1 INTRODUCTION

1.1 Objectives

Since making mistakes is for human, detecting and correcting them should be the responsibility of the human. Developing a software project in a limited time is a tough work and prone to be defected. Although every member of the project is aware of the importance of the project and tries to generate well-written, flawless code, the errors are inevitable. Therefore if we do not get out of it, we have to review and check our modules and codes closer to detect and correct any error in our project by applying a careful and systematic testing plan.

In the progress of our project "CoreAccess", the testing phase is the last important step before our product goes to the customer. The main objective of our group is to release our product with minimum error and bugs. By applying our testing plan on our project, we will achieve these goals and produce a product with high quality.

1.2 Scope of the Document

Actually the testing is present in every phase of implementation, that is, every member is responsible of his/her code and tests whether any error or bug exists in his/her code. The first tester of every code is the person who generated it. After the codes are combined, the other members are also testing each other's code.

Although testing includes all the implementation progress as we mentioned above, in this document we mainly told about the test plan after a release of the project is done. There are lots of testing methods in software engineering, and we tried to show which testing methods are applicable to our project and to which methods we will pay attention much. We have also included a test scenario for the three main modules of our project: PDA Modula, Administrator Module and Content Manager Module.

The tools we will use throughout testing phase is mentioned and explained clearly. In addition to that the duties of each member in testing process are clarified and the schedule for testing is done.

1.3 Major Constraints

1.3.1 Time

Since our project should be finished in a month's time, time is the main constraint for us. We had a schedule for the testing phase, therefore the deadlines of each specific work are known beforehand. If we manage to obey the deadlines according to schedule, our project "CoreAccess" is released after proper testing and in time.

1.3.2 Hardware

For testing phase to be successful, we have to have enough hardware to accomplish our project. We have 4 PDA's for all mobile GIS groups however this is not a problem for us since we have a proper schedule. We can carry on our implementation and testing on PDA, when we have it. On the other hand, the GPS hardware is still a problem for us. We do not have any GPS device to get needed location information. When this problem is solved, we will be able to do and test GPS-related parts of the project.

1.3.3 Software

Since we were able to show the vector map of METU on PDA, we got rid of a big problem. Testing each modules (pda module, administrator module and content manager module) separately will be done and ease our job with a considerable amount.

1.3.4 Performance

Performance testing is a main issue for us. For PDA module, the performance of the project can vary between Microsoft Visual Studio Pocket PC Emulator and our PDA. The processes at PDA are far faster on PDA than on emulator. Therefore for performance testing, we need PDA ready for fast processing. The performance test of administrator's and content manager's GUIs are easier since we created them on Java platform.

1.3.5 Staff

Our 5 people group is a major constraint, since we have all the responsibilities for the testing phase. However, since we have a good harmony in our group, the testing of the project will be completed successfully.

2 TEST PLANNING

2.1 Unit Testing

Since CoreAccess is mobile GIS application, its main components are related with GIS. Therefore, unit testing is applied to two main areas of "CoreAccess" project; GIS engine and GIS data. Unit testing is based on procedures to write test cases for all of the functions and methods and for this reason it is a very costly operation. So, we have decided to apply unit testing to very critical modules most of the time in order to save the resources of the development team.

2.1.1 GIS Engine Tests

"CoreAccess" is a big mobile GIS project which consists of a desktop application, a PDA application and a web application. In order to satisfy these requirements, "CoreAccess" has 12 main modules and these modules have further sub-modules. Considering the implementation time of the project, it is very hard to make unit testing for each module in a complete manner. Because, unit testing needs writing some extra code and sometimes it becomes very complex to separate some modules which are deeply connected such as our “Transportation module” and “Pathfinder module”. For these cases, mock objects should be generated in order to satisfy perfect independency of modules. As a result, we apply unit tests only to the modules which are critical for the rest of project and which work clearly independent from the other modules.

GIS engine related unit tests are done using the black-box method; outputs are examined according to related inputs.

Up to now, we have applied unit testing to eight modules considering the already implemented parts. These modules are namely;

- ➢ PDA module,
- ➢ Content Manager module,
- ➢ GIS Engine module,
- ➢ Activity module,
- ➢ Logger module,
- ➢ Administrative module,
- ➢ Map module,
- ➢ Database module.

These modules are nearly finished and most of them have already been integrated. Before the integration phase, a unit testing phase was inescapable. Some extra code has been written for unit testing and various kinds of data (both vector data and object data) have been supplied. For example, it can be easily said that map module is the core component of our project. Success of the whole project depends very much on its reliability; therefore we have spent so much effort to be sure that it can handle all instances of vector map format that we have preferred as a standard (ESRI shale file: .shp). It was a hard process because output is not a written data, only a visible map if successful. We had some problems with the METU campus vector maps taken from GGIT and tried hard to produce debug and error codes. Finally, we became sure that our map module is perfect but METU campus maps have some problems based on their coordination systems. (The problem with METU maps will be addressed in the next section: GIS Data Tests.) The other modules were tested in the same manner. According to test results, corrections and adjustments were done. Then the integration of existing modules was completed.

Two main modules, namely pathfinder module and transportation module are waiting for unit tests to be applied. The development of these modules is still in progress. In the pathfinder module, Dijkstra’s shortest path algorithm is being implemented. This algorithm can be tested with unit testing independently. Correctness of the implementation has the first priority; however performance issues should also be resolved with the help of unit testing. When we only think of testing of pathfinder module’s algorithm, we are safe because it is obviously independent. On the other hand, considering the broad aspect, pathfinder depends on map module because it must take the nodes and edges. This problem should be resolved in integration testing but it is much better if unit tests are constructed by thinking about the probable future problems.

One of the 12 modules is the GPS module. At the beginning of the project, we were told that one GPS receiver will be provided for all MOBGIS groups or some simulator input will be given. We made our research and started to implement a bit of the GPS module. However, for some reasons we did not get any receiver or data. As a result, the implementation of GPS module is idle now because we have shifted the resources to other parts' implementation. If any progress is achieved in the following days, unit testing should also be applied to GPS module. It can be an easy to do so, because GPS module is pretty much a standalone module.

2.1.2 GIS Data Tests

"CoreAccess" includes several kinds of data most of which are visual and crucial for the sake of the application. All of these data must be verified with unit tests. Correctness of each data must be checked in detail. Especially in the case of vector maps, the verification becomes tough because they are usually huge maps and contain lots of detail. As stated in the previous section, we had serious problems during the unit testing of map module because of the METU campus vector maps supplied by the GGIT department. Whatever we tried, we could not be able to display the METU map on PDA. On the other hand, there was no problem with the original world or USA maps taken from ESRI map data. As a result, our map module passed the unit test and we have concentrated on unit testing of METU vector maps. Discovering the problem took about 1 week and resolving it took about 2 days. The problem was, GGIT had used the meter-based planar coordinate system for METU maps. However our map module works well with latitude-longitude-based global coordinate system. We resolved this critical issue with the help of ESRI ArcMap tool by changing the coordinate system of METU maps. The rest of the data usually resides in relational database (we prefer MySQL) and we can clearly say that unit testing of data that are in the database is a simple process.

Unit testing of data is easier than unit testing of modules because no extra code is written and testing is done by the help of external tools. In the following table the tools that were used for unit testing of data is shown:

Data	**External Tool(s)**	**Purpose**
Vector Map (ESRI .shp format)	ESRI ArcView, ESRI JoViewPlus	Check coordinate system, view every detail
Vector Map (.dbf Database file)	MapInfo, ESRI ArcReader, Microsoft Excel	Check written data, control localization problems
Data in MySQL	phpMyAdmin	Check written data

2.2 Integration Testing

When we are confident that the all individual modules are working well, we will begin the integration test of the modules. Namely, after unit testing, we are going to make some commitment about this phase of testing.
In integration test we are using bottom up modeling. The reason why we determined the bottom up testing is that it is easy to combine the little pieces. We divided our system into the several modules. They are implemented independently by each member.

Moreover, we are not able to do integration test from beginning of the implementation to the end of the implementation. We determined some milestones for each module to be finished.

Only when they are finished we can integrate them and test them in this phase. However in some cases, in order to test some modules we have to use the other modules.

In the following, we listed the modules which are interacted to other modules. From the table, it is easily seen that which module will be integrated to which one.

1	GPS Module	PDA Module	Second module gets the position data of the user from first module
2	Database Module	Content Manager Module	Second Module inserts some values to the database by the help first module
3	Database Module	Activity Module	Second Module inserts some values to the database by the help first module
4	Administrative module	Map Module	First Module interface depends on the second module implementation.
5	Transportation Module	Map Module	Only after Map module is completed, transportation module can be tested.
6	Path Finder Module	Map Module	Only after Map module is completed, Path Finder module can be tested.
7	Logger Module	Database Module	First module writes the logs of the search result to the database by the help of second module.

2.3 Validation Testing

Validation test can be named as system test. This testing phase is done in the actual working environment. Therefore, we are planning to simulate the typical working environment for each different type of users which are PDA users, Admin and Content Manager.

We divided validation test into two. One of them is interface validation test, the other one id data flow validation test. In the following, you can find the detailed information about these test phases.

2.3.1 Interface Testing

In interface testing, we act according to the user requirements. For different users, we follow the below steps for validate the system to work:

- **Administrator:** Administrator runs an application on his/her desktop. The menu items should be reachable easy. The time while passing from one tab to another tab is important.

- **Content Manager:** There will be different content managers in different locations. Therefore, they will use web application which is available in every computer. The menu items are more important for content manager. Moreover, since it is a web application the connection is important. Timeout will be handled for our system.

- **PDA Users:** Our actual customers are PDA users. The comfort of the user is crucial. Users have to see the search items in one look. Furthermore, for hasty users, zooming will be tested in several times consecutively, in order not to collapse the system. The last concern is the actual working environment of the users. Developers generally implement the application in the emulator. They have to try out to deploy the system into the Pda and should control properties of the screen.

2.3.2 Data Flow Testing

Sequence of the data is significant. Therefore, the flow of data should be controlled in validation test phase.

- **Administrator:** This user only interacts with the database. Transaction execution should be regarded.

- **Content Manager:** Content Manager Module interacts with the web service module and web service module interacts with database. As it is mentioned, the transition between these modules should be controlled by using different datasets for each scenario.

- **PDA User:** This is very similar to the Content manager part. Triple data flow is valid for PDA User.

2.4 Higher Order testing

As at the end of this project we will come with a full software package and we intend to release this software to costumers, it is required to apply higher level test to our software package. We separated our high level tests as, performance, stress and alpha-beta tests.

2.4.1 Performance Testing

Since hardware is the common limitation in software performance we have to consider the hardware limitation although we implement a working program. Our software package has two main sides and we have to consider each separately for our performance testing. As PDA hardware is more limited than PC we decided to compute complex procedures like query activities, search places and finding path in our server application and send the result through the communication line between our server and costumer's PDA. In this point of view, although these complex procedures can be computed fast in PC, we have to determine the maximum number of costumer who can do these procedures concurrently. At the end of this test we will come up with a respond time bound with respect to number of costumers.

2.4.2 Stress Testing

In the server side we will connect as many PDA as we can and we intend to see the maximum number of PDA that we can server without any bottleneck. On the other hand we intent to determine the maximum number of maps and layers that we can load in the PDA side. Since PDA hardware configurations are very limited we have to know this bound so that we can force users not to load maps over the critic limit and save our PDA application from possible crashes.

2.4.3 Alpha-Beta Testing

Every time we add a new feature to our software package, we run both server and PDA applications and see the results. Moreover, we test the reliability and consistency with other features of this new feature. Consequently, our alpha testing is continued as we continue to add new features to our system. However, as we intend to come up with a reliable software package we have to a second test- beta test, to see whether we have any bug or inconsistency which have not recognized before. Since PDA is not popular as PC, we can not release a free beta version in the internet and look forward the feedback. Therefore, we decided to do our beta test with our friend MOBGIS project groups. We thought that if every project member and group accepts to involve our beta test (also these beta test can be applied to other MOBGIS project with the same point of view), we intend to reach to 20 people who have both user and programming experience in PDA which will increase the effectiveness of our beta test.

3 TEST SCENARIO

For being able to test the system as a whole we prepared a test scenario for each main module of the system. In this way, we can see the missing parts of the program for the sake of both functionality and usability. We included nearly all the functionalities into the scenario that we stated in our requirements analysis. By the help of this scenario, we will identify what is difficult and what is easy for a user. We will explain the scenario to a number of people and we will request them to use CoreAccess for this purpose.

3.1 Scenario Description

In a sunny and brilliant Saturday afternoon, after their fight against Shredder ninja turtles want to see the theater festival in METU. Although they visited METU several times, they have no idea about where such activities are performed. However, technology geek, Donatello has installed CoreAccess software into his mobile device a few days ago and he is aware of the benefits of the program. They all together drive their balloon to METU. Since the air traffic is not allowed in METU, they leave the balloon at A4 door. Donatello uses his device to make a "Category Search". He just selects the theatre category and leaves the date as the default which is the current day. Then he/she clicks on the search button. In a few seconds a list of results is returned which is sorted by distance, "Kültür Kongre", "MM-25" and "Mimarlık Anfisi". For each of them they look at the position of the places on map. They realize that Mimarlık Anfisi is quite far compared to the others. They decide to go to the nearest one “Kültür Kongre”. In order to see the way to the Kültür Kongre, they active another query for the transportation details. In just a few seconds the path to the destination is highlighted. Using this path, they reach to the kültür kongre and enjoy the theatre.

While leaving the saloon, they hear two students speaking about the CengCup’06. One student persuades the other for watching the semi-final match. Leonardo get curious about the match and suggests to see the next match. Donatello makes another search but this time “Activity Search” since they know the name of the cup. For the name of the activity he enters “Ceng Cup” and again leaves the date and time unchanged. Only one result as a place is returned which is “Odtu Halısaha”. Next, following the path, they reach at the “halısaha”.

They were really enjoying the match. However, through the end of the second match, a heavy rain starts. However, they do not want to go home since they want to take some time in METU. An idea appears in Michelangelo’s mind: going to the cinema. In this way, maybe at the end of the movie, the rain might stop. Others also loved the idea of going to the cinema. But he does not know which films are available and which one is worth to watch. Again CoreAccess helps them. Donatello begins to use his PDA. This time he selects the search for activity type. He chooses the cinema activity. He fills the necessary fields on the PDA. They especially want to go action film. Finally, list of the places having cinema is returned: “U3” and “MM-125”. They click on both U3 and MM-125 to see the detailed information of the activity. They see that the movie on U3 is in the evening and there is too much time. Therefore, they decide to go to MM-125 to watch “Lucky number Sleven”. Then they view the map path on the map to MM-125.

4 TESTING TOOLS

We will use the testing tools and environments during the tests:

- Eclipse for compile and debugging: Eclipse has very good features for debugging. At a specific time, by breakpoints, content of any variable can be seen easily.
- Visual Studio for PDA: Since our development language for PDA is .Net we will use Visual Studio for tracking the instruction flow.
- Log4J for logging: Without changing any line of code and without a need to recompile, the log level can be changed to any of the followings: DEBUG, INFO, WARN, ERROR, FATAL.
- JUnit for java modules: JUnit is a regression testing framework which implements unit tests in Java and open source software.
- TCPMon for web service communication: It is a tool to follow the messages over TCP communication.

5 STAFFING

The staff of our project is mainly responsible for the parts showed below:

Testing Java Modules, Testing Database	Sercan Gök
Testing Java Modules, Testing Database	Eda Kılıç
Test Coordinator, Testing Web Service, Testing Database	Mehmet Olduz
Testing PDA Modules, Testing Map Modules	Onur Tekdaş
Testing PDA Modules, Testing Map Modules	Mustafa Yüksel

6 TESTING SCHEDULE

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

Test Plan Delivery: (Deadline) 23.04.2006

Unit Test and Integration Tests: (Start) 23.04.2006 - (Deadline) 01.05.2006

Validation Tests: (Start) 01.05.2006 - (Deadline) 04.05.2006

Performance and Stress Tests: (Start) 04.05.2006 - (Deadline) 06.05.2006

Beta and Alpha Tests: (Start) 06.05.2006 - (Deadline) 13.05.2006

Results (i.e. Bugs) Tracing and Correction: (Deadline) 16.05.2006