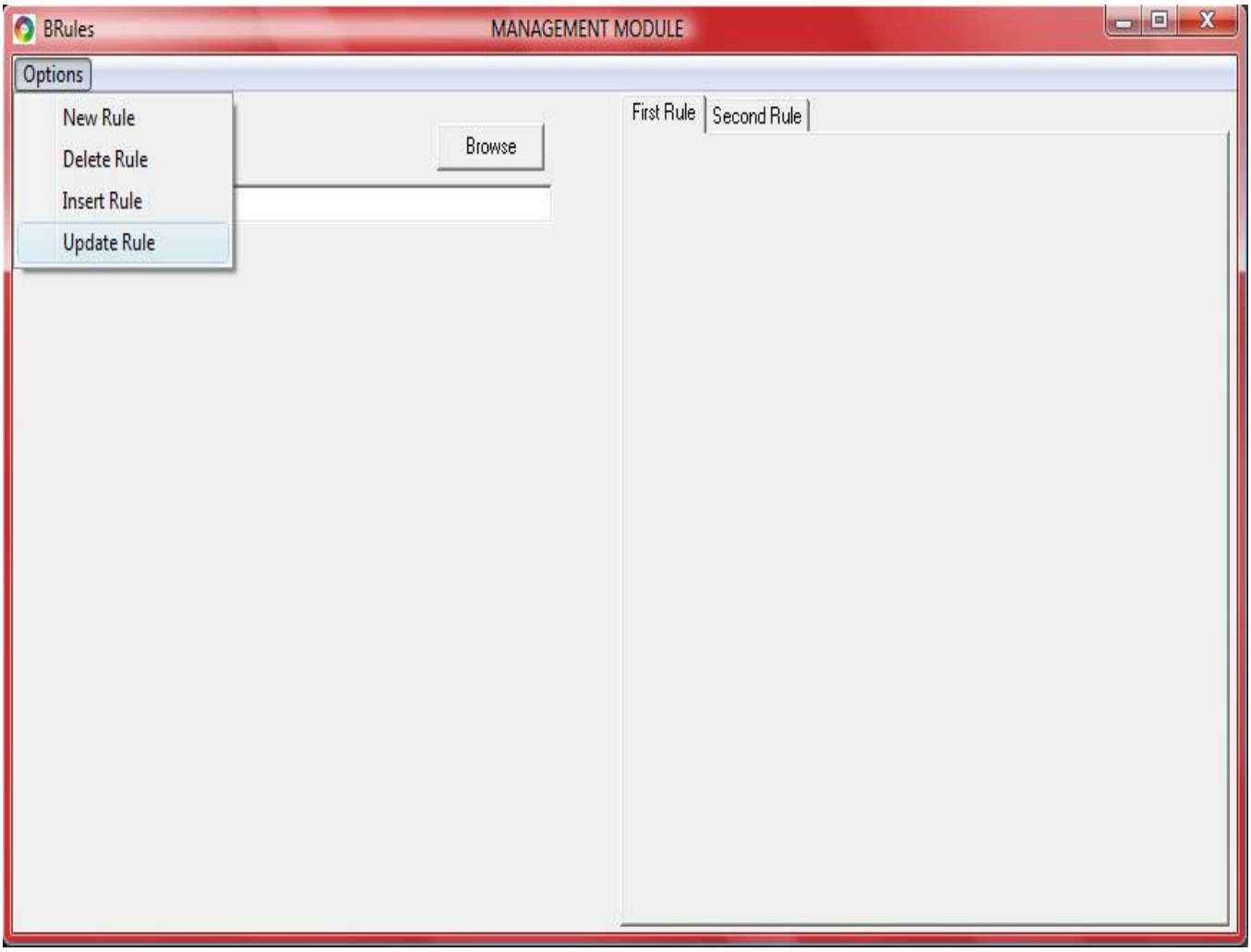


Figure 6.5- Update Rule in Management Module of Brules

Update Rule option in the Management Module make the user be capable of updating an existing rule in the Rule Base. After the detailed design these options and modules will be developed.

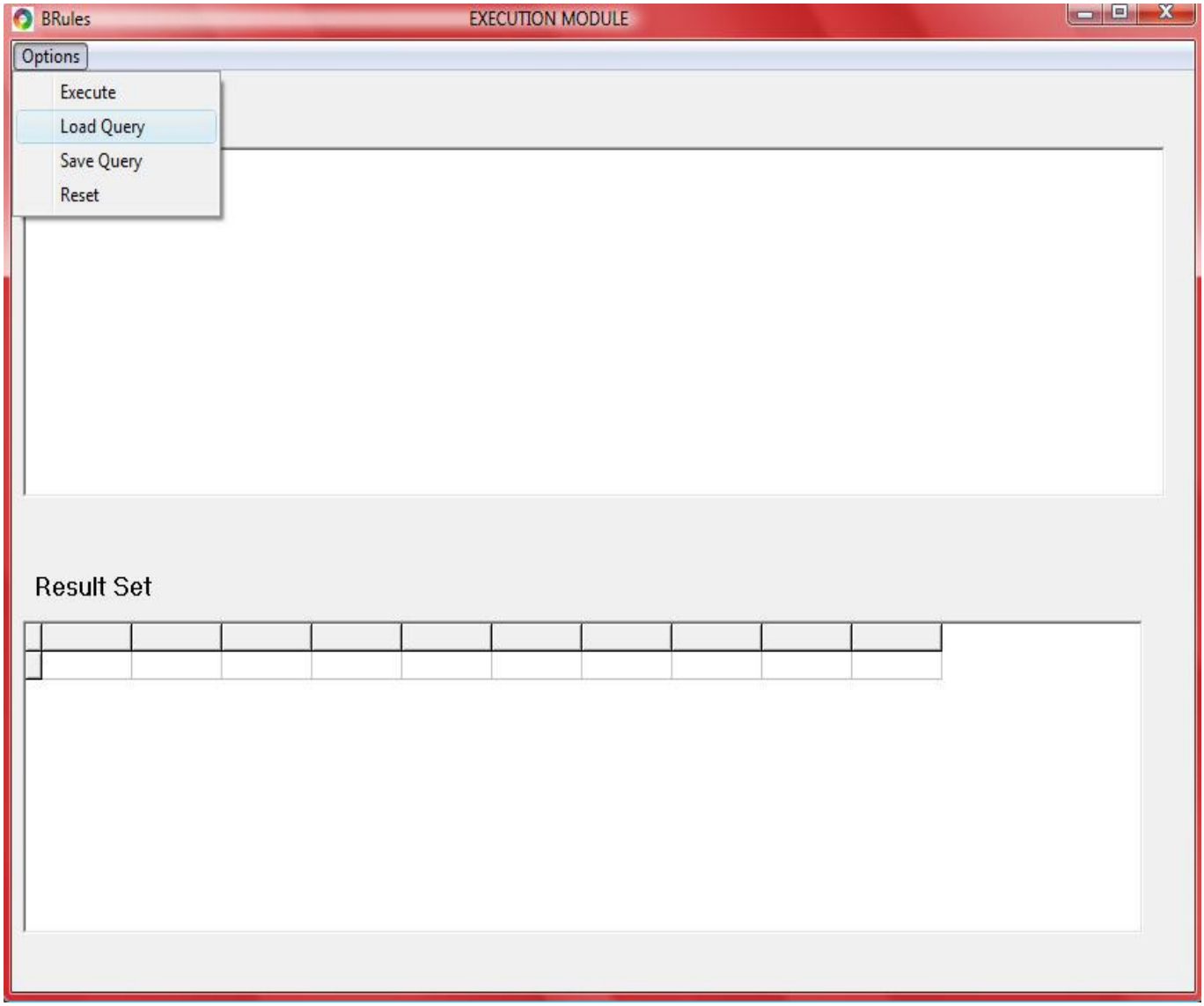


Figure 6.6- Execution Module of Brules

With Execution Module, users can test and make queries to get a result set as a table.

This module has three main part , namely, Options, Query, and Result Set.The users will write a query with our specific markup language ,MML to the Query part and get the result back at the Result Set part of Execution Module.

6. PROCESS MODEL

The most important elements that define a process are process model and team organization choices. They are explained below in detail.

6.1. TEAM STRUCTURE

The most proper team organization category for our group is *Democratic Decentralized* (DD).

Our reasons to choose DD are:

- Consensus plays an important role in our decision making process
- We have no permanent team leader
- Every member of the team should understand how things are handled
- Communication is horizontal
- We have coordinators for tasks, namely rotating task coordinators, but these coordinators may change as tasks change

6.2. PROCESS MODEL

After discussing among group members, we decided to use waterfall software model depending on the characteristics of our project. This software model was the most suitable one for our project since in this model software evolution proceeds through an orderly sequence of transitions from one phase to the next in order. Furthermore, time constraints had affects to choose this model since there is a strict schedule with deadlines that is given at the beginning of the project.

Waterfall model was best matched model with our case. No overlap or iteration is allowed during any period of the process. After a phase is completed we will not turn back for large scale modifications. This classic model has been widely characterized as both a poor descriptive and prescriptive model of how software development "in-the-small" or "in-the-large" can or should occur. Alternatively, the most criticized property of waterfall model is the absence of evolutionary approach and supplying

the product only after all the phases are over. But in the implementation phase we will provide several prototypes in order to get rid of the mentioned disadvantage.

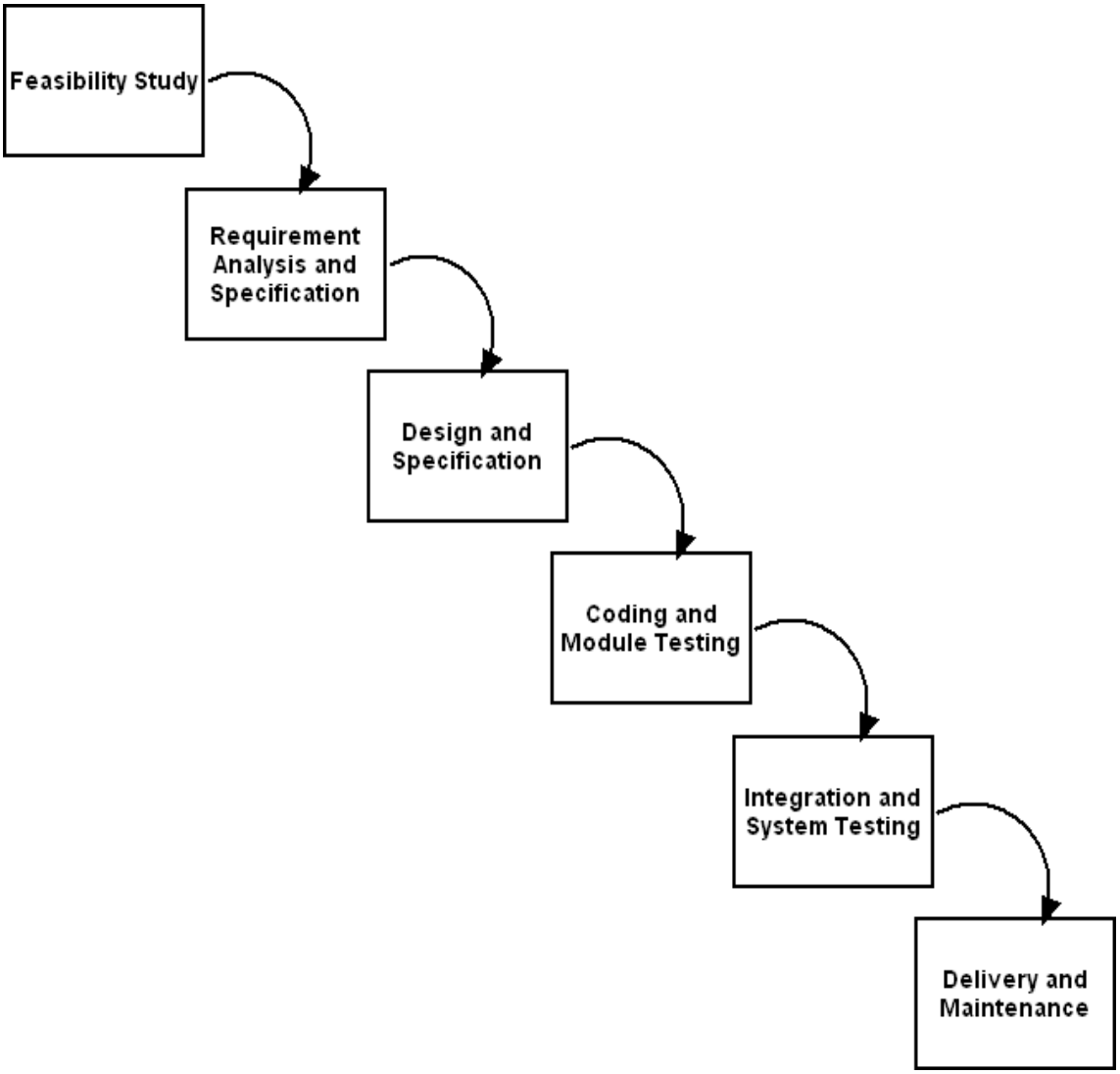


Figure 1 - Modified Waterfall Model

6.3. GANTT CHART

GANTT CHART

Number	Task	Start	End	Duration	9/15	9/22	9/29	10/6	10/13	10/20	10/27	11/3	11/10	11/17	11/24	12/1	12/8	12/15	12/22	12/29	1/5	1/12	1/19	1/26
1	GROUPING AND SELECTING TOPIC	9/19/2008	9/25/2008	6																				
2	PLANNING	9/23/2008	10/3/2008	9																				
3	TOPIC RESEARCH	10/1/2008	10/8/2008	7																				
4	PROPOSAL REPORT	10/3/2008	10/10/2008	7																				
5	INTERNET RESEARCH	10/5/2008	10/25/2008	20																				
6	CYBERSOFT MEETING	10/20/2008	10/21/2008	1																				
7	REQUIREMENT ANALYSIS REPORT	10/23/2008	11/16/2008	24																				
7.1	OUTLINE OF REPORT	10/23/2008	10/25/2008	2																				
7.2	MARKET RESEARCH	10/24/2008	10/27/2008	3																				
7.3	DETERMINING USER REQUIREMENTS	10/26/2008	11/3/2008	8																				
7.4	DETERMINING SYSTEM	11/1/2008	11/6/2008	5																				
7.5	DATA&BEHAVIORAL MODEL	11/3/2008	11/10/2008	7																				
7.6	FUNCTIONAL MODEL	11/9/2008	11/14/2008	5																				
7.7	LAST VERSION OF REPORT	11/13/2008	11/16/2008	3																				
8	CYBERSOFT MEETING	11/17/2008	11/18/2008	1																				
9	INITIAL DESIGN REPORT	11/18/2008	12/10/2008	22																				
9.1	OUTLINE OF REPORT	11/18/2008	11/19/2008	1																				
9.2	DEFINING CONSTRAIN	11/19/2008	11/25/2008	6																				
9.3	DESIGN OF CORE MODEL	11/23/2008	11/28/2008	5																				
9.4	DESIGN OF SUPPORTING MODELS	11/27/2008	12/2/2008	5																				
9.5	INITIAL GUI DESIGN	11/30/2008	12/7/2008	7																				
9.6	MODELING	12/5/2008	12/9/2008	4																				
9.7	UPDATE SCHEDULE	12/8/2008	12/9/2008	1																				
9.8	LAST VERSION OF REPORT	12/8/2008	12/10/2008	2																				
10	CYBERSOFT MEETING	12/9/2008	12/10/2008	1																				
11	INITIAL PROTOTYPE PREPARATION	12/8/2008	12/18/2008	10																				
12	FINAL DESIGN REPORT	12/15/2008	1/5/2009	21																				
12.1	OUTLINE OF REPORT	12/15/2008	12/17/2008	2																				
12.2	REVIEW OF MODULES	12/16/2008	12/23/2008	7																				
12.3	FINAL GUI DESIGN	12/21/2008	12/28/2008	7																				
12.4	FINAL DESIGN VERIFICATION	12/27/2008	1/1/2009	5																				
12.5	LAST VERSION OF REPORT	12/30/2008	1/5/2009	6																				
13	CYBERSOFT MEETING	1/5/2009	1/6/2009	1																				
14	PROTOTYPE PREPARATION	1/5/2009	1/20/2009	15																				
15	WEB SITE DESIGN	1/19/2009	1/24/2009	5																				
16	PRESENTATION	1/20/2009	1/30/2009	10																				

GANTT CHART

Number	Task	Start	End	Duration	2/16	2/23	3/2	3/9	3/16	3/23	3/30	4/6	4/13	4/20	4/27	5/4	5/11	5/18	5/25
1	IMPLEMENTATION OF SUBMODULES	2/16/2009	5/7/2009	80															
2	IMPLEMENTATION OF MANAGEMENT MODULE	2/16/2009	3/6/2009	18															
2.1	ALGORITHM SEARCH	2/16/2009	2/19/2009	3															
2.2	IMPLEMENTATION	2/19/2009	3/3/2009	12															
2.3	TESTING	3/1/2009	3/6/2009	5															
3	CYBERSOFT MEETING	3/6/2009	3/7/2009	1															
4	IMPLEMENTATION OF EXECUTION MODULE	3/6/2009	3/24/2009	18															
4.1	ALGORITHM SEARCH	3/6/2009	3/9/2009	3															
4.2	IMPLEMENTATION	3/9/2009	3/21/2009	12															
4.3	TESTING	3/19/2009	3/24/2009	5															
5	CYBERSOFT MEETING	3/24/2009	3/25/2009	1															
6	IMPLEMENTATION OF INTERACTION MODULE	3/24/2009	4/11/2009	18															
6.1	ALGORITHM SEARCH	3/24/2009	3/27/2009	3															
6.2	IMPLEMENTATION	3/27/2009	4/8/2009	12															
6.3	TESTING	4/6/2009	4/11/2009	5															
7	CYBERSOFT MEETING	4/11/2009	4/12/2009	1															
8	IMPLEMENTATION OF USER INTERFACE MODULE	4/11/2009	4/29/2009	18															
8.1	ALGORITHM SEARCH	4/11/2009	4/14/2009	3															
8.2	IMPLEMENTATION	4/14/2009	4/26/2009	12															
8.3	TESTING	4/24/2009	4/29/2009	5															
9	CYBERSOFT MEETING	4/29/2009	4/30/2009	1															
10	WHOLE SYSTEM TEST	4/30/2009	5/14/2009	14															
11	OPTIMIZATION	5/12/2009	5/22/2009	10															
12	PRESENTATION	5/20/2009	5/30/2009	10															

7. CONCLUSION

At the beginning of the project we didn't know anything about the project. After making so much research we improved our knowledge about the project and got idea about the case. From beginning to now we gained too much experience and improved our skills.

We had a great motivation for initial design report. After checking our early works, especially in requirement analysis report, we saw our missing parts. We fulfilled our lack of knowledge by doing so much research. Especially finding corresponding classes and methods was very hard and diagrams took most of our time. Keywords was also a big deal. Creating user interface also took much time. However, we believe that we overcomed all problems and did a good job.

We are much aware of th signifigance of initial design since it is the main step to detailed design. So we did our best while working on initial design and created this document. We changed some parts in our plans that we did before while preparing requirement analysis report. We know that this initial design report reflects our idea and knowledge about the project better than early works. Although having some small problems in some details of project, we are now confident about the project.

8. REFERENCES

RuleML Homepage, Realize your knowledge, September 2008.

URL:http://www.ruleml.org/

jDrew Homepage, A Java Deductive Reasoning Engine for the Web, January 2004

URL:http://www.jdrew.org/jDREWebsite/jDREW.html.

CLIPS: A Tool for Building Expert Systems, May 2008

URL:http://clipsrules.sourceforge.net/

Jess, the Rule Engine for the Java Platform, November 2008

URL:http://herzberg.ca.sandia.gov/

Business Rules Management System, Enterprise Decision Management: Visual Rules, 2008

URL: http://www.visual-rules.com/business-rules-management-enterprise-decision-management.html? utm_source=GoogleAd&utm_medium=PPC&utm_content=G_W-Klicker_BRMS&utm_campaign=G_W-Klicker&gclid=CLqe3IfJyJcCFUwb3godU0cUSA

Business Rules Engine, Microsoft Corporation, 2008

URL: http://msdn.microsoft.com/en-us/library/aa561216.aspx

IBM Education Assistant-WebSphere software, November 2008

URL: http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp? topic=/com.ibm.iea.wpi_v6/wpswid/6.0/BusinessRules.html

CodeProject: Externalize your business rules, May 2007

URL: http://www.codeproject.com/KB/dotnet/BRMS.aspx

Jboss Drools, How do Jboss projects work together? 2008

URL: http://downloads.jboss.com/drools/docs/4.0.3.15993.GA/html/index.html