

Seapine Software

Style Guide 2014

Preface

Bringing Control Proof Insight Control to Complex Product Development

Seapine Software's process-centric solutions equip product development and IT organizations with tools, technologies, and best practices to deliver quality software products on time and within budget. As a company focused exclusively on the needs of product development organizations, Seapine Software is uniquely positioned to deliver innovative, tightly-integrated solutions that improve product quality, traceability, and compliance.

Contents

1.0 Logo Design.3

2.0 Typography6

3.0 Color Pallete.8

4.0 Graphics11

5.0 Collateral13

Style Guide

1.0 Logo Design

1.1 Logo Design

Stacked Logo



Primary Logo



1.2 Logo Design

The logo should be reflected on all internal and external communication tools. It is made up of two components, the Seapine company name and the tree—which together make up the company signature.

For quality purposes and brand recognition it is important that the Seapine logo maintain the following guidelines.

Trademark: Required on printed material only.

Clear space: Do not place content within area of implied segmented line.

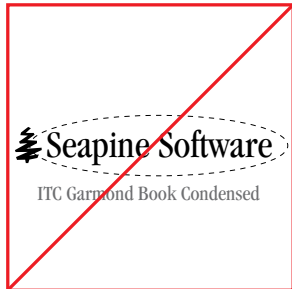
Minimum Width: Do not make logo smaller than 1.5" wide.



1.3 Logo Design

The following are examples of how not to use the Seapine logo.

- (1.1) Do not use ITC Garmond Book Condensed for the logo. Only use original outlined logo.
- (1.2) Do not alter or rearrange logo.
- (1.3) Do not condense the logo.
- (1.4) Do not stretch the logo.



(1.1)



(1.2)



(1.3)



(1.4)

Style Guide

2.0 Typography

2.1 Typography

Helvetica Neue is the official Seapine font family used for all marketing materials. For all office applications on PC, Arial is the corporate typeface. It should be used on all PC-generated and laser printed materials, as well as in Powerpoint presentations, if Helvetica Neue or Helvetica is not available. In order to maintain consistency across marketing documents, a standard set of paragraph styles has been created.

Primary Font:

Helvetica Neue

Substitute Fonts:

Helvetica, Arial (*only when absolutely necessary*).

- Do not use condensed or stretched fonts.

Helvetica Neue Black Italic

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789**

Helvetica Neue Bold

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789**

Helvetica Neue Bold Italic

***ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789***

Helvetica Neue Medium

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789**

Helvetica Neue Medium Italic

***ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789***

Helvetica Neue Regular

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789**

Helvetica Neue Italic

***ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789***

Helvetica Neue Light

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789**

Helvetica Neue Light Italic

***ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789***

Helvetica Neue Thin

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789**

Helvetica Neue Thin Italic

***ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789***

Helvetica Neue UltraLight

**ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789**

Helvetica Neue UltraLight Italic

***ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789***



Style Guide

3.0 Color Pallete

3.1 Color Palette

The personality of Seapine Software is embodied and communicated through the color palette. Seapine blue is the primary color used in all Seapine identity and should not be replaced by any other color or variant of blue. The PMS color for Seapine Blue is PMS 285.

The Seapine Software color system also includes a secondary color palette which is primarily used for emphasis or as an accent color.

Primary Colors

PMS 285
CMYK: 89-43-0-0
RGB: 0-125-195
HEX: 007dc3

K 100
CMYK: 50-40-40-100
RGB: 0-0-0
HEX: 000000

Gray 70%
CMYK: 58-49-46-14
RGB: 109-110-113
HEX: 6d6e71

Secondary Colors

PMS 297
CMYK: 48-1-0-0
RGB: 115-205-243
HEX: 73ccf3

PMS Bright Orange
CMYK: 0-69-83-0
RGB: 243-114-60
HEX: f3713c

PMS 2597
CMYK: 85-100-0-0
RGB: 82-46-145
HEX: 522e91

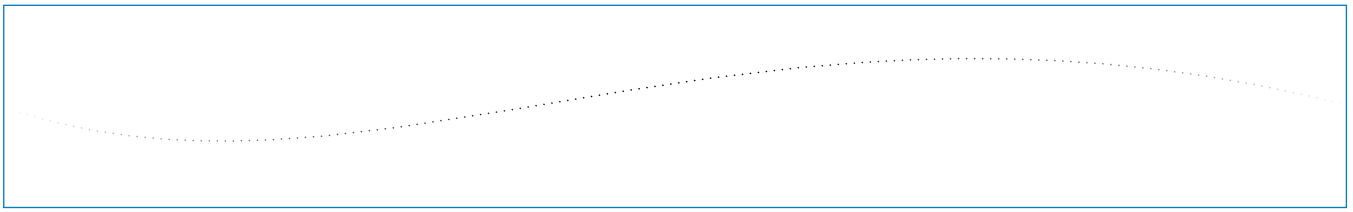
PMS 376
CMYK: 48-1-0-0
RGB: 115-205-243
HEX: 73ccf3

Style Guide

4.0 Graphics: Curve

4.1 Graphics: Curve

Seapine's curve element plays a crucial role in the look and feel of Seapines brand. The dashed curve line and stroke style should not be altered.



Traceability Curve

Style Guide

5.1 Collateral: White Paper

6 Tips for Successful System Integration Testing

Best Practices to Improve Your Testing Process

Whether you are upgrading your human resources implementation in an enterprise, adopting a new electronic health record (EHR) solution in a hospital, or replacing the billing system behind your online commerce site, a key set of best practices can help your team avoid common challenges and conduct a successful system integration test every time.

The goal of a system integration testing (SIT) project is to verify that an existing software system continues to function properly after the introduction of one or more new software components. SIT is also done for completely new systems, but without the additional challenges of updating a system that's already running in production.

Why Is System Integration Testing Important?

System integration testing validates that the system as a whole matches stakeholder requirements. Perhaps most importantly, it's the last chance to find defects and issues before the new system goes live and problems start to negatively affect the users and, ultimately, the business.

Prior to this point in the software development lifecycle, testing is focused on individual system components, often relying on test harnesses or other artificial means to simulate the interaction between the component under test and other system components. When testing reaches the SIT stage, the interaction with other components is usually no longer simulated. The component is placed in a test environment that mirrors the production system, to see if it properly handles business processes that span components.

Small, simple system projects can get away with taking short cuts. Good SIT processes are critical for the bigger, more complicated systems typically found at the enterprise level. These six SIT best practices will help you establish a better process for performing testing on large, complex systems.

What Is a System?

For the purposes of this paper, we'll use "system" as shorthand for a collection of software programs in an enterprise, which consists of many disparate applications and services that must communicate and exchange information to support larger business objectives. When one or more of the system's components is changed, the correct functioning of the entire integrated system must be verified.

System integration testing, then, is a phase in the software development lifecycle where the goal is to determine if the modified system works as expected before the changes are put into production.

1. Make Sure Your Test and Production Environments Match

It goes without saying that you shouldn't conduct your initial system integration testing in production. All kinds of problems can arise when releases are tested in the production environment—database corruption, critical module failures, system crashes, and so on. Even making "minor" changes to the production system can cause serious problems.

Obviously, testing should be conducted in a separate test environment that mirrors the production system. Test environments allow teams to better manage releases, because they can complete basic performance and load testing earlier, while there is still time to make adjustments to the implementation approach. Test environments also provide more control over which version of a component or system is installed.

For a test environment to be effective, it's important for it to be up to date. If changes have been made to the production environment that aren't reflected in the test system, you risk problems cropping up in one environment that aren't in the other, which only complicates and slows down the testing effort. Make it a part of your process to update your test environments whenever you update the production environment—even if it's just a “tweak” to production.

Virtual Test Environments

Sometimes, teams feel like they have to test in the production environment because they don't have enough machines to establish a mirrored test environment. However, advances in virtual machine technology have made it harder to justify not having a test environment.

Virtual machines offer a great way to run automated scripts in a separate, customized environment, which can exist without interfering with normal computer operations. The virtual machine can be configured to ensure the environment remains constant on each script run.

2. Use Good Test Data

To know if a system integration test is successful, you have to know what output to expect from the test data being input. For example, if you're testing the integration of a hospital billing system, you should know what kind of billing statement that the test data, “Adult, age 50, no insurance, appendectomy” should generate. By identifying data in/expected data out, you can confidently define test success or failure. Otherwise, you may end up with poorly written or vague test cases that don't help you successfully validate system requirements.

Another benefit of creating good test data is that test automation becomes a realistic possibility for basic regression tests and for driving test harnesses. Automating these practices can take a large burden off the testing team and significantly speed up your testing, recouping the time you spent creating the test data.

Don't Forget to Reset

Along with good test data, it's important to have a way to reset the systems that are part of the testing effort to some known starting point. This can mean resetting a virtual machine, dropping database tables and running create-and-load scripts, or deleting directories and their contents from servers and restoring files from a source code control system. Regardless of the reset method used, controlling the starting conditions means issues are more likely to be reproducible.

3. Use a Common Repository for Issues and Tests

A common repository ensures issues are visible across teams and makes it easy to reassign responsibility for investigation and issue resolution. Repositories provide a system-wide view of the state of issues and of production readiness.

Your issue and test repository should be available to all SIT participants to reduce the chance that issues will fall through the cracks. It also allows for better decision-making, because an individual issue can be viewed within the context of other issues. If you have different teams working on a common issue list, you need to make sure they are all seeing the most up-to-date list.



To ensure you've captured the information you'll need to communicate efficiently and make the best decisions, we recommend that you document the following in your issue and test repository:

- A brief description of the issue
- Who found the issue, and when
- Steps to reproduce (steps from the test case or the specific actions and data that resulted in the issue)
- Affected component or subsystem
- Who is currently responsible for issue resolution
- Issue status (open, blocked, in progress, resolved)
- Fix priority
- Environment and build information

4. Establish an Issue Triage Process

In addition to establishing a common issue and test repository, you should also have an issue triage process that includes a triage schedule, identifies the team member who owns triage, and lists all participants. An effective triage process can help you balance the demands of decision makers, ensure high priority issues are fixed first, resolve priority conflicts, and maintain a history of issues fixed for a release. It can save you many headaches if you have it in place ahead of time, or really hurt you if you don't.

A defined triage process is an effective way to provide a flexible, group-based approach to issue prioritization:

- **Decision-makers** can quickly find issues and assign a priority.
- **Moderators** can view priority conflicts and resolve them with the appropriate decision makers.
- **All team members** will have a clearer picture of the work to complete before a product release, which helps ensure high priority issues are fixed first.

For information on establishing an effective triage process, read our "[Triaging Issues in TestTrack](#)" (PDF) article.

5. Have a Communication Plan

Establishing a plan for communicating between teams can keep projects moving smoothly through the SIT process, especially if some individuals or teams are geographically distributed. A communication plan becomes critical if the testing team contains remote, temporary, or inexperienced testers.

At a minimum, the communication plan should address how testers will be notified when their part is ready for testing, and how the team will manage handoff notifications to ensure the testing effort is conducted in the proper order.

Inexperienced and temporary testers may lack the basic knowledge more experienced testers have. Make sure the communication plan answers basic questions, such as:

- How do I know what to test?
- How do I let you know how it went?
- What do I do if it doesn't work?
- Who do I notify, and how (call, email, IM, etc.)?
- Where do I save my results?

A test management solution can simplify your communication plan by automating alerts, notifications, reporting, and other project-critical communications. However, you will also want to make sure your key contact list or lead matrix is updated and shared with the team.

6. Leverage Automated Reporting

Management is often eager for results, especially as deadlines approach and expectations for pushing changes into production increase. Without automated reporting, it can take hours just to answer a simple question like, “How did testing go today?”

Avoid this time sink before testing begins by leveraging a tool that can automatically deliver high-level status reports to management. Typically, management will want to see reports that include the following data:

- A list of the highest risk items
- The number of tests run that day
- The number of tests that have passed or failed
- The number of tests left to be run
- A breakdown by system, to identify what modules are causing trouble

Conclusion

These six best practices should help establish a better process for performing system integration testing on large, complex projects. Good test data and up-to-date test environments will help you establish a strong foundation for SIT. A centralized, integrated test management solution—combined with an efficient triage process—will help improve visibility and avoid errors during testing. And, leveraging automatic reporting can save time when communicating test and issue status to your team and stakeholders.

About Seapine Software

With over 8,500 customers worldwide, Seapine Software, Inc. is the leading provider of quality-centric product development solutions. Headquartered in Cincinnati, Ohio, with offices in Europe, Asia-Pacific, and Africa, Seapine’s development solutions help organizations ensure the consistent release of high quality products, while providing traceability, metrics and reporting, and compliance.

www.seapine.com

Style Guide

5.2 Collateral: Customer Success & Data Sheet

QA Wizard Pro has enabled RSA to find issues very early in its test cycles. RSA used QA Wizard Pro to create automated scripts for one of its most complex applications. They run their automated regression scripts at the very start of each new code release into the test environment. This process has caught a number of defects very early in the test cycle, whereas before they would not have found the defects until much later in the testing cycle.

Result

RSA has seen an improvement in productivity with Seapine because it has streamlined communications between development, QA, and RSA's different divisions. Seapine solutions have become an integral part of RSA's testing toolset. "In 2014, we plan to have all projects use TestTrack, which helps us standardize all test artifacts," said Tom.

In addition, Seapine's built-in email tracking allows RSA to send and receive emails at a Seapine item level. All emails related to a defect, test case or test execution are tracked and stored with that particular item. "You do not have to go in your mail client and search for these emails. The integration of TestTrack Client with Outlook Client has greatly improved the collaboration between our departments and partners," added Valentin.

Another benefit is productivity gains in testing. While the automated scripts run, RSA executes its manual test cases in parallel. The increased flow has allowed them to absorb time delays in project deliveries to the testing team.

"We are pleased with our results and are looking to expand automation to other applications," said Tom.

"Before the use of Seapine, it would take great effort to compile enterprise reports, let alone a single project report."

Valentin Blaga,
Senior Quality Management Analyst,
RSA

QA Wizard Pro has enabled RSA to find issues very early in its test cycles. RSA used QA Wizard Pro to create automated scripts for one of its most complex applications. They run their automated regression scripts at the very start of each new code release into the test environment. This process has caught a number of defects very early in the test cycle, whereas before they would not have found the defects until much later in the testing cycle.

Result

RSA has seen an improvement in productivity with Seapine because it has streamlined communications between development, QA, and RSA's different divisions. Seapine solutions have become an integral part of RSA's testing toolset. "In 2014, we plan to have all projects use TestTrack, which helps us standardize all test artifacts," said Tom.

In addition, Seapine's built-in email tracking allows RSA to send and receive emails at a Seapine item level. All emails related to a defect, test case or test execution are tracked and stored with that particular item. "You do not have to go in your mail client and search for these emails. The integration of TestTrack Client with Outlook Client has greatly improved the collaboration between our departments and partners," added Valentin.

Another benefit is productivity gains in testing. While the automated scripts run, RSA executes its manual test cases in parallel. The increased flow has allowed them to absorb time delays in project deliveries to the testing team.

"We are pleased with our results and are looking to expand automation to other applications," said Tom.

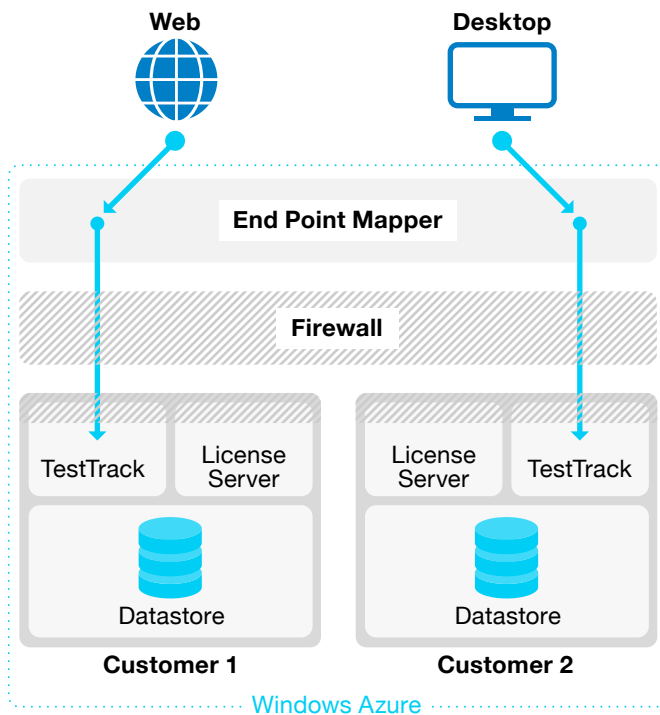
"Before the use of Seapine, it would take great effort to compile enterprise reports, let alone a single project report."

Valentin Blaga,
Senior Quality Management Analyst,
RSA

Seapine Cloud: TestTrack Security and Privacy Overview

Seapine Software takes the security and privacy of customer data seriously. We've partnered with Microsoft to use their Windows Azure hosting environment to ensure security and application reliability.

TestTrack and the Seapine Cloud environment also include a number of features to help minimize risk to your data, such as isolated environments, multiple firewalls and restricted access, stringent TestTrack security, and isolated data.



Isolated Environment

Each TestTrack server and its data are in an isolated environment. A unique virtual machine with separate security, backup, and endpoints is defined for each company, ensuring there will not be any data leakage between customers.

Firewalls and Restricted Access

There are multiple levels of defense provided between your TestTrack server and the Internet. First, Microsoft Azure does not allow any inbound network traffic from the Internet except through a set of Seapine defined endpoints. These endpoints are further restricted to only allow inbound traffic from a restricted set of network addresses.

Remote management of the virtual machine is also restricted to only allow access from Seapine's corporate network. TestTrack access from the desktop client can be restricted to only allow connections from your company's internal network. Finally, TestTrack web access is restricted to SSL (HTTPS) access only. In addition to these external restrictions, we also configure the operating system firewall to limit access to only those services.

TestTrack Security

The TestTrack server is configured to limit access. As mentioned, the only web access allowed is over SSL. Desktop client access can be limited within your corporate network. We also require encryption between the TestTrack server and the native client to secure your data in transit.

Isolated Data

Your data is isolated from other customers by using separate database instances for each project in your virtual machine. Backups are performed daily and stored at an off-site facility to allow for disaster recovery.