ABSTRACT

REPORT ON A MTSC INTERNSHIP AT SEAPINE SOFTWARE

by Jessica L. Warren

This four-chapter report describes the Master of Technical and Scientific Communication internship I completed as a member of the technical publications team at Seapine Software, Inc. during the fall of 2008. It provides an overview of the company, my role as a technical writing intern, and the projects I worked on; a description of a Frequently Asked Questions (FAQs) set I created as my major project; and an analysis of how my internship experience compared to my expectations generated from my coursework at Miami University. In Chapter 4, I also address my development from intern to full-time technical writer.
REPORT ON A MTSC INTERNSHIP AT SEAPINE SOFTWARE

An Internship Report

Submitted to the
Faculty of Miami University
in partial fulfillment of
the requirements for the degree of
Master of Technical and Scientific Communication

Department of English

by

Jessica L. Warren

Miami University

Oxford, Ohio

2012

Advisor: ____________________________
Jean A. Lutz, Ph.D.

Reader: ____________________________
W. Michele Simmons, Ph.D.

Reader: ____________________________
Glenn Platt, Ph.D.
# Table of Contents

List of Figures ................................................................................................................................. iv
Dedication ......................................................................................................................................... v
Acknowledgements ......................................................................................................................... vi

## Chapter 1 | Introduction ............................................................................................................... 1
  About Seapine Software ................................................................................................................ 1
  Organizational structure and culture ............................................................................................. 1
  Work environment ......................................................................................................................... 2
  Internship role ............................................................................................................................... 3
  Mentorship ....................................................................................................................................... 4

## Chapter 2 | Internship Projects .................................................................................................... 5
  Timeline .......................................................................................................................................... 5
  Startup plan ..................................................................................................................................... 5
  Knowledgebase articles .................................................................................................................. 8
  Surround SCM training material ...................................................................................................... 10
  QA Wizard Pro user documentation and FAQs ............................................................................ 11

## Chapter 3 | Major Project: QA Wizard Pro 2009 FAQs .......................................................... 15
  Planning the approach .................................................................................................................... 15
  Researching the product and user tasks ......................................................................................... 16
  Drafting the FAQs .......................................................................................................................... 17
  Sending content through review .................................................................................................... 20
  Revising content ............................................................................................................................. 20
  Delivering the final FAQs ................................................................................................................. 21

## Chapter 4 | Internship Analysis .................................................................................................. 24
  Classroom tools used in the workplace ........................................................................................... 24
  Need for specialization ..................................................................................................................... 25
  User analysis for a task-oriented approach .................................................................................... 25
  Usability testing ............................................................................................................................... 27
  Value of the technical communicator ............................................................................................. 29
  Experience gained over time ............................................................................................................ 29
  Looking forward ............................................................................................................................... 31
  Conclusion ....................................................................................................................................... 32

References ........................................................................................................................................... 33

Appendix A | Knowledgebase Articles ............................................................................................ 34
Appendix B | Changes Made to Surround SCM 2009 Training Guides ............................................ 35
Appendix C | QA Wizard Pro 2009 Java Application Support Documentation ..........38
‘Adding Java applications’ topic in Adobe PDF.................................................................38
‘Adding Java applications’ topic in Microsoft HTML Help..............................................40

Appendix D | Single-Sourced Topic in Madcap Flare .........................................................41
User guide PDF preview ..................................................................................................42
Training guide PDF preview .........................................................................................43
Differences between outputs .........................................................................................45

Appendix E | QA Wizard Pro 2009 FAQs ..................................................................46
FAQs in Adobe PDF.......................................................................................................46
FAQs in Microsoft HTML Help......................................................................................49
LIST OF FIGURES

Figure 1: Seapine Software’s organizational structure

Figure 2: Distribution of work during internship

Figure 3: TestTrack Pro defect record used to request a new knowledgebase article

Figure 4: Seapine Software’s documentation workflow

Figure 5: QA Wizard Pro interface

Figure 6: Drop-down text formatting in QA Wizard Pro FAQs
DEDICATION

This work is dedicated to my best friend and love, Matthew Jones, for bringing more hope and happiness to my world than I could have ever wished for.

Our diplomas are going to look so great hanging next to each other.
ACKNOWLEDGEMENTS

I would like to thank the following people for offering me their help and support in this accomplishment: my advisor, Jean Lutz, for providing the encouragement and guidance I needed to complete this degree and believe in myself; my internship committee members, Michele Simmons and Glenn Platt, for their time and valued feedback; my Seapine colleagues, Sarah Wigser and Amy Kearns, for welcoming me to my first tech writing team, making it an awesome team to join, and letting me stick around after my internship; and, last but far from least, my parents, James and Melissa Warren, for encouraging me to set lofty goals and achieve them (even if they take me too far away from home).
CHAPTER 1 | INTRODUCTION

Between August 18 and November 13, 2008, I completed a professional internship to fulfill one of the requirements of the Master of Technical and Scientific Communication (MTSC) program at Miami University. For the internship, I was employed as a technical writing intern at Seapine Software, Inc. in Mason, Ohio. In this chapter, I introduce the company, its organizational structure and culture, and my role as a technical writing intern.

About Seapine Software

Seapine Software is a leading developer of application lifecycle management (ALM) solutions for product development and IT organizations. With over 8,500 customers worldwide, Seapine offers the following development tools to “help organizations of all sizes streamline communication, improve traceability, achieve compliance, and deliver quality products.”

(Seapine Software website, 2011)

- **TestTrack Pro** for workflow and issue tracking to identify and resolve defects in application development and functionality.

- **TestTrack TCM** for test case management to identify specific conditions applications should work under or functionality they should provide; to manage all aspects of the software testing process.

- **TestTrack RM** for requirements management to identify business, functional, and other requirements an application or specific feature must meet before it is delivered to customers.

- **Surround SCM** for software configuration management to track changes in application source code and make assets available to all development team members.

- **QA Wizard Pro** for automated functional and regression testing to determine if applications work as expected under test case scenarios.

Organizational structure and culture

Seapine was established in 1995 and has grown to approximately 100 full-time employees among its headquarters in Mason, Ohio and international offices in London, Munich, and Melbourne. This growth is considerable, but Seapine is still a relatively small company, making
it easy to keep communication open across development, quality assurance, sales, and marketing departments. Figure 1 shows the overall organizational structure at Seapine, including interaction between employee roles.

![Figure 1: Seapine Software’s organizational structure. Technical writers work with members across various teams and report to the Director of Corporate Communications.](image)

**Work environment**

Seapine provides a very comfortable work environment for its employees. I worked at Seapine’s headquarters in Mason, Ohio, which is spread among only two floors in the same office building. The dress code is almost as casual as it gets, allowing employees to wear shorts and sandals in the summer and jeans year round. Each floor includes a large kitchen where employees can enjoy unlimited coffee, tea, and soda free of charge; kick back and take a break in front of one of the big screen televisions; get a sugar fix from the M&M’s dispensers; or play a quick game of foosball or Wii bowling. It is clear that Seapine’s founders and head executives care a great deal about making sure their employees enjoy coming in to work every day and are not overworked or stressed.
Despite the casual environment and relatively small office space, there appeared to be a divide between departments. Employees spend a large amount of their time working individually in their own cubicles or offices. I was surprised on my first day on the job to learn that I would be part of the Marketing department. I was concerned I would be working on projects that were not technical in nature, but that turned out not to be the case. I expected the technical writing team to be either its own department or in a department involved in product development. Being part of the Marketing department meant I sat on the second floor, while most of the development and QA team members sat on the third floor. This separation may have made it more difficult to remember to consult or include technical writers in early stages of the development process because we weren’t considered part of that team, and the separation presented some obstacles for me as an intern trying to learn about coworkers and how they did (and do) their jobs.

**Internship role**

I joined the technical publications team in Seapine’s Marketing department for my internship. In this role, I performed the following tasks to contribute to Seapine’s product documentation:

- Reviewed feature design documents and interviewed subject matter experts to learn about changes in upcoming product releases documentation and collect the necessary information for future documentation.

- Created text for new product features based on customers’ needs and existing knowledge while adhering to the corporate style and tone.

- Learned how to use Seapine’s products to test and verify written content as a user.

- Used help authoring and screen capture tools to structure content for print and online user and training guides.

- Followed the established documentation workflow for edits, reviews, and revisions until tasks were completed.

- Produced updated versions of help files and PDFs to be installed with products.

- Wrote knowledgebase articles based on caveats in new functionality and requests from support team members.
Mentorship

When I joined Seapine, there was only one other technical writer on the technical publications team, Amy Kearns. The Director of Corporate Communications, Sarah Wigser, served as the team’s manager and the company’s corporate editor for all external communications. Amy and Sarah were the two Seapine employees I worked with most closely as they taught me what I needed to know about becoming a technical writer for a software company.

Sarah formally signed on to be my internship mentor. She was responsible for setting the overall documentation schedule and distributing tasks between Amy and me. As I worked on tasks, Sarah and I would meet at least once a week, if not more often, to discuss my progress and any general questions I had. Sarah’s open door policy also made it easy to keep her updated on my progress because she encouraged me to drop by her office anytime I needed anything, and I did that frequently. If she did not have an answer to my question, she always knew who to direct me to so I had everything I needed to do my job.

While Amy was not my official mentor, there is no doubt she played a major role in my professional development during my internship. She was the coworker who helped me set up my test builds to use when documenting features, showed me how to use Seapine’s products in my day-to-day work, and provided advice based on her own experience when I was unsure of the best way to approach a task. The technical publications team was small, and I relied heavily on Amy’s guidance as I learned about Seapine’s products and documentation.

In the remainder of this report, I discuss the documentation projects I worked on during my internship and analyze the technical communication strategies I applied in my work.
CHAPTER 2 | INTERNSHIP PROJECTS

When I began my internship, Surround SCM 2009 development was nearing completion and QA Wizard Pro 2009 was in the early stages of development. The specific release dates for these products were staggered, meaning one would be released before the other, and that development schedule shaped my internship projects and timeline. In this chapter, I briefly describe the projects I worked on, and the schedule for when I worked on them, during my 14-week internship.

Timeline

In addition to a technical writer co-op startup plan, I worked on four projects for the two 2009 product releases:

- Surround SCM 2009 Admin and User Training Guides
- QA Wizard Pro User Guide
- QA Wizard Pro FAQs
- QA Wizard Pro Training Guide

Figure 2 shows how these projects were distributed during my internship.

![Figure 2: Distribution of work during Seapine Software technical writing internship.](image)

Startup plan

I was provided with a four-week startup plan the first day of my internship to help acclimate me to Seapine’s employee policies and products. At the end of each week, I emailed Sarah a
progress report of what I tasks I had completed and questions or concerns that had arisen. Following is a week-by-week overview of the activities included in my startup plan.

**Week 1**

- Reviewed Seapine’s technical writing policies and the corporate style guide to make sure I would be able to incorporate them into my writing process.

- Read recent blog posts from product managers; the three most recent editions of the corporate monthly newsletter; white papers about quality-centric application lifecycle management; and articles about product licenses, support, and server architecture to gain a basic understanding of how Seapine’s products worked.

-Installed Adobe Framemaker on my computer and began online Framemaker training to learn how to use the authoring tool the technical publication team was using at the time. The team was in the process of moving their documentation files to Madcap Flare, but I was still required to work with some Framemaker files to meet documentation deadlines.

**Week 2**

- Attended TestTrack TCM, TestTrack Pro, Surround SCM, and QA Wizard Pro product demonstrations. These demos were web meetings provided to new users or potential customers. As someone who had never used a Seapine product before my internship, this introductory tour was essential before I began documenting the products and using them in my day-to-day work routine.

- Read Seapine articles to familiarize myself with best practices for using the TestTrack suite and Surround SCM.

- Continued Framemaker training. The bulk of the training was completed during the second week of my internship.

- Discussed Seapine’s documentation workflow and review process with Sarah before I began any writing assignments. I discuss this process in more detail in Chapter 3 of this report.

- Installed two versions of TestTrack and Surround SCM on my computer. One version was the version I used to access project files and assignments from Seapine’s internal server. The other version was the latest beta build I would be documenting for the upcoming releases, which I used to capture usage tasks and images for the user help and training material.
• Reviewed Seapine’s checklist for working with Framemaker files and an overview of how the technical publications team uses TestTrack to monitor assignments. These documents were references to prepare me for my first project, which was to update the Surround SCM 2009 training material in Framemaker.

**Week 3**
• Reviewed the technical publication team’s internal documentation for using Surround SCM to manage documentation files and wiki articles about Surround SCM to understand how it is used for software configuration management and source file version control.

• Retrieved local copies of training guide files from the Surround SCM server. These are the files I updated for the 2009 release, and I used Surround SCM to manage the changes I made.

• Read and followed the existing Surround SCM and TestTrack training guides to learn how to use the products. This task also helped familiarize me with the Surround SCM training guide, which I would be updating shortly after learning how to use the product.

• Reviewed my first knowledgebase article assignment. I discuss knowledgebase article requests in more detail later in this chapter.

**Week 4**
• Reviewed the steps for creating and managing knowledgebase articles to learn about the format and tone used in these articles and how to actually post one to Seapine’s website.

• Met with Sarah to discuss how I felt at this point in my startup, if I was getting to know my coworkers, and what projects I would work on.

• Began Surround SCM 2009 training material updates.

It was not until the middle of my fourth week at Seapine that I began my direct work on any user documentation project. As explained in *Hodges’ Harbrace Handbook*, writing is a process and everything cannot be done at once (392). Time to generate, organize, develop and clarify ideas is required, and it is important to remember that specific activities may be repeated or revisited during the process. My startup plan tasks were not directly part of the formal writing process, but I do feel they played a critical role in preparing me for beginning the writing process and forming general ideas of everything it would include during my internship. I appreciated the time
to learn about my coworkers, Seapine’s products, and how openly the technical publications team communicates because this knowledge was critical to my being able to do my job.

**Knowledgebase articles**

The technical publications team is responsible for maintaining a knowledgebase of articles related to troubleshooting tasks or content only applicable to specific situations outside of the scope of a product’s documentation. During my internship, I wrote seven new articles to add to the knowledgebase (see Appendix A). I created some of these articles to address information in a feature design document that was unique to usage environments too specific to include in the user documentation but could be helpful to the users in specific situations, such as a specific issue that might be encountered when configuring a Seapine product on a system where another commonly used application like an antivirus program may affect how the product works. Other articles were created to address specific requests from the product management and support teams based on user feedback or from the development team to address known product issues.

To request a knowledgebase article, my coworkers added a defect record to the TestTrack Pro database that corresponded with the related product. While a typical defect is an issue or problem in a product’s functionality, these databases are also used to track customers’ questions and requests for new features, graphics, or knowledgebase articles. When a knowledgebase article is requested, the defect record is assigned to Sarah, who reviews it and determines its priority based on the documentation schedule and subject matter. Sarah then assigns the knowledgebase article to the writer, who works it into her schedule accordingly. While these articles are typically short, they are a lower priority than documentation because the documentation must be completed in correlation with the product development schedule so the help files can be installed with the product. On the other hand, the articles are often for features in an upcoming release so the content cannot be published until after a product version is released. Figure 3 shows an example defect record used to request a knowledgebase article.
Each knowledgebase article request was unique; some included attached articles already written by a development or support team member, and I would have to revise them to match the corporate style; some provided only a brief description of the topic. If the knowledgebase article request included an attached draft, I would begin by reviewing the content and marking notes on areas I felt were unclear, too detailed, or too vague. I would then discuss any questions I had, whether working with an existing draft or creating a draft from scratch, with the subject matter expert (SME), or in this case, the individual who submitted the article request. After I completed a draft, I sent it to the SME for technical review. I edited the article based on the technical review
feedback and then submitted it to Amy for a peer review. Again, I edited the article based on any peer review feedback and then submitted it to Sarah for final review. I made any final revisions based on Sarah’s feedback and then added the article to the knowledgebase database to be published either immediately or after the corresponding product released. The entire article creation process could take anywhere from half a day to an entire week, depending on how much research I needed to conduct and feedback turnaround.

The final step in the knowledgebase article creation workflow is to close, or in TestTrack terminology, fix, the defect record after the article was published. All the historical information about the article is managed in the TestTrack database and any Seapine employee can access the record at any time.

**Surround SCM training material**

My first product-related assignment was to update the Surround SCM Administrator and End User Training Guides for the 2009 release. This task required me to put my newly developed Framemaker knowledge to use. Amy assisted me with setting up the Surround SCM 2009 client and server on my computer so I could capture screenshots for changes in the user interface (UI); she also provided a list of specific updates that would be needed in the existing training guides. This project not only taught me more about using Framemaker and the style used in Seapine’s training guides, but it also helped me learn how to use Surround SCM as both an administrator and user.

This project was also the first where I used Surround SCM to track changes made to documentation files. All of Seapine’s training and user documentation source files are stored on an internal Surround SCM server. I first checked out the training files, which meant I retrieved local copies from the server and locked the server files so no other user could make changes to them while I was working with them. I used the latest product build to update screenshots and made any necessary content updates based on UI changes or text I felt was unclear or inconsistent.

Because training changes are based on the user guide updates, which had already been through the review process and approved, training guide updates did not go through as rigorous a review
as the user guides. Nevertheless, to avoid overwhelming Sarah with a large review request, I would check in the changes for each module as I completed them and change their status to “Needs Review” to notify Sarah that the updates were ready for review. When she completed the review, she would change their status to either “In Revision” if I needed to make changes and resubmit for review or “Approved” if there were no or only minor revisions needed.

When I had completed updating all of the training topics and screenshots, I followed the technical publication team’s instructions for generating PDFs and spot-checked the entire guides. If I saw any mistakes, bad page breaks, or incorrect formatting, I made the corrections in the files and regenerated the output until the PDF was ready for review. The review process for final PDFs was the same as all other drafts. I saved a copy of the final, approved training guide PDFs on our network server so Sarah could access them after the product released. I wrote a list of all the changes I made to the training to provide our services team members with an overview, and a “heads up,” of what changed so they would be prepared to deliver the training to Surround SCM 2009 customers. When Surround SCM 2009 released, Sarah provided the services team with the updated training materials and the list of changes. Refer to Appendix B for the complete list of changes.

**QA Wizard Pro user documentation and FAQs**

I worked on three separate documents for the entire QA Wizard Pro 2009 user documentation project: the user guide, the frequently asked questions, and the training guide.

**User guide**

After the Surround SCM 2009 training updates were completed, I moved on to the QA Wizard Pro 2009 User Guide. Amy and I both worked on this project because I was not familiar enough with the product or our documentation set to manage all the updates myself. Due to time constraints, this project was more about learning Seapine’s documentation process instead of learning how to use the product.

QA Wizard Pro 2009 broadened its testing capabilities with the introduction of support for testing Java applications (Seapine Software, 2008), which was the feature I was assigned to document. To understand this new feature, I reviewed the design document provided by the
development team to determine what information was relevant to the documentation. Because I had never used QA Wizard Pro before, I was a little confused by the details of the new feature and decided to interview the subject matter expert, the feature developer in this case, for a better understanding of the new support and how it is beneficial to users. The developer was happy to take the time to answer my questions and teach me how to add a Java application and run tests against it.

With a better understanding of the new feature, I installed a build of QA Wizard Pro on my computer to work with. A sample Java application, WysiCRM, was installed with QA Wizard Pro so that is what I used to test the functionality and capture screenshots. Before I made any changes to the user guide files, I reviewed the existing guide to determine if I should add new topics or make changes in existing ones. Java support required its own topic to go along with the existing topics for adding web and Windows applications, the other two supported application types. Java support also affected the ‘Getting Started’, ‘Setting general options’, ‘Adding applications’, and ‘Adding versions’ topics, which all referenced supported application types and UI elements that requested information specific to application type.

Like all other documentation projects, these user guide files were stored on the Surround SCM server. I checked out only the topics I needed to update and began making changes using Madcap Flare. To draft the text for the new topic about adding Java applications, I performed the task using the test build and wrote down all the steps as I performed them. After I had the procedure complete, I considered the additional information I collected from the SME interview that might be useful for users who needed to add Java applications. I added the information to my draft and then compared it to the existing topics for adding web and Windows applications and pulled wording from them to ensure consistency in the guide. Finally, I applied formatting to the topic to make it look like the other topics and printed a copy for technical review. The review process for this topic followed the same workflow as all projects. When the topic was approved and the wording associated with the task was tacked down, I updated the existing topics with the necessary information related to Java support and checked in all the changes. Refer to Appendix B for the completed ‘Adding Java applications’ topic that I added to the user guide.
My final task for this user guide project was to use Flare to generate the help files and user guide to be installed with the product. The digital help files were generated as Microsoft HTML Help and the printed user guide was generated as an Adobe PDF file. Amy provided the procedure for generating the help, but I had to repeat the process a few times as I noticed formatting errors in the output. This process taught me new-to-me Flare functionalities and exactly how Seapine produces the physical documentation provided with its products.

**Training guide**

After the QA Wizard Pro user guide updates were complete, Amy and I worked together to update the training guide. Training is provided as a separate service, so the training guide is not included in the product installation. Because we were no longer working against the development team’s build schedule, we had a little more time to work with the training guide files. This time allowed us to single-source the training content into the user guide project files.

As stated in Sarah O'Keefe’s *Single-Sourcing Overview*, “The term ‘single-sourcing’ refers to the process of creating multiple output formats (usually printed and online) from a common set of source files” (2). Our goal in single-sourcing the QA Wizard Pro user help and training topics was to better manage the source content and efficiently utilize text that overlapped documents. At this point, we had completed all user guide updates and moved all that content from Framemaker into a Flare project. The next step was for us to merge the training content that was still in Framemaker into the user guide Flare project and figure out how to use the single Flare project as a source for both the user guide and training guide.

Before we actually moved the training content into the Flare project, Amy and I worked with Risa, who is also the QA Wizard Pro trainer, to discuss what information she usually covers in training and how she uses the guide. We learned that some of the content did not follow the procedural order to naturally advance with the program and some of the content simply was not used at all. How a training course is delivered is typically customized to meet the customer's needs, but the training material needed to be general enough to apply to all user scenarios. This discussion provided us with suggestions to restructure the content and remove unused content while merging training text and images into the Flare project that combined training material and material in the user guide.
Actual updates needed for the QA Wizard Pro 2009 training were minimal, as we used the updates already in the user guide. The bulk of this project entailed copying and pasting content from Framemaker topics into the Flare project, editing each topic accordingly, and ensuring the correct content is included in the separate outputs. Some topics included more content about every option on a dialog for the training as well as cross-references to other topics for the user guide. We applied the training condition tag to the full details and the user guide tag to the cross-references. Because the training guide is only available in print, we wanted all the information to be sequential and available in topics for high-level tasks so users could follow the entire process before moving on. The user guide is more referential and users can move back and forth between topics, in print or HTML help, and skim information more easily because they are most likely looking for a quick answer and are not trying to learn a complete process.

In Flare, I applied conditional text tags to indicate which text is only to be included in the user guide (highlighted in blue) and which text is only to be included in the training guide (highlighted in pink). Content that is not highlighted is included in both documents. The separate targets to generate the user guide and training guide were saved in the Flare project and each was configured to exclude text that is marked with the other guide’s conditional tag. Refer to Appendix D for an example of a single-sourced topic I worked on and previews of the two different PDF outputs generated from the same topic.

**Frequently asked questions (FAQs)**

To provide a unique project I could work on and manage predominantly on my own, Sarah and I decided that I would create a Frequently Asked Questions (FAQs) section to incorporate into the QA Wizard Pro 2009 User Guide. The main goal of this project was to provide new users with an overall summation of the automated testing process and create a point of reference for the main tasks associated with using QA Wizard Pro. In Chapter 3 of this report, I provide an in-depth explanation of this project and the steps I took to complete it.
CHAPTER 3 | MAJOR PROJECT: QA WIZARD PRO 2009 FAQs

For the QA Wizard Pro 2009 release, I worked on a set of Frequently Asked Questions (FAQs) to include in the user documentation installed with the product. I consider this the major project of my internship because it was a task I was individually responsible for, it was a unique communication format that did not previously exist in the user documentation, and I learned a great deal about Seapine’s overall communication, development, and documentation processes. In this chapter, I describe my process for creating and delivering the FAQs. In Chapter 4, I provide an analysis of how my process at the time contradicted the task-oriented approach taught in the MTSC program, why I took the following approach, and what I now observe after reflecting on the process applied to this project.

Planning the approach

Seapine’s technical publications team follows a typical technical writing workflow for all writing tasks. The specific tasks shown in Figure 4 correspond with working on the user documentation for new product features, but I followed the same general workflow when creating the FAQs.

![Documentation Workflow](image)

**Figure 4:** Seapine Software’s documentation workflow.

I created a documentation plan before working on the QA Wizard Pro FAQs. According to *Writing Software Documentation: A Task-Oriented Approach*, a documentation plan consists of two parts (208):

- The design plan defines what the document will contain (content) and what it will look like (layout, language, graphics, etc.).
The project plan explains how you will produce the document including details about the schedule and resources needed.

The plan was informal and began with Sarah and me meeting to discuss the goal and approach for the project. This discussion was necessary before getting too far in to the project to ensure that Sarah and I agreed upon the individual tasks I would complete for the project, how long each task might take, and what could be expected in regards to the final deliverable.

The QA Wizard Pro development schedule allotted approximately three weeks for me to work on the FAQs. We determined the target audience was new QA Wizard Pro users, and the project goal was to create a list of common questions/answers to provide an overall reference for the complete automated testing process using QA Wizard Pro functionalities. The next step was for me to research the product and the automated testing process to determine what information should be included in the FAQs and then meet with Sarah again to discuss the chosen content before I began any drafting.

**Reseaching the product and user tasks**

QA Wizard Pro is a tool that quality assurance analysts use to automate functional and regression testing of their own applications. It allows users to create scripts that perform specified actions on features in an application, which can then be run to test the applications’ functionality in less time than would be required for the analyst to manually test the application.

To understand using QA Wizard Pro in the testing process, I worked with the sample applications and scripts provided with the product to experiment with its functionalities. As a new user myself, I listed questions that came to mind as I worked through different areas of the UI (see Figure 5). I consulted the existing user documentation to clarify terminology and drafted a list of questions based on each task users perform during the automated testing process.
In addition to working with the application and existing documentation, I met with the QA Wizard Pro sales engineer, Risa Cohen, who served as the subject matter expert for this specific project. As a sales engineer on Seapine’s services team, Risa delivers QA Wizard Pro training to customers and assists customers in customizing QA Wizard Pro to best fit their testing needs. Risa and I discussed the list of questions I composed during my initial research of the product and some of the most common questions she encounters when working with new users. After we agreed upon the questions that would best serve as a good starting point for new users, I moved on to drafting the FAQs.

**Drafting the FAQs**

The initial FAQs draft included 21 questions divided into six subsections. These subsections provided a high-level organization for the FAQs based on the main tasks associated with using QA Wizard Pro. I wanted to encourage users to use the FAQs, so I used slightly more informal wording for some of the questions to more closely match the phrasing a user might use in conversation. The answers in the FAQs were intended to be complete yet brief so the user could
quickly move on; however, when applicable, the answer ended with a cross-reference to another topic in the user help to provide additional information if the user wanted to learn more.

In the following sections, I describe the subsections used to organize the FAQs based on actions performed when creating automated tests and the questions initially included in each subsection.

**Getting started**
A new QA Wizard Pro user may want to begin with learning about the features and functionalities of the product. I used a general heading for this subsection to identify that the grouped questions were targeted at users with no prior experience working with QA Wizard Pro. The following questions were included in the first draft of the ‘Getting started’ subsection.

- Where do I begin?
- What is a workspace?
- What is the application repository?
- Can I use QA Wizard 4 scripts with QA Wizard Pro?
- Are there any sample QA Wizard Pro Scripts?

**Application repository**
When a user is familiar with the basics of QA Wizard Pro’s user interface and terminology, the first step in the testing process is to add the test application to the repository in QA Wizard Pro. The following questions were included in the first draft of the ‘Application repository’ subsection.

- How do I add an application for testing?
- What is the difference between the local application repository and the global application repository?
- What is an application version?
- What do I do when I need to test a new application version?
Script recording
Once the test application is added to the repository, users can begin recording scripts, which are the files used to run automated tests. The following questions were included in the first draft of the ‘Script recording’ subsection.

- What happens during recording?
- What’s in a script?
- What are window and control properties used for?
- Why does my script have low-level actions?

Script playback
After a script is recorded, users run it to see the automated test in action. This event is referred to as script playback. The following questions were included in the first draft of the ‘Script playback’ subsection.

- What happens during playback?
- Can I monitor scripts during playback?
- Where do I see the results?

Script modification
Users may need to modify their scripts if changes are made in the application, errors or unexpected results are returned during playback, or additional factors need to be tested. The following questions were included in the first draft of the ‘Script modification’ subsection.

- What if I want to add more stuff to a script?
- What are the differences between Grid View and Text View?
- Can I add notes to a script?
Script troubleshooting

If anything unexpected happens during the automated testing process, the troubleshooting questions were intended to serve as a quick reference for the most common problems that could be encountered. The following questions were included in the first draft of the ‘Script troubleshooting’ subsection.

• What if a script fails? How do I tell if it’s a problem with my script or if the application changed?

• What types of checkpoints does QA Wizard Pro support?

Sending content through review

Seapine’s typical review process begins with sending a complete draft to the SME for a technical review of the content’s accuracy and brevity. The SME could be the product manager, the development manager, a services engineer, a QA analyst, or a technical support specialist, depending on the document’s subject matter. For the FAQs, I sent the draft to Risa, the services engineer (trainer) who initially helped me compile the list of questions, and requested that, based on her experience working with our customers, she confirm the accuracy of the answers and provide feedback on anything that was missing.

The next step in the documentation review process includes a peer review from Amy to make sure the document follows Seapine’s style. However, due to the short project deadline and Amy’s simultaneous work on the QA Wizard Pro documentation, that step was skipped for this project. I incorporated Risa’s feedback into the revised draft and printed a clean copy of the FAQs to submit to Sarah for review.

Revising content

As this was my first major project and I was still learning Seapine’s style, the FAQs draft required quite a few revisions. Some of the edits pertained to the language I used being too informal in spots. For example, contractions I used in questions were expected to be separate words and my reference to the variety of elements that could be added to scripts as “stuff” was not specific or, in my opinion, very professional.
Other edits needed pertained to the organization of the questions. Sarah felt that some of the questions were grouped in the wrong subsection. For example, she suggested that I move ‘What is the application repository?’ from the ‘Getting started’ subsection to the ‘Application repository’ subsection because it was specifically about explaining what the application repository is and not necessarily a question that needed to be answered to start using the product. Sarah also felt that some of the questions were beyond the FAQs scope and should be deleted altogether. For example, the question about the types of included checkpoints was about a specific concept that might not be performed often or by the typical user.

After I made the requested edits, I submitted the revisions to Sarah for another review. She returned them to me with some other minor change requests but also informed me that after I made those final changes, the FAQs would be approved to add to the user guide.

**Delivering the final FAQs**

The QA Wizard Pro user documentation is generated in two formats: Adobe PDF as the print output and Microsoft HTML Help as the online help file accessible from the Help menu. To ensure the FAQs were included in the documentation, I needed to add a new topic to the existing user guide Flare project. I entered the approved FAQs text in the new topic and applied the appropriate paragraph and heading styles from Seapine’s stylesheet so the text would match the other text in the generated documentation outputs. I also applied some additional drop-down text formatting to group the questions and corresponding answers and utilize a design element that could make the FAQs easier to use. Figure 6 shows an example of the drop-down text formatting.
FAQs

Getting started

☐ Where do I begin?

☐ What applications can I test with QA Wizard Pro?

☒ What is a workspace?

A workspace is where you organize scripts and related datasheets. You must create at least one workspace before you can record scripts. See Creating workspaces.

☐ What are the differences between Grid View and Text View?

☐ Can I use QA Wizard 4 scripts with QA Wizard Pro?

☒ Are there any sample QA Wizard Pro scripts?

Figure 6: Drop-down text formatting in QA Wizard Pro FAQs. The “drop-down” effect is displayed when clicking a link in the Microsoft HTML Help file that opens from QA Wizard Pro’s Help menu.

As shown in Figure 6, the drop-down text effect only displays the questions until the user clicks the text to reveal the answer. This effect creates a less cluttered design for the FAQs in the HTML Help and allows users to view only the information they want without having to scroll through a long list of questions and answers. Because the PDF version of the user guide is intended for print purposes, the drop-down text effect is not applied in that output. However, the project stylesheet does apply boldfacing to the questions to help make them distinguishable and easier to scan in print.

After I completed the updates and formatting in Flare, I added a link to the FAQs topic to the table of contents file, which is used to compile all the help files as one output. I sent the FAQs through one final review to ensure I correctly applied the formatting and that it worked as expected in the generated outputs.

Sarah approved the FAQs in the Flare project and my final task was to generate the user documentation after all the updates were checked in to Surround SCM. I had generated documentation from Flare for the Surround SCM 2009 training updates, but this project was the first where I had to make sure the user documentation was checked in to the product’s source code for the next scheduled build. Amy helped me with generating and spot-checking the documentation, and she showed me where I needed to check in the files on the server. The check
in process involved my updating the version of the complete user guide PDF and HTML Help file installed with QA Wizard Pro with the final changes, including the new FAQs topic, for the 2009 release.

I notified Sarah and QA Wizard Pro’s development manager that the user documentation updates were complete and checked in, and that was the final step to complete the FAQs project. It was possible I would need to make edits after the QA team checked the documentation files included in the next build; however, no requests for changes came up. Refer to Appendix E for the complete and final version of the QA Wizard Pro FAQs in PDF and HTML Help format.
CHAPTER 4 | INTERNSHIP ANALYSIS

My technical writing internship at Seapine Software was my first real-world, professional experience working on technical documentation. It was an exciting opportunity to apply the knowledge and skills I gained from my studies in the MTSC program to new and challenging projects outside an academic setting. After my internship ended, I accepted a full-time position as a technical writer at Seapine, and I continue to work in that role now as I complete this report. In this final chapter, I provide an analysis of how my internship experience compared to my expectations and reflect on my professional development from intern to full-time employee.

Classroom tools used in the workplace

Courses in the MTSC program taught me how to use a variety of tools—Adobe InDesign, Photoshop, and Dreamweaver; advanced features of Microsoft Office suite; and Madcap Flare—to create different types of communications. Many of these tools aid in the design of technical documents, which is an aspect of a technical communicator’s job I personally enjoy. I expected to use these tools more regularly and create new documents from scratch, but that was not the case in my role at Seapine. At the time of my internship, the technical writers did not have much responsibility in the overall design of the documents they produced. Sarah managed the corporate stylesheet used to ensure all Seapine documentation looked cohesive, and it was simply my responsibility to apply the styles and preconfigured page layouts correctly in the guides.

Though I did not use many of the programs I learned about in the MTSC program during my internship, I know that understanding how to use them is extremely beneficial to my career. One program I did learn about and apply was Madcap Flare, which I was introduced to in ENG 514: Designing and Testing User Documentation. Experience with a help authoring tool —Madcap Flare, Author-It, Robohelp, and Framemaker—is practically mandatory for technical writers in the software industry. If I had not taken the user documentation elective, I think I would have been at a disadvantage when trying to obtain an internship in my specialized field. This course was an elective that I managed to take by extending my MTSC coursework an extra semester. In it, I learned about analyzing the needs of software application users, using a task-oriented approach to help users apply software to their professional goals, and single-sourcing and
managing content for various documents, which are all skills I applied first-hand and continue to expand upon while working at Seapine.

**Need for specialization**

In the MTSC program, students needed to select a technical field to specialize in and tailor their supplemental coursework to that field. To prepare myself for a career in writing for the software industry, I chose computer science as my specialty area and took three different computer language courses: Visual Basic (VB), Java, and Actionscript. At the time, I found these computer science classes to be very challenging, and I didn’t expect to ever be in a situation where I would be working with code. It was my internship at Seapine that helped me understand just how important a strong technical background is to writing in this field.

QA Wizard Pro’s coding language is based on VB. Taking that VB course at Miami made jumping in to the program a little less intimidating for me. By no means do I understand the language well enough to create complex scripts, but I knew enough at the time I started my internship to work unassisted with the sample scripts in the application and follow the examples in the existing documentation. This knowledge was a time saver in regards to learning about the tool, but it also helped better prepare me for communication with members of the development, services, and support teams. The “developer” language is used in internal design documents and emails frequently, and I often have to translate it to terminology that users with a less technical background can understand. If I had not taken specialized courses at Miami, I think it would have taken me much longer to pick up on terminology at a software development company. I was glad to have what basic computer science knowledge I did during my internship because it made me feel slightly more on par with my coworkers, the majority of whom are software developers.

**User analysis for a task-oriented approach**

My MTSC coursework taught me that the first step in any technical documentation process is to analyze the target user. The needs of the user must be understood and used to shape a documentation plan. However, it’s not just as simple as define the user’s role and then explain the operations they can perform with the application being documented. That limited scope
results in what is referred to as the default manual in *Writing Software Documentation: A Task-Oriented Approach*.

The default manual consists of descriptions of the interface elements and operations the user can perform with them. But presenting information just at that level ignores the situated nature of software use and can often leave the user frustrated and anxious. The solution, then, is to examine the user in enough detail at the level of the situated workplace context and develop from the investigation a number of documentation themes, design ideas, organizational strategies, and so on so that it reflects the user integrating the program instead of just the program. (139-140)

Performing a sound user analysis was one of the biggest challenges I faced in my internship. I was unfamiliar with the tasks software developers, testers, and quality analysts typically perform or the general processes they follow in their roles. That’s not to say I did not analyze the users before developing my documentation plan for the QA Wizard Pro FAQs. In fact, because I was working on adding content to existing documentation instead of creating a new document from scratch, I had to quickly develop a general understanding of the users. I formed a user definition by reviewing the existing documentation, which included examples of how the applications are used by different professional roles, and discussed the common user roles and activities with Sarah and Amy so I would have the same ideas about our users as the rest of the technical publications team. I felt this process was a good starting point for me because it helped me familiarize myself more with the documentation, and I received clear descriptions of our users from my fellow technical communicators. Sarah and Amy understood what I was trying to achieve and their experience made them excellent references when creating my user definition.

However, a proper user analysis can take a great amount of time and resources, and I simply did not have the time in my internship to understand our users’ informational needs as well as the tacit knowledge, or the “motivations, behaviors, values, and knowledge pertaining to […] users that might not be visible on the surface” (Barker, 123) needed to apply a fully task-oriented approach in my deliverables. What technical communicators need to remember is there is a difference between the activities users performs and the tasks they are trying to complete. The activities are the basic steps they follow on the path to their goal. The task is the goal itself. For
example, in QA Wizard Pro, a user needs to set up their application repository before they can test the applications. That would be the main task they focus on at a time. Actions like adding the applications to the repository, capturing the initial control properties, and creating testing variables are all activities that are part of setting up an application repository.

Collecting more tacit knowledge might have helped me more easily identify user tasks for the FAQs instead of just addressing the activities they can perform in QA Wizard Pro. I do believe if I had had more time during my internship to better understand the tasks our users are really trying to accomplish, or to see the bigger picture, the FAQs organization may have been categorized by task rather than application functionality and they would have been, overall, more effective.

**Usability testing**

In addition to the importance of a strong user analysis, usability testing after a document is created can also help determine just how effective the document really is while it's still in a draft state. For example, “A manual writer might test the manual by arranging to have people representative of the target audience attempt to use it under realistic conditions. He or she would note the difficulties they encountered and revise the manual to eliminate the sources of the problem.” (Anderson, 162) When working on MTSC projects, a great emphasis was put on planning, conducting, and reporting on usability tests; I cannot think of a single project where some form of usability testing with an actual user was not conducted. Even when I could not work with an actual representative of the target audience, I was still able to set up a test environment and find participants to test my work to some degree.

It’s fair to say I was a little surprised to learn that Seapine does not conduct formal usability testing on its documentation. Instead, the writers rely on technical reviews from subject matter experts – product managers, sales engineers, and support team members who serve as the voice of the customer because they directly communicate and interact with customers and have the strongest understanding of how Seapine’s products are being used. Sometimes documents go through multiple tech reviews, and the subject matter experts are always willing to help answer questions whenever necessary to help ensure the documentation delivered is accurate and efficiently meets the needs of the user. This method may not meet the standard definition of
usability testing in the technical communication world, but it certainly is better than nothing at all. I almost always find new ways to revise my drafts after technical reviews, so the information I receive from them is comparable to what I might collect from a formal usability test.

I knew I would not be testing my content with users as I worked on the FAQs project, so I had to improvise when it was time to draft and then self-review the document. I tried to put myself in the role of a new user and focused on the information I would need or expect if I were trying to figure things out. It wasn’t too difficult to manage as I was a new user, so I asked myself questions as I drafted the FAQs such as ‘Is the answer thorough yet brief?’, ‘Why would I use the FAQs?’, ‘Did the question and answer make sense based on the categories?’, and ‘Where could I find additional information if I wanted to learn more?’.

Using that informal testing approach, I completed the initial FAQs draft and sent it to Risa. I provided her with the questions I created to test the FAQs. I also requested a technical review to make sure the content was accurate and everything we agreed upon during our first meeting was included as expected and in a manner that our customers would understand. After the technical review and some edits, the FAQs were ready for Sarah to review. The FAQs went through a couple of reviews before they were approved and added to the user guide, but I feel the absence of proper usability testing resulted in them not fully meeting our users’ needs.

“User testing supports task orientation because it helps the documenter build a clear picture of the user's work environment and encourages a broader view of software use than mere learning of functions” (Barker, 253). The FAQs I created did answer the 'what' and 'how' of basic QA Wizard Pro functionality and guided users to more information that could better address their goals, but they were not task-oriented as they seemed to be centered on user activity instead of broader tasks. Being able to run the FAQs by an actual QA analyst would have been a much more effective test to see if users would actually be motivated to use them. Instead of focusing on the features in QA Wizard Pro, I should have asked some of Seapine's QA analysts how they use the application in their day-to-day tasks and what the ultimate objectives of their tasks were. Obtaining that knowledge may have helped me catch the limitation of the content before delivering it, but its absence was a lesson learned for larger, future projects.
Value of the technical communicator

It seemed like a recurring theme across various MTSC courses I took that technical writers are stereotypically undervalued in the workplace. It may be true that some people don’t really understand the technical writer’s role and think that what we do can be done by anyone, but I consider myself lucky in that I haven’t witnessed that attitude first-hand at Seapine.

As a company, Seapine acknowledges the value useful documentation adds to its products and overall success. This understanding is what made it easy for me to decide to accept a full-time position after my internship. Project managers, developers, quality assurance analysts, support specialists, sales managers, and services engineers all find examples of information they want to ensure is documented in some way, so they turn to the technical publications team. Our contribution is to ensure that the information our customers need is made available and delivered in an efficient and effective fashion that helps our customers use our products to achieve their goals.

While internal feedback and input is readily available for the technical publications team, I believe Seapine could further demonstrate its appreciation for high-quality documentation by allowing the technical publications team more direct access to customers. User involvement is critical to delivering “better” documentation. And better documentation results in better products and experiences for users.

Experience gained over time

Transitioning from intern to full-time employee was a seamless effort. Technically I was a full-time employee during my internship, just on a temporary basis. Sarah understood that regardless the amount of previous experience I had, I needed to take some time to learn about processes and products before I could jump right in to projects. I’ve seen the same attention given to other writers who joined our team after me, and the only real difference between my introduction to Seapine as an intern vs. a full-time employee was my startup plan. I appreciated the startup plan but I would have preferred working with the products sooner or getting to know my coworkers better during the early weeks of my internship. Working with the tools and talking face-to-face with people are more active and effective ways to collect knowledge, and my continued experience at Seapine has given me plenty of time to do just that.
My internship experience was invaluable. But I couldn’t learn everything in 14 weeks. (Luckily I wasn’t expected to.) I feel the work I did complete during my internship was an excellent starting point to a long-term career. Since I became a full-time employee, I have had the opportunity to learn more about Seapine’s products, documentation, customers, and employees. I use our products in my day-to-day activities and just by using the product more to complete my own tasks, I have become a target user to some extent. Granted, the tech writers use some products differently than other teams, but overall, it’s a company-wide standard to use Surround SCM to store source files and TestTrack to maintain defect records, use cases, and functionality requirement documents. Now that I know how to use these complex tools, I am more confident with identifying defects for our developers to correct, with requesting feature requests I think could enhance existing functionality, and with finding tacit knowledge that is entered in defect records that I might not find when reading design documents. For example, our support team often enters defects after a customer encounters a specific problem trying to accomplish a task. The developers might fix the feature causing the issue and report the fix or workaround to the technical publications team to address, but they might not explain what the customer was doing when they encountered the problem. Reviewing these types of records allows us to use a task-oriented approach in our writing and provides suggestions for index and meta keywords, help topic titles, and even blog posts. I didn’t work much in TestTrack during my internship, so I simply wasn’t exposed to this knowledge because it relates to specific tasks that I was not working with at the time.

When I look back on the FAQs now, it’s easier to see ways they could have been improved. As I worked on them during my internship, though, I didn’t see that I wasn’t taking the task-oriented approach taught in MTSC. Instead, I was more focused on joining a writing team with an established process and documentation set and making sure the work I contributed “fit in.” I tried to create a topic that I believed was cohesive with the existing user guide, and I feel I achieved that. The user guide to which I needed to add the FAQs did incorporate some elements of task-orientation, but there was room to improve. If I had possibly requested that the FAQs be a separate document or page on Seapine’s website, I might have applied a task-oriented approach without worrying if the information was presented in a way that differed from other QA Wizard Pro documents.
It also didn’t occur to me during my internship that I could ask other people in the company for input on the FAQs both before and after I drafted them. I worked with Risa because I was told she was my best point of reference. While she did provide valuable input and contributed a great deal to my being able to deliver the FAQs, I wish I had dropped in on other coworkers and asked what their jobs were like to gain more insight into what questions new QA Wizard Pro users might commonly ask. Now this approach is part of my research before documenting new features, and it is simply easier for me because I know my coworkers and our products better now than I did as an intern. I am able to ask questions that are less technical in nature and more contextual, which helps me gain the knowledge I need to deliver task-oriented work.

Looking forward
Right now, the team is still very small—just Amy, who is the technical writing manager, and me—and that can make it challenging to complete everything that needs to be done for releases and still make larger changes to our documentation. However, we have currently reworked the Surround SCM User Guide using a task-oriented approach that allowed us to revise the document in a way that invites users to find information about tasks they need to perform and to learn how different activities apply to a single task without needing familiarity with the user interface terminology. The decision to revisit our approach came after noticing that a large number of support case records included information that was in the documentation but that was overlooked or was simply missing altogether. Since the document has been revised, fewer documentation defect records have been submitted, which we take as indication that the task-oriented approach did enhance the usability and effectiveness of the guide. Because of the success of this document, we plan to incorporate similar changes in the TestTrack and QA Wizard Pro documentation when time is available.

Just within the past two years, Seapine’s technical writers have moved from the Marketing team to Development. This change may seem small, but I believe just making everyone aware that we are on the team and have input to offer has increased the open communication between the technical writers and developers, development managers, and even co-ops. The smaller development teams for individual products are gradually shifting to Agile development methods, which puts a great emphasis on including the technical writers earlier in the incremental
development process. I think our involvement in product development will increase more and more over time, and as I become more familiar with development processes, I believe I will have a more active role in suggesting changes or raising questions to consider about the usability of our products.

**Conclusion**

Reviewing my work samples and notes from my internship has allowed me to reflect on how I have progressed as a professional writer, which is something I might not have taken the time to do if not for this final degree requirement. With more time in my role, I have collected a greater arsenal of knowledge about our users and their tasks that I apply beyond just the activities I could document, and our entire documentation set for all our products has taken a noticeable turn towards a more task-oriented approach. Using a task-oriented approach is helping Seapine’s technical writing team make the documentation better fit the ever-evolving needs of our customers and is enabling us to contribute to Seapine’s continuous goal of delivering quality-centric products. And while that may be Seapine’s goal, I think it’s a good goal for all technical writers to focus on as well.
REFERENCES


APPENDIX A | KNOWLEDGEBASE ARTICLES

The following list includes the seven knowledgebase articles that I wrote during my internship and a brief description of each.

- **Moving the Seapine License Server** (www.seapine.com/kb/questions/1511)---Describes how to move the Seapine License Server from one location to another and includes links to references about preparing server files stored in Relational Database Management System (RDBMS) databases for the move and file permission and capitalization caveats.

- **Starting the PostgreSQL Server** (www.seapine.com/kb/questions/1509)---Explains how to manually start the PostgreSQL server on supported Windows, Mac OS X, and Linux operating systems.

- **Uninstalling PostgreSQL** (www.seapine.com/kb/questions/1510)---Explains how to manually uninstall the PostgreSQL Relational Database Management System (RDBMS) server from a computer after the Surround SCM server has been uninstalled.

- **LDAP Users and Controlled Hyperlinks** (www.seapine.com/kb/questions/1501)---Describes limitations of Lightweight Directory Access Protocol (LDAP) users accessing TestTrack projects through a controlled hyperlink.

- **Using Test Variant Field Codes in Test Cases** (www.seapine.com/kb/questions/1493)---Explains what test variant field codes are and how to use them to in TestTrack TCM test cases to automatically replace test variant values in test runs generated from the same test case.

- **Generating a TestTrack Workflow Transitions Report in Excel** (www.seapine.com/kb/questions/1507)---Provides steps for TestTrack users to create a report of their current project workflow and transfer the content to a Microsoft Excel file, which can then be used to share workflow information with team members or stakeholders who do not have access to TestTrack or to modify the workflow content in an application users may be more familiar working with.

- **Apache Support in the TestTrack SDK** (www.seapine.com/kb/questions/1508)---Provides a list of requirements for using Apache web server software with the TestTrack Software Development Kit (SDK).
APPENDIX B | CHANGES MADE TO SURROUND SCM 2009 TRAINING GUIDES

Surround 2009.0 Training Changes
Most of the changes to the training materials are cosmetic/screenshot changes. The user training manual is a subset of the admin training. Any changes we made in the following admin modules are also made in the corresponding user modules:

- How to Add Files
- How to Check Out and Edit Files
- How to Share Files and Add Notifications
- How to Create Branches
- How to Promote and Rebase Changes
- How to Create Reports

The following lists the changes to the admin training by module.

Course Preview – no changes

How to Add Files

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to create a repository</td>
<td>Updated Create Repository dialog box screenshot to show progressive disclosure design</td>
</tr>
<tr>
<td>How to add files</td>
<td>Updated Add Files dialog box screenshot to show progressive disclosure design</td>
</tr>
<tr>
<td>Getting a file</td>
<td>Updated Get Files dialog box screenshot to show progressive disclosure design</td>
</tr>
<tr>
<td>Practice, Adding Files</td>
<td>Updated Add Files and Get Files dialog box screenshots to show settings for examples</td>
</tr>
</tbody>
</table>

How to Check Out and Edit Files

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to check out a file</td>
<td>Updated Check Out Files dialog box screenshots to show progressive disclosure of Changelist options and Advanced Options areas</td>
</tr>
<tr>
<td>Setting Check Out Files options</td>
<td>Updated User Options dialog box screenshot to show current options</td>
</tr>
</tbody>
</table>

How to Check In and Merge Files

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checking in files</td>
<td>Updated Check In Files dialog box screenshot to show progressive disclosure design</td>
</tr>
<tr>
<td>Creating a changelist during check in</td>
<td>Updated Check In Files dialog box screenshot and text on expanding Changelist Options to show how to create a new changelist</td>
</tr>
<tr>
<td>Committing a changelist</td>
<td>Updated Changelists dialog box screenshot to show new Label button</td>
</tr>
<tr>
<td>How to view file history</td>
<td>Updated History dialog box screenshot to show new Label button</td>
</tr>
<tr>
<td>Setting Check In Files options</td>
<td>Updated User Options dialog box screenshot to show current option</td>
</tr>
</tbody>
</table>

Integrated Practice: Checking Files Out and In

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution</td>
<td>Updated Add Files, Check In Files, and Check Out Files dialog boxes screenshots to show progressive disclosure design</td>
</tr>
</tbody>
</table>

How to Share Files and Add Notifications

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to create email notifications</td>
<td>Updated My Email Notifications dialog box screenshot to display Shared, Pre-Condition, and On Event columns</td>
</tr>
<tr>
<td>How to edit an email template</td>
<td>Updated My Email Notifications dialog box screenshot to display Shared, Pre-Condition, and On Event columns</td>
</tr>
</tbody>
</table>
### How to Create Branches

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to create a baseline branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
<tr>
<td>How to create a snapshot branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
<tr>
<td>How to create a workspace branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
<tr>
<td>Practice: Creating Branches</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
</tbody>
</table>

### How to Promote and Rebase Changes

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to promote changes</td>
<td>Updated Promote Branch dialog box screenshot and updated text since OK button has been removed. Text now guides admins/users through previewing the promote.</td>
</tr>
<tr>
<td>How to rebase changes</td>
<td>Updated Rebase Branch dialog box screenshot to show Rebase Branch Based On options list and updated text since OK button has been removed. Text now guides admins/users through previewing the rebase.</td>
</tr>
</tbody>
</table>

### How to Manage Branches

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a mainline branch</td>
<td>Topic added for Admins</td>
</tr>
<tr>
<td>Adding a branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
<tr>
<td>How to freeze a branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
<tr>
<td>How to hide a branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
<tr>
<td>How to inactivate a branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
<tr>
<td>How to destroy a branch</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas.</td>
</tr>
</tbody>
</table>

### Integrated Practice: Branching, Promoting, and Rebasimg

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution</td>
<td>Updated Create Branch dialog box screenshot to show Create branch based on and Custom fields options areas. Updated Check Out Files dialog box screenshot to show progressive disclosure design. Updated Promote Branch dialog box screenshot to show new label field.</td>
</tr>
</tbody>
</table>

### How to Add Custom Fields

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to decide when to create a custom field</td>
<td>Updated Properties dialog box screenshot since Comments field has been removed</td>
</tr>
</tbody>
</table>

### How to Configure the Workflow

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assigning a workflow to a repository</td>
<td>Updated Properties dialog box screenshot since Comments field has been removed</td>
</tr>
</tbody>
</table>

### How to Create Triggers

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to add a trigger</td>
<td>Updated Triggers dialog box screenshot to display When, Preconditions, On Event, and Action columns</td>
</tr>
<tr>
<td>How to define a trigger</td>
<td>Note added that refers admins/users to advanced trigger examples available at Seapine Labs</td>
</tr>
</tbody>
</table>

### How to Create Reports

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to create a history report (Try It)</td>
<td>Updated text and screenshot for redesigned Filter By Action dialog box</td>
</tr>
<tr>
<td>Practice: Creating Filters and Reports</td>
<td>Updated text and screenshot for redesigned Filter By Action dialog box</td>
</tr>
</tbody>
</table>
### How to Add Users and Security Groups

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to add global users to the Seapine License Server</td>
<td>Updated license server screenshots to show new icons</td>
</tr>
</tbody>
</table>

### How to Set Server Options

<table>
<thead>
<tr>
<th>Section</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>Updated all Server Options dialog Box screenshots to show Server Database, RDBMS Connections, and Caching categories</td>
</tr>
<tr>
<td>How to set server database options</td>
<td>Section added</td>
</tr>
<tr>
<td>How to set RDBMS connections</td>
<td>Section added</td>
</tr>
<tr>
<td>How to set caching options</td>
<td>Section added</td>
</tr>
</tbody>
</table>
Adding Java applications

Before you record scripts, you must add the application to the repository so QA Wizard Pro can connect to it. After an application is added, it can be accessed from any workspace.

1. Choose File > New > New Application
   The Add New Application dialog box opens.

2. Select Java as the Type of application you are adding.
3. Enter the Java application file name or click Browse to locate the file.
4. Enter the Working directory path to indicate the path that contains the application. This field is automatically populated if you browsed to locate the file.
5. Enter the Java launcher to indicate the executable that will run the application. This option is only enabled for jar files. The field is automatically populated if you browsed to locate the jar file.
6. Enter any Command line arguments.
   The command line arguments allow you to control how the application operates when a script starts it. For example, some applications allow you to pass a file name as an argument when the application starts.
7. Enter the application Name. The application name must be unique within the repository.
8. Enter the application **Version**.
   We recommend entering the version of the application you are testing. Each application in the repository contains one or more versions. Each version contains information for starting the application, and the data captured during recording.

   **Note:** If you are testing a new version, add a version to the repository instead of an application. See *Adding versions*, page 1.

9. Select the **Repository** to create the application in. See *About application repositories*, page 1.

10. Enter and confirm a **Password** to password protect the application.

    Use a password if you want to control access to application data. You may not want all users to have access to add and modify application data because changes may cause scripts to fail.

11. **Click OK.**

    The application is added to the repository.

    **Tip:** You can also add applications when you record a script. See *Recording new scripts*, page 1.
Adding Java applications

Before you record scripts, you must add the application to the repository so QA Wizard Pro can connect to it. After an application is added, it can be accessed from any workspace.

1. **Choose file > New > New Application.**
   The Add New Application dialog box opens.
2. **Select Java** as the Type of application you are adding.
3. **Enter the Java application file name** or click **Browse** to locate the file.
4. **Enter the Working directory path** to indicate the path that contains the application. This field is automatically populated if you browsed to locate the file.
5. **Enter the Java launcher** to indicate the executable that will run the application. This option is only enabled for .jar files.
   The field is automatically populated if you browsed to locate the .jar file.
6. **Enter any Command line arguments.**
   The command line arguments allow you to control how the application operates when a script starts it. For example, some applications allow you to pass a file name as an argument when the application starts.
7. **Enter the application Name.** The application name must be unique within the repository.
8. **Enter the application Version.**
   We recommend entering the version of the application you are testing. Each application in the repository contains one or more versions. Each version contains information for starting the application, and the data captured during recording.
   **Note:** If you are testing a new version, add a version to the repository instead of an application. See Adding versions.
9. **Select the Repository** to create the application in. See About application repositories.
10. **Enter and confirm a Password** to password protect the application.
    Use a password you want to control access to application data. You may not want all users to have access to add and modify application data because changes may cause scripts to fail.
11. **Click OK.**
    The application is added to the repository.

**Tip:** You can also add applications when you record a script. See Recording new scripts.
APPENDIX D | SINGLE-SOURCED TOPIC IN MADCAP FLARE

A Adding applications

Before you record scripts, you must add the application to the repository so that MadCap Flare can connect to it. You will generally perform this task when you are preparing to test a new application. After an application is added, it can be accessed from any workspace.


The Add New Application dialog box opens. The fields vary depending on the type of application you are adding.

Note: Required fields are marked with an exclamation point icon.

2. Select the Type of application you are adding.

   - If you are adding a Windows application, see Adding Windows applications, page 1.
   - If you are adding a Web site, see Adding Web applications, page 1.
   - If you are adding a Java application, see Adding Java applications, page 1.

3. If you are adding a Windows application, enter the following information:

   - Enter the Executable path or click Browse to locate the executable.
   - Enter the Working directory path to indicate the path that contains the application. This field is automatically populated if you browse to locate the executable.
   - Enter any Command line arguments. The command line arguments allow you to control how the application operates when a script starts it. For example, some applications allow you to pass a file name as an argument when the application starts.

4. If you are adding a Web application, enter the following information:

   - Enter the URL.
   - Select Create repository variable to create a repository variable that contains the address for the server hosting the Web site. You may want to select this option if you are testing against multiple web servers.
   - The \$URL variable in the script is automatically populated using the server portion of the URL you entered (e.g., www.mywebsite.com). This is the default variable value used when running and recording scripts against the application version.

5. If you are adding a Java application, enter the following information:

   - Enter the Java application file name or click Browse to locate the file.
   - Enter the Working directory path to indicate the path that contains the application. This field is automatically populated if you browse to locate the file.
   - Enter the Java launcher to indicate the executable that will run the application. This option is only enabled for .jar files. The field is automatically populated if you browse to locate the .jar file.
   - Enter any Command line arguments. The command line arguments allow you to control how the application operates when a script starts it. For example, some applications allow you to pass a file name as an argument when the application starts.

6. Enter the application Name. The application name must be unique within the repository.

7. Enter the application Version.

   We recommend entering the version of the application you are testing (e.g., 7.2). Each application in the repository contains one or more versions, and each version contains information for starting the application and the data captured during recording.

Note: If you are testing a new version, add a version to the repository instead of an application. See Adding versions, page 1.

8. Select the Repository to create the application in. See About application repositories, page 1.

9. Select Local as the Repository. You can rename controls and change the search criteria for the controls stored in the local repository. After the application and controls are configured the way you want them, you can optionally promote them to a global repository to share them with other users.

10. Enter a password to protect the application.

   Use a password if you want to control access to application data. You may not want all users to have access to add and modify application data because changes may cause scripts to fail.

   You will spend some time renaming and configuring controls in the repository. After the repository is configured the way you want it, a password ensures the properties of the controls are not changed.

11. Click OK.

   The application is added to the repository.

   Tip: You can also add applications when you record a script. See Recording new scripts, page 1.
Adding applications

Before you record scripts, you must add the application to the repository so QA Wizard Pro can connect to it. You will generally perform this task when you are preparing to test a new application. After an application is added, it can be accessed from any workspace.

   The Add New Application dialog box opens. The fields vary depending on the type of application you are adding.
2. Select the Type of application you are adding:
   - If you are adding a Windows application, see Adding Windows applications, page 1.
   - If you are adding a Web site, see Adding Web applications, page 1.
   - If you are adding a Java application, see Adding Java applications, page 1.
3. Enter the application Name. The application name must be unique within the repository.
4. Enter the application Version.
   We recommend entering the version of the application you are testing (e.g., 7.2). Each application in the repository contains one or more versions, and each version contains information for starting the application and the data captured during recording.
   Note: If you are testing a new version, add a version to the repository instead of an application. See Adding versions, page 1.
5. Select the Repository to create the application in. See About application repositories, page 1.
6. Enter a Password to password protect the application.
   Use a password if you want to control access to application data. You may not want all users to have access to add and modify application data because changes may cause scripts to fail.
7. Click OK.
   The application is added to the repository.

Tip: You can also add applications when you record a script. See Recording new scripts, page 1.
Adding applications

Before you record scripts, you must add the application to the repository so QA Wizard Pro can connect to it. You will generally perform this task when you are preparing to test a new application. After an application is added, it can be accessed from any workspace.

1. **Choose File > New > New Application.**
   
   The Add New Application dialog box opens. The fields vary depending on the type of application you are adding.

   ![Add New Application dialog box](image)

   **Note:** Required fields are marked with an exclamation point icon.

2. Select the **Type** of application you are adding.
3. If you are adding a Windows application, enter the following information:
   - Enter the **Executable path** or click **Browse** to locate the executable.
   - Enter the **Working directory** path to indicate the path that contains the application. This field is automatically populated if you browsed to locate the executable.
   - Enter any **Command line arguments**. The command line arguments allow you to control how the application operates when a script starts it. For example, some applications allow you to pass a file name as an argument when the application starts.
4. If you are adding a Web application, enter the following information:
   - Enter the **URL**
   - Select **Create repository variable** to create a repository variable that contains the address for the server hosting the Web site. You may want to select this option if you are testing against multiple web servers.

1
Adding applications

- The %URLROOT% field is automatically completed using the server portion of the URL you entered (e.g., www.wysiscorp.com). This is the default variable value used when running and recording scripts against the application version.

5. If you are adding a Java application, enter the following information:
   - Enter the Java application file name or click Browse to locate the file.
   - Enter the Working directory path to indicate the path that contains the application. This field is automatically populated if you browsed to locate the file.
   - Enter the Java launcher to indicate the executable that will run the application. This option is only enabled for jar files. The field is automatically populated if you browsed to locate the jar file.
   - Enter any Command line arguments. The command line arguments allow you to control how the application operates when a script starts it. For example, some applications allow you to pass a file name as an argument when the application starts.

6. Enter the application Name. The application name must be unique within the repository.

7. Enter the application Version.
   - We recommend entering the version of the application you are testing (e.g., 7.2). Each application in the repository contains one or more versions, and each version contains information for starting the application and the data captured during recording.

8. Select Local as the Repository. You can rename controls and change the search criteria for the controls stored in the local repository. After the application and controls are configured the way you want them, you can optionally promote them to a global repository to share them with other users.

9. Enter and confirm a Password to password protect the application.
   - You will spend some time renaming and configuring controls in the repository. After the repository is configured the way you want it, a password ensures the properties of the controls are not changed.

10. Click OK.
    - The application is added to the repository.
Differences between outputs

The single-sourced ‘Adding applications’ topic produces different results in the QA Wizard Pro 2009 user guide and training guide PDF outputs. The following list includes the differences between the outputs generated from the same topic.

- A screenshot of the dialog is only included in the training guide example to provide a point of reference for users proceeding step-by-step through QA Wizard Pro training.

- Application type information (step 2) lists cross-references to other topics in the user guide example. This structure is used to map context-sensitive help, which is displayed when the user presses the F1 key when the dialog is open, to a general topic. Users can click the cross-reference to the topic that applies to the application type they are adding and bypass unrelated information. The training guide example provides details for all application types in subsequent steps (steps 3 - 5) to provide a general overview of all the options available as they learn about supported application types.

- Tips about adding versions instead of new applications and adding applications while recording script steps, with corresponding cross-references, are only included in the user guide example to provide users with potential alternatives that might better suit the task they are trying to accomplish or to introduce an easier approach for users who are more familiar with QA Wizard Pro.

- The information about selecting a repository is general with a cross-reference to more detailed information in the user guide example. This structure is used to accommodate the different situations that could apply to all users. This information is kept specifically related to the local repository in the training guide example because that is the only application repository used in the training and additional repository information is beyond the training’s scope.
FAQs in Adobe PDF

FAQs

Getting started

Where do I begin?
Before creating scripts in QA Wizard Pro, you might want to take some time to familiarize yourself with the program's user interface and basic features. See QA Wizard Pro interface, page 17.

What applications can I test with QA Wizard Pro?
You can test Web, Windows, and Java applications with QA Wizard Pro.

What is a workspace?
A workspace is where you organize scripts and related datasheets. You must create at least one workspace before you can record scripts. See Creating workspaces, page 8.

What are the differences between Grid View and Text View?
In Grid View, scripts are displayed in a graphical, easy-to-use table format, and each step is displayed in a row. In Text View, scripts are displayed in a text-based editor, and each statement is displayed on a line. Text View provides direct access to the scripting language, which makes it easy to create and edit statements, modify scripts, and add advanced logic to scripts.

Can I use QA Wizard 4 scripts with QA Wizard Pro?
QA Wizard Pro does not support QA Wizard 4 scripts. You can run existing scripts in QA Wizard 4, while creating new scripts in QA Wizard Pro.

Are there any sample QA Wizard Pro scripts?
A sample workspace that includes both Web and Windows scripts is installed with QA Wizard Pro in the C: \Documents and Settings\<username>\My Documents\QA Wizard Pro Workspaces\Sample Workspace directory. To open it, choose File > Open Workspace. Browse to the directory and open the Sample Scripts.qawworkspace file.

Application repository

What is an application repository?
Tested applications, versions, and windows and controls are stored in the application repository. When you record a script, QA Wizard Pro captures information about the application's windows and controls and adds it to the corresponding repository. During script playback, QA Wizard Pro compares the window and control properties in the repository to the tested application to locate objects. See About application repositories, page 37.

How do I add an application for testing?
Choose File > New > New Application. The Add New Application dialog box opens. Select the application type and enter the requested information. See Adding applications, page 27.

What is the difference between the local application repository and the global application repository?
The local application repository is specific to one computer and is not shared. The local repository is generally
used when one tester is recording and running scripts against an application version or when setting up an application before it is promoted to a global repository.

The global application repository is hosted on a server, can be shared among multiple users and computers, and is generally used when multiple testers are running scripts against the same application version. You can promote an application to a global repository from a local repository to share applications with other users.

What is an application version?
Each application in the application repository contains one or more versions. Each version contains information for starting the application as well as window and control data that is captured during recording. Maintaining versions allows you to continue testing previous versions of an application while creating new scripts to perform tests on new versions. See Adding versions, page 34.

What do I do when I need to test a new version of my application?
Create a new version in the application repository when significant changes have been made to the tested application. Maintaining versions in the application repository allows you to continue testing previous versions of an application while creating scripts for new versions. See Adding versions, page 34.

Script recording
What happens during recording?
QA Wizard Pro captures the actions you perform as you interact with the application. When you stop recording, you can review any new or changed windows and controls that were found and decide to add them to the application repository. See Recording new scripts, page 46.

What are window and control properties used for?
Each window and control in the tested application has several properties that are used to uniquely identify the object in QA Wizard Pro. Different properties are captured for different types of windows and controls. See Managing Windows and Controls, page 59.

Why does a script include low-level actions?
In some instances, QA Wizard Pro records low-level mouse actions when it cannot recognize an object as a supported control. Low-level actions use X and Y coordinates relative to the control to interact with an application.

Script playback
What happens during playback?
QA Wizard Pro runs the script and compares the test application to the information stored in the application repository. If differences are found, the script generally fails. See Running Scripts, page 51.

Can I monitor scripts during playback?
Yes. Use the QA Wizard Pro Status Tool to monitor scripts during playback. See Using the Status Tool, page 167.

Where do I see the run results?
You can view the results of a test in the run report. Run reports include details on what happened when the script ran.
Script modification

What if I want to add more steps or data to a script?
You can modify scripts to meet your testing needs. Adding statements, creating checkpoints, and using data sources for input data are all ways to add flexibility to a basic recorded script. See Modifying Scripts, page 73.

You can also add actions by recording steps into an existing script, which ensures that actions are performed on the correct windows and controls. See Recording additional script steps, page 48.

Script troubleshooting

What if a script fails? How do I tell if it is a problem with my script or if the application changed?
To determine whether it was your script or a change in the application that caused the failure, run the script in debug mode, which allows you to follow the script actions step-by-step to locate the problem and make corrections. See Debugging Scripts, page 111.
FAQs in Microsoft HTML Help

Note: This screenshot shows the drop-down text used in the HTML Help version of the QA Wizard Pro 2009 FAQs. Only three questions are expanded to give an impression of what the topic looks like the first time it is accessed.