Example
User Acceptance Test Plan

Version 0.6

Revision History

Date	Version	Description	Author
7 July 2010	0.1	First draft	Geri Winters
8 July 2010	0.2	Added functional components and tools to the functional requirements tables. These still need priorities and acceptance criteria assigned.	Geri Winters
9 July 2010	0.3	First draft of priorities for Functional tests; first draft of acceptance criteria numbers.	Geri Winters
20 Sept 2010	0.4	Reformatted to match IEEE test plan template and added processes for test design and development, test progress reporting, and acceptance negotiation. Also added templates for test cases and test scripts to appendix a.	Greg Fournier
26 Sept. 2010	0.5	Updated based on response from reviewers.	Greg Fournier
28 Sept. 2010	0.6	Updated based on additional comments from reviewers.	Greg Fournier

Table of Contents

Test Plan

1. Introduction

1.1 Purpose

The purpose of this test plan is to describe the User Acceptance Testing that will take place for Release 1 of the project. The plan covers testing from a user perspective and will include functional Use Case based testing and Usability testing. This plan will include the following information:

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

1.2 Background

For the entire product being developed, major features were identified and scoped out. Five primary Use Cases were identified representing major scenarios of how the system will be used. From these, 72 functional components were identified from the 5 Use Cases. These functional components along with the 5 Use Cases will be the primary focus of User Acceptance testing. Additional testing of non-functional requirements and usability requirements will be addressed as well.

The version of the product developed for release 1 focuses on three major features listed in section 1.3.1 below which are associated with 18 functional components.

1.3 Scope

1.3.1 Features to be Tested

The scope of release 1 encompasses multiple features supported by the functional components and realized by the 5 Use Cases. The following Features will be targeted for test.

Time Series: This feature along with Scenario Management makes up the bulk of the functionality delivered in release 1. This feature corresponds to about 1/3 of the functional components and about 20 tools targeted for release 1.

GIS Management: This feature includes importing, editing, visualizing, and evaluating GIS data.

Adaptor Interface: This is a component interface that is newly developed and allows tools to interact with the product via adaptors. This functionality spans Functional features and is presented in many of the Use Cases at various levels. This is specified here because it is considered an important component and will be tested accordingly. This component will be tested as part of the functionality testing of the other features.

1.3.2 Features Not to be Tested

Currently the test team plans on testing all features related to release 1 which are listed above. Any specific functionality related to release 2 and 3 will not be tested.

1.4 Approach

The purpose of User Acceptance Testing is for the end users to accept delivery of the product. The users test the software to confirm that the planned work is complete.

User Acceptance Testing is not destructive testing, that is the purpose is not to break the software. Rather the purpose is to show that the software meets the needs of the users by verifying that it conforms to requirements. At the same time the test team plans on doing negative testing to ensure the important aspects of the product are covered and work correctly.

The overall approach to acceptance testing release 1 will be to get users involved in the testing of the delivered product. Use Case based testing will be performed on functional components with focus on areas of interest for the sponsor. Regression testing and Usability testing will also be performed to ensure the product complies with specifications and is acceptable to the client.

Usability testing will be performed using a combination of testing against usability constraints, exploratory testing, and user observation. These are explained in more detail in section 3.6.

The detailed test strategy is outlined in detail in section 3 of this document.

1.5 Assumptions

The following is a list of assumptions made during the development of this document that may impact the design, development or implementation of testing.

- There will be sufficient user representation and involvement throughout the testing process
- There are enough existing Data and Models that can be imported to the product to conduct sufficient testing
- There will be sufficient hardware and support to perform UAT.
- Resources will be available to set up the test environment.
- The release 1 product will be delivered from the development team on schedule.
- Licenses for the test managements software will be sufficient.
- The product will stable enough to be tested.
- It will be clear which functional components are in scope for release 1.
- The development team will not find any gaps in testing when completing traceability in the test management tool.
- Communication between the development team, the test team, and the project stakeholders will go smoothly during defect resolution

1.6 Risks

The following is a list of risks or contingencies that may affect the design, development or implementation of testing.

- Use Cases may not accurately represent how the system will work. Mitigate this risk by elaborating the Use Cases by adding steps and reviewing the software to refine the steps.
- The code delivered from the development team may be unstable and incomplete when initially delivered based on defect tracking reports and analysis which indicates some functions may not be covered by test cases. Mitigate by planning for

interim deliveries of code from the development team and add extra tests where tests are incomplete.

1.7 Test Plan Resources

The following is a list of documents used for developing this test plan.

- The Software Requirements Specification version 1.4
- The User Interface Requirements and Design version 1.2
- The Feature release Plan version 1.0
- The Software Architecture document version 1.7

2. Requirements to be Tested

Functional Components to be tested in Release 1

The following table lists the functional components by Feature which are targeted for test in release 1.

Functional Component	Description
Time Series Components	
UC001 Import Time Series	Importing time series data to the database. Activities involved comprise formatting the input time series file to a supported layout, specifying the data type and unit for the input data, providing the time series with a name and associating it with GIS features.
	Import Time Series By a Wizard from a CSV file – including linking to an object and not linking to an object.
	Import Time Series by a Wizard from an ACSII file (optionally with header – to facilitate "loading") – formats are yet to be defined (body part is anticipated to be flat) - including linking to an object and not linking to an object.
	Import Time Series by a Wizard from an XLS file - including linking to an object and not linking to an object.
	Import Time Series by a Wizard from a DBF file - including linking to an object and not linking to an object.
UC002 Use Time Series Tool – Process a time series (i.e. into another time series, a number, or other output)	Processing time series. Activities involved comprise selecting of one or more time series to process, selecting the appropriate tool(s), setting tool specific parameters, executing the tool to modify the time series' data and storing the output in the database.
	Mean Value Tool - Calculates the Mean value of a time series.
	Minimum Value Tool - Calculates the Minimum value of a time series.
	Maximum Value Tool - Calculates the Maximum value of a time series.
	Standard Deviation Tool - Calculates the Standard Deviation of time series values.
	Median Tool - Calculates the Median of the time series values
	Mode Tool - Calculates the mode of the time series values.
	Skewness Tool - Calculates the skewness of the time series values.
	Kurtosis - Calculates the Kurtosis of the time series values.
	Resample Tool – Re-samples the time series into a new time step.
	Synchronize Tool - Synchronizes the time axes of two or more time series.
	Gap Filling Tool - Performs gap filling of time series, based on a multi-site correlation analysis.
	Extract Period Tool - Extracts a sub-period from a time series.
	Moving Average Tool - This tool computes the moving average for a time series.
	Double Mass Tool - This tool is used to perform a double mass analysis on the input time series.

	Extreme Value Analysis Tool - This tool is used to extract extreme values from one or more time series, fitting them to appropriate extreme value distributions, and allows the user to compute the values for different return periods (or vice versa).
	Outlier Analysis Tool - This tool allows the user to identify outliers based on a number of statistical methods.
	Residual Mass Tool - Calculates the cumulated deviation from the mean for a time series.
	Residual Series Tool - Calculates the difference between two time series.
	Empirical Frequency Tool - Calculates the Empirical Frequency of time series.
	Cumulated Frequency Tool - Calculates the Cumulated Frequency for time series.
	Duration Curve Tool - Calculates the duration curve for time series.
	Correlations Analysis - This tool computes the correlation between time series.
	Rate of Change Tool - This tool computes the time derivative of a time series.
	Test for Stationarity Tool - This tool tests for stationarity of time series at user specified significance levels
	Test for Homogeneity Tool - This tool tests for homogeneity of time series at user specified significance levels
	Test for Randomness Tool - This tool tests for randomness of time series at user specified significance levels
UC003 Visualize time series.	Display time series Activities involved comprise selection of time series, adding time series to an existing or new chart or table and customizing the chart or table.
	Plot multiple time series on the same graph with different y-axis
	Plot multiple time series with one trending up and the other trending down on the same graph
	Plot multiple time series on different graphs
	Show time series in tabular format
	View time series meta information
UC008 Filter time series	Defining search criteria for looking-up time series in the data-base. Activities involved comprise specifying search parameters like name, type of time series, time series generated from specified scenarios, associated with selected features etc.
UC009 Inspect time series	Looking at time series attributes and meta-data Activities involved comprise looking-up time series, selecting time series, showing time series attributes and showing associated meta-data.
	Select a time series and view its meta data
	Select a time and view its data in a graph and in a tabular format – See "Visualize Time Series"
	Select a time series and view its properties
UC 52 Edit time series	Modifying a time series Activities involved comprise selecting the time series, adding it to a data table, editing time stamps and/or values and saving it. This can happen as a manual or automated process.
UC074 Create time series	Creating new time series in the Database. This can be

	accomplished by importing time series, using time series tools, running models or directly by creating them within the application. This use case is concerned with the latter. Activities involved include the definition of a time axis, data type, data units, entering values, saving the time series under a specified name and associating it with GIS features.
UC077 Link time series to feature –	Associating time series with features Activities involved comprise selection of one or more time series, selection of GIS features and executing the association process.
UC088 Export time series	Export time series data from the database to a native modeling tool or the file system. Activities involved comprise selection of the time series that shall be exported and selection of output format
UC075 Create Hydroobject	Creating a new hydro object and store it in the database. Activities involved could comprise selection of hydro object, type, entering characteristics, associating with GIS feature and saving it.
UC078 Edit Hydroobject	Modifying an existing hydro object Activities involved comprise searching for hydro objects, selecting a hydro object, modifying characteristics and saving it
GIS Comnponent	
UC021 Import GIS data	Transfer from other sources (e.g. shape files) GIS information into the database. The import may include projection of data, transformation of data, sub-selection of data and aggregation of data. The outcome can be new layers/feature classes and/or new features in existing layers.
	Import GIS Vector Data (Point, Polyline, Polygon) By a Wizard from a GML file
	Import GIS Vector Data (Point, Polyline, Polygon) By a Wizard from a KML file
	Import GIS Vector Data (Point, Polyline, Polygon) By a Wizard from an SHP file
UC023 Edit GIS data – Edit properties of GIS layer and features. Edit meta data for GIS layer.	Perform GIS operations on existing GIS data (extract, combine, modify properties, etc.) on exiting GIS layers to create new or modified layer information in the database.
UC092 Visualize GIS layer	Show GIS features as map layers Activities involved comprise selection of GIS layers, adding the layer to a new or existing map and customizing the map (e.g. symbology, labels, scale, projection)
UC007 Inspect GIS layers	Looking at GIS features, their attributes and meta-data. Activities involved comprise looking-up GIS feature layers, selecting features, visualizing features, showing features as attribute tables and showing feature associated meta-data.
UC026 Use GIS tool	Establishing catchments based on GIS data Activities involved include selecting GIS feature layer, selecting the delineation tool, setting delineation parameters, performing the geo-processing and storing the delineated catchments.
UC089 Export GIS data	Export GIS data from the database to a native model-ling tool or the file system. Activities involved comprise selection of the features that shall be exported and selection of output format.
UC005 Publish In Report	Creating reports from different data sources – references to existing or "to be generated" data - within the database. Activities involved comprise defining report templates, by

	grouping controls on reports, associating controls with data sources and "running" the report template in order to create the report.

3. Test Strategy

This Test Strategy presents the recommended approach to acceptance testing of release 1.

3.1 Strategy Approach

The test team plans to conduct Use Case based functional testing and usability testing for release 1. Regression testing will be used as needed as defect fixes are migrated to code. The test team will use a subset of the functional tests for regression testing.

Although Stress testing and Load testing are not the focus of User Acceptance Testing for release 1, the plan is to perform those tests as needed if problem areas are identified. With the short turn-around time for release 1, it wouldn't be feasible to procure and use tools to do this. It is suggested that the test team use existing models and data related to the Use Cases of potentially problem areas and run large volumes of data simultaneously. The development team must provide those problem areas. This all depends on the potential problems. Functional component related problem areas will be identified and the corresponding Functional (Use Case based) tests will be used along with large volumes of related data or a large number of users.

Use Case based testing will be used for functional testing based on the functional components describe in section 2 above and the corresponding delivered functionality of the 5 Primary Use Cases.

Usability testing will be conducted based on specifically defined constraints that will be defined as part of usability test design. The test team will identify constraints and have them approved by the sponsor as a precursor to writing usability test cases. Constraints will outline usability features such as number of clicks to reach a solution; ease of use based on type of user, and overall screen layout and functionality. Usability will employ existing test cases used for functional testing to test areas based on Users. These test cases will be modified for expected results based on usability constraints. Exploratory testing and user observation will also be employed as part of Usability testing. The process is explained in greater detail in section 3.6.

The following table describes the overall types of testing that may take place as part of User Acceptance Testing.

The types of tests that can be used to support milestones, phases and iterations include:

Test Type	Comments
Use Case Based Functional testing	The objective is to ensure proper target-of-test functionality, including navigation, data entry, processing, and retrieval. This will be conducted based on Functional Components and the 5 Primary Use Cases.
Data and Database Integrity testing	As part of Use Case Based testing data integrity will be tested.
Regression Testing	The test team will use regression testing as

	needed to ensure defect fixes do not break code that is functioning properly.
User Interface Testing	This will be part of Usability testing. The goal is to verify that Navigation through the target-of-test properly reflects requirements in the User Interface Requirements and Design document and window objects conform to standards.
Performance Testing	This will be performed only as specified by the sponsor. The goal is to verify that system performance meets required standards under normal and worst case conditions. While establishing specific performance criteria is out of scope for User Acceptance testing it is expected that such criteria exists and the testing team is prepared to perform some performance testing in areas of concerned specified by the sponsor.

3.2 Risk areas requiring test focus

The project team has identified risk areas that should be focused on as part of testing. While the entire system will be tested, these areas will get extra consideration when selecting tests from the potential tests. More work is required by the test team and the sponsor to elaborate these. The risk areas are as follows:

- Robustness of Software:
- Integration to external tools
- Capacity
- Data Integrity

3.3 Functional Test Design and Development

Since Use Case based testing is employed for functional testing, the overall process for test design and development will be to start with the functional components, determine test coverage needs, and create the test cases. From those deliverables test cases for the 5 Primary Use Cases will be created. Test cases will be used to develop test scripts. Scripts will eventually be run to test two versions of the product, the Professional, and Corporate versions.

The activities performed in test design and development are as follows:

- Elaborate on the functional components
- Identify potential tests
- Evaluate and determine test coverage and select tests
- Write Test Cases
- Write test scripts

Although the bulk of the test development will be done early, it must be noted that tests can and will be added iteratively as test reports are analyzed and it is found that new tests are needed.

Elaborate the Functional Components

The current functional components have good information but must have steps associated with them in order to be testable. The focus of this activity is to add steps to the functional component descriptions to show a sequence of events associated with the functional component objectives. Although the functional components are elaborated for testing purposes, these will be given to the sponsor and development team so that the SRS may be updated if with steps if it deemed feasible.

The inputs to this activity are:

- The SRS containing functional components
- The User Interface Requirements and Design Document
- A copy of the product UI

The result of this activity will be an interim deliverable of a document containing Elaborated Use Cases for each functional component.

For each functional component the information contained in the SRS will be read to understand the goal of the functional components and associated activities. The test team will use the User Interface Requirements and Design document to understand actual steps and sequence of events for the associated activities. Steps will be written for the corresponding functional component. If needed the steps will be further refined using the product UI for clarification.

The following is an example of an elaborated Use Case for the Import Time Series functional component.

Use Case	Description	Steps
UC001 Import Time Series	Importing time series data to the database. Activities involved comprise formatting the input time series file to a supported layout, specifying the data type and unit for the input data, providing the time series with a name and associating it with GIS features.	1. User select Import Time Series 2. System start Wizard 3. User select the type of the file to import ALT: Import from CSV file 4. User repeat steps 5-7 as long as desired. 5. Browse to location of one or more time series files 6. Add files to list of files to be imported ALT: file is of wrong type ALT: remove file from list 7. For each file selected, choose which GIS feature to associate it with ALT: choose no association 8. Select Import 9. The system imports the time series, creates the associations, and saves the time series and associations in the database.

Identify Potential Tests

This activity uses the Elaborated Use Cases to identify all potential tests for a functional component in order to identify full test coverage. The result will be a table for each functional component identifying all potential tests with a beginning test state, variable inputs to decision points, and expected results for each potential test.

First, the steps of a Use Case are analyzed and the decision points identified. Inputs required for each decision point are identified and become operational variables. Specific operational variables are combined in a row identifying unique test combinations with expected results. Once all unique combinations are completed the result is a table of unique tests with the intent of covering all decision points of a Use Case. Each row in the table represents a unique test.

The following is an example of a partially filled out potential test table for the Import Time Series functional component.

Var.	Sys State	Node Level	File Type	File Name	File Path	GIS Feature	Expected Results
1	Time Series Explorer is open	Root Node	N/A	N/A	N/A	N/A	Wizard opens
2	Time Series Explorer is open	Time Series Group	N/A	N/A	N/A	N/A	Wizard opens
3	Time Series Explorer is open	Invalid node	N/A	N/A	N/A	N/A	The Wizard doesn’ come up. (no way select import tim series at that no
4	Time Series Explorer is open	Valid Node	Time Series	Name Entered	Valid Path	N/A	File List present
5	Time Series Explorer is open	Valid Node	CSV	Name Entered	Valid Path	N/A	File List present
6	Time Series Explorer is open	Valid Node	Invalid file type	Name Entered	Valid Path	N/A	A message is displayed indicat an error and the is not added to t File List.
7	Time Series Explorer is open	Valid Node	Valid File	Name Entered	Valid Path	Feature Selected	Feature added to list
8	Time Series Explorer is open	Valid Node	Valid File	Name Entered	Valid Path	No feature selected	User is prompted select a group
9							

Evaluate and determine Test Coverage and Selecting Tests

For each Use Case potential tests will be selected from the list of potential tests identified in the activity described above. Coverage will be determined as a team involving the sponsor and development team. The stakeholders and development team will assist the test team in test selection base on the following:

- Confidence in test coverage for corresponding functionality by development team testing
- Importance and priority of functionality
- Perceived functional component stability

For Confidence in test coverage: The test team along with the sponsor will look at test coverage in development team testing by examining the test management software and looking for components or functionality with little or no tests linked to them. For these areas that have little coverage, the functional components related to the problem areas will be identified and greater emphasis will be given to the corresponding potential test tables. A large number of tests will be selected from the potential tests with a greater emphasis placed on negative testing.

For Importance and priority of functionality: Important functionality is determined by the sponsor and other expert users. Functional components for this important functionality will be identified, corresponding potential test tables will be scrutinized and test will be selected from the tables accordingly.

For Perceived functional component stability: The test team along with the sponsor will look at defect rates and defect severity levels for each functional component taking into consideration clustering and the types of errors present. Based on the analysis, the functional components related to the problem areas will be identified and greater emphasis will be given to the corresponding potential test tables. From there it will be determined which extra tests should be added.

The above considerations will be balanced with the coverage needs of all functional components. Since most issues of stability and priority relate to the functional components, Use Case test selection will be conducted for all Use Cases that trace to a functional component.

Write Test Cases

Once tests are selected, test cases can be written based on the information in the potential test table. Input and output information can be moved into the Test Case template. Selected tests should be able to be combined into one or two test cases per Use Case.

The resulting test cases will be considered a test deliverable and will be included in the Inception Report.

Please see Appendix A for an example of a Test Case template.

3.4 Write Test Scripts

Each test case will correspond to at least one test script which will execute multiple tests. The scripts will identify set up, execution, and analysis steps required for testing. The scripts will be used by the testers to perform manual testing.

The test scripts are also considered test deliverables and will be included in the Inception Report.

Please see appendix A for an example of the Test Script template that will be use.

3.5 Test Case and Test Scripts creation for the 5 Primary Use Cases

Test cases and test scripts written for the functional components can be combined to form tests that cover delivered functionality of the 5 Primary Use Cases. This activity will be performed by test personnel and approve by the sponsor.

3.6 Usability Test Design and Development

Test design for Usability testing will focus on first identifying usability constraints that can be tested and then writing tests. Usability constraints that are testable will be identified by reviewing functionality stated in the SRS, the User Interface Requirements and Design document, and Use Case steps to get a feel for total functionality and ease of use needs. User sessions may be needed to elaborate constraints. Constraints will be communicated in quantifiable terms and will outline usability features such as number of clicks to reach a solution; ease of use based on type of user, and overall screen layout and functionality.

These will be presented to the sponsor for review. Upon review and approval, tests will be written.

There will be three types of Usability tests performed which are:

- Test against usability constrains
- Exploratory testing
- User Observation

Testing against constraints: For this testing, test team members will modify existing functional tests. Based on the User and the group of constraints identified, functional components will be identified and corresponding functional tests will be reviewed. From those tests new tests will be created which will have different expected results based on the identified constraints. The scripts will be performed by individuals representing the target user type. The test team will also perform these tests.

Exploratory Testing: This type of testing will be conducted by user type representatives with the intent of evaluating the ease of use. The users will be presented with general scripts prepared by the test team to use to guide through certain functionality. The individual testers are give liberty to stray from the test scripts as needed. Any perceived difficulties in navigation and ease of use will be identified and evaluated further by the test team and the sponsor as needed. Defects will be logged on identified problems.

User Observation: This testing will be conducted by the test team along with specific users. The Users will be given tasks to perform (general scripts). The test team will observe the users interacting with the system and take notes. These sessions will be followed up by interviews of the users by the test team to gather feedback on noted areas. From these interviews, defects will be identified and logged.

3.7 Test Execution

The following activities will take place in the test execution phase:

- Distribute Tests
- Execute Tests
- Record Results
- Track Progress
- Retest Defects

Test Distribution

Tests will be distributed to test execution participants by the test team based on functional area expertise. The testing effort will be coordinated by the testing team. It is expected that the test team also perform the bulk of the tests so overlap in test execution is expected. A work package of test scripts will be delivered to each test participant.

Execute Tests

The bulk of the testing will take place in XYZ city with some testing possibly being performed in other user locations. Testing will be conducted using test scripts. Results will be recorded in the scripts in the space allotted for Actual Results. Completed test scripts will be returned to the test team.

Record Results

The test team will be responsible for recording results for each test step and opening defects as needed using the test management tool.

Track Progress

The Lead Software Engineer is responsible for reporting test results and tracking test progress. This includes tracking defects and defect resolution progress. Most of the reporting will be done using the test management tool.

The Lead Software Engineer will provide a weekly status report on progress in testing and findings. This will be synchronized with reporting done by the sponsor to other project stakeholders.

Retest

Retesting will take place as defects are resolved and migrated. Regression testing will be conducted as needed based on the nature of the defect and code changed. Because updated code is expected during the testing process, regression testing will be performed during UAT.

3.8 Defect Tracking and Resolution Process

Defects will be entered and tracked using the test management tool. For each defect the severity level will be initially determined by the test team and test participants. The sponsor and area experts may be consulted to help determine severity level as needed. Note that the severity level and priority of a defect may be changed as necessary.

Defect severity levels.

The following is a description of defect severity levels.
Severity classification for UA

Defect severity	Project Definition
Critical	Defects that cause serious and disastrous consequences of the system in question or prevents a person from completing a task or one that blocks another task from being completed. These are defects that prevent further testing of the product or function under test. Examples include: An error that prevents other critical functionality from being performed or tested (If a time series path can’t be selected, functionality won’t be available to import a time series); loss of data; system or functionality unavailability; inappropriate or loss of security .
High/Major	Defects that cause significant consequences for the system in question. Defects that cause the functionality to fail to behave as expected and/or designed and no workarounds can be found or the cost of a work around is prohibitive either in terms of cost or effort. Examples of this include inaccurate calculations; the wrong field being updated; the inability to import data for a specific format which is deemed to be important to the project.
Medium	Defects that cause no major consequences for the system in question or the user can complete the task as described, but has some difficulty in doing so. With these defects the function tested does not function as expected/designed but workarounds may exist to achieve functionality. Examples include misleading error messages; displaying output in a format other than expected.
Minor	Defects that cause no negative consequences for the system in question. These are cosmetic defects that do not affect the functionality of the system. Such defects normally do not cause erroneous outputs. Examples include simple typos in documentation; bad layout or misspelling on screen.

Defect Resolution Process

Defects are expected to be worked on, resolved, and retested within a given timeframe based on severity. The following table shows the initial expectations.

Defect Category	**Development Team Response**	**Test Response**
Critical	Evaluate and give a response with a plan to fix within one business day. It is expected that the sponsor or test team will respond to the evaluation quickly with any feedback.	Test the defect within 1 business day of code migration for such a defect.

	Once a critical defect is fixed, communication should take place immediately in order to facilitate code migration.	
Major	Evaluate and give a response with a plan to fix within three business days. Migration of code to fix Major defects can be coordinated with the test team with the emphasis on moving multiple fixes in the same migration as opposed to immediate migration.	Test the defect within 2 business days of code migration for such a defect.
Medium	Response can be negotiated with the test team and sponsor based on work load of critical and major defects and the volume of medium defects.	Retesting of defects will be based on priority. The goal is to retest as soon as possible while concentrating on Critical and Major defects first.
Minor	Response can be negotiated with the test team based on to work load of other defects with higher severity.	Retesting of defects will be based on priority with defects of higher severity tested first.

Defect Status Meetings

Defect status meetings will be held periodically based on necessity. Participants will include developers, testers, the sponsor, and area experts as needed. The frequency of these sessions will be determined by the sponsor and the lead software engineer (test lead). The meeting will be facilitated by the test lead and is expected to last no longer than an hour. The focus of these sessions will be on the status of outstanding defects and issues related to testing such ass agreeing on defect criticality.

The preferred format for such meetings is face-to-face. This will not be possible for all individuals so the plan is to meet in a room whenever possible, with the people unable to meet in the room being present either by phone or Skype. On the occasions where most participants are unable to attend in person, Skype sessions will be employed.

4. Resources

This section presents the recommended human and environmental resources for the project.

4.1 Roles and Responsibilities

The following is a list of roles related to the testing process.

Test Lead

The Test Lead is a professional software test expert who plans all the testing. The Test Lead will interact with counterparts in the development team, database team, and QA to coordinate the test activities. The Test Lead is responsible for this UAT Plan document, for planning the testing, scheduling the work, and managing the test sessions. The Test Lead will also perform the same work as the Tester and will direct the work of the Tester. The Test Lead will prepare the final acceptance test report.

Tester

The Tester is a professional software test expert who prepares test scripts and executes the tests. The Tester reports on the results of the tests and collaborates with the Test Lead to make the final acceptance test report for each phase.

Technical Data Entry Person

The Technical Data Entry Person is an end user who may be asked to load a lot of data to support testing. This data will be used by other people in the study team, such as the GIS Expert. The Technical Data Entry Person will run tests in areas such as loading GIS data, loading time series data, and loading raster data.

GIS Expert

The GIS Expert is an end user who has significant expertise in working with GIS data and using software tools. The GIS Expert will run tests in areas such as loading shape files, viewing GIS data, and combining layers.

Manager

The Manager is an end user who wants to browse the data and see various reports or summaries of results. The manager will run tests in areas such as plotting time series and viewing simulation results.

4.2 Environmental Needs

Hardware and Software Requirements

Two versions of the software must be set up for test. The first is the professional version which includes laptops or desktops running the client application, and a server running the server application and the database on the same machine.

The second set up will be the corporate version where the application is running on one laptop and the database running on another machine.

There will be enough systems configured with the latest version of the software to support all test participants.

Location

Most testing will take place in XYZ city. A room will have to be set up with enough machines configured with the application to support testing with all participants. At this time it is unknown how many individuals will be performing tests.

Test Data Set Up

Data will have to be set up in order to perform tests. The test lead will coordinate with the development team and the sponsor to facilitate gathering and loading sufficient data.

Time and Availability of Testing

The bulk of the testing will take place in the first three weeks of November. Afterwards the number of people participating will be reduced.

5. Deliverables

The test lead will be responsible for a number of external deliverables that will be delivered to the project team. Power point presentations will be presented to the sponsor periodically showing progress on deliverables. These deliverables include:

- This test plan
- Test Cases
- Test Scripts
- Test Progress reports
- A User Acceptance Test report

Interim deliverables will also be created in the process of creating the above deliverables. These too will be submitted to the sponsor for review. Those include:

- Elaborated Use Cases
- Potential test tables

6. Release Acceptance Negotiation

This section discusses the process for accepting the release 1 product. The term

“Acceptance negotiation” takes into consideration that although acceptance criteria is set with the expectation of adhering to them, some level of compromise will take place. This section covers:

- Acceptance Criteria
- The Acceptance Process
- A process for handling outstanding defects accepted with a release

6.1 Acceptance Criteria

Defects will relate directly to test cases for functional testing. If a step in a test case fails, a defect is written for it without exception. So there will never be a situation where a test case fails but there are no defects. Since the test scripts are related directly to Use Cases, it corresponds that implemented functionality to support a Use Case will not be successful until all linked tests cases and scripts pass completely meaning all related defects are resolved.

The following acceptance criteria is a single set of criteria based solely on defect criticality and the number of defects. Component maturity importance was intentionally left out because those considerations will have been taken into consideration in the test selection process, and in defect assignment. Also, priorities will change as testing and discovery progress causing changes to criticality of defects.

The following are the release acceptance criteria.

- All test case will have been run
- No Critical Defects
- No Major Defects
- 10 or less medium defects with no more than four associated with a single functional component
- 15 Minor defects with no more than 6 associated with a single component

The reasoning for the numbers above goes as follows:

The product should not be released with any critical or major defects. These types of defects imply that the customer is associating critical or major issues to the release. If there are no issues of that nature related to acceptance of the product it should also be reflected in the criticality of the defects.

The numbers for acceptance of medium defects is based on the calculation that 10 defects equals less than 5% of all tests executed depending on the total number of test cases and the number of individual tests per test case. This number can be adjusted to a value the sponsor feels comfortable with when final test counts are available. The number of less than 4 defects associated with a given component implies stability of components and a lack of trouble spots.

The numbers for acceptance of Minor defects is based on the calculation that 15 defects equals less than 7% of all tests executed depending on the total number of test cases and the number of individual tests per test case. This number can be adjusted to a value the sponsor feels comfortable with when final test counts are available. The number of less than 6 defects associated with a given component implies stability of components and a lack of trouble spots.

6.2 The Acceptance Process

As the go-live date approaches, focus will be on the number of outstanding defects, the criticality of the defects, and how that relates to the acceptance criteria. Depending on the perceived value of certain functionality, the sponsor in conjunction with the users may agree to change the criticality of a defect. In other cases the actual acceptance criteria may be changed as more information feeds comfort or discomfort with the product.

In order to change the criticality of a defect, the suggestion can come from the sponsor, development team, or the test team, but the final decision will have to come from the sponsor with input from users. This is seen as an informal process that would not happen very often. The expectation is that there will not be a large number of outstanding defects that the customer is uncomfortable with.

In order to facilitate such a process an Acceptance Committee will be formed.

Examples of Acceptance Negotiation

Example 1: A defect is logged as Critical because it blocks functionality (and the ability to test) related to a specific tool that was initially deemed important at the beginning of the release. After further review, the system users and the sponsor agree that the functionality associated with the defect and blocked tool are not important for release 1 but necessary for release 2. The defect is reduced to Medium. When the release takes place the defect will be covered by a Change Request (CR) that will be scheduled into release 2.

Example 2: A defect was originally classified as a Major because the functionality is important and the work around for the functionality is time consuming. After review it is determined that the functionality is still important but will be used infrequently in release 1 making the work around feasible. Under those circumstances the decision is to change the defect to a Medium.

Example 3: After reviewing the existing defects the users and the sponsor agree that they are comfortable with the amount of minor defects although the count is over the limit outlined in the acceptance criteria. They agree to accept the release with the higher count of minor defects.

6.3 A Process for Handling Outstanding Defects Accepted with a Release

There will likely be some defects in the product when it is accepted. In order to keep track of those defects in later releases, it is recommended that the defects are changed to requirements. Those requirements would become change requests that are reviewed by the Change Control Board and scheduled for development like any other requirement.

When release 1 is ready to go live, and acceptance has been agreed upon based on the acceptance criteria and process outlined above, all outstanding defects must be dealt with. The existing defects that were agreed to be accepted as part of the release will be evaluated by the sponsor with assistance from the Lead Software Engineer and a CR will be written for each. These CRs will be scheduled for delivery in a future release. Each CR must be created, approved by the Change Control Board, and scheduled before the corresponding release 1 defect can be closed in test management tool. In essence these CRs become requirements of future releases. Once all defects have been closed for release 1, the product will be deemed ready to go live by the sponsor.

7. Testing Tasks

7.1 Tasks

Testing Tasks include functional and administrative tasks required to perform test and should incorporate test activities for each of the test efforts identified in the previous sections. The following is a list of milestones. Please note that this is to illustrate milestones and expectations. The project plan has a more detailed breakdown.

Milestone Task	Effort	Start Date	End Date
Plan Test	Review Existing test plan, the SRS, and the	20-09-2010	01-10-2010

	design documents. Add test process to the plan. Fill in details related to IEEE standards and what is known about release 1. Send for review Refine Gain approval		
Design Test	Elaborate functional components Create potential tests tables for each functional component Identify Coverage of each functional component and select tests	24-09-2010	08-10-2010
Implement Test	Write Test Cases Write Test Scripts	04-10-2010	15-10-2010
Execute Functional Tests	Run Tests Monitor testing Report	15-11-2010	23-11-2010
Execute Usability Tests	Run Tests Monitor testing Report	24-11-2010	30-11-2010
Evaluate Test		15-11-2010	19-01-2011

7.2 Staffing & Training

The testing team will consist of the Lead Software Engineer (LSW) and Software tester. Test participants will include individuals from the different types of users expected to use the software.

The only training associated with release 1 will be on the application and will be conducted on the first week of November.

7.3 Schedule

A detailed schedule has been worked out with the sponsor and will be part of the overall detailed project plan.

8. Appendix A: Templates

Test Case Template

Test Case Name/Identifier

Description:
This Test Case validates …. <This can be things such as nominal (happy) path of a UC, Negative tests, etc.>
Add info on how the TC starts and what happens at a general level

Objective:
List high level functionality verified or references to requirements>

- Verify the <system> software performs the functionality of the requirements allocated to this Test Case, per <reference>.
- Verify the <system> software performs the functionality of the <reference use case, sequence diagrams, functions, etc.

Test Items/Requirements Addressed:
List items to test by feature and corresponding requirements. Use requirement ID and description.
For each item, consider supplying references to the following test item documentation:
a) Requirements specification;
b) Design specification;
c) Users guide;
d) Operations guide;

Prerequisite Conditions:
List conditions including other test cases that must run, system states, etc. that must be in place for this test case to run

Test Inputs (Input Specifications)
Specify each input required to execute the test case. Some of the inputs will be specified by value (with tolerances where appropriate), while others, such as constant tables or transaction files, will be specified by name. Identify all appropriate databases, files, terminal messages, memory resident areas, and values passed by the operating system.

Expected Test Results (Output Specifications):

Specify all of the outputs and features (e.g., response time) required of the test items. Provide the exact value (with tolerances where appropriate) for each required output or feature
Criteria for Evaluating Results:
List any details related to result evaluation. Pass/fail.

Environmental needs
Hardware
Specify the characteristics and configurations of the hardware required to execute this test case.
Software
Specify the system and application software required to execute this test case. This may include system software such as operating systems, compilers, simulators, and test tools. In addition, the test item may interact with application software.
Other
Specify any other requirements such as unique facility needs or specially trained personnel.

Test Script Template

Test Script for <test case Name>

Test Script 1

Script Specific Files and Set up
List the files that need to be set up and their formats including specific values for this procedure. These can include configuration files and specific data files that may simulate specific situations.
Test Environment Set Up
Test environment setups are described here. Detailed instructions can be kept in other documents and referred to here.

Test Script Steps
The table below will list the steps taken to set up and run the tests. It documents manual activities.

Step#	TEST Step	EXPECTED RESULT	Pass/Fl Req	Partial	Comments
1	Script Step 1				
2	Script Step 2				
3		Expected Results for prior steps	Requirement covered		
4					

Test Evaluation Instructions