

An Intelligent Information Retrieval System based on Meta-Search Engine

Software Development Plan (Example)

Drumm, Paul M

Change History

Date	Summary	Person
3/31/07	Document started. Added Section 5 & 6	Paul Drumm
4/1/07	Added sections 2.4-2.11, excluding 2.6	Paul Drumm
4/2/07	Added Section 1	Paul Drumm
4/2/07	Added 7.1.1 to 7.1.9, Section 8	Paul Drumm
4/2/07	Added 2.6. Merged section 1, 7.1.1-7.1.9, and 8	Paul Drumm
4/2/07	Outlined section 12, Added section 10, modified section 2 to include allocated times and summary of resource hours.	Paul Drumm
4/3/07	Modified Section 7, Added draft of Section 9	Paul Drumm
4/3/07	Added section 11	Paul Drumm
4/4/07	Added several risks to section 5	Paul Drumm
4/4/07	Document format and style changes to various sections	Paul Drumm
4/4/07	Update minor changes to section 9 and 12	Paul Drumm
4/22/07	Minor changes made to 1.1, 1.2, 1.3, 2.5, 2.6, 2.7, 2.8, 2.9, 2.10 and 2.11.	Paul Drumm
4/23/07	Edits made to sections 5 through 12	Paul Drumm
4/24/07	Added Section 4	Paul Drumm
4/24/07	Added section 3	Paul Drumm
4/24/07	Edit various sections adding infrastructure descriptions. 2.6, 2.8,6.1,9, 11.1	Paul Drumm
4/25/07	Edited various sections dealing with infrastructure environment, added Diagram A and B	Paul Drumm
4/26/07	Modified section 11	Paul Drumm
4/27/07	Modified section 4. Added ASDP references	Paul Drumm
4/27/07	Modifications to various sections, including 2 and 3. Format changes	Paul Drumm
4/27/07	Edits made to various sections	Paul Drumm
4/30/07	Added Glossary.	Paul Drumm

Table of Contents

1	Project Objectives	1
1.1	Objective	1
1.2	Intended Key Features	1
1.3	Goals	1
2	Defining Project Artifacts and Deliverables	1
2.1	Vision and Scope Document.....	1
2.2	SRS	2
2.3	Architectural and Detailed Design Documents.....	2
2.4	List of COTS used	2
2.5	Source and executable code	2
2.6	Test Plan.....	2
2.7	Acceptance Plan.....	7
2.8	Periodic Reports Generated by Reporting System.....	10
2.9	Deployment Plan.....	12
2.10	User and operational Manuals	14
2.11	Customer Training Plan	15
3	Software Development Process Model.....	16
3.1	Model Selection	16
4	Defect Prevention Process	18
4.1	Defect Methodology	18
4.2	Defect Reporting	19
5	Risk Management Plan	19
5.1	Design complexity	20
5.2	Creeping requirements	20
5.3	Excessive schedule pressure	20
5.4	Low quality	21
5.5	Cost overruns	21
5.6	Inadequate configuration control	22
5.7	Inadequate user documentation.....	22
5.8	Excessive time to market	23
5.9	Team members not able to allocate time to the project	23
5.10	Technology changes and needs.....	23
5.11	Acceptance testing mechanisms are unclear.....	24
6	Change Management Plan	24
6.1	Change Management Plan	24
7	Iterative Work Breakdown Structure	26
7.1	Iteration 0 – IIRS-MSE Prototype WBS.....	26
7.2	Iteration 1- IIRS-ME WBS	26
8	Project Effort.....	29
9	Development Schedule (52 weeks).....	30
9.1	Iteration 0 – IIRS-MSE Prototype (7 weeks).....	30
9.2	Iteration 1- IIRS-MSE (45 weeks).....	30

10	Additional Resource Requirements	33
10.1	Development System	33
10.2	Staging System.....	34
11	Technology Infrastructure Extensions	35
11.1	What is the Base Infrastructure.....	35
11.2	Base and Extended Infrastructure	35
11.3	Resources at Each Tier.....	36
12	People Infrastructure Extensions	38
12.1	SQA.....	38
12.2	Configuration Management	38
12.3	Test Engineer	38
12.4	Deployment Engineer	38
12.5	Systems Engineer.....	39
12.6	DBA	39
12.7	Business Analyst.....	39
Appendix A:	Diagrams	A1
	Diagram A - People and Infrastructure Diagram.....	A1
	Diagram B – Infrastructure Responsibilities.....	A2
Appendix B:	Glossary.....	B1
Appendix C:	References	C1

1 Project Objectives

1.1 Objective

To develop a system that assists the fund managers and investors in gaining better knowledge and understanding of the rapidly changing financial markets by providing indexed and searchable information related to investment strategies.

1.2 Intended Key Features

- Both browser and stand alone graphical user interface
- Professional fund manager mode
- Simplified Investor mode
- Indexing system for fast text searches by keywords
- Indexes manually entered text documents and web documents.
- Text mining and data mining subsystems
- Ontology and dictionary used for text mining and better natural language support
- Integrates with third party search engine services
- Ability to send alerts via email after setting thresholds on research entities

1.3 Goals

The project artifacts are to be developed in incremental steps. Each increment will lay the foundation for succeeding iterations. Each iteration is not to exceed 4 weeks. Major features shall be broken down into smaller work items that should fit into four-week time intervals.

During each iteration requirements will be frozen before the architecture design phase shall begin. Requirements revealed or revised in the middle of an iteration shall be recorded into the change management system and addressed in a subsequent increment. Requirements that could potentially break the architecture shall be evaluated immediately and if necessary work will be halted to revise the requirements and the project plan before work could be resumed.

The relatively short iterative time intervals will allow for frequent artifact releases and running code produced early in the process and growth in functionality in designated increments. This will also allow for frequent customer demos and workshops that will provide valuable feedback to the process.

2 Defining Project Artifacts and Deliverables

2.1 Vision and Scope Document

See separate document.

2.2 SRS

See separate document.

2.3 Architectural and Detailed Design Documents

TBD

2.4 List of COTS used

The following list of commercial software packages will be included in the IIRS-MSE system.

1. SQL 2005 server
 - a. Database Engine
 - b. Analysis Engine
 - c. Integration Services
 - d. Reporting Services
 - e. Notification Services
2. Windows 2003 server
3. Windows 2003 IIS 6 web server services
4. Windows 2003 SMTP services
5. Nokia IPSO 530 Checkpoint Firewall

2.5 Source and executable code

The source code shall be delivered to the customer after final payment is made.

2.6 Test Plan

85 hours are allocated for each test iteration, in addition a maximum of 427 hours are set aside to debugging over the life of the project.

Test plans will be generated by the Microsoft Visual Studio Team System for Software Testers in association with Microsoft Team Foundation Server using the unit test code generator and authoring tools built in to the system (See Diagram A and B). Tests that are more complicated will be built first and then automated.

Microsoft Load Test Agent shall be automated and setup to test the desired number of users every time the software has to be re-tested which gives a repeatable test environment under load. The Load Test Agent in conjunction with Team System will be able to auto generate reports on a daily basis for casual analysis of any load and test issues.

Microsoft Visual Studio Team System for Testers, Developers and Architects in conjunction with Visual Studio Team Foundation Server, was chosen due to the following:

- Possibility to integrate the product into the Software Development Lifecycle (SDLC)

- The product's ability for configuration management
- Test generation and automated testing
- Work item tracking
- Build generation and tracking
- Automated reporting of all features that are measured
- Project tracking and integration with Microsoft Project Server

The system also contains templates for agile software development and CMMI.

All tests will be automated and scheduled to run after the automated build occurs on a nightly basis.

TI-1: SRS 5.1.1 The IIRS-MSE System shall not take more than 15 seconds to fully display a page.

Conceptual Test Case:

- a) Website - Click a link to another page or
- b) Website - select an object on that page that produces a post back event or
- c) Window Form - select menu item from list or
- d) Window Form - Click on object that causes event

Time Allocated for Testing: Estimated at 10 hours.

Time Allocated for Debugging: Estimated at 60 hours total.

Stop-Test Criteria: Test Pass Rate of 95 % of pages/forms to meet the 15-second criteria and Test Coverage to include 80% of forms and web pages.

TI-2: SRS 5.1.2 The IIRS-MSE System shall return query results in 15 second or less.

Conceptual Test Case:

- a) Website – Enter query parameters or
- b) Window Form - Enter query parameters

Time Allocated for Testing: Estimated at 2 hours.

Time Allocated for Debugging: Estimated at 25 hours total.

Stop-Test Criteria: Test Pass Rate of 95 % of pages/forms to meet the 15-second criteria and Test Coverage to include 100% of forms and web pages.

TI-3: SRS - 5.1.3 The IIRS-MSE system will be designed to support 200 concurrent users with estimated average session duration of 30 minutes.

Conceptual Test Case: An automated load agent will simulate 200 users on the system running automated regression tests.

Time Allocated for Testing: Estimated at 5 hours.

Time Allocated for Debugging: Estimated at 30 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage at 100% of 200 users.

TI-4: SRS- 5.1.4 Data cube builds, re-indexing, or large data imports shall not affect the other performance requirements.

Conceptual Test Case: TBD – We will have to select the data jobs and the quality performance requirements to perform the test.

Time Allocated for Testing: Estimated at 20 hours.

Time Allocated for Debugging: Estimated at 10 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 50 % of data jobs.

TI-5: SRS - 5.3.1 Access to the IIRS-MSE system shall be based on a secured role based system with encrypted passwords between 7 to 14 characters

Conceptual Test Case: a) If you add a user to the system with a password outside those parameters, it should prompt you. b) The system should force you to login if your session does not exist and you try to go to a password-protected page.

Time Allocated for Testing: Estimated at 1 hour.

Time Allocated for Debugging: Estimated at 10 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-6: SRS - 5.3.2 The IIRS-MSE system shall encrypt all sessions that are web based.

Conceptual Test Case: If you navigate to the site you should see that it forces SSL by seeing the “https://” prefix in the address bar.

Time Allocated for Testing: Estimated at 1 hour.

Time Allocated for Debugging: Estimated at 10 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-7: SRS - 3.1 Create/View Watch List - The user maintains a list of companies that are currently the focus of her research and investigation.

Conceptual Test Case: The a) web form and b) window form shall be navigated to that contains the Watch List functionality. A watch list will be created, modified, and added to.

Time Allocated for Testing: Estimated at 2 hours.

Time Allocated for Debugging: Estimated at 5 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-8: SRS - 3.2 Create/View Alerts - The user maintains a list of companies and triggers which will result in alerts to be issued to her.

Conceptual Test Case: The a) web form and b) window form shall be navigated to that contains the Alert functionality. An alert will be created based on the user’s criteria. The criteria then will be artificially set in order to trigger an alert. The email account will be checked to make sure an alert was generated.

Time Allocated for Testing: Estimated at 16 hours.

Time Allocated for Debugging: Estimated at 16 hours total.

Stop-Test Criteria: Test Pass Rate 80% and Test Coverage 50%.

Special Notes: There are too many variations of parameters and thresholds to validate every single scenario even under automated conditions. The system will have 25 different parameters checked and artificially set to a post threshold level.

TI-9: SRS - 3.3 Investigate Markets & Sectors - The user can categorize and further pinpoint her research and information gathering on particular industries that are part of her funds or are candidates to be added to the funds under her management.

Conceptual Test Case: The a) web form and b) window form shall be navigated to that contains the Research View functionality. A Research View will be created or modified. A sub-view will then be created under a main view.

Time Allocated for Testing: Estimated at 1 hour.

Time Allocated for Debugging: Estimated at 25 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

Time Allocated for Testing: Estimated at 10 hours.

Time Allocated for Debugging: Estimated at 60 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-10: SRS - 3.4 Searches for Financial Data

This feature is used to view current and historical financial data to analyze market position of the company that is the focus of her research.

Conceptual Test Case: A single query will be issued to search for financial data.

Time Allocated for Testing: Estimated at 1 hour.

Time Allocated for Debugging: Estimated at 40 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-11: SRS - 3.5 Search internal documents - this feature is used to search for information stored in the investment company's internal file storage and database.

Conceptual Test Case: The a) web form and b) window form shall be navigated to that contains the Search functionality. The control labeled "Search internal resources only" will be selected. A query will then be issued. The resultant data set will be reviewed to validate it only contains internal data sets. A series of 10 queries will be run and validated against for internal data sources.

Time Allocated for Testing: Estimated at 10 hours.

Time Allocated for Debugging: Estimated at 30 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-12: SRS - 3.6 Search and store analytical articles on the Web

This feature is used for a more thorough and deeper investigation of targeted companies using outside sources that would include financial magazines, university research material etc.

Conceptual Test Case: The a) web form and b) window form shall be navigated to that contains the Search functionality. The control labeled “Search external web resources only” will be selected. A query will then be issued. The resultant data set will be reviewed to validate it only contains external web data sets. A series of 10 queries will be run and validated against for external web data sources.

Time Allocated for Testing: Estimated at 4 hours.

Time Allocated for Debugging: Estimated at 15 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-13: SRS - 3.7 Search and store news.

This feature provides the fund manager with the ability to search and store any news published by online newspapers or domestic and international news organizations.

Conceptual Test Case: The a) web form and b) window form shall be navigated to that contains the Search functionality. The control labeled “Search external news feed resources only” will be selected. A query will then be issued. The resultant data set will be reviewed to validate it only contains external news feed resources. A series of 10 queries will be run and validated against for news feed resources only.

Time Allocated for Testing: Estimated at 1 hour.

Time Allocated for Debugging: Estimated at 15 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100 %.

TI-14: SRS - 3.8 Search and store research views.

This feature provides the fund manager or investor with the ability to group and store various metadata (hyperlinks, feed, and document id) about her various searches in a search panel.

Conceptual Test Case: The a) web form and b) window form shall be navigated to that contains the search functionality. A search query shall be issued. An individual query result will then be selected. The “Add to Research View” control will then be selected. A dialog shall give the user the ability to add it to an existing Research View or create a new one. The “Research View” page will then be navigated to and the specific Research View will be selected. This will show the available meta-data associated with that view. The meta-data will then be examined to ascertain that the correct link exists.

Time Allocated for Testing: Estimated at 1 hour.

Time Allocated for Debugging: Estimated at 60 hours total.

Stop-Test Criteria: Test Pass Rate 100% and Test Coverage 100%.

2.7 Acceptance Plan

All testing and reporting automation environment was mostly laid out in section 2.6. Again, this uses the Microsoft Visual Studio Team System and Team Foundation Server for test coverage and automated reporting (Diagram A and B). This is referred to as the automated test system and/or automated reporting system.

2.7.1 Scope - This plan covers the acceptance of the system based on the requirements laid out in the SRS sections 3, 5, and 6. All system requirements must be accepted for full acceptance of the system.

2.7.3 Testing Approach – Requirements that are in SRS sections 3, 5 and 6 shall be tested using an automated test generator. These results shall be automatically entered into to the acceptance test database and reported on by the automated test system.

2.7.4 Test Schedule – The acceptance test shall run through one complete cycle to ensure adequate testing. Total time for Acceptance Testing is 27 hours per iteration.

2.7.5 Problem Reporting – Problem reporting will automatically be recorded in the automated reporting system.

2.7.6 Resource Requirements – The plan shall use the Data Manager and the Security Officer as test verifiers. This will use 8 hours of their time.

2.7.7 Test Environment – Testing shall be carried out in the staging environment.

2.7.8 Post-Delivery Tests - Testing shall be repeated in the production environment.

2.7.9 Test Equipment – The only required test equipment outside of the environment is the automated test generator scripts. These automated scripts shall run through all the required acceptance criteria as specified in the SRS sections 3, 5, and 6.

2.7.10 Software Requirements - The Acceptance test plan shall take place in the staging environment with all software requirements from release 1 in place.

2.7.11 Hardware Requirements - The Acceptance test plan shall take place in the staging environment.

2.7.12 Test Identification – Mapping to quality and functional requirements along with the test method.

TI-1: SRS 5.1.1 The IIRS-MSE System shall not take more than 15 seconds to fully display a page.

Method: The automated test generator shall step through every page to test the page turn of the system.

Allocated Acceptance Time: 1 hour.

TI-2: SRS 5.1.2 The IIRS-MSE System shall return query results in 15 seconds or less.
Method: The automated test generator shall generate random natural language queries, straight text based queries, and meta-data retrieval queries for each type. The resulting data shall be put in the database and automatically reported on when all tests are completed.

Allocated Acceptance Time: 1 hour.

TI-3: SRS - 5.1.3 The IIRS-MSE system shall be designed to support 200 concurrent users with estimated average session duration of 30 minutes.

Method: Microsoft's Load Test Agent shall be run to simulate 200 concurrent users.

Allocated Acceptance Time: 1 hour.

TI-4: SRS- 5.1.4 Any cube builds, re-indexing, or large data imports shall not affect the other performance requirements.

Method: Several large data imports and cube builds will be run while testing Test ID's TI-1 through TI-3.

Allocated Acceptance Time: 1 hour.

TI-5: SRS - 5.3.1 Access to the IIRS-MSE system shall be based on a secured role based system with encrypted passwords between 7 to 14 characters.

Method – The automated test generator shall generate low, high, and inbounds passwords to test the functionality.

Allocated Acceptance Time: 1 hour.

TI-6: SRS - 5.3.2 The IIRS-MSE system shall encrypt all sessions that are web based.

Method: Network Monitor shall be run on a server to evaluate the packets coming inbound and out bound.

Allocated Acceptance Time: 1 hour

TI-7: SRS - 3.1 Create/View Watch List - The user maintains a list of companies that are currently the focus of her research and investigation.

Method: From the Watch List screen the automated test generator shall generate multiple watch lists and add various data to each watch list.

Allocated Acceptance Time: 1 hour.

TI-8: SRS - 3.2 Create/View Alerts - The user maintains a list of companies and triggers which will result in alerts to be issued to her.

Method: From the Create Alerts screen the automated test generator shall generate multiple alerts and then artificially set the threshold at an abnormal value.

Allocated Acceptance Time: 1 hour.

TI-9: SRS - 3.3 Investigate Markets & Sectors - The user can categorize and further pinpoint her research and information gathering on particular industries that are part of her funds or are candidates to be added to the funds under her management.

Method: TBD

Allocated Acceptance Time: 1 hour.

TI-10: 3.4 Search for Financial Data

This feature is used to view current and historical financial data to analyze market position of the company that is the focus of the research.

Method: An arbitrary query shall be generated based on company name to check report of company data.

Allocated Acceptance Time: 1 hour.

TI-11: SRS - 3.5 Search internal documents - this feature is used to search for information stored in the investment company's internal file storage and database.

Method: An arbitrary document search criteria will be entered into the system. Documents based on that search will be entered into the file system. The data will be replicated in the file system to see if the document shows up.

Allocated Acceptance Time: 1 hour.

TI-12: SRS - 3.6 Search and store analytical articles on the Web

This feature is used for a more thorough and deeper investigation of targeted companies using outside sources that would include financial magazines, university research material etc.

Method: The automated test generator shall retrieve a document from the web and store it in the database. The article will then be retrieved from the database system.

Allocated Acceptance Time: 1 hour.

TI-13: SRS - 3.7 Search and store news

This feature provides the fund manager with the ability to search and store any news published by online newspapers or domestic and international news organizations.

Method: The automated test generator shall retrieve a news article from the web and store it in the database. The news article will then be retrieved from the database system.

Allocated Acceptance Time: 1 hour.

TI-14: SRS - 3.8 Search and store research views.

This feature provides the fund manager or investor with the ability to group and store various metadata (hyperlinks, feed, and document id) about her various searches in a search panel.

Method: The automated test generator shall create a new view and then add various articles to it.

Allocated Acceptance Time: 1 hour.

2.7.13 Acceptance Test Report – A test report shall be generated upon completion of all acceptance tests.

Allocated Acceptance Time: 5 hours.

2.7.14 Corrective Action - If a test fails it will be sent to the Causal Analysis and Resolution committee to determine further corrective action and how to retest the defect.

Allocated Acceptance Time: TBD

2.7.15 Summary of Results - A resultant summary shall be automatically generated and put into the reporting system upon final entry of the CAR Committee.

Allocated Acceptance Time: 8 hours.

2.8 Periodic Reports Generated by Reporting System

The following reports will be automatically generated and emailed to selected project team members. The Automated Reporting System (ARS) is the Visual Studio Team Foundation Server and the Team Foundation Server report generator, which uses Microsoft SQL 2005 Reporting Services. All reports will be generated daily or weekly. Weekly reports that involves an overvalue threshold will be emailed to the proper team members as shown below. All documentation packages, requirement changes, work item changes, and schedule changes are put into the Team System. Estimated time to create the reports is 75 hours.

Type	Documentation Related	
Stakeholders	Business Analysts Project Managers Architect Lead developers.	
Reports	Changes to SDP Changes to SRS Changes to V&	Schedule Weekly Weekly Weekly

Type	Project Plan Related	
Stakeholders	Business Analysts Project Managers Architect	
Reports	Milestone Achievement and Progress Indicator Individual Task Achievement and Progress Indicator Overall Schedule Status Progress Indicator Risk Assessment – Change in Risk Factor Key Performance Indicators (KPI) for the Project Customized Practices Implemented based on defects found	Schedule Weekly Daily Weekly Daily Weekly Weekly

Type	Requirements Management	
Stakeholders	Business Analysts Project Managers Architect Lead Developers SQA	
Reports	Requirements changes and additions Use case changes and additions Regression test changes per day Use cases with correlated test case Functional Requirements correlating with test cases	Schedule Weekly Weekly Weekly Weekly Weekly

Type	Construction and Implementation Related
Stakeholders	Business Analysts Project Managers Architect Lead Developers Configuration Management Test Engineer Developers

Reports		Schedule
	Automated Code build statistics	Daily
	Static Code Analyzer statistics	Daily
	Source code check	Weekly
	Implementation statistics based on functional requirements	Weekly
	Use case implementations	Weekly
	Functional Requirement implementation	Weekly
	Test cases generated per use case and functional requirement	Weekly
	Cumulative SLOC generated	Weekly
	Cumulative Classes generated	Weekly
	Cumulative Functions generated	Weekly
	Automated Build Scripts generated	Weekly

Type Test and Defect Related
Stakeholders Project Managers
 Architect
 Lead Developers
 Test Engineer
 Developers
 Configuration Management

Reports		Schedule
	Defects found and categorized	Daily
	Defects resolved and categorized	Daily
	Overall Problem and defect tracking	Daily
	Automated Testing results based on code builds	Daily
	Causal Analysis and Resolution to Automated Code Testing	Weekly

1.

Type Deployment and Transition Related
Stakeholders Project Managers
 Architect, Lead Developers
 Test Engineer
 Configuration Management

Reports		Schedule
	Deployment scripts generated	Weekly
	Deployment test case builds	Weekly
	Deployment defects generated	Weekly

2.9 Deployment Plan

The product will be deployed to the customer site in five phases. Each phase will use the automated build and testing tools incorporated into Microsoft Visual Studio Team

System in order to reduce human error and also to automatically record and generate reports of each part of every phase.

The five deployment stages are:

1. Deploy staging and production hardware.
 - a. The staging hardware will consist of:
 - i. Two Microsoft Windows 2003 servers.
 - ii. One Windows 2003 Server for testing and deploying code.
 - b. The production hardware will consist of:
 - i. Four Microsoft Windows 2003 servers.
 - ii. One EMC SAN for shared storage.
 - iii. One checkpoint firewall.
 - c. Automated Testing
 - i. An automated test script from Microsoft Visual Studio Team System will run to test various features of the hardware. These scripts will use windows management instrumentation objects to validate RPC calls can be made between the servers and their services.
2. Deploy COTS components
 - a. The staging environment will have all the features of SQL server 2005 installed using a scripted deployment file. This includes:
 - i. Database Engine
 - ii. Analysis Engine
 - iii. Integration Services
 - iv. Reporting Services
 - v. Notification Services
 - b. The staging environment will have Microsoft IIS 6 installed on one of the web servers.
 - c. The production environment will have all components of SQL 2005 installed but in a clustered environment on two Microsoft Windows 2003 servers.
 - d. The production environment will have IIS 6 set up on two Microsoft Windows servers configured as Network Load Balanced.
 - e. Automated Testing – uses Microsoft Visual Studio Team System
 - i. An automated test script will run to test various features of the SQL 2005. Various T-SQL scripts will run to test the functionality of the server.
 - ii. Automated post and get statements will be scripted to test against the IIS server to test web service functionality.
 - iii. An automated test script will shutdown one of the clustered SQL servers and another part of the automated test script will see if it can still query the server.
3. The staging workstation will run MSBUILD (part of MS Visual Studio Team System) to automatically compile, configure, and push the web site code to the staging environment. The Java based custom windows application will be built using ANT.

- a. The code will be read out from source control.
 - b. An automated batch file will run using the Microsoft MSBUILD tool to compile, configure, and deploy the web site.
 - c. An automated batch file will institute ANT to build, configure, and deploy the java code. Feedback from the build logs will be imported into MS Visual Studio Team System. The network administrator will configure the local software distribution policy in the Microsoft Active Directory network to automatically push the software package to the desktop for any staging machines.
 - d. An automated test script will then run against the web server to validate that the application is running and that several test cases are successful when queried.
 - e. An automated test script will then be run against the user interface to determine if several high impact use cases will run correctly.
4. The production workstation will run MSBUILD to automatically compile, configure, and push the web site code to the staging environment. The Java based custom window application will be built using ANT.
 - a. The code will be read out from source control.
 - b. An automated batch file will run using the Microsoft MSBUILD tool to compile, configure, and deploy the web site.
 - c. An automated batch file will institute ANT to build, configure, and deploy the java code. The network administrator will configure the local software distribution policy in the Microsoft Active Directory network to automatically push the software package to the desktop for any staging machines.
 - d. An automated test script will then run against the load balanced web server to validate that the application is running and that several test cases are successful when queried.
 - e. An automated test script will then be run against the user interface to determine if several high impact use cases will run correctly.
5. The Data Manager, Security Officer, and SOX officer will be taken through an operational run through of the system. This is in addition to the training session they have already had on it. After they are satisfied with the operational run through and the system passes acceptance testing the system will then be released to them.

2.10 User and operational Manuals

User manuals will be developed and maintained throughout the software development life cycle. This includes:

- User Manual for each module of the project
- a training manual to be used during training sessions
- Data Manager's Guide to be used by the team responsible for server and data maintenance
- Security and Sox Officer guide to be used by the system administrators

The requirements and use cases are used as input to the documentation. For the technical writers to fully understand the system it is important that the architects, lead developers, configuration managers and SQA personnel guide and help the writers through the process.

Estimated work efforts for the different manuals are:

- User manual – 40 hours to prepare
- Training manual – 10 hours to prepare
- Data manager's guide – 5 hours to prepare
- Security and Sox Officer guide – 20 hours prepare

The user manual will serve as the foundation for the generated online HTML based help system.

2.11 Customer Training Plan

Customer Training shall be conducted two weeks prior to deployment. The intended audience for the training is fund managers, but there will also be additional training for the Security and SOX officers. The training session will run from 9 AM to 4 PM with one lunch break and two personal breaks. Training is scheduled for one day only.

The training shall be located off-site at a location within 30 minutes of the customer's corporate office. No company laptops will be allowed in the training sessions.

Materials shall be provided by the speaker. The user manual will be presented and a copy will be handed out to each participant. Preparation time for generating the meeting materials, not including the user manual, is 12 hours. That brings the estimated total for conducting the customer training to 20 hours.

The one-day training session shall be divided into five parts.

1. Brief overview of the system:
This section shall provide an overview on the physical and logical structure of the system at an abstract level. It will also mention the various sources of data, how data is stored and the quality of the data. Basic user interactions, such as logging into the system, will be covered in this section.
2. Search Features:
This will cover how to use the search feature. This will include how to use natural language searching, what types of words to put into the system for best results and how to select what areas of the system to search (documents, database, web)
3. Saving and grouping search meta-data:
This section will cover how to save metadata from a search and save it in logical groupings. The "Research View" feature will allow the end user to

create and modify meta-data and retrieve the previously searched for documents from the appropriate data source

4. Setting up alerts:

This section will cover how to setup alerts on various meta-data retrievals. If the previously searched item has changed based on some threshold the system can notify the end user about it

5. Adding documents:

This section will cover the steps involved in adding a document to the system. The end user, not including investors, may add documents to the data store using the centralized scanner. Pending approval from the SOX officer, the document will only be visible and searchable to the user who added it.

There will be a separate training session scheduled to immediately follow the fund manager session. The focus for the later session shall be user setup, auditing and monitoring. The session is aimed at the security and SOX officers. The fund managers will not be required to attend this session.

3 Software Development Process Model

3.1 Model Selection

After conducting an evaluation of several process models, the project team concluded that the Agile Unified Process (AUP) model would best fit the needs of the project and the background of its members.

Due to a high level of collective technical and domain knowledge possessed by the project team, it was decided to follow the AUP model. There was a common understanding among the team members that they did not need to read detailed process documentation, though they acknowledged that they preferred to have everything outlined concisely using a handful of pages. The team also decided that the best approach was to focus on the activities that directly produce project deliverables and not every possible thing that could happen or go wrong before and after the product's implementation. Thus it was decided to minimize the efforts put into risk management and risk analysis.

The Agile Unified Process is a simplified version of the Rational Unified Process . It describes a simple, easy to understand approach to developing business application software using agile techniques and concepts.

To improve productivity, AUP applies agile techniques including test driven design (TDD) which is a technique that involves repeatedly first writing a test case and then implementing only the code necessary to pass the test. It also applies agile modeling, agile change management, and database refactoring which involves making a simple

change to a database schema that improves its design while retaining both its behavioral and informational semantics.

AUP is made up of four phases:

1. Inception

Identify the initial scope of the project, a potential architecture for the system, and obtain initial project funding and stakeholder acceptance.

2. Elaboration

Prove the architecture of the system.

3. Construction

Build working software on a regular, incremental basis that meets the highest-priority needs of project stakeholders.

4. Transition

Validate and deploy the system into the production environment.

In addition it offers seven disciplines:

- 1. Model.** Understand the business of the organization, the problem domain being addressed by the project, and identify a viable solution to address the problem domain.
- 2. Implementation.** Transform model(s) into executable code and perform a basic level of testing, in particular unit testing.
- 3. Test.** Perform an objective evaluation to ensure quality. This includes finding defects, validating that the system works as designed, and verifying that the requirements are met.
- 4. Deployment.** Plan for the delivery of the system and to execute the plan to make the system available to end users.
- 5. Configuration Management.** Manage access to project artifacts. This includes not only tracking artifact versions over time but also controlling and managing changes to them.
- 6. Project Management.** Direct the activities that take place within the project. This includes managing risks, directing people (assigning tasks, tracking progress, etc.), and coordinating with people and systems outside the scope of the project to be sure that it is delivered on time and within budget.
- 7. Environment.** Support the rest of the effort by ensuring that the proper process, guidance (standards and guidelines), and tools (hardware, software, etc.) are available for the team as needed.

Based on previous experiences with the AUP model the team felt that it would ultimately give the best chance to deliver a working product given the project schedule. The team considered the need for explicit AUP training to be non-existing.

4 Defect Prevention Process

4.1 Defect Methodology

The team decided to apply the principles, policies and practices of the ASDP to help prevent defects.

The goal of the ASDP is to help produce high quality products, increase productivity and operational quality by enforcing a controlled, improved and sustainable process. This can be accomplished by satisfied employees who use automated processes and make decisions based on information stored in and reported by the infrastructure in place.

By establishing a defect prevention mindset in the team, and by analyzing the root causes of the defects, the defects are prevented from re-occurring in future iterations.

The first principle of the ASDP is to “establish a infrastructure that integrates people and technology”[Huzinga & Kolawa] By defined roles, providing the required technology, a structured and defined interaction between people and technology, the team hopes to reduce bugs by detecting and fixing the root causes of the problems as early as possible. As a part of fixing the root cause of a problem, it will be necessary to change the process itself. If possible, these changes shall be introduced to a subset of the project before applied to the overall process. The goal is to gain acceptance from key members before presenting the change to the masses.

As an all Microsoft shop, the team has decided to use the Microsoft Team System and the Microsoft Team Foundation Server (TFS) (Diagram A and B) as the heart of the infrastructure. The team foundation server provides the team with the necessary configuration management system, automated build and automated test system to meet the infrastructure, automation and reporting principles of the ASDP.

The TFS system will be the collaboration HUB for the team members. By gathering fragmented information and making it available on the WEB through tight integration with the IDE it will help track bugs, documents and source code. As such, the TFS system will provide build automation, change management, reporting, work item tracking, version control and process and applications metrics, and master project integration via a connector to a MS Project Server.

Incremental introduction of best practices and tailoring the best practices to the development process and the project is considered a vital point to success. What methods and best practices to be introduced, will be decided at a later stage.

A four hour training and introduction to ASDP shall be conducted prior to entering the elaboration phase.

Prior to writing any business logic, the developers are required to write unit tests. These tests must then be automated and run daily as part of the nightly build.

4.2 Defect Reporting

The TFS (Diagram A and B) will provide the team and the project management with tracking and process and application metrics. As part of the nightly build, the unit tests are run and the results logged into TFS.

On a daily basis, the developers are required to evaluate the reports on the build results and the report generated from the automated unit tests.

The project manager is on a weekly basis required to evaluate the following reports:

- Unit tests results
- Open and closed bugs
- Time spent vs. estimated time
- Open and close requirements
- Requirements by priority
- Source code statistics
- Project process

The SQA team is not only responsible for defect detection but they are also involved in the defect prevention process. When defects are encountered, they must be reported to the project's defect tracking system - TFS. Then entry shall include who detected the defect, what build the defect was detected in, a brief description of the defect and if possible a description on how to reproduce the defect.

Every morning the project team shall view and analyze the defects via reports from the TFS. Defects shall be prioritized and activities shall be assigned to team members in TFS.

5 Risk Management Plan

The Risk Management Plan (RMP) will be put under source control in the Visual Studio Team System. The RMP shall be re-evaluated at every major milestone and at priority checkpoints in deemed necessary. The changes to the RMP will be reported on a weekly basis via the Visual Studio Team System report generator.

5.1 *Design complexity*

5.1.1 Impact

In case of complex design, the project might experience delay in delivery and cost overrun.

5.1.2 Probability of Occurrence

The team consists of experienced developers who have been involved in similar projects in the past. Probability of occurrence is thus rated low.

5.1.3 Mitigation Plan

The team will re-use modules developed and designed in other projects. The team will also use COTS where applicable.

5.2 *Creeping requirements*

5.2.1 Impact

Creeping requirements could lead to increased cost, delayed delivery and in the end project cancellation if the project is never able to produce the final deliverables.

5.2.2 Probability of Occurrence

Being able to discover all the requirements upfront is nearly impossible. New and revised requirements will be expected to emerge at any point during the project. Probability of occurrence is high. Requirement changes will be monitored via the automated report generator in Visual Studio Team System. Thresholds will be set for a high rate of change to alarm stakeholders.

5.2.3 Mitigation Plan

Develop a horizontal prototype early in the project. Both the customer and end-user representatives will be allowed to experiment with the prototype. By allowing the customer and end-user representatives to use the prototype it is expected that new and unimplemented requirements will be discovered.

Introduce an iterative development process with short iterations where each iteration results in a set of deliverables together with a sound requirements management plan.

5.3 *Excessive schedule pressure*

5.3.1 Impact

Excessive schedule pressure could result in unmotivated employees and unrealistic customer expectations. This could in turn lead to high employee turnover and an unsatisfied customer.

5.3.2 Probability of Occurrence

Despite the fact that the company has developed several similar projects in the past and has a highly skilled staff, over commitment is still considered a danger. Probability of occurrence is rated medium. Work Item shortages and project plan missed goals will be generated on a weekly basis by the Visual Studio Team system automated report generator. Thresholds will be set for a high rate of missed deadlines to alarm stakeholders.

5.3.3 Mitigation Plan

Several project estimation methods and best practices will be used including Elements of Wideband Delphi and COCOMO II.

5.4 Low quality

5.4.1 Impact

Low quality could lead to an unsatisfied customer and potentially cost and schedule overruns.

5.4.2 Probability of Occurrence

Several quality attributes exist, what is important to one customer might not be important to another. Since the team has extensive knowledge from working on similar projects in the past, the probability of occurrence is rated medium.

5.4.3 Mitigation Plan

Quality Attribute Scenarios shall be developed as a part of the inception and elaboration phase.

Defect prevention best practices, both automatic and manual, shall be implemented as part of the software development process model chosen for the project.

5.5 Cost overruns

5.5.1 Impact

Cost overruns will reduce the company's profit on the project and could potentially lead to a financial loss. Severe overruns will lead to cancellation of the project and in the worst-case scenario bankruptcy for the company.

5.5.2 Probability of Occurrence

Software projects are notoriously difficult to estimate and control. Prior experience and metrics are important input parameters to the project effort estimation. The company's record of accomplishments and its employees' experience denote that the probability of occurrence is medium.

5.5.3 Mitigation Plan

The architectural design and the project skeleton shall be developed early in the project, COTS to be used shall be identified and project effort estimation shall be performed using methods such as Elements of Wideband Delphi and COCOMO II.

5.6 *Inadequate configuration control*

5.6.1 Impact

Inadequate configuration tool or a lack of a configuration tool would make it impossible to keep track of changes, and provide process metrics. It would be nearly impossible to deliver a high quality product on time and on budget on a large project.

5.6.2 Probability of Occurrence

The team will be using the Microsoft Team Foundation server and the Team System as its configuration tools. The tools are new to the market and have never been used by Global Inventory Inc. Since it is very hard to predict how well suited these products are for the project team members and the development model chosen for the project, the probability of occurrence is rated high.

5.6.3 Mitigation Plan

A small team will be assembled to evaluate the new Team Foundation Server prior to project startup. One employee will be assigned the role of Configuration Tool Manager. This person will go through extensive training held by the product vendor.

The old configuration control system shall be kept intact through for older projects. The Team Foundation Server shall be used in the inception and elaboration phase for both documentation, source code of the prototype, test generation, build generation, and automated reporting. This will allow for a gradual introduction of the system to other projects in the organization.

5.7 *Inadequate user documentation*

5.7.1 Impact

In case of inadequate user documentation, the end-user might find the system hard to use and the system could potentially be used incorrectly. End user understanding of the system would be limited and the product's quality and its adherence to the requirements might be perceived as unsatisfactory.

5.7.2 Probability of Occurrence

Discrepancy between requirements, implementation and the documentation might occur.

5.7.3 Mitigation Plan

Documentation team will be included early in the project. The documentation team will be included in the inception and the elaboration phase to increase their understanding of the requirements and the business. The documentation team will also be included in the

design of the test scripts and in the integration testing. They will be designing and authoring all test scripts in the Visual Studio Team System with Team Foundation Server.

5.8 Excessive time to market

5.8.1 Impact

As described in other risks excessive time to market could lead to project cancellation, cost overruns, and profit loss and in a worst-case scenario bankruptcy.

5.8.2 Probability of Occurrence

Several identified risks could lead to a delayed project and the probability of occurrence is considered high.

5.8.3 Mitigation Plan

Introduction of an acknowledged software development model, best practices, formal requirements gathering process, change management process and configuration management process are just some of the actions to be implemented to mitigate the risks of a slipping schedule.

5.9 Team members not able to allocate time to the project

5.9.1 Impact

The company has other projects that require maintenance and ongoing development. It is thus essential for the project's success that the right resources are able to contribute to the project at the scheduled time. If not the project could experience a slipping schedule.

5.9.2 Probability of Occurrence

The project requires specialized knowledge that are a limited resource, thus the availability becomes a bottleneck. Probability of occurrence is rated high.

5.9.3 Mitigation Plan

Revise team composition if necessary, have finer grained tasks and consider backup resources.

5.10 Technology changes and needs

5.10.1 Impact

System delivery shall be delayed or perhaps impossible to develop. Increases in the cost of COTS or the hardware.

5.10.2 Probability of Occurrence

The computer and software industry is continuously changing. Changes to the technology used must be expected and accepted. Usually changes introduce cost savings or process

improvements, and will have a positive impact on the project. Probability of occurrence is thus considered low.

5.10.3 Mitigation Plan

Proof-of-concepts and prototypes developed early before we commit. Upfront review of market and merging technologies.

5.11 Acceptance testing mechanisms are unclear

5.11.1 Impact

In case of unclear acceptance testing mechanisms, the whole defect prevention process and the test procedures and scripts might be incorrectly designed and implemented. If the system is not accepted by customer, additional time must be spent on testing.

5.11.2 Probability of Occurrence

High

5.11.3 Mitigation Plan

Make sure that we can do acceptance testing as part of the elaboration phase.

6 Change Management Plan

6.1 Change Management Plan

The change management plan defines responsibilities, policies, guidelines and procedures necessary for controlling and managing technical changes to the product. All changes to documentation will be tracked in the Visual Studio Team System and Team Foundation Server in order to automatically track, mitigate risk, and report on changes.

The guidance outlined in this plan must be incorporated into the development process and management decision-making processes throughout the entire system's life cycle from concept and requirements identification to development, system operations and maintenance.

Change management is a strict discipline in which all stakeholders must participate and co-operate. Effective Change Management requires active participation from all stakeholders and the process needs to be fully supported and endorsed by top management

The goal of the change management plan is to ensure that all changes are

- Necessary
- Documented correctly
- Evaluated to consider interfaces
- Evaluated against available resources

- Evaluated to deliberate cost vs. benefit, schedule and performance trade-offs
- Communicated to the Help Desk and the user community

The project manager is responsible for overseeing the change management process. He is responsible for tracking changes and making sure that the process is conducted according to the plan.

Infrastructure:

The heart of the change management process is the online change management application based on Microsoft Team Foundation Server.

When entering a new Change Request (CR) the stakeholder is required to fill in the following fields:

- CR originator name and organization
- Product(s) affected
- Change type including its severity
- Change description
- Change priority
- Related changes

In addition, the project manager and developers can enter

- Implementation responsibility
- Estimated and actual LOC (new, changed and deleted)
- Programmer hours (estimated and actually spent)
- Test requirements

Other data tracked in the CR:

All changes are logged and stored in an online database, including CR number, date opened, date closed.

The Change Control Board (CCB) is responsible for properly considering and coordinating the change requests. The board consists of members not only from development but also from test, documentation and support teams. Depending on the project's phase and stakeholders' feedback, the workload can become excessive. Therefore, it is recommended that the CCB have at least weekly meetings. Meetings that are more frequent should be considered in case of extreme workloads.

By recording and tracking the change request, problem reports, CCB actions and activities the CR system can be used to provide process metrics. The system can provide metrics on pending requests, closed requests, number of and what changes currently assigned to whom, changes by module, number of CRs closed during the last week, month, year etc.

7 Iterative Work Breakdown Structure

Our WBS is defined using a hierarchical tree structure. The WBS will define a set of planned outcomes, major deliverables and accomplishment that will collectively represent a 100% of the project scope to the point where the only details remaining are actions. Our WBS details will be progressively refined before work begins on an element of work.

7.1 *Iteration 0 – IIRS-MSE Prototype WBS*

7.1.1 Prototype Data-Mining

7.1.2 Prototype Indexing via Full Text Search and Text Mining

7.1.3 Prototype Data Transformation of Incoming Data into Database and Analysis Service Cubes/Fact Tables

7.1.4 Prototype Ontology Schema in Snowflake Fact Tables and Data-Mining

7.1.5 Prototype Web Site

7.1.6 Prototype Window Based UI

7.1.7 Prototype Validation

7.2 *Iteration 1- IIRS-ME WBS*

7.2.1 Requirements and Planning

7.2.1.1 Vision and Scope Document

7.2.1.2 SRS Document

7.2.1.3 SDP Document

7.2.1.4 Project Management

- Project initiation
- Project planning or design
- Project execution
- Project monitoring and controlling
- Project completion

7.2.2 Design

7.2.2.1 Financial data feeder

- Provide current price/position
- Provide historical prices/position

7.2.2.2 Database that stores retrieved data

- Provide necessary hardware/software and system configuration specification.
- Search Local Database
- Store data into the Database

7.2.2.3 Data mining capability

- Search for data patterns using meta-search engine
- Index database sourced data patterns found

7.2.2.4 Text mining capability

- Search for news text sources using meta-search engine
- Search for analytical web documents using meta-search engine
- Index retrieved documents

7.2.2.5 General search engine

- Research providers of meta-search engine and text mining capability
- Evaluate search engine products
- Select a meta-search engine product.

7.2.2.6 Ontology capability

- Build basic ground level objects
- Identify/build classes of objects
- Identify/build attributes that objects can have
- Identify/build relationships and ways object can relate to one another

7.2.2.7 Browser and stand alone GUI presentation

- Browser based/stand alone presentation of financial data
- Browser based/stand alone presentation of indexed new articles
- Browser based/stand alone presentation of data patterns
- Browser based/stand alone presentation of analytical text documents

7.2.2.8 System Integration

- An incremental integration plan
- Well defined component interfaces
- Architectural design tactics that support integrability

- Production plans for making final product from core assets
- Pre-integrating as many components as possible

7.2.3 Implementation and Test

7.2.3.1 Financial data feeder

- Provide current price/position
- Provide historical prices/position

7.2.3.2 Database that stores retrieved data

- Provide necessary hardware/software and system configuration specification
- Search Local Database
- Store data into the Database

7.2.3.3 Data mining capability

- Search for data patterns using meta-search engine
- Index database sourced data patterns found

7.2.3.4 Text mining capability

- Search for news text sources using meta-search engine
- Search for analytical web documents using meta-search engine
- Index retrieved documents

7.2.3.5 General search engine

- Research providers of meta-search engine and text mining capability
- Evaluate search engine products
- Select a meta-search engine product

7.2.3.6 Ontology capability

- Build basic ground level objects
- Identify/build classes of objects
- Identify/build attributes that objects can have
- Identify/build relationships and ways objects can relate to one another

7.2.3.7 Browser and stand alone GUI presentation

- Browser based/stand alone presentation of financial data
- Browser based/stand alone presentation of indexed new articles
- Browser based/stand alone presentation of data patterns
- Browser based/stand alone presentation of analytical text documents

7.2.3.8 System Integration

- An incremental integration plan

- Well defined component interfaces
- Architectural design tactics that support integrability
- Production plans for making final product from core assets
- Pre-integrating as many components as possible

8 Project Effort

We decided to use Effort Analogy to estimate the effort required to complete this project due to our company's extensive experience in development of search engine products. We used two particular projects as a baseline for estimating efforts.

Project "Search Engine" which was done for the University of Oceanside research library with 95% similarity in core functionalities. This project completed in 135 Person Months. We assigned a project manager, a lead architect and two development teams one of which with three developers and the other with four. Each team had one lead developer member. The project completed in 15 months from the time contracts was signed.

The other project used was project "Ultimate Search Engine" which was done for ABC investments with 95% similarity in core functionalities also. This project completed in 128 Person Months. For this project, we had assigned a project manager, a lead architect and two development teams with three developers in each team. Each team had one lead developer member. This project completed in 16 months from the time contracts was signed.

For project "IIRS-MSE", we estimate that in order to complete this project in 132 Person Months. In order to deliver this product in 12 months we have decided to assign one project manager, one lead architect, and three teams of developers with three members each. Each team will have a lead developer as a member.

<i>Project: Operation Search Engine</i>	<i>No of persons Assigned</i>
<i>Project manager:</i> 15 PM	1
<i>Architect:</i> 15 PM	1
<i>Developer:</i> 105 PM	7
<i>Total:</i> 135 PM	9

<i>Project: Ultimate Search Engine</i>	<i>No of persons Assigned</i>
<i>Project manager:</i> 16 PM	1
<i>Architect:</i> 16 PM	1
<i>Developer:</i> 96 PM	6
<i>Total:</i> 128 PM	8

<i>Project: IIRS-MSE</i>	<i>No of persons Assigned</i>
<i>Project manager:</i> 12 PM	1
<i>Architect:</i> 12 PM	1
<i>Developer:</i> 108 PM	9
<i>Total:</i> 132 PM	11

9 Development Schedule (52 weeks)

The development schedule will be closely tracked using the work item tracking features of the Microsoft Team Foundation Server. This brings order and consistency to the various tasks lists, project plans, issues and requirements that are part of the software project with integrated Work Item Tracking. All work item progress is reported and updated every time an item is checked into the system or when a percentage of a work item is updated in the system. This is also integrated into the project plan via Microsoft Project server using the Team System-Project Server connector, which allows for automated updating and tracking of the project plan as the work items are completed. Each completed section marks a relevant milestone where a checkpoint is made and the resources and schedule reassessed.

9.1 *Iteration 0 – IIRS-MSE Prototype (7 weeks)*

9.1.1 Prototype Data-Mining – 1 week

9.1.2 Prototype Indexing via Full Text Search and Text Mining – 1 week

9.1.3 Prototype Data Transformation of Incoming Data into Database and Analysis Service Cubes/Fact Tables – 1 week

9.1.4 Prototype Ontology Schema in Fact Tables for Snowflake Cube – 1 week

9.1.5 Prototype Web Site – 1 week

9.1.6 Prototype Window Based UI – 1 week

9.1.7 Prototype Validation – 1 week

9.2 *Iteration 1- IIRS-MSE (45 weeks)*

9.2.1 Requirements and Planning – 5 weeks

9.2.1.1 Vision and Scope Document - 1 week

9.2.1.2 SRS Document – 2 weeks

9.2.1.3 SDP Document – 1 week

9.2.1.4 Project Management – 1 week

- Project initiation – 1 day
- Project planning and design -2 days

- Project execution – 2 days
- Project monitoring and controlling – 1 day
- Project completion - 1 day

9.2.2 Elaboration and Design – 8 weeks

9.2.2.1 Financial data feeder – 2 day

- Provide current price/position – 1 day
- Provide historical prices/position – 1 day

9.2.2.2 Database that stores retrieved data – 2 days

- Provide necessary hardware/software and system configuration specification. – 1 day
- Search Local Database and Store data into the Database – 1 day

9.2.2.3 Data mining capability – 2 weeks

- Search for data patterns using meta-search engine – 1 week
- Index database sourced data patterns found – 1 week

9.2.2.4 Text mining capability – 1 week

- Search for news text sources using meta-search engine – 3 days
- Search for analytical web documents using meta-search engine – 3 days
- Index retrieved documents – 1 day

9.2.2.5 General search engine – 3 days

- Research providers of meta-search engine and text mining capability – 1 day
- Evaluate search engine products – 1 day
- Select a meta-search engine product. – 1 day

9.2.2.6 Ontology capability – 2 weeks

- Build basic ground level objects – 6 days
- Identify/build classes of objects – 4 days
- Identify/build attributes that objects can have – 3 days
- Identify/build relationships and ways object can relate to one another – 4 days

9.2.2.7 Browser and stand alone GUI presentation – 1 week

- Browser based/stand alone presentation of financial data – 2 days
- Browser based/stand alone presentation of indexed new articles – 1 day
- Browser based/stand alone presentation of data patterns – 1 day
- Browser based/stand alone presentation of analytical text documents – 3 days

9.2.2.8 System Integration – 1 week

- An incremental integration plan – 2 days
- Well defined component interfaces – 1 day
- Architectural design tactics that support integrability – 2 days
- Production plans for making final product from core assets – 1 days
- Pre-integrating as many components as possible – 1 days

9.2.3 Construction - Implementation and Test – 27 weeks

9.2.3.1 Financial data feeder – 1 week

- Provide current price/position – 4 days
- Provide historical prices/position – 3 days

9.2.3.2 Database that stores retrieved data – 3 weeks

- Provide necessary hardware/software and system configuration specification – 1 week
- Search Local Database – 1 week
- Store data into the Database – 1 week

9.2.3.3 Data mining capability – 6 weeks

- Search for data patterns using meta-search engine – 4 weeks
- Index database sourced data patterns found – 2 weeks

9.2.3.4 Text mining capability – 5 weeks

- Search for news text sources using meta-search engine – 1 week
- Search for analytical web documents using meta-search engine – 3 weeks
- Index retrieved documents – 1 week

9.2.3.5 General search engine – 5 weeks

- Evaluate search engine products – 3 weeks
- Select a meta-search engine product. – 2 weeks

9.2.3.6 Ontology capability – 6 weeks

- Build basic ground level objects – 2 weeks
- Identify/build classes of objects – 2 weeks
- Identify/build attributes that objects can have – 1 week
- Identify/build relationships and ways objects can relate to one another – 1 week

9.2.3.7 Browser and stand alone GUI presentation – 16 weeks

- Browser based/stand alone presentation of financial data – 8 weeks
- Browser based/stand alone presentation of indexed new articles – 1 week
- Browser based/stand alone presentation of data patterns – 4 weeks

- Browser based/stand alone presentation of analytical text documents – 3 weeks

9.2.3.8 System Integration – 2 weeks

- An incremental integration plan – 2 days
- Well defined component interfaces – 3 days
- Architectural design tactics that support integrability – 4 days
- Production plans for making final product from core assets – 4 days
- Pre-integrating as many components as possible – 1 day

9.2.4 Transition – 3 weeks

9.2.4.1 Customer Training – 1 day

9.2.4.2 System Acceptance Testing – 1 week

9.2.4.3 Deployment – 2 weeks

10 Additional Resource Requirements

IIRS-MSE will require two additional working environments beside that of the production environment. One environment will be used to stage the system and the other to develop the system. This in addition to the infrastructure requirements called out in sections 11 and 12. The development system and the staging system are both based on Visual Studio Team System with Team Foundation server in order to record and report on any issues in the development or staging systems. It is also used to build and deploy code and report any defects during build or deploy.

10.1 Development System

The development system will be used for

- Development – This will be a shared development system with all developers using the various web, database, and application servers
- Qualification for moving to the Staging System – code will be staged in this environment in preparation for moving to the STAGING environment
- Initial Test – All initial testing will be done in the development environment

The development environment is on similar hardware and software as compared with production but there will be no Clustered Services or Load Balanced Services.

- a) Physical Environment
 - a. 3 Compaq DL 380 servers
- b) Operating Environment
 - a. Windows 2003 STD SP2 on the servers

- c) Software and Applications
 - a. Server DEV_D - SQL server 2005 Developer Version with all features installed
 - b. Server DEV_D - Visual Studio 2005 Team Edition Load Test Agent
 - c. Server DEV_W - IIS 6 Web and SMTP services
 - d. Server DEV_W - Visual Studio 2005 Team Edition Load Test Agent
 - e. Server DEV - Visual Studio 2005 Team Edition for Software Developers
 - f. Server DEV - Visual Studio 2005 Team Edition for Database Developers
 - g. Server DEV - Visual Studio 2005 Team Edition for Software Testers
 - h. Server DEV - Visual Studio 2005 Team Edition Load Test Agent
 - i. Server DEV - Visual Studio 2005 Team Edition for Architects
 - j. Server DEV - Visual Studio 2005 Team Edition Foundation Server
 - k. Server DEV – NetBeans IDE
- d) Estimated time to build out environment is 80 hours

10.2 Staging System

The staging system will be used for

- Staging Code – Code will be brought out of source control and automatically compiled via MSBUILD or ANT, and deployed to the various servers and workstations
- Final Test and SQA – All final testing and SQA will be done in staging area
- Acceptance Testing – Acceptance Testing will be done in the staging environment
- Deployment To Production – After code has gone through acceptance testing it can then be automatically deployed to the production environment via the automated build system MSBUILD or ANT

The staging environment is on similar hardware and software as compared with production but there will be no Clustered Services or Load Balanced Services.

- a) Physical Environment
 - a. 3 Compaq DL 380 servers
- b) Operating Environment
 - a. Windows 2003 STD SP2 on the servers STAGE_D and STAGE_W and STAGE
- c) Software and Applications
 - a. Server STAGE_D - SQL server 2005 Developer Version with all features installed
 - b. Server STAGE_D - Visual Studio 2005 Team Edition Load Test Agent
 - c. Server STAGE_W - IIS 6 Web and SMTP services
 - d. Server STAGE_W - Visual Studio 2005 Team Edition Load Test Agent

- e. Server STAGE -Visual Studio 2005 Team Edition for Software Developers
 - f. Server STAGE -Visual Studio 2005 Team Edition for Database Developers
 - g. Server STAGE -Visual Studio 2005 Team Edition for Software Testers
 - h. Server STAGE -Visual Studio 2005 Team Edition Load Test Agent
 - i. Server STAGE -Visual Studio 2005 Team Edition for Architects
 - j. Server STAGE -Visual Studio 2005 Team Edition Foundation Server
 - k. Server STAGE – NetBeans IDE
- d) Estimated time to build out environment is 80 hours

11 Technology Infrastructure Extensions

11.1 *What is the Base Infrastructure*

We will use Microsoft's Visual Studio Team System as our primary integrated development environment and Team Foundation Server for a multitude of other features as shown in Diagram B. At a minimum, the base infrastructure is a source control system, automated reporting system, work item tracking, and an automated build system.

Team Foundation Server will enable our development teams to manage changes to source code, issues, and requirements effectively. Team Foundation Server's integrated approach will enable us to make decisions based on real-time information from a single comprehensive data source. Microsoft Project Server will be integrated into the Team Foundation Server using the Team System-Project Server connector. Diagrams A and B show a graphical view of the entire infrastructure.

11.2 *Base and Extended Infrastructure*

We have been using a four-tier approach in our software development model, which uses extended technologies to promote automated defect prevention practices.

Individual Development: This is the individual working environment for our individual developers and small teams. Working in isolation with the rest of the tiers, the developer(s) can apply any changes to the code on their own workstations without adversely affecting the rest of the development team.

Development: This is a common environment where all developers commit code changes. The purpose of this environment is to combine and validate the work of the entire project team so it can be tested before being promoted to the Staging Environment.

Staging: The staging tier is an environment that is as identical to the production environment as possible. The purpose of the Staging environment is to simulate as much

of the Production environment as possible. We use the staging environment as a Demonstration/Training environment also.

Production: The production tier is defined in Diagram A.

These tiers represent “environments” rather than “machines” or “servers.” It is certainly possible for multiple Development environments and the Staging environment to be on the same physical machine but not desirable. The team development and staging environments shall mimic production as much as possible. The production environment will not be shared with any of the other environments.

11.3 Resources at Each Tier

11.3.1 Staging

- Identical software configuration as the production machine and a complete copy of the production database so that it is a true basis for QA testing.
- Comparable or similar hardware configuration to the production system so an accurate forecast of capacity by performance testing against it and then multiplying its performance by the number of machines that will be deployed in production.

11.3.2 Development

- Identical software configuration as the production machine and a complete copy of the production database so that it is a true basis for development testing.
- Comparable or similar hardware configuration to the production system so an accurate forecast of capacity by performance testing against it and then multiplying its performance by the number of machines that will be deployed in Staging.

11.3.3 Individual Development

- Limited subset of data that is useful for testing “boundary conditions” in the application. We refresh this subset of data frequently to remove the artifacts of software development and testing on the Integration environment. Each developer has a Visual Studio Team System installed on his or her own workstation along with associated IIS 5.1 and SQL 2000 developer services.

11.3.4 Moving Between Tiers

The software developer writes code on their Individual Development environment and checks it into the source code repository. They check out other modules or stubs to let them simulate the real environment. The developer works out the majority of the logic errors on their local Individual Development Environment.

Developers only make changes to the Individual Development and Development environments. If a bug fix is to be made, the developer makes it in source code at the Individual Development stage. In order to maintain the integrity of the source code repository at no point does a developer make changes directly to the Staging or Production environments.

When the developers are satisfied with the behavior of their Individual Development environment, they check it in to Visual Studio Team System. The configuration manager pushes it to the main development code line, “tags” the code in source control, and updates the main development environment to this tag. The nightly automated build process will test the build with all the newly modified source code and then deploy to the Development web and data servers. After this the automated unit and functional testing begins. The automated reporting system generates reports to the developers. The quality assurance (QA) testers start their review. QA testers can be both internal staff and external reviewers. The automated QA reports go back to the developer who fixes any logic/code errors and the process starts anew in the developer’s individual development environment and back into the main development environment. After the functional testing and unit testing show an acceptable pass level, the configuration manager promotes the source code to the Staging environment.

The staging environment has an automated nightly build, deploy, and test. The staging area is where the customer will sign off on final acceptance testing and where final SQA approval is given. Automated testing is done on demand to show all automated testing can pass at any given time. The report system automatically generates test results for final acceptance testing and SQA. Any failure in Staging raises an automated chain of emails to a Causal Analysis team to determine why defects made it this far into the testing cycle. The cause of the defect is noted and new processes are tailored and put into the software development process so the same defects and cause of defects do not reappear in the software development process. Correct action policies will also be introduced into the automated testing system, the automated static code checker, testing documentation, and if necessary, the schedule is adjusted to allow more time for the new tailored process.

This process continues until the QA team declares the staging version is “okay to release”. The configuration manager promotes the source code to a mainline release version from the Staging environment source code control, deploys it on the production servers in the automated nightly build, and deploys cycle.

For each deployment to Production, there are multiple versions in Staging and for each deployment into Staging, there are multiple versions in Development/Individual Development. By design, end users are isolated from the rapid and occasionally buggy process of developing software. It is assumed that most bugs will be caught early and repeated versions at the early stages will find bugs faster.

Only the configuration manager can deploy versions to the next stage. There can be different configuration managers for deployment from Development-to-Staging and Staging-to-Production; the release manager can even change from version to version. The

company policy is to make sure that the developer is not the same person to release the code to production, as this would be a SOX violation. Permissions are set in Visual Studio Team System/Team Foundation Server to assure that only the configuration managers can release and promote code to the next step.

11.3.5 Coding Standards

To facilitate transfer of applications from the development server through the staging and into production, the code should be free of any server-dependent variables.

- To ensure code is portable across file systems and environments, the automated build process (MS Visual Studio Team System – MS Build) shall have an automated configuration set up for each environment to alleviate concerns about hard coded variables using server names, path names, database server name, web server names, data sources, and any other configuration variable that will be setup. For example, the Visual Studio Project file for a module will have a mandatory minimum of four configurations selection. A debug build for each environment and a final release versions.
- The architect shall make a base project and module template for all Visual Studio projects create. This policy prohibits developers from adding external objects that may deviate from company standards.
- Static code checkers available in MS Visual Studio Team System will be used to enforce coding standards.

These standards are enforced by MS Visual Studio Team System policies and discrepancies are reported to the developer via the automated reporting system.

12 People Infrastructure Extensions

Due to the nature of the business, most of the team had to be assigned to multiple roles in order to fulfill all the infrastructure requirements.

12.1 SQA

It was decided that one developer should spend 50% of her allocated time in a SQA role.

12.2 Configuration Management

It was decided that one developer should spend 50% of her allocated time in a SQA role.

12.3 Test Engineer

It was determined that one developer should spend 25% of her allocated resources in a Test Engineer role.

12.4 Deployment Engineer

It was decided that one developer should spend 15% of her allocated time in a Deployment Engineer role.

12.5 *Systems Engineer*

It was decided that the architect should spend 10% of her allocated time in a System Engineer role.

12.6 *DBA*

It was decided that one developer should spend 15% of her allocated time in a DBA role.

12.7 *Business Analyst*

It was decided that the same developer should spend 20% of their allocated time in a Business Analyst role

Appendix A: Diagrams

Diagram A - People and Infrastructure Diagram

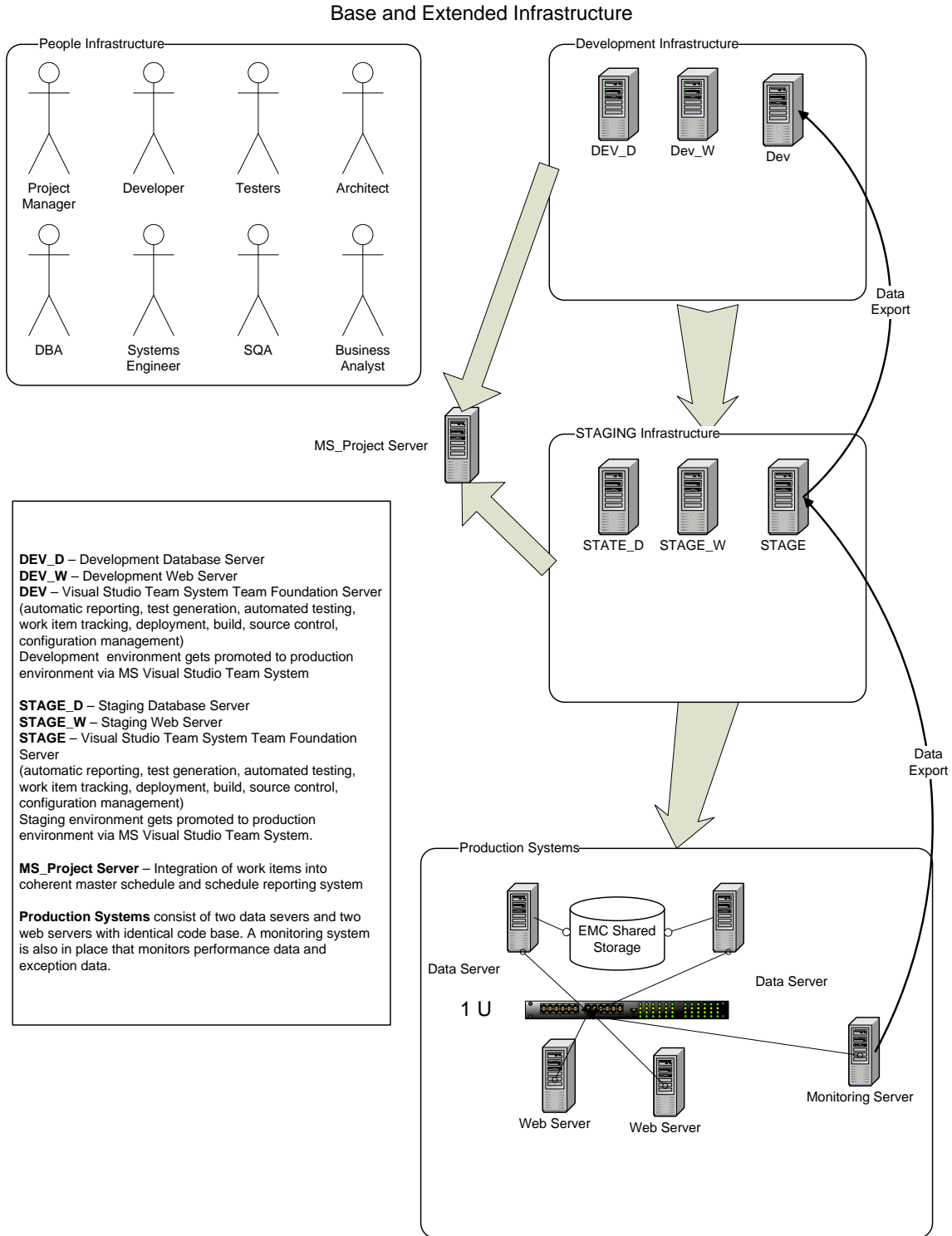
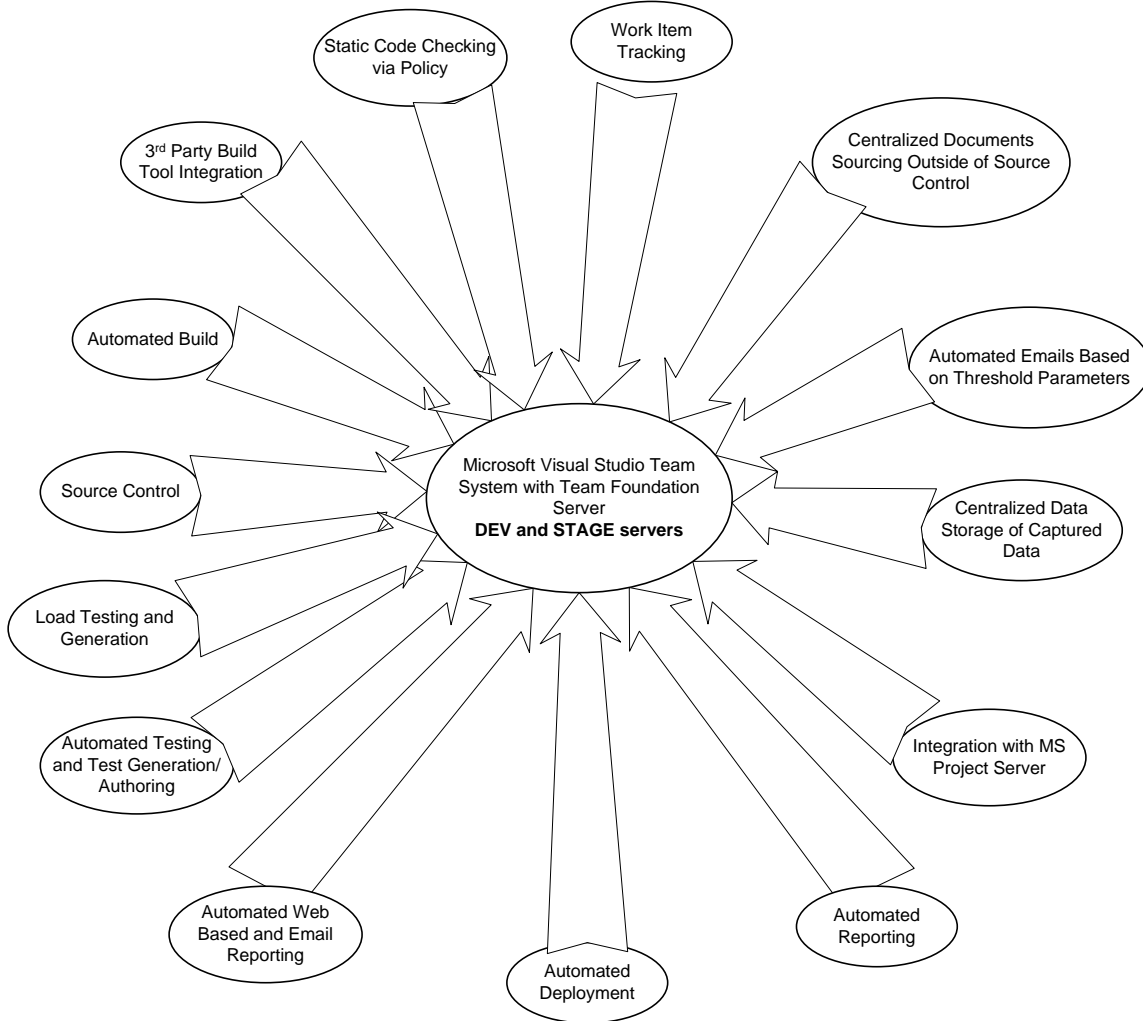


Diagram B – Infrastructure Responsibilities

Infrastructure responsibilities associated with Microsoft Visual Studio Team System with Team Foundation Server on DEV and STAGE Servers



Appendix B: Glossary

Term	Definition
ANT	Ant is a software tool for automating software build processes.
Browser	A web browser is a software application that enables a user to display and interact with text, images, and other information typically located on a web page at a website on the World Wide Web or a local area network. <i>(See Web Browser)</i>
COCOMO II	COCOMO is an effort estimation method designed to give an estimate of the number of person-months it will take to develop a software product.
COTS	Commercial off-the-shelf (COTS) is a term for software products that are ready-made and available for sale, lease, or license to the general public. [Wikipedia]
GUI	Graphical User Interface; an output interface used to display images, animation and text.
HTTP and HTTPS	Hyper Text Transport Protocol is a protocol for transferring information across the network over TCP/IP. HTTP also specifies the formatting of such data as it flows. HTTPS is the same but the data is encrypted by virtue of a set of key exchanges that encrypts and decrypts the data. An asymmetric key exchange takes place between the server and client which then includes the symmetric key that is used to encrypt and decrypt the remaining data exchange.
LOC	Lines of code, a software metric used to measure the amount of code in a software program
MSBUILD	MSBuild is a build tool for Visual Studio project files, and is available for free.
SLOC	Source lines of code. See LOC
SMTP	Simple Mail Transfer Protocol is a protocol that specifies how transfers of electronic messages are sent and the formatting of messages between client and servers over TCP.
SQA	Software Quality Assurance
Wideband Delphi	Wideband Delphi is an estimation consensus-based estimation method for estimating effort.

Appendix C: References

Ambler, S.W., The Agile Unified Process (AUP),
<http://www.ambysoft.com/unifiedprocess/agileUP.html> , Ambysoft

Bass, Clements, Kazman, Software Architecture in Practice Second Edition, Addison-Wesley, 2003

Huzinga & Kolawa, Automation and Defect Prevention in Software Management, Wiley Inc, 2007