

2009

CEng 492

DSK4BRMS

TEST SPECIFICATION REPORT



Ghassan ALSHANA

Yetkin KARIŞ

Metin BARIŞ

Erkan AKYOL

Bahattin YALCIN

MOCKWARE

Table of Contents

INTRODUCTION	3
Goals and Objectives	3
Scope of Document	3
Statement of Testing Plan Scope	3
Major Constraints.....	4
Time	4
Staff.....	4
Testing Plans and Strategies.....	5
Testing Plan.....	5
Testing Strategy.....	5
Testing User Interface Module	5
Testing Engine	6
Testing Connector Module	6
Testing Procedure	7
Unit Testing.....	7
Integration Testing	7
Validation Testing	7
TESTING TOOLS AND ENVIRONMENTS.....	8
NetBeans.....	8
Testing Staff	8
Testing Schedule	8

INTRODUCTION

This section gives a general overview of the test specification for Domain Specific Kit Implementation for Business Rules Management project.

Goals and Objectives

Domain Specific Kit Implementation for Business Rules Management project has several modules including user interface, manager module, executor module, connector module and database modules. We are assured that processing and duties of each module should be verified so that all of these modules work in harmony and correctly when they are integrated. As a result, we give great importance to testing our project in order to obtain a bug free, and logically correct product with high performance.

Scope of Document

The purpose of this document is to describe the testing process of our project. In fact, while developing our project up to now, we did testing for each module, so we will explain about the testing process that took place since the beginning of the project until now.

Statement of Testing Plan Scope

Testing process of our project includes unit testing, integration testing, validation testing, performance testing and alpha- beta testing.

Unit testing

In our project we implemented unit testing for the modules of user interface and database.

Integration Testing

This is the most important part of our testing because getting a whole program that works correctly is the aim of this project. We have to test all the units of the project as a whole and test for any collision between them

Validation Testing

We did validation testing in two categories: requirements validation and design validation.

Performance testing

In this part we tested the accuracy of results and the speed of program.

Alpha-Beta testing

Whenever a new feature was added to the project someone in our group made alpha testing so we could be sure that the added parts work correctly. After all of the parts were added all group members made beta testing for being sure that our project was bug-free.

Major Constraints

In this part, we have divided that topic into two constraints which are very important and will affect us during testing our project. These constraints are: Time constraints; since we had to finish the final package on time while we were testing our software. The second one was the number of staff. We had 5 group members and it made difficult to do all tests in time. Since our project could work on a standard computer we had no hardware constraints.

Time

Since our project should have been finished due to 15 June, time was the main constraint for us. We had a schedule for the testing phase, therefore the deadlines of each specific work were known beforehand. If we managed to obey the deadlines according to schedule, our project was released after proper testing and in time.

Staff

As I mentioned above, the number of staff was a major constraint for our testing plan since there were only five people, who had also other responsibilities in our project, as a staff.

Testing Plans and Strategies

Testing Plan

In our testing process, our aim is to find as many bugs and design errors as possible, as early as possible in the development lifecycle. In order to do so, we needed to plan the testing beforehand, and make sure we have different test paths for different tasks. Our testing plan extends to both endpoints in a testing environment: from unit tests to scenario tests.

Testing Strategy

Our project consisted of three main testing modules. Those modules are:

- ✓ Testing User Interface Module.
- ✓ Testing Engine.
- ✓ Testing Connector Module.

Testing User Interface Module

In our Graphical User Interface (GUI), when we open the program, we have four interface screens namely; Login screen, Management screen, Request screen and Execution screen.

In the Login screen, the users can open the module according to their permission. While in Management screen, it has one user namely, IT manager who creates new rule and ruleset, updates, deletes rules and rulesets. We have three tabs in this screen namely; Rule Manager, Fact Manager and Requests.

In the request screen, the user namely; Business Analyst can request creating new rule and ruleset, updating, deleting rules and rulesets from IT Manager. We have two tabs in this screen namely; Rule Requests and Fact Requests.

In the Execution screen, the regular users can execute queries and search for some information. In this screen, we have text box for entering SQL queries and a button to execute this written query.

Testing Engine

our engine is responsible for executing queries by using rulestes and factsets in its database. In other words, the engine will do reasoning. We will test our engine by using execution module. When the user enters a query and executes it at the background, the engine will work on this query and find the result by applying reasoning.

Testing Connector Module

Since we must connect to external sources to get the required facts, the connector module is an important part of the program. There are three different types of external sources, APIs, External Databases and web services. After finishing our user interface part, we will work on testing connector module.

Testing Procedure

Unit Testing

Unit testing is the testing of the each individual components of the project separately to see whether application will suffer from errors arising from these components. This testing should be handled before the integration testing. It should be certain that errors arising in integration testing are not about individual components. Moreover, there will be one scenario to test each component. It is a white-box type of testing which is every branch in the code is taken and every combination of input values is tried. For example, in our project we implemented unit testing for the modules of user interface and database.

Integration Testing

This is the most important part of our testing because getting a whole program that works correctly is the aim of this project. We are going to test all the units of the project as a whole and test for any collision between them. Also, we will test all the expected features of the system such as related data of the users of the program is managed, user interface is prepared for easy use, and the module functionalities work properly. Moreover, we will carry out the integration test from the perspective of the possible users of our program. Finally, we will look for if all the modules are integrated successfully and the functionalities are interacting properly.

Validation Testing

Validation test asks for if the product is going in the right way or not and the answer specifies whether our program will be preferred by the form users or not. Therefore validation is important. So, we performed a black box testing too in order to specify all the needed

requirements and obtained possible errors and solve them. Also, we did validation testing in two categories: requirements validation and design validation.

TESTING TOOLS AND ENVIRONMENTS

NetBeans

At the beginning of the semester, we were using eclipse but after that we switched to NetBeans after we noticed the needs for preparing a graphical user interface and debugging it. Although most team members switched to NetBeans, we sometimes still use Eclipse, especially when working on department computers.

Testing Staff

Yetkin KARIŞ : Data Engine, Integration Testing

Metin BARIŞ : Testing user interface

Erkan AKYOL : Integration Testing Data Engine, Query Engine

Ghassan ALSHANA: Testing user interface, Unit Testing.

Bahattin YALCIN: Testing database modules.

Testing Schedule

The following is the schedule for the testing plan that was presented in this report:

Test Name	Deadline
Test Plan Delivery	28.05.2009
Unit Test and Integration Tests	01.06.2009
Performance Test	06.06.2009
Alpha and Beta Tests	13.06.2009
Tracing and Correction	15.06.2009

