Sumatra (SW 3.1) Integration and System Test Plan

Overview

The following test plan describes the formal testing to be performed by the Integration and System Test (IST) Team (Test Team) within the Engineering Services Group (ESG) against Sumatra. This test plan covers the included items in the test project, the specific risks to product quality we intend to address, timeframes, the test environment, problems that could threaten the success of testing, test tools and harnesses we will need to develop, and the test execution process. Two specific testing activities occur outside of the Test Team's area: 1) Unit/component testing, which will be handled by Jenny Kaufmann's Development Team; and, 2) Alpha/Beta testing, which will be performed by actual users of the Sumatra system under the direction of Kate Hernandez.

This document also lays out the strategies, resources, relationships, and roles involved in performing Integration and System Testing as a distinct testing subproject within the larger Sumatra project. The Sumatra project is about adding a Document Management System (DMS) to the SpeedyWriter product, along with fixing various small bugs and adding some customer-requested enhancements. The Project Team intends to follow an incremental lifecycle model, with the major functionality showing up in the first two increments, then the lesser functionality showing up in the last three. The release, which is tentatively labeled 3.1, will go out to customers at the end of Q1/2003.

To test Sumatra, we need to receive several releases of each increment into the test environment and run multiple passes of tests against each increment in both Integration and System Test Phases. The Integration tests will focus on bugs between components, while System tests will look for bugs in the overall system. Component testing, which is to be run by the Development Team, is essential to get Sumatra ready for these test phases. Because of the incremental lifecycle, all three phases will overlap. Integration and System Testing will be primarily behavioral; i.e., looking at the system externally, as a user would, based on what behaviors the system is to exhibit for the users. The Test Team will develop both automated and manual tests to cover the quality risks identified in the FMEA, as well as augmenting that test set to the extent possible under resource and time constraints with customer based tests (from tech support, sales, and marketing) along with structural coverage gap analysis.

Bounds

The following sections serve to frame the role of the independent test organization on this project.

Scope

Table 1 defines the scope of the Sumatra Integration and System Test effort.

IST Testing for Sumatra…	
IS…	IS NOT…

Positive and negative functionality Load, capacity, and volume Reliability and stability Error handling and recovery Competitive inferiority comparison Operations and maintenance Usability Data quality Performance Localization Compatibility Security and privacy Installation and migration Documentation Interfaces Distributed (outsource usability and localization testing) Black-box/behavioral testing 3.x regression testing 3.1 regression testing (across increments)	Date and time handling Standards and regulatory compliance Code coverage verification Set-use pair or data flow verification Database table overflow error handling User interface offensiveness User interface performance under maximum load OfficeArrow integration Migration and media load fine-tuning Legacy browsers, client, or server compatibility OS login identity override Legacy (pre 3.0) migration Unit or component testing (except supporting development team) White-box/structural testing

Table 1: IST for Sumatra IS/IS NOT (Scope)

Definitions

Table 2 defines some test terms and other terms found in this document.

Term	Meaning
Black Box Testing	Testing based on the purposes a program serves; i.e., behavioral testing.
Bug	Some aspect of the system under test that causes it to fail to meet reasonable expectations. “Reasonable” is defined by iterative consensus if it is not obvious.
Build	A collection of software objects of known revision levels compiled into one or more software executables for installation on the system under test.
Test Release	
Confirmation Test	A selected set of tests designed to find ways in which a bug fix failed to address the reported problem fully.
Entry Criteria	The parameters that determine whether one is ready and able to enter, continue, or exit a particular test phase.
Continuation Criteria	
Exit Criteria	
Integration Test	A set of tests designed to find bugs in the interfaces (control and data flows) between tested components of the system.

Term	Meaning
Quality Risk	The possibility of a specific system failure mode, either localized, caused by subsystem interactions, or a repercussion effect of a remote system failure, that adversely affects the system's user.
Reference Platform	A "known correct" system against which one can compare the results and behaviors of the system under test.
Server Cluster	A set of servers configured to perform a particular role in terms of development, testing, or production.
Regression Test	A set of tests run to find new failures, or regressions, that changes have caused in component, interface, or system functionality.
Smoke Test	A limited set of regression tests designed to determine, though a random sample of critical functions, whether a given build is ready for testing.
System Test	A set of tests designed to find bugs in the overall operation of the integrated system.
SUT	System under test. In this case, the Sumatra software running in the test environment.
Test Escape	A field failure that could reasonably have been found by IST executing this test plan but for errors in execution, attentiveness, interpretation of observed behavior, or other reasonably foreseeable and preventable test errors.
White-Box Testing	Testing based on the way a program performs its tasks; i.e., structural testing.

Table 2: Definitions

Context

Figure 1 shows the various stakeholders and participants in the test effort and how they interact.

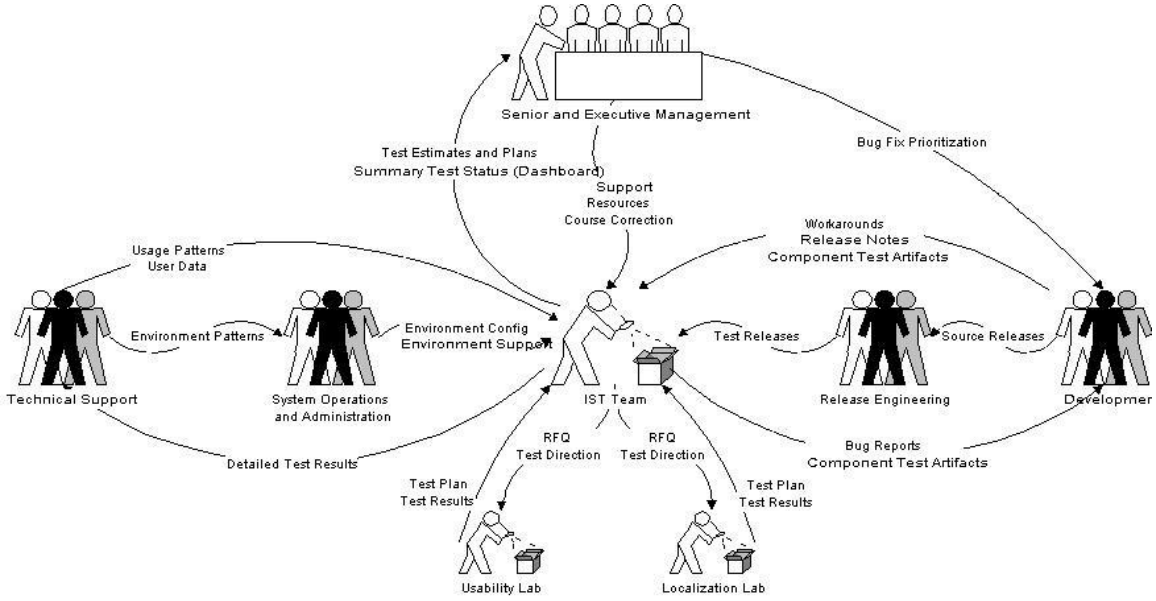

Figure 1: Sumatra IST Context

Quality Risk Test Coverage Strategy

At the highest level, the Sumatra test strategy can be summarized as follows:

- Develop automated and manual tests to cover all the quality risks identified as needing extensive or balanced testing, focusing on behavioral factors observable at some user interface or accessible API.
- Add test data or conditions within existing cases or new test cases to cover critical customer usage profiles, customer data, or known defects in our product or competitors' products.
- Use exploratory testing in areas that were not addressed previously and that appear, due to test results or intuition, to be at high risk of bugs. Update or define new test cases to cover found bugs.
- Run tests across a "customer-like" mix of server, network, and client configurations (the test environment).
- Repeat all Integration Test and System Test cases, including SpeedyWriter 3.0 functional tests, multiple times across each phase to detect regression.
- Possibly, should time and resources allow or should concern about unidentified quality risks dictate, use structural coverage techniques to identify untested areas, then add tests to cover critical test gaps.

To give a slightly more detailed discussion, Table 1 describes the strategies used to cover each quality risk within scope of the testing effort.

Quality Risk Category	RPN	Phase	Test Strategy	Test Environment
Positive and negative functionality	2	Intgrtn System	Extensive automated (GUI) behavioral testing	All clusters All topologies All browsers (2 at a time)
Load, capacity, and volume	1	System	Extensive automated (GUI and API) behavioral testing	All clusters All topologies All browsers (5 at a time)
Reliability and stability	4	Intgrtn System	Extensive automated (GUI and API) behavioral testing	Clusters East and North 1 Gbps, 100 Mbps, 10 Mbps Ethernet All browsers (5 at a time)
Error handling and recovery	4	Intgrtn System	Extensive scripted manual behavioral testing	All clusters All topologies All browsers (2 at a time)
			Complete desk-check review of error messages	N/A
Competitive inferiority comparison	36	Intgrtn System	Isolate bugs against competitive software in test environment	All reference platforms (1 at a time)
Operations and maintenance	10	System	Balanced scripted (following Operator's Guide and Release Notes) manual behavioral testing	All clusters All topologies All browsers (2 at a time)
Usability	10	System	Balanced scripted and exploratory manual behavioral testing (as part of all manual test cases)	All clusters All topologies All browsers
			Usability study	Offsite test lab
Data quality	4	Intgrtn System	Extensive manual and automated behavioral and structural testing (as part of all test cases that access the database)	All clusters All topologies All browsers

Quality Risk Category	RPN	Phase	Test Strategy	Test Environment
Performance	9	System	Balanced automated behavioral testing	All clusters All topologies All browsers (5 at a time)
Localization	24	System	Balanced manual behavioral testing	Offsite test lab
Compatibility	4	Intgrtn System	Include recent browsers, clients, and servers in test environment	All clusters All browsers
Security and privacy	3	System	Extensive exploratory and scripted manual behavioral and structural testing, including through the GUI, APIs, macros, command-line, databases, and other interfaces	All clusters All topologies All browsers (2 at a time)
Installation and migration	2	System	Extensive scripted and exploratory manual behavioral testing, including the GUI and command-line methods	All clusters All topologies All browsers (2 at a time)
Documentation	10	System	Balanced documentation-driven manual behavioral testing of examples (macros, commands, etc.)	Any or all, according to the judgment of the tester
			Complete desk-check review of all printed materials	N/A
Interfaces	2	Intgrtn	Extensive automated API structural testing.	All clusters All topologies All browsers (2 at a time)
Previous release regression	2	Intgrtn System	Repeat of 3.x automated functionality tests (mostly GUI behavioral)	All clusters All topologies All browsers (2 at a time)

Quality Risk Category	RPN	Phase	Test Strategy	Test Environment
3.1 regression testing (across increments)	N/A	Intgrtn	4 week test passes (repeat of Integration Test cases)	All clusters All topologies All browsers (2 at a time)
		System	2 week test passes (repeat of System Test cases)	
Unidentified	N/A	Intgrtn System	Exploratory testing of areas seen as promising by testers ("Guerilla Friday")	Any and all, as judged appropriate by each tester

Table 3: Quality Risk Test Coverage Strategy

Schedule of Milestones

The following shows the scheduled events that affect this test effort.

Milestone/effort	Start	End
Planning	9/22/2002	9/25/2002
Plan approved by project management team	9/25/2002	9/25/2002
Staffing	9/16/2002	10/22/2002
Test environment acquisition/configuration	9/20/2002	10/18/2002
First integration test release delivered/installed	10/7/2002	10/7/2002
Integration test phase entry criteria met	10/7/2002	10/7/2002
Integration test phase begins for increment one	10/7/2002	10/7/2002
Integration test pass one	10/7/2002	11/1/2002
Integration test phase begins for increment two	11/1/2002	11/1/2002
Integration test pass two	11/4/2002	11/29/2002
Integration test phase begins for increment three	11/29/2002	11/29/2002
Integration test pass three	12/2/2002	12/27/2002
Integration test phase begins for increment four	1/3/2003	1/3/2003
Integration test pass four	1/6/2003	1/31/2003
Integration test phase begins for increment five	1/31/2003	1/31/2003
Integration test pass five	2/3/2003	2/14/2003
Exit criteria met	2/14/2003	2/14/2003
Integration test exit meeting	2/14/2003	2/14/2003
System test entry criteria met	10/21/2002	10/21/2002
System test entry meeting	10/21/2002	10/21/2002
Increment one delivered	10/21/2002	10/21/2002
System test pass one	10/21/2002	11/1/2002
System test pass two	11/4/2002	11/15/2002
Increment two delivered	11/15/2002	11/15/2002
System test pass three	11/18/2002	11/29/2002
System test pass four	12/2/2002	12/13/2002
Increment three delivered	12/13/2002	12/13/2002

Milestone/effort	Start	End
System test pass five	12/16/2002	12/27/2002
System test pass six	12/30/2002	1/10/2003
Increment four delivered	1/10/2003	1/10/2003
System test pass seven	1/13/2003	1/24/2003
System test pass eight	1/27/2003	2/7/2003
Increment five delivered	2/7/2003	2/7/2003
System test pass nine	2/10/2003	2/21/2003
System test pass ten	2/24/2003	3/7/2003
Golden candidate delivered	3/7/2003	3/7/2003
System test pass eleven	3/10/2003	3/21/2003
System test exit meeting	3/21/2003	3/21/2003
System test exit criteria met	3/21/2003	3/21/2003

Table 4: Scheduled System Test milestones

Transitions

The following subsections define the factors by which project management will decide whether we are ready to start, continue, and declare complete the Integration Test and System Test phases.

Integration Test Entry Criteria

Integration Test shall begin when the following criteria are met:

1. The bug tracking system is in place and available for use by the engineering services and system engineering teams.
2. The System Operations and Administration Team has configured the Integration Test clients and servers for testing. The Test Team has been provided with appropriate access to these systems.
3. The Development Team has placed at least two communicating components to be released to IST for Integration Testing under formal, automated source code and configuration management control.
4. The Development Team or the Release Engineering team has prepared a test release, containing at least two communicating components, both of which have completed Component Testing.
5. The Development Team has prepared release notes that document the functionality in the test release and any known bugs and limitations.

Integration Test Continuation Criteria

Integration Test shall continue provided the following criteria are met:

1. Each Integration Test release contains only components under formal, automated source code and configuration management control.
2. The Development Team or the Release Engineering team prepares test releases from communicating components that have completed Component Testing.

3. The Development Team accompanies each release with release notes that document the functionality in that release and any known bugs and limitations.
4. Some test release built from relatively up-to-date source code can be installed in the test environment in such a way that the release functions in a stable fashion. Furthermore, the Test Team can execute planned or exploratory tests against this test release in a reasonably efficient manner to obtain further meaningful test results.

Integration Test Exit Criteria

Integration test shall successfully conclude when the following criteria are met:

1. The Test Team has performed all planned tests against all planned integration builds.
2. The final Integration Test release contains all components that will be part of the customer-released (General Availability) version of Sumatra.
3. The Sumatra Project Management Team agrees that sufficient Integration Testing has been performed and further Integration Testing is not likely to find more integration-related bugs.
4. The Sumatra Project Management Team holds an Integration Test Phase Exit Meeting and agrees that these Integration Test exit criteria are met.

System Test Entry Criteria

System Test shall begin when the following criteria are met:

1. The bug tracking system is in place and available for all project participants.
2. The System Operations and Administration Team has configured the System Test clients and servers for testing. The Test Team has been provided with appropriate access to these systems.
3. The Development Team has completed all features and bug fixes scheduled for Increment One and placed all of the underlying source components under formal, automated source code and configuration management control.
4. The Development Teams has completed Component Testing for all features and bug fixes scheduled for Increment.
5. Less than fifty (50) must-fix bugs (per the Project Management Team) are open, including bugs found during Component Test and Integration Test.
6. The Sumatra Project Management Team holds a System Test Phase Entry Meeting and agrees that Increment One is ready to begin System Test.
7. The Release Engineering Team provides a revision-controlled, complete software release to the Test Team as described in the “Release Management” section.
8. The Development Team has prepared release notes that document the functionality in the test release and any known bugs and limitations.

System Test Continuation Criteria

System Test shall continue provided the following criteria are met:

1. All software released to the Test Team is accompanied by Release Notes. These Release Notes must specify the bug reports the Development Teams believe are resolved in each software release and any known bugs and limitations.
2. No change is made to Sumatra system, whether in source code, configuration files, or other setup instructions or processes, without an accompanying bug report or as part of the planned features or bug fixes for a subsequent Increment.
3. The Release Engineering Team provides weekly a revision-controlled, complete software releases to the Test Team as described in the "Release Management" section, built from relatively up-to-date source code. These releases can be installed in the test environment in such a way that the release functions in a stable fashion. Furthermore, the Test Team can execute planned or exploratory tests against this test release in a reasonably efficient manner to obtain further meaningful test results.
4. Less than fifty (50) must-fix bugs (per the Project Management Team) are open, including bugs found during Component Test and Integration Test.
5. Twice-weekly bug review meetings occur until System Test Phase Exit to manage the open bug backlog and bug closure times.

System Test Exit Criteria

System Test shall successfully conclude when following criteria are met:

1. No panic, crash, halt, wedge, unexpected process termination, or other stoppage of processing has occurred on any server software or hardware for the previous four (4) weeks.
2. The GA-candidate Increment (currently targeted to be Increment Five) has been in System Test for six (6) weeks.
3. The Test Team has executed all the planned tests against the GA-candidate System Test release.
4. The Development Team has resolved all "must-fix" bugs.
5. The Test Team has checked that all issues in the bug tracking system are either closed or deferred, and, where appropriate, verified by regression and confirmation testing.
6. The Product Quality Dashboard Gauge indicates that Sumatra has achieved an acceptable level of quality, stability, and reliability.
7. The Quality Risk Coverage Dashboard Gauge indicates that all Quality Risks have been adequately covered.
8. The Project Management Team agrees that the product, as defined during the final cycle of System Test, will satisfy the customers' and users' reasonable expectations of quality.
9. The Project Management Team holds a System Test Phase Exit Meeting and agrees that these System Test exit criteria are met.

Test Configurations and Environments

There are ten clients, running various browser and operating system combinations. There are four sets of different database, Web, and app servers (clusters).

System	Name	IP Address	OS	Other SW	Status
Server Cluster East (SrvE)					
DB1	Kursk	192.168.6.10	Solaris	Oracle 9i	Available
Web1	Leningrad	192.168.6.20	Solaris	Netscape	Available
App1	Stalingrad	192.168.6.30	HP/UX	Oracle 9AS	Available
Server Cluster West (SrvW)					
DB2	Dunkirk	192.168.6.11	AIX	Sybase	Available
Web2	Bulge	192.168.6.21	AIX	Domino	Available
App2	Normandy	192.168.6.31	OS/400	WebLogic	Available
Server Cluster North (SrvN)					
DB3	Narvik	192.168.6.12	NT	SQL Server	Available
Web3	Oslo	192.168.6.22	W2K	IIS	Available
App3	Trondheim	192.168.6.32	NT	iPlanet	Available
Server Cluster South (SrvS)					
DB4	ElAlamein	192.168.6.14	OS/2	DB2	Order
Web4	Tobruk	192.168.6.24	Linux	Apache	Order
App4	Anzio	192.168.6.34	NT	WebSphere	Order
Hierarchical Storage Management Servers					
HSM1	Midway	192.168.6.101	NT	Legato	Order
HSM2	Bataan	192.168.6.102	Linux	Veritas	Order
Network Topologies and Communications					
1Gb EN	N/A	N/A	N/A	N/A	Available
100 Mb EN	N/A	N/A	N/A	N/A	Available
10 Mb EN	N/A	N/A	N/A	N/A	Available
16 Mb TR	N/A	N/A	N/A	N/A	Available
Dial-Up	Verdun (Mdm Bank)	192.168.6.112	Solaris	N/A	Available
Browser Stations					
BS1	Patton	DNS	MacOS	Netscape	Available
BS2	Eisenhower	DNS	Linux	Netscape	Available
BS3	DeGaulle	DNS	Win2K	IE	Available
BS4	Montgomery	DNS	Win98	IE	Available
BS5	Clark	DNS	Win95	IE	Available
BS6	Mannerheim	DNS	MacOS	IE	Order
BS7	Truman	DNS	Solaris	Netscape	Order
BS8	Bradley	DNS	Linux	Netscape	Order
BS9	Komorowski	DNS	WinNT	IE	Order
BS10	Nimitz	DNS	Solaris	Netscape	Order
Reference Platforms					
RP1	Roosevelt	DNS	Solaris	Star Office	Available

System	Name	IP Address	OS	Other SW	Status
RP2	Chiang	DNS	WinMe	MS Office	Available
RP3	Churchill	DNS	Linux	Corel Suite	Available

Table 5: Test Environments

A graphical view of the test network configuration is shown in Figure 2.

The System Operation and Administration team, as discussed below, will support the test environment. Nevertheless, the environment's configuration remains under the control of Jamal Brown, Test Manager. No change is to be made to any hardware, software, or data that is part of, installed on, or stored by the test network without the express permission of Jamal Brown or one of the three Test Engineers. Any changes made are to be communicated back to the entire test team in writing via e-mail.

Following initial configuration and verification of the test environment, Jamal Brown shall request Keith Lee to have his SOA team take "snapshots" of the boot devices of all systems in the network. This shall be done by copying an exact image of the drive to a duplicate of the drive. These drives shall be labeled and locked in a cabinet in Jamal Brown's office. Only Jamal, Keith, and the designated Sumatra SOA support team members (see below) shall have keys to this cabinet.

At the end of each test cycle—generally on Saturday evening—an automated backup shall run that copies all the data on all the drives of all the systems in the test environment (servers, clients, and references platforms) to tape storage. Lin-Tsu shall label these tapes with the date and give them to Jamal for locked storage as well.

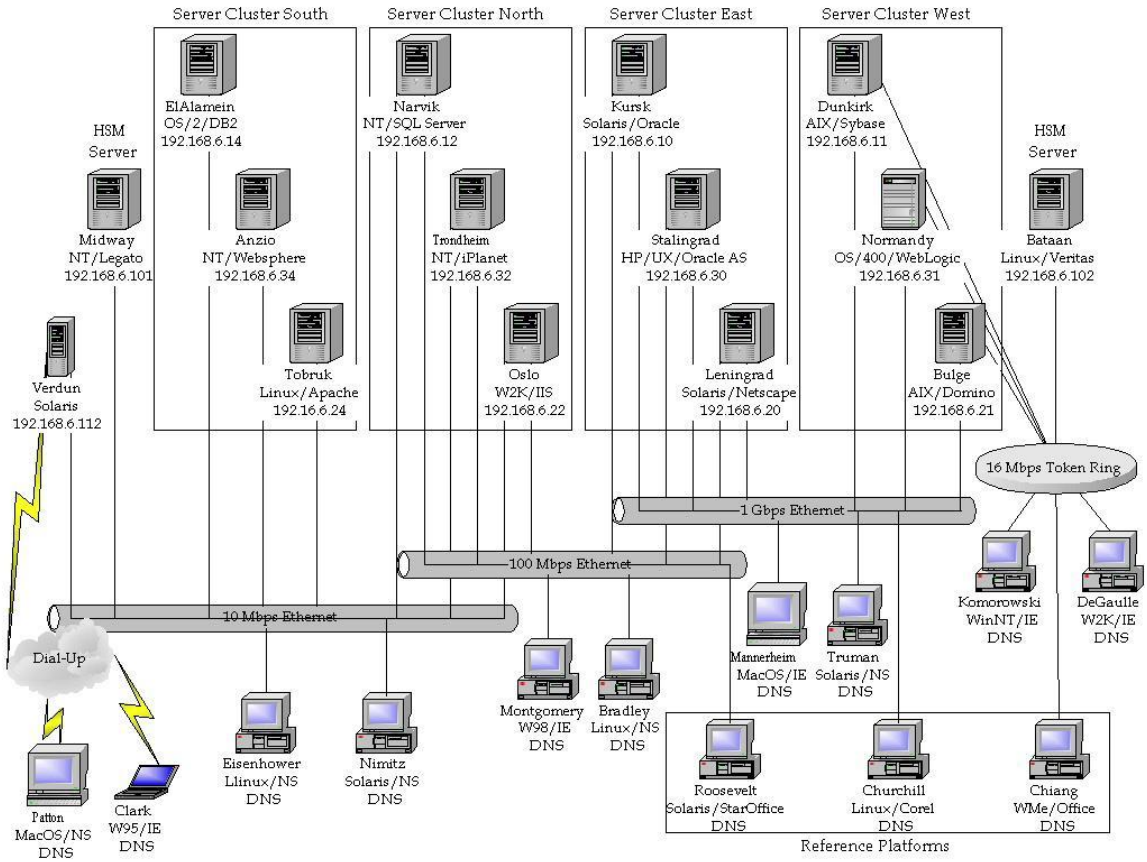

Figure 2: Test Environments

Test Development

The Integration and System Testing is based on a collaborative risk prioritization process, the failure mode and effect analysis performed earlier in the project. This analysis identified major areas, mostly from the requirements, that are: 1) subject to severe failure; 2) a priority for the users and customers; or, 3) likely to suffer from technical problems that would be widely encountered in the field. Based on this analysis, the test stakeholders have identified and prioritized areas of testing. Some of these areas require new or updated test data, cases, procedures, scripts, and so forth.

The following test case development and/or test maintenance activities are included in the scope of this project.

- Updating the SpeedyWriter 3.0 automated GUI behavioral functional test data, cases, and scripts for the Sumatra features and behaviors.
- Creating new manual GUI behavioral functional test data, cases, and, if need be, procedures for the Sumatra DMS features. Automating those test cases.
- Creating (with development) new automated API structural test data, cases, scripts, and, if necessary, harnesses for the component interfaces.

- Updating the SpeedyWriter 3.0 manual GUI and command-line behavioral installation, maintenance, and operations test data, cases, and, if need be procedures for Sumatra.
- Creating new automated GUI behavioral reliability and stability test data, cases, and scripts for the Sumatra product as a whole, including the DMS features.
- Creating new manual GUI, API, macro, command-line, database, and other interface behavioral and structural security and privacy test data, cases, procedures, macros, scripts, and so forth, including researching known vulnerabilities for browser-based systems.
- Updating the SpeedyWriter 3.0 manual GUI, API, command-line, and other interface behavioral and structural error handling test data, cases, procedures, and so forth for Sumatra-specific features and behaviors.
- Updating the SpeedyWriter 3.0 manual GUI behavioral localization test data, cases, procedures, and so forth for Sumatra-specific prompts, messages, help screens, features, and behaviors.
- Updating the SpeedyWriter 3.0 automated GUI behavioral performance, load, capacity, and volume test data, cases, and scripts for the Sumatra features and behaviors, and creating any API or GUI load generators or other test harnesses required.
- Updating the SpeedyWriter 3.0 manual GUI, macro, command-line, and database behavioral documentation test data, cases, procedures, and so forth for Sumatra-specific documentation, user's guide, packaging, corporate Web site content, warranty documentation, published claims, messages, help screens, features, and behaviors.

Those automated tests that work through the GUI shall use the Office Arrow standard GUI automation tool, XXX. Automated API tests shall be built using standard freeware testing harnesses such as ZZZ.

The developing test engineer shall document all test objects created or updated at the appropriate level of detail—including the actions to be taken, the data to be used, and the expected results to be looked for—for use by the test engineers and test technicians who will execute tests with them as well as for use by subsequent test engineers who will maintain and update them. At this point, the Test Team does not foresee the development of "throw-away" test objects for Sumatra testing only.

The developing test engineer, upon completion of the object and successful peer review of that object, shall place that test object under the same level of configuration management and change control as the other project objects. The repository contains a separate subfolder called "Test" in the Sumatra folder. See "Software Cafeteria Integrated Development Environment User's Guide" for more details on how to check objects in and out of this repository.

Test Execution

The following subsections define the activities and participants involved in test execution.

Test Execution Hours

When only Integration Test or only System Test execution is underway, the IST team will test primarily between the hours of 9:00 AM and 6:00 PM, (all times Central), Monday through Friday.

During the period of overlapping Integration and System Test execution, the IST team will test primarily between the hours of 8:00 AM and 12:00 midnight (all times Central), Monday through Friday. This will entail a day shift and an evening shift. The engineers and the test manager shall be available on-site at least from 10:30 AM to 7:30 PM to support both shifts.

Test execution may start earlier and continue later, as the workload requires. Work may also occur over the weekends.

Test Passes and Regression Strategy

A test pass consists of execution of all the test suites (the test cohort or set) defined for a given test phase. For the Integration Test Phase, a test pass requires four weeks. For the System Test Phase, each test pass will require two weeks. The Test Team shall repeat these passes multiple times over the course of each test phase. This repetition of substantially the same set of tests constitutes the regression strategy for this project. However, the test cases in each test suite will improve in coverage as the product grows in each Increment, the number of test cases may also increase, and test engineers may repair defects in test objects.

Test Cycles

A test cycle consists of some subset of a test pass executed against a single, identifiable test release. For Integration Test, we will run a test cycle every day. For System Test, we will run a test cycle every week. Test execution will begin as soon as the new releases are appropriately installed. The Test Team will carry out testing in the following order:

- **Confirmation testing.** The Test Team will retest all bugs reported as fixed in the release notes. For those bugs fixed, the testers will close the appropriate bug reports. They will re-open reports for any bugs not fixed.
- **Scheduled testing.** The Test Team will run all test cases scheduled for that test cycle, starting with those test cases considered high risk. Test cases never executed before, especially against newly-developed functionality, are typically the highest risk.
- **Guerrilla testing.** Every Friday, at least one tester shall run exploratory tests against the system, based on where that tester suspects that current test coverage is light and bugs remain undiscovered. Should the tester(s) discover a substantial number of bugs in these areas, especially where those bugs indicate a major omission in the risk-based test design strategy, a test engineer will add tests to fill the identified gap.

Should the Test Team complete these tests, it will perform any other test cases not scheduled for that cycle but which have not been run against the current build. If those cases are then completed, the test team will perform exploratory testing until the following release begins the next cycle. Conversely, due to delays of deliverables, schedule pressures, high bug find rates, or the size (duration or effort) of the test suites,

the Test Team may not complete all the scheduled test cases during every cycle. In that case, Jamal shall reschedule the test cases not completed as first priority for the next cycle, immediately following the confirmation testing.

Test Execution Process

The objective of the Integration Test and System Test phases is to find bugs in the system under test, in the interfaces between components and in the system as a whole, respectively. We intend to do this by running a set of manual and automated test cases against each test release (build).

At the beginning of each test cycle, Jamal Brown shall assign a "basket" of test cases to each tester. Each tester's basket shall be different from one test cycle to the next, to ensure that any invalid assumptions on one tester's part do not result in a test escape. Once assigned their basket of tests, these testers will follow the test steps outlined in the test case to execute each test case, and repeat the process until they have exhausted their list. If they empty their basket prior to the end of a cycle, they shall assist other testers by coordinating a reassignment of some of those testers' test cases to themselves. If all tests are completed, the test team shall proceed as described in the section above.

Key People Involved in the Test Effort

Table 6 describes the human resources need to execute this plan. Key support and liaison roles are also defined.

Title	Roles	Name
Test Manager	Plan, track, and report on test design, implementation, and execution Secure appropriate people, test environments, and other resources Provide technical and managerial leadership of test team	Jamal Brown
Manual Test Engineer	Design and implement manual tests Supervise and review technician manual testing Report results to the Test Manager	Lin-Tsu Woo
Automated Test Engineers	Design, implement, and execute automated tests Supervise and review technician automated testing results Report results to the Test Manager	Emma Moorhouse [To Be Hired (1)]
Test Techni-cians	Execute tests at the direction of Test Engineers Report results for review and approval by test engineers	[To Be Hired (4)]
(Offsite) Project Manager	Plan and manage external lab usability testing	[RFQ Out to Lab]
(Offsite) Project Manager	Plan and manage external lab localization testing	[RFQ Out to Lab]

Title	Roles	Name
Project Manager	Manage the development of the Sumatra system Work with the Release Engineering manager to ensure smooth flow of system under test into test environment Coordinate development/test hand-off issues with the Test Manager Handle escalations of "stuck" test-blocking bug processes	Jenny Kaufman
Programming Engineer	Respond to reports of test-blocking bugs Build and install (if necessary) the Integration Test builds	Bob Chien
Product Manager	Manage the overall SpeedyWriter product line Ensure development and testing is aligned with the long-term product goals	Kate Hernandez
Release Engineering Manager	Handle escalations of "stuck" release processes	Yasuhiro Kanagawa
Release Engineer	Deliver the Sumatra System Test builds from the source repository to the Manual Test Engineer for installation	Sam Nighthorse
SOA Manager	Handle escalations of "stuck" test environment support process	Keith Lee
Server Support Engineer	Support the server clusters in the test lab	Petra Fahimian
Client Support Engineer	Support the browser and other clients in the test lab	Inder Vaneshwatan

Table 6: People involved in the test effort

Resolution and Escalation Processes

Unexpected failures, logistical challenges, and strange behaviors occur during Integration and System Test execution. Some of these events can block forward progress through the test sets. The following are the resolution and escalation process when such blocking events occur during testing.

Non-functional Test Environment

1. An IST team member, upon discovering a problem blocking tests that appears to result from a non-functioning test environment, shall get a "second opinion" from another IST team member.
2. Should both IST members concur the blocking issue relates to the environment, the test team member discovering the problem shall escalate to Petra Fahimian (for servers) or Inder Vaneshwatan (for clients) for resolution.

3. Petra or Inder shall ascertain if the problem is an environment issue. If so, she or he shall resolve the problem. If not, she or he shall pursue one of the other escalation paths discussed in this section until a clear hand-off of responsibility is achieved.
4. Once Petra or Inder has resolved the problem, she or he shall notify at least one member of the IST team that the problem is resolved.

Should resolution prove impossible or this process break down, IST team members shall escalate to Jamal Brown, Keith Lee, and Jenny Kaufman.

Test-Blocking Bug

1. An IST team member, upon discovering a problem blocking tests that appears to result from a test-blocking bug in the Sumatra system, shall get a "second opinion" from another IST team member.
2. Should both IST members concur the blocking issue relates to the environment, the test team member discovering the problem shall escalate to Bob Chien for resolution.
3. Bob shall ascertain if the problem is a Sumatra bug. If not, he shall pursue one of the other escalation paths discussed in this section under a clear hand-off of responsibility is achieved. If so, he shall assess whether any change (allowed under the Release Management policy describe below) can unblock the testing or provide a workaround to the problem. If such a change will unblock testing, he shall document and implement the change after consulting with the test team. If not, he shall work with Sam Nighthorse as described below to revert to a working previous test release after consulting with Jamal Brown.
4. Once Bob has unblocked the testing, he shall notify at least one member of the IST team that the problem is resolved.

Should resolution prove impossible or this process break down, IST team members shall escalate to Jenny Kaufman and Jamal Brown.

Missing/Bad Release

1. An IST team member, upon discovering a new release has either not been delivered or is non-functional (in a way that blocks most tests) shall get a "second opinion" from another IST team member.
2. Should both IST members concur the blocking issue relates to a missing or bad test release, the test team member discovering the problem shall escalate to Sam Nighthorse for resolution.
3. Sam shall ascertain if the problem is a bad or missing release. If not, he shall pursue one of the other escalation paths discussed in this section until a clear hand-off of responsibility is achieved. If so, he shall resolve the problem, either by locating and installing the new release or by reverting to a previous release that will allow forward progress in testing.
4. Once Sam has resolved the problem, he shall notify at least one member of the IST team that the problem is resolved.

Should resolution prove impossible or this process break down, IST team members shall escalate to Yasuhiro Kanagawa and Jamal Brown.

Fall-Back Escalation

Should any person have difficulty executing the appropriate escalation process, or should the process prove incapable of unblocking the problem within four hours, the IST staff shall escalate the situation to Jamal Brown. Evening or weekend testing staff who become blocked may escalate to Jamal immediately and discontinue their shift, and shall escalate to Jamal and discontinue their shift within two hours of inactivity.

Assumptions about Key Players

Commitment by all key players in the testing process to fulfill their roles is essential to success of the Integration and System Test efforts.

- All key players shall carry a mobile phone or pager, turned on with adequately charged batteries and sufficient coverage range, at all times during Integration and System Test execution. For those key players without phones or pagers, Software Cafeteria will provide either for use on project-related business for the duration of test execution.
- All key players shall load from the Software Cafeteria Intranet and keep up to date the office phone, mobile phone, pager, e-mail, and home phone contact information for each other key player into a PDA or mobile phone.
- Key players shall be available on-site or by mobile phone during test hours, and shall respond within half an hour of contact. Escalation to management occurs if response does not occur within that time period. A key player who is off-site temporarily or for the day may hand off a request for resolution or escalation to a competent peer.
- Automated and manual testing will occur continuously during Integration and System Test execution. Any key player requiring periods of unavailability shall communicate these periods to Jamal Brown and to their manager, at least one week in advance, and the person making himself or herself unavailable shall arrange for alternate coverage.

Orderly, efficient, and effective testing implies that each key player shall carry out her role crisply, eschewing both individual heroics on the one hand and avoidance of responsibility and accountability on the other hand.

Test Case Tracking

Test cases shall be tracked using a set of Excel worksheets. These shall provide both detail level and summary level information on test cases for each test pass. As discussed elsewhere, we intend to complete multiple passes through all the tests, so the Excel spreadsheet file will have two worksheets for each Integration Test Pass and two worksheets for each System Test Pass. Jamal Brown shall maintain and update these tracking documents with the assistance of the test engineers. For each test case, the Test Team shall track the information shown in the following table.

Column	Meaning
State	The state of the test case. The possible states are: **Pass:** The test case concluded successfully. **Warn:** The test case concluded with an error, which the project management team has either deferred, closed as external to the product, or closed as unavoidable. **Fail:** The test case revealed a defect that development will address. **Closed:** The test case previously revealed a failure that is now resolved. **In Queue:** The test remains to be executed (indicated by a blank in the column). **Skip:** The test will be skipped (explanation required in "Comment" column). **Blocked:** The test cannot be run (explanation required in "Comment" column).
System Configurations	In most cases, the workstation, network, and server cluster IDs from the "System Config" worksheet.
Bug ID	If the test failed, the identifier(s) assigned to the bug by Tracker when the tester entered the report.
Bug RPN	The risk priority number (severity multiplied priority) of the bug(s), if applicable, in a column next to each bug ID.
Run By	The initials of the tester who ran the test.
Plan Date	The planned date for the first execution of this test case.
Actual Date	The actual date it was first run.
Plan Effort	The planned effort (person-hours) for this test case.
Actual Effort	The actual duration (person-hours) required.
Test Duration	The actual clock time required to run the test.
Comment	Any comments related to the test case, required for those test cases in a "Skip" or "Blocked" state.

Table 7: Test tracking

As the test organization runs each test, the state of each case will change from "In Queue" to one of the other states noted in the table above. In the ideal situation, at the System Test Phase Exit meeting, all the test cases will be in a "Pass," "Warn," or "Closed" state on the Test Case Summary worksheet for the final pass.

Bug Tracking

For each test that identifies a problem and enters a "Fail" or "Warn" state, the tester will open a bug report in the bug tracking system. For each defect, the bug tracking system will store (at a minimum) the information shown in the following table.

Field	Meaning
Bug ID	A unique identifier for each bug.
Summary	A one- or two-sentence summary of the failure observed.

Field	Meaning
Failure Description	A text field, free-format, consisting of three sections: **Steps to Reproduce:** A detailed, numbered process that will recreate the bug. **Isolation:** The steps performed to isolate the problem, including verification on other platforms, bad-unit checking, and other pertinent tests. **Regression:** A short description of whether this failure is a regression, and why or why not.
Severity	The seriousness of the effects of each potential failure, from one (most damaging) to five (least damaging), on the following scale. 1. Loss of data: Bug causes loss of user (end-user, operator, etc.) or system data. 2. Loss of functionality: Bug blocks use of a major functional area (can include non-functional problems like performance that impose unacceptable delays in functionality). 3. Loss of functionality with a workaround: Bug blocks use of a major functional area, but a reasonable affected-user workaround exists. 4. Partial loss of functionality: Bug blocks some unessential portion of a functional area. 5. Cosmetic error: Bug allows normal functionality but with significant blemishes (especially in the user interface or system responsiveness).
Priority	The importance of fixing the problem, based primarily on the ability of the delivered system to meet customer needs, though also on logistical project issues, regulatory or standards compliance, or other business considerations, from one (most important to fix) to five (least important to fix). 1. Urgent: Bug requires immediately resolution. 2. Essential: Bug is must-fix for release. 3. Valuable: Bug significantly reduces the value of the system to one or more customers or users. 4. Desirable: Bug should be resolved in this release if possible within feature, budget, and schedule constraints; otherwise in next scheduled release. 5. Discretionary: Bug can be fixed whenever possible in some future release, allowing for other priorities.
Resolution Notes	Once the bug is closed, this should include a description of the final resolution.

Field	Meaning
Submitter	The name of the tester or other engineer who identified the problem, defaulting to the current user. For remotely generated reports, this will specify either the contact name or the outsource test organization itself.
Submit Date	The date on which the bug report was opened.
Owner	The person responsible for moving the bug report to its next, and ultimately terminal, state.
State	The state of the issue, as follows: **Review:** Awaiting a peer review by another tester. **Rejected:** Review failed; behavior is not a bug. **Reported:** The problem is deemed by the test engineer fully characterized and isolated. **Assigned:** The problem is accepted as fully characterized and isolated by development, and an owner, responsible for fixing the problem, is assigned. **Build:** The problem is believed fix and the corresponding changes are checked into the source code repository awaiting the new test release. **Test:** The new test release, containing the fix, is installed in the test lab, and someone owns testing the problem to evaluate the fix. **Reopened:** The fix failed the retest. **Defer:** Do not fix for this release. **Closed:** The fix passed the retest.
Quality Risk	A classification of the symptom against the quality risk involved as follows: Functionality Performance, Load, Capacity, or Volume Usability Reliability or Stability Installation, Maintenance, or Operations Localization Security or Privacy Documentation Error Handling and Recovery Integration Other N/A

Field	Meaning
Subsystem Affected	A classification of the subsystem most impacted by the bug as follows: User Interface Edit Engine Tools File Installation or Configuration Documentation or Packaging Document Management System Other Unknown N/A
Test ID	The test identifier (from the test-tracking spreadsheet described above) corresponding to the test case the engineer ran that uncovered the issue. Also allowed are “Ad Hoc”, “Other” and “Unknown”.
Version Number	The release identifier against which the bug was identified.
Closed Date	The date on which the issue was confirmed fixed or put on hold, which is used only when the issue is in a “closed” or “deferred” state.

Table 8: Bug tracking

Test case execution and bug reporting are two critical internal testing processes that each member of the test team must master. To make sure that a consistent and high-quality job is done on these tests, all new hires shall attend a one-day training course within two weeks of their start date. Test technicians shall each have an engineer assigned to mentor them. The mentor shall comprehensively and completely review each technician’s test results and bug reports prior to those results entering the tracking systems.

The test team shall write accurate, concise, thoroughly-edited, well-conceived, high-quality bug reports. Comments, questions, or concerns about the bug reporting process or the quality of the bug reports should be directed to Jamal Brown.

To avoid test escapes, the testers shall adopt **an active bias towards bug reporting**. This implies the following attitudes and behaviors:

- The Test Manager shall inform the team that its role is to find bugs, and shall reinforce this message.
- If in doubt, testers shall assume the observed behavior is incorrect until they satisfy themselves otherwise.
- If Sumatra and any reference platform disagree about the correct product behavior, a bug exists and the tester discovering the discrepancy shall report it.
- If the requirements specifications, design specifications, on-screen help, or any other official document and Sumatra disagree about correct behavior, a bug exists and the tester discovering the discrepancy shall report it.

- The tester shall report as a bug any event leading to loss of data or the crash, wedge, hang, or other availability incident involving any client, server, or network, even if the bug appears to result from misconfiguration of the environment.
- If, in the professional opinion of the tester, the behavior of Sumatra does not conform to reasonable expectations of Web and/or GUI program behavior and/or quality, or is otherwise confusing, misleading, or ambiguous, the tester shall report a bug.
- Questions about which component a bug resides in do not affect whether anomalous behavior is incorrect. Testers shall report such bugs regardless of whether the assignment of responsibility for repair is clear.
- Disagreements between the IST team and any other group about whether a behavior is incorrect shall be escalated to Jamal Brown, Jenny Kaufman, and Kate Hernandez for resolution. Jamal, Jenny, and Kate shall have the sole authority to cancel or defer any bug report for whatever reason.

Release Management

During Integration Test and Prior to the Start of System Test

IST will accept a test release as often as daily from either the Development Team or the Release Engineering team. The releases may be built using harness, stubs, drivers, and other scaffolding software to allow execution. The source code must be under configuration management and the release built entirely from check-in code, but no revision numbering is required. Should the installation process for these releases differ from the customer installation process, then the Development Team shall install the test release. The Development Team may provide emergency patches to be applied in the test environment, but these must be built on source code checked into the configuration management system. Because the integration strategy is a parallel one, subsequent releases may contain different and non-overlapping functionality from previous releases. The Development Team shall include Release Notes that document the functionality in each Integration Test release.

During System Test

IST shall receive test releases every Monday morning from the Release Engineering team. These releases shall consist only of source code that is intended to be delivered in the GA release to customers; i.e., it may not contain any harnesses, stubs, drivers, or other scaffolding software, with the possible exception of probes inserted by dynamic analysis tools such as code coverage and memory leak detectors. The Sumatra system will support querying for a specific build number through the Help\About menu selection. Build numbers shall be designated as “3.1.XXX” where “XXX” is a three-digit number starting with “001” for the first build. As builds will occur nightly, we expect the final build to be somewhere around 200; i.e., labeled “3.1.200”.
IST shall install the test release in the same manner as a customer. IST shall not accept emergency patches or upgrade releases into the test environment, except as part of a

specific test case designed to test that functionality. The Development Team shall provide Release Notes that document the functionality and bug fixes in each System Test release. Since the test environment is shared, once System Test starts, the release management process for System Test, being stricter than that for Integration Test, prevails.

Outsourced Test Execution

During System Test, external test labs shall perform usability testing and localization testing. Because test releases delivered for System Test shall be in customer-like format, each external lab is expected to be able, with some level of phone support, to install the Sumatra system in their lab.

Each test lab shall perform testing according to their own process, but the test cases and bugs shall be tracked in the same reporting process as bugs found by IST. Jamal Brown shall give the external labs a copy of his test tracking spreadsheet, which shall list the test cases they shall run. Jamal Brown shall work with the System Operations and Administration team to provide access, via VPN or other secure mechanism, to the bug tracking system.

Jamal Brown shall serve as the point of contact for escalation of any problems that impede test execution or results reporting at the external labs. However, since it is important that we not waste the money spent on outsourced testing, we shall deliver to the external labs a test release that has already gone through one cycle of System Test and that Jamal Brown has deemed stable enough for external testing.

Risks and Contingencies

The following table describes the key risks to success of this plan, and contingency and/or mitigation plans to address them.

Risk	Contingency/Mitigation
Breakdown in resolution and escalation process(es).	Attempt to work-around blocking issue, accept inefficient progress and delays in plan fulfillment.
	OR
	Halt testing under continuation criteria, resolve process problem.
	OR
	Continue testing via workaround meanwhile resolving process problem for next incident.
External lab(s) can't install Sumatra.	Prevent problem by thoroughly testing installation before sending a test release to an external lab.
	OR
	File bug report(s) against installation process.
	AND
	Send an experienced test engineer to install.

Risk	Contingency/Mitigation
Last-minute, large-impact change in requirements, design, features, or other portions of development plan.	Accept increased quality risks due to incomplete testing of change.
	OR
	Accept increased budget risk by staffing up or outsource to do sufficient testing at last minute
	OR
	Accept increased schedule risk by delaying ship date.
Bad test release discovered after test cycle has begun.	Institute automated smoke testing in Release Engineering to detect bad releases.
	OR
	Stop testing during "blown" cycle, revert to the "last known good" test release, and continue testing, accepting the reduced fulfillment of planned tests and loss of efficient progress.
Test environment incomplete on Integration Test or System Test phase entry date.	Start testing with those tests that can be run on the environment available, accepting limited test fulfillment, inefficient progress, and significant gaps in test coverage.
	OR
	Slip the planned test phase entry and exit dates day-for-day until the environment is available.
Test environment incomplete for one or two whole phases.	Rework plan to lengthen phases as required, drop any tests that were dependent on the missing configuration(s), identify increased quality risks to project.
	OR
	Institute a third (graveyard) shift. (NB: This decision must be made before retaining technicians and may not provide total mitigation of the risks.)
Test environment system support unavailable or not proficient.	Accept inefficient progress and unfulfilled planned tests.
	OR
	Retain a contractor for the short-term to provide support.
Buggy deliverables impede testing progress, reduce overall test coverage, or both.	Prevent problem via thorough Component Testing by Development Team.
	OR
	Adhere to continuation criteria and stop testing if problem becomes too bad, accepting delay of schedule.
	OR
	Attempt to continue testing on current schedule and budget, accepting a poor quality system release to customers.
Gaps in test coverage.	Exploratory (guerilla) testing allows testing of areas not covered by planned tests.
	AND

Risk	Contingency/Mitigation
	Possibly, we can use structural coverage analysis techniques, field-failure analysis, and customer data to identify gaps to be filled. This should probably be decided on a resources-available basis during test design and development.
Unclear customer usage profiles/ environments results in incomplete or incorrect testing.	Get information on actual customer usage from Technical Support as well as from Marketing's Alpha and Beta efforts and the Sales Team, then create tests to cover those areas.
	OR
	Accept, due to schedule or resource limitations, the possibility of testing being misaligned with customer usage.
Slips in development schedule affect entry criteria readiness on scheduled dates.	Hold to entry criteria, reducing the number of features delivered by slipping the test and overall project schedules and letting the final increment(s) of functionality drop off.
	OR
	Violate the entry criteria and accept the increased risk of poor quality due to insufficient time to find and fix bugs, which can include the risk of utter project failure.
Unanticipated resource needs for complete testing.	Acquire the resources and exceed the test budget.
	OR
	Delete some other testing and reallocate resources to the missing testing area. (NB: May not be possible for all situations).
	OR
	Decide to skip the tests for which sufficient resources were not budgeted and accept the increased quality risks associated with that test coverage gap.
Test team attrition.	Accept temporary slowdown in planned test fulfillment.
	AND
	Backfill ASAP with contract tester having appropriate skills.
Key player (outside test team) attrition.	Identify alternate or back-up people for each non-test key player.
	OR
	Backfill the role within the test team upon loss of the supporting person, resulting in inefficiency and slower planned test fulfillment.
Can't hire appropriate automated test engineer.	Use all available means to locate the ideal candidate, including recruiters.
	OR
	Increase pass duration to allow for slower test fulfillment.
Debugging in test environment	Provide customer-like configurations for development as well as test, manage the configurations to ensure reproducibility of bugs in development environments.
	OR

Risk	Contingency/Mitigation
	Accept the slowdown or stoppage of test plan fulfillment along with the reduction in efficiency caused by the need to restore the test environment after debugging.

Table 9: Risks and contingencies

Change History

The following table outlines the change history for this document.

Rev.	Released	Description/Changes	Author/Editor
0.1	9/18/2002	First draft to Sumatra team.	Jamal Brown
0.2	9/20/2002	Second draft for final review.	Jamal Brown
1.0	9/24/2002	Plan of record agreed to by stakeholders.	Jamal Brown

Table 10: Change history

Referenced Documents

See the various Sumatra documents in "Projects/SpeedyWriter/Sumatra/" on Jupiter, especially the FMEA chart, project schedule, and budgets in "Projects/SpeedyWriter/Sumatra/Test". These documents are Read-Only for reference, with the source documents in the XYZ repository.

See the "Software Cafeteria Integrated Development Environment User's Guide" on the Intranet Web site's Development Standards page for more details on how to use the XYZ repository.

This test plan format complies with ANSI/IEEE Standard 829-1983. Please see *IEEE Standard for Software Test Documentation*, published by the Institute of Electrical and Electronics Engineers.