ÉCOLE
POLYTECHNIQUE
MONTRÉAL

Le génie sans frontières

Time Monitoring Tool

Test Plan

Version 3.0

Revision History

Date	Version	Description	Author
15/01/2001	1.0	First version	Sandra Lee
26/01/2001	2.0	Revised majors points with Lab attendant	Sandra Lee
01/04/2001	3.0	Final revision and addition of the Timestamps Validation functionality	James Prevost

Table of Contents

Test Plan

1. Introduction

1.1 Purpose

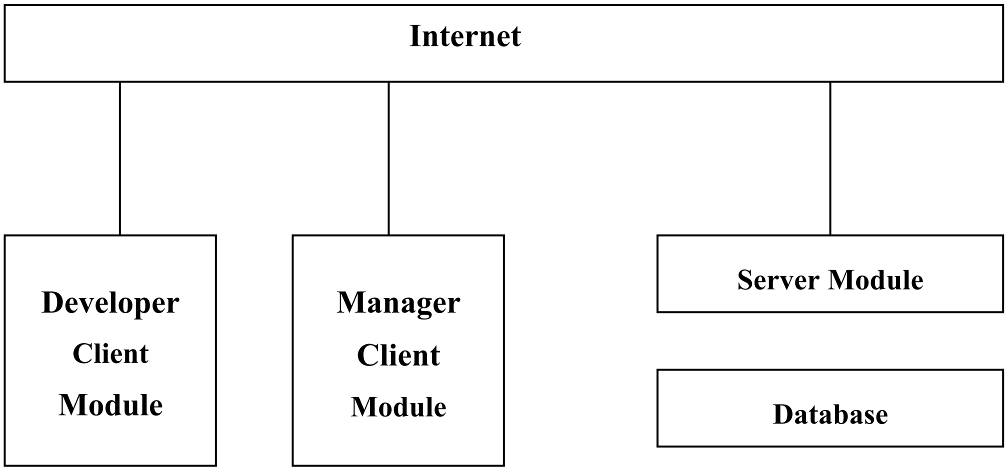
This Test Plan document for the *Time Monitoring Tool* supports the following objectives:

1. Identify existing project information and the software components that should be tested
2. List the recommended Requirements for Test (high level)
3. Recommend and describe the testing strategies to be employed
4. Identify the required resources and provide an estimate of the test efforts
5. List the deliverable elements of the test project

1.2 Background

The Time Monitoring Tool allows developers working within a defined software development process to record the time spent on the various activities, in a database. The TMT will also allow a manager to derive analyses and produce reports based on the data entered in the system.

The TMT system is a stand-alone tool that is integrated within the organization's Intranet. It consists in four major components: a Developer Client Module, a Server Module, a Database, and a Manager Client Module.



The Developer Client Module allows developers to log onto the TMT system and to record timestamps corresponding to their activities in a convenient way.

The Manager Client Module allows a manager to retrieve timestamp information from the database to produce analyses and reports. The Developer and Manager Client Modules must provide a user interface that is available through a WWW browser.

The Server Module is a daemon accepting connections from Developer and Manager Client Modules and serves as an interface between these modules and a database. The Server Module does not have a user interface other than a command to launch it.

The Database component can be any type of database. It does not have to be developed as a part of the TMT system, as long as the Server Module can interface with an available database system. The Database module does not have a user interface.

1.3 Scope

This Test Plan applies to the unit and integration tests that will be conducted on the Time Monitoring Tool System Release 1.

Unit tests will address functional quality, while system testing will address issues of performance.

The following systems interfaces will be tested:

- **Developer Client Module Interface**
- **Manager Client Module Interface**

1.4 Project Identification

The table below identifies the documentation and availability, used for developing the test plan:

Document (and version / date)	Created or Available	Received or Reviewed	Author or Resource	Notes
Requirements Specification	■ Yes □ No	■ Yes □ No		
Use Case Reports	■ Yes □ No	■ Yes □ No		
Design Specifications	■ Yes □ No	■ Yes □ No		
Prototype	■ Yes □ No	■ Yes □ No		
Users Manuals	■ Yes □ No	■ Yes □ No		
Project Plan	■ Yes □ No	■ Yes □ No		

2. Requirements for Test

The listing below identifies those items (use cases, functional requirements, and non-functional requirements) that have been identified as targets for testing. This list represents what will be tested. All interfaces actions; database accesses and internal calculus are also listed.

Functionality	Interface	Database	Calculus
1.1 The user shall be able to load the Developer Client Module within Netscape.	The DCM is loaded upon a Developer's validated login.	Access the *Users* table (SELECT) and verify submitted data (username and password)	None
1.2 The Developer Client Module shall support the logging of users.			
1.2.1 The initial window of the DCM shall contain a field for a user name, a field for a password, and a button labeled login. The password field shall be a "secret" field, which does not display what the user types.	Auto-verification and validation of the submitted data and ensure that the password is visually encrypted	None	None
1.2.2 When a user presses the login button, the DCM shall send a request to the SM to login the user.	Upon clicking the *LOGIN* button, the DCM sends the right login request to the SM	Access the *Users* table (SELECT) and verify submitted data (username and password)	None
1.3 If the logging of a user is successful (see 1.2.2), the DCM shall display the Time Monitoring Window.	The DCW (Developer Client Window) of the TMT loads.	None	None
1.3.1 The Time Monitoring Window shall always display the identifier for the current week.	Ensure that there is a Date Identifier in the DCM window header.	None	Get the Current date from the server
1.3.2 The Time Monitoring Window shall always display the username of the user currently logged in.	Ensure that there is a Username Identifier in the DCM window header.	Get the right username from the data validated while processing the login request (SELECT)	None
1.3.3 The Time Monitoring Window shall display all the validated and non-validated time stamp records previously entered for the current week by the user currently logged in.	Ensure that the DCM shows correctly all timestamps (all validated and non-validated timestamps of the current week)	A database SELECT query retrieves all validated and non-validated timestamps of the current week.	Verify the date's day and get the current week's boundaries in terms of dates. Sends this date with the database query as a filter.
1.3.3.1 The validated and non-validated time stamp records shall be displayed	Ensure that all data is shown in a dynamic HTML table having one timestamp per row.	None	None

in a tabular fashion, with every record being on a separate row.			
1.3.3.2 The fields should be identified with labels. Intuitive and non-confusing abbreviations can be used if necessary.	Ensure that every column's title is appropriate to the data contained within that column.	None	None
1.3.3.3 The fields should be non modifiable to prevent modification of the timestamp records.	Ensure that all non-modifiable fields are preventing the user from any type of edition through his internet browser.	None	None
1.4 The existing Time Monitoring Window timestamps shall be updateable.	Ensure that all modifiable fields allow the user to update the timestamps through his internet browser.	None	None
1.4.1 It shall only be possible to modify the Project, Task, Activity, and Artifacts fields with values that are allowed for the user currently logged in.	Ensure that all available projects (Shown and listed in the DCW) are the user-specific projects.	A database SELECT query retrieves from the *Users* table all active projects from the selected user.	None
1.4.2 It shall only be possible to modify the time fields for every day of the week using a positive numerical value of maximum 24.	Ensure that all dates all editable and shown in the 24:00 format.	None	None
1.4.3 Modifications to the records shall not be synchronized automatically with the server, that is, it should be possible to modify several values and then to click a button or perform another mechanism to update the server.	All modifications must be saved when a user submits the timestamps forms (new timestamps or corrected timestamps)	INSERT and UPDATE queries to the database in the *Timestamps* table to save all modifications	Ensure that all data has been modified.
1.5 The Time Monitoring Window shall support the entry of new timestamps.	The user must be able to add new timestamps.	None	None
1.5.1 It shall be possible to add new timestamps according to predefined fields.	Ensure that upon adding a new timestamp, all fields have a default value.	A SELECT query in the Users table ensures that the default values for this user are shown.	None
1.5.2 It shall be possible to create up to 100 new timestamps for a given user	Ensure that at least 100 timestamps can be added to the database within the same week for a given user.	None	None

and a given current week.			
1.5.3 All new timestamps should come with the predefined selections for a given user.	Ensure that upon adding a new timestamp, all fields have a default value.	A SELECT query in the Users table ensures that the default values for this user are shown.	None
1.5.4 All time entry shall be in hours and minutes H:mn.	All time fields must be shown in the HH:mm format. The interface scripts ensure that more complex formats (e.g.: retrieved from the database) are downsized to this format.	None	None
1.5.5 All empty time fields shall be at 0.	By default all time fields in the forms are set to the 00:00 default value	None	None
1.6 The Time Monitoring Window shall support entry of timestamps.	The DCW interface shows an "Add new timestamps" form which is operational.	None	None
1.6.1 All fields of a timestamp shall have predefined values for the logged in user.	Ensure that upon adding a new timestamp, all fields have a default value.	A SELECT query in the Users table ensures that the default values for this user are shown.	None
1.6.2 Project, Task, Activity, Artifact and at least one time field must be filled out before the record is sent to the database.	The DCW interface ensures that the Project field, the Task field, the Activity field and the Time field must be filled before submitting the form	None	Internal scripting to the interface verifies the non-filled fields and generates error messages, if needed.
1.6.3 Time entry could be in duration or taxi mode.	The interface allows the Taxi mode activation and the simple duration calculus.	None	When taxi mode is activated, Internal scripting sets a timer until the Timestamp's End Time is entered. Then the timer is set off.
1.6.4 Time Monitoring Window shall provide a running clock for the taxi mode.	Ensure that the Taxi mode timer is visible when this mode is selected.	None	Simple timer calculus: current time minus start time. Always refreshes the timer display.

1.7 The Time Monitoring Window shall provide cumulative totals.	**Ensure that the statistical module is showing all available totals (by column, by day, by project)**	**A SELECT query retrieves the current week's timestamps in the *Timestamps* table**	**A sum of all available timestamps data**
1.7.1 Each column day of the current week shall have the total number of hours recorded.	**Ensure that every day of the current week has its total number of hours shown**	**A SELECT query retrieves the current week's timestamps in the *Timestamps* table**	**A sum of every day's timestamps data**
1.7.2 Each project and task line of the current week shall have the total number of hours recorded.	**Ensure that every task and project of the current week have their total number of hours shown**	**A SELECT query retrieves the current week's timestamps in the *Timestamps* table**	**Horizontal sum**

Functionality	Interface	Database	Calculus
2.1 The manager shall be able to load the Manager Client Module within Netscape.	**The MCM is loaded upon an Administrator's validated login.**	**Access the *Users* table (SELECT) and verify submitted data (username and password)**	**None**
2.2 The Manager Client Module shall support the logging of managers.			
2.2.1 The initial window of the MCM shall contain a field for a user name, a field for a password, and a button labeled login. The password field shall be a "secret" field, which does not display what the user types.	**Auto-verification and validation of the submitted data and ensure that the password is visually encrypted**	**None**	**None**
2.2.2 When a user presses the login button, the MCM shall send a request to the SM to login the user.	**Upon clicking the *LOGIN* button, the MCM sends the right login request to the SM**	**Access the *Users* table (SELECT) and verify submitted data (username and password)**	**None**
If the logging of a user is successful (see 2.2.2), the MCM shall display the Manager Client Window.	**The MCW (Manager Client Window) of the TMT loads.**	**None**	**None**
2.3.1 The Manager Client Window shall always display the username of the manager currently logged in.	**Ensure that there is a Date Identifier in the MCM window header.**	**None**	**Get the Current date from the server**

2.3.2 The Manager Client Window shall display two groups of icons, which are for the user management and the project management.	Ensure that both menus are shown and that all icons and options are present for each menu.	None	None
2.4 Manager client Window shall support user's management.	See 2.4.x	None	None
2.4.1 The user management icons shall include: add a user, display/modify/disable users, and validate timestamps.	Ensure that all User Management functionalities are shown in the appropriate menu.	None	None
2.4.2 Clicking add user icons shall display a fill in form for adding a user to the project.	Ensure that clicking the *Add User* option loads the Adding a new user form web page.	None	None
2.4.2.1 Add user form shall enable the recording of the user id, the projects, the user's supervisor identification, and the selection of predefined fields for this user.	Ensure that the Adding a new user form contains and shows all the needed fields and that default values are present.	A SELECT query to the database, retrieving the available project and supervisors	Ensure that all fields are filled correctly.
2.4.2.2 Add user form shall be validated for completeness before being sent to the Server Module.	Externally query the database (NOT in TMT, for example a Database management software) and verify that the same values are retrieved	Same SELECT statement as above to the database, retrieving the available project and supervisors	None
2.4.3 Clicking Display all users icon should display the list of all users in alphabetical order with the identification of the projects that they are involved in and their supervisor.	Ensure that upon clicking the *List Users* option, the list is sorted alphabetically and that all mentioned fields are shown beside each user entry.	A SELECT query in the *Users* table to retrieve all users data using a simple ORDER BY clause	None
2.4.3.1 Display all users could provide a list ordered alphabetically by last name of the user or by project or by supervisor.	Ensure that the interface has multiple sort order options available and operational (by name, project, or supervisor)	A SELECT query in the *Users* table to retrieve all users data using a simple ORDER BY clause	None

2.4.3.2 By selecting a user, his profile can be changed or disabled.	Ensure that the status of a given user is changed.	A SELECT query in the *Users* table to retrieve all users data using a simple ORDER BY clause	None
2.4.4 Clicking validate timestamps should display the list of all timestamps Records submit by users.	Ensure that upon click on the *Validate Timestamps* option, list contains only the new (non-validated) timestamps submitted by the TMT users.	A SELECT query in the *Timestamps* database.	None
2.4.4.1 The timestamp records shall be displayed in a tabular fashion, with every record being on a separate row.	Ensure that the interface's dynamic table shows each timestamp on a separate row.	None	None
2.4.4.2 The fields should be identified with labels. Intuitive and non-confusing Abbreviations can be used if necessary.	Ensure that the dynamic columns have a header row with appropriate and clear titles.	None	None
2.4.4.3 The manager can validate a timestamp record, refuse a timestamp record or leave timestamp record non-validated.	Ensure that all timestamp validation options are present and operational. Ensure that the interface allows the timestamp's status modification	An UPDATE query in the *Timestamps* table modifies the status	None
2.4.4.4 The validated timestamps records should be inserted in the database.	Ensure that all validated timestamps are not shown in the validation table.	A SELECT query filters the *Timestamps* database and retrieves the right timestamps	None
2.4.4.5 The refused timestamps records should be returned to the user to allow correction.	Ensure that all rejected (sent back to user for correction) timestamps are not shown in the validation table.	A SELECT query filters the *Timestamps* database and retrieves the right timestamps	None
2.4.4.6 The non-validated timestamps record should be stayed in the table to allow manager to validate or refuse a timestamp record another time.	Ensure that, at any time, all non-validated timestamps are shown in the validation table regardless of the different actions performed on the table data.	A SELECT query filters the *Timestamps* database and retrieves the right timestamps	None
2.5 Manager client Window shall support project's management.	See 2.5.x	None	None
2.5.1 The project's management icon shall include the add project, add task, MS EXCEL and MS Project icons.	Ensure that all Project Management functionalities are shown in the appropriate menu.	None	None

2.5.2 Clicking the MS EXCEL icons shall display a form to identify the project, users, week's data and the EXCEL file where the corresponding data from the database should be transferred.	Ensure that upon clicking on the *Export to EXCEL* option, the export form is shown with the appropriate fields and data selection options. Ensure that the exportation form includes a functional file selection input for the exportation destination.	A SELECT query in all TMT database tables to retrieve all exportable data.	None
2.5.3 Clicking the MS-Project icons shall display a form to identify the project; users, week's data and the MS-Project file where the corresponding data from the database should be transferred.	Ensure that upon clicking on the *Export to MSPROJECT option*, the export form is shown with the appropriate fields and data selection options. Ensure that the exportation form includes a functional file selection input for the exportation destination.	A SELECT query in all TMT database tables to retrieve all exportable data.	None
2.5.4 Clicking add project icon shall display a form to identify the new project to add.	Ensure that upon clicking the *Add Project* option, the appropriate form is loaded.	None	None
2.5.5 Clicking the add task icon shall display a form to add a new task in a project selected.	Ensure that upon clicking the *Add Task* option, the appropriate form is loaded.	None	None

Functionality	Test to execute
3.1 The Server module shall be the only intermediate between the two client modules and the database.	None test A simple architecture verification and validation will do
3.2 The Server Module shall receive all the requests and format the pages.	Start with basic SELECT queries to the database and generate a simple dynamic HTML table in a web page. If that result is successful, all further requests can be based on this model. See 3.6 & 3.8 below.
3.3 The Server Module shall accept all connections from developers and manager client modules.	Overdrive the SM will multiple connections made simultaneously and ensure that all made requests have been accepted.
3.4 Upon log in request from the DCM the server module shall produce the Time Monitoring Window as specified in 1.2.	Ensure (login as a Developer) that the DCM is loaded. See 1.2 above.
3.5 Upon request for updating time stamp records from the DCM the server module shall update the database to reflect the new state of the DCM.	When a developer modifies timestamps data, the SM sends an UPDATE or an INSERT query to the database following simple SQL rules. The database should accept the queries and internally perform the modifications.

3.6 The Server Module shall validate and execute all requests coming from the DCM.	**For all functionalities specified in requirements 2.1 to 2.5.4.1, the SM executes the requested actions. Use a multi-connections environment and query all MCM actions on each connection. All actions should be accepted.** **The SM returns the requested action, interacts with the database and returns all database actions details, then gets the results and returns these results also.**
3.7 Upon log in request from the MCM the server module shall produce the **Manager Client Window as specified in 2.2.**	**Ensure (login as an Administrator) that the MCM is loaded. See 2.2 above.**
3.8 The Server Module shall validate and execute all requests coming from the MCM.	**For all functionalities specified in requirements 2.1 to 2.5.4.1, the SM executes the requested actions. Use a multi-connections environment and query all MCM actions on each connection. All actions should be accepted.** **The SM returns the requested action, interacts with the database and returns all database actions details, then gets the results and returns these results also.**
3.9 The Server Module shall display the TMT status.	**In both DCM and MCM use the SM; query the database with wrong entries which should create an abnormal query termination. The status of the TMT should change and an error message will be displayed.**
3.9.1 Any error of execution, communication, validation or else shall be identified and appropriate comment display.	**Launch any action or query then, interrupt internet connection on the server side for a short time. Remove database files or tables.**
3.9.2 The server module shall try to recovery from most common errors.	**None**

3. Test Strategy

The Test Strategy presents the recommended approach to the testing the target-of-test. The main considerations for the test strategy are the techniques to be used and the criterion for knowing when the testing is completed.

3.1 Testing Types

3.1.1 Function Testing

Test Objective:	**Ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval.**
Technique:	**Execute each use case, use case flow, or function, using valid and invalid data, to verify the following:** • **The expected results occur when valid data is used.** • **The appropriate error / warning messages are displayed when invalid data is used.** • **Each business rule is properly applied.**
Completion Criteria:	• **All planned tests have been executed.** • **All identified defects have been addressed.**
Special Considerations:	

3.1.2 User Interface Testing

Test Objective:	**Verify the following:** • **Navigation through the target-of-test properly reflects requirements, including window to window, field to field, and use of access methods** • **Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.**
Technique:	**Create / modify tests for each window to verify proper navigation and object states for each application window and objects.**
Completion Criteria:	**Each window successfully verified to remain consistent with benchmark version or within acceptable standard**
Special Considerations:	**Not all properties for custom and third party objects can be accessed.**

3.1.3 Data and Database Integrity Testing

Test Objective:	**Ensure Database access methods and processes function properly and without data corruption.**
Technique:	**Invoke each database access method and process, seeding each with valid and invalid data (or requests for data).** **Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved (for the correct reasons)**
Completion Criteria:	**All database access methods and processes function as designed and without any data corruption.**
Special Considerations:	**Testing may require a driver to enter or modify data directly in the database.** **Processes should be invoked manually.**

3.1.4 Performance Profiling

Test Objective:	**Verify performance behaviors for designated transactions under the following conditions:** **- normal anticipated workload** **- anticipated worse case workload**
Technique:	**Use Test Procedures developed for Function Cycle Testing.** **Modify data files to increase the number of transactions, or the scripts to increase the number of iterations each transaction occurs.**
Completion Criteria:	**Single Transaction / single user: Successful completion of the test scripts without any failures and within the expected / required time allocation (per transaction)** **Multiple transactions / multiple users: Successful completion of the test scripts without any failures and within acceptable time allocation.**
Special Considerations:	**Methods that can be used:** **• "Drive transactions" directly to the server, usually in the form of SQL calls.** **• Use multiple physical clients, each running test scripts to place a load on the system.** **Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.** **The databases used for Performance testing should be either actual size, or scaled equally.**

3.1.5 Load Testing

Test Objective:	**Verify performance behaviors time for designated transactions or business cases under varying workload conditions.**
Technique:	**Use tests developed for Function Cycle Testing.** **Modify data files (to increase the number of transactions) or the tests to increase the number of times each transaction occurs.**
Completion Criteria:	**Multiple transactions / multiple users: Successful completion of the tests without any failures and within acceptable time allocation.**
Special Considerations:	**Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement.** **The databases used for load testing should be either actual size, or scaled equally.**

3.1.6 Stress Testing

None

3.1.7 Volume Testing

Test Objective:	**Verify that the target-of-test successfully functions under the following high volume scenarios:** • **Maximum (actual or physically capable) number of clients connected (or simulated) all performing the same, worst case (performance) for an extended period.** • **Maximum database size has been reached (actual or scaled) and multiple queries / report transactions are executed simultaneously.**
Technique:	**Use tests developed for Performance Profiling or Load Testing.** **Multiple clients should be used, either running the same tests or complementary tests to produce the worst case transaction volume / mix (see stress test above) for an extended period.** **Maximum database size is created (actual, scaled, or filled with representative data) and multiple clients used to run queries / report transactions simultaneously for extended periods.**
Completion Criteria:	**All planned tests have been executed and specified system limits are reached / exceeded without the software or software failing.**
Special Considerations:	**What period of time would be considered an acceptable time for high volume conditions (as noted above)?**

3.1.8 Security and Access Control Testing

Test Objective:	Application-level Security: Verify that an actor can access only those functions / data for which their user type is provided permissions. System-level Security: Verify that only those actors with access to the system and application(s) are permitted to access them.
Technique:	Application-level: Identify and list each actor type and the functions / data each type has permissions for. Create tests for each actor type and verify each permission by creating transactions specific to each user actor. Modify user type and re-run tests for same users. In each case verify those additional functions / data are correctly available or denied. System-level Access (see special considerations below)
Completion Criteria:	For each known actor type, the appropriate function / data are available and all transactions function as expected and run in prior function tests
Special Considerations:	Access to the system must be reviewed / discussed with the appropriate network or systems administrator. This testing may not be required as it maybe a function of network or systems administration.

3.1.9 Failover / Recovery Testing

None

3.1.10 Configuration Testing

Test Objective:	Verify that the target-of-test functions properly on the required hardware / software configurations.
Technique:	Use Function Test scripts Open / close various non-target-of-test related software, such as the Microsoft applications, Excel and MS Project, either as part of the test or prior to the start of the test. Execute selected transactions to simulate actor's interacting with the target-of-test and the non-target-of-test software Repeat the above process, minimizing the available conventional memory on the client.
Completion Criteria:	For each combination of the target-of-test and non-target-of-test software, all transactions are successfully completed without failure.

Special Considerations:	What non-target-of-test software is needed is available, accessible on the desktop? What applications are typically used? What data are the applications running (i.e. large spreadsheet opened in Excel, large document in MS Project). The entire systems, NetWare, network servers, databases, etc. should also be documented as part of this test.

3.1.11 Installation Testing

None

3.2 Tools

The following tools will be employed for this project:

	Tool
Project Management	Microsoft Project

4. Resources

The resource for the *Time Monitoring Tool* test effort is the development team.

4.1 Workers

This table shows the staffing assumptions for the project.

Human Resources		
Worker	Resources Recommended (number of workers allocated full-time)	Specific Responsibilities/Comments
Test Manager	1	Provides management oversight Responsibilities: • Provide technical direction • Acquire appropriate resources • Management reporting

Test Designer	1	Identifies, prioritizes, and implements test cases Responsibilities: • Generate test plan • Generate test model • Evaluate effectiveness of test effort
Tester	1	Executes the tests Responsibilities: • Execute tests • Log results • Recover from errors • Document change requests
Designer	1	Identifies and defines the operations, attributes, and associations of the test classes Responsibilities: • Identifies and defines the test class(es) • Identifies and defines the test packages
Implementer	1	Implements and unit tests the test classes and test packages Responsibilities: • Creates the test classes and packages implemented in the test model.

4.2 System

All system tests will be executed on (at the most) 2 or 3 different computers: one from the CISCO laboratory and personal developers' computer to manage DCM and MCM accesses. The Server Module will be mainly tested on a computer located in the Ecole Polytechnique School (C-211).

The server tests stations must have the following softwares installed and properly configured:

JAVA 2 SDK v1.3

Apache v1.3

Tomcat v3.2.1 (Apache)

mySQL Server v3.23

mySQL JDBC driver

The main test station for the Server Module is a PIII-900Mhz, 256MB RAM running under Windows 2000. The computer is operational at anytime and available for testing purposes 24 hours a day and 7 days a week. The IP address is: 132.207.114.212 (static)

In order to make all Server Module tests more reliable, multiple tests will be executed at different traffic times: by day, by night, by morning, by the peak hour.

5. Project Milestones

Milestone Task	Effort	Start Date	End Date
Plan Test	10h	10/01/01	26/01/01
Design Test	15h	28/01/01	12/02/01
Implement Test	15h	12/02/01	20/02/01
Execute Test	30h	22/02/01	20/03/01
Evaluate Test	10h	24/03/01	30/03/01

6. Deliverables

6.1 Test Model

The Test Model will define all the Test Cases and will reference the test procedures and test scripts which are associated with each test case.

6.2 Test Results

For each test executed, a test result form will be created. This shall include the name or ID of the test, the use case or supplemental specification to which the test relates, the date of the test, the ID of the tester, required pre-test conditions, and results of the test.

6.3 Test Evaluation Report

A final evaluation of test activities will be presented.