Acceptance Testing Case Study

Version 6.0

Table of Contents

I. Company/Software Early History……3

II. The Software……...3

III. Business Model……..3

IV. The Task……5

V. General Testing Strategy…….....7

VI. Original Contract Negotiations……..7

VII. Requiem Acceptance Testing…….10

VIII. Revised Contract……...11

IX. Acceptance Testing Requirements……..12

X. Selecting a Partner for the Test Plan and Pre-Acceptance Testing…….……..14

XI. Test Plan Document……..……...15

XII. Pre-Final Run Acceptance Testing Workflow…….……..16

XIII. Selecting a Partner for Final Acceptance Testing……18

XIV. Acceptance Testing Protocol……18

XV. Acceptance Testing and Results……19

I. Company/Software Early History

Greene Software Systems (GSS) was founded in 1997 as a privately owned software company. Between 1997 and 2000, GSS became involved in several software development areas, one of which was a security software package called SecNet.

II. The Software

SecNet's client front-end was developed in C and C++. The back-end server software used for billing and authentication of SecNet was developed using a combination of Java, Perl, and C.

SecNet was the most important product made by GSS. The SecNet package included anti-spyware software with a spyware scanner, real-time spyware protection, and spyware cleanup. It also contained antivirus software, the most important component of the software package, and firewall, parental control, and popup blocker software, as well as a campaign manager (see: campaign manager). The antivirus software, anti-spyware software, and parental control software were licensed from other software suppliers. A major new version of SecNet was released two to three times each year; a new version of the software was constantly in development. The version of the software on the market in early 2004 was SecNet 4.2.

III. Business Model

The business model for SecNet was highly profitable for both GSS and the ISPs to which it licensed its security software. Before 2004, GSS licensed SecNet to several significant Canadian ISPs. SecNet was branded separately for each ISP licensing it (see:

branding). When SecNet was deployed for a new ISP or a major change occurred at a current ISP customer, a development team from GSS worked with the ISP to take the current release of SecNet and adapt it to the ISP's requirements.

These ISPs then distributed SecNet to end users. This was done in several ways: as part of a promotion of the ISP's internet services, as part of the base cost of the ISP's internet services, or as part of a monthly subscription service on top of the end user's basic internet service fees owed to the ISP. The ISPs licensing SecNet were often receiving an additional $5 CDN each month from each end user, a significant increase in revenue for the ISPs.

The typical profile of an end user of SecNet was that of a non-technical personal computer user. These users downloaded SecNet transparently from their respective ISPs once they signed up for the security services; the software was then installed and registered with the user's ISP. GSS received updates from the suppliers of its software and would only deploy them to end users' PCs after validation and testing procedures were performed. The users received software updates and virus definition updates from GSS directly when such updates became available; this was part of the Service Level Agreement (SLA) GSS had with each ISP (see: virus definitions, SLA).

The end users believed that they were dealing only with the ISP. They would call their respective ISPs for technical support. SecNet was only one of several services the ISP's helpdesk supported. GSS offered training to each ISP's support technicians, but had its own helpdesk for issues that the ISP's support staff was not qualified to manage. Such issues were forwarded to SecNet's support department as they arose.

IV. The Task

Robert Sabourin was hired on a consultant contract at GSS, and was given a broad mandate to improve internal processes at GSS and to assist in preparing SecNet 5.0, the version of SecNet that was to replace SecNet 4.2, for release. His project with GSS started on January 1st, 2004, although he was in communication with GSS about the project as early as September 2003. All of his consulting time for 2004 and a short amount of time in 2005 were purchased by GSS, and he was given the position of Director of Software Engineering Services. This involved working on software quality assurance, software testing, software requirements, software workflow, and configuration management (see: software workflow, configuration management). His main task was to get SecNet 5.0 released to a large American ISP.

The Canadian ISPs that GSS had as its customers as of January 2004 were telecom customers that had limited experience in reselling software. By a combination of good fortune and a good business model, the Canadian ISPs were still able to make a profit on the software, but their SLA agreements with GSS were naïve. Because of the revenues involved and because the Canadian ISPs didn't know better, problems that arose with SecNet 4.2 and below were not considered major issues. These problems included a release being deployed on an end user's computer and causing serious harm to the user's PC environment. The end users who experienced damage to their PCs either had another computer purchased for them by GSS or had their disks reformatted depending on the severity of the problem encountered, but this way of dealing with severe issues was only made fiscally possible by large profits and the relatively small size of the Canadian ISPs.

The American ISP that GSS was aggressively pursuing a contract with, Lepton, was much larger than the Canadian ISPs that GSS currently had as its customers. Lepton was not as trusting as its Canadian counterparts, and its agreement with GSS gave GSS many more obligations.

Lepton stated that SecNet had to work with Lepton's own software. If support calls got too high, the SLA agreement between GSS and Lepton included penalties to be exacted upon GSS. Any issues with the software would have to be resolved within timeframes proscribed in the SLA depending on their severity level, as set out in a severity scheme based on the amount of damage caused to users' PCs and the number of users affected. If the problem was not resolved in the prescribed timeframe, penalties would result. Updates had to be deployed to many more users than GSS was used to, so the availability of updates had to be increased. Finally, SecNet had to pass acceptance testing, as specified in the contract with Lepton (see: acceptance testing).

There were a considerable number of compatibility bugs in SecNet 4.2 that needed to be resolved in SecNet 5.0 before the software would pass acceptance testing. In SecNet 4.2, patches had to be pushed frequently to resolve field-reported incompatibilities with other products installed on end users' computers. SecNet 4.2 interacted with a variety of low-level third-party drivers (see: third-party drivers). As many of these conflicts as possible needed to be identified in testing and dealt with before the software was deployed to the hundreds of thousands of end users subscribing to Lepton.

Of the three quantities that must be balanced on a software project – time, quality, and money – time and quality were the critical factors. Quality was a great concern to

GSS because GSS did not want to be fined by Lepton for support calls, and Lepton wanted to deploy SecNet 5.0 by April or May 2004, so there was only a short time frame in which to complete the testing project. The contract with Lepton was worth several million dollars to GSS, so the budget was generous enough to allow the time and quality goals to be met.

V. General Testing Strategy

The main goal of the project for SecNet 5.0 was to anticipate calls to Lepton's technical support and prevent penalties for breaching Lepton's SLA. GSS wished to find as many bugs that would trigger support calls as possible. A many-pronged testing approach was used to ferret out different types of bugs in the software and to confirm that SecNet would run normally on the average end user's computer. The testing strategies implemented included unit testing, system testing, acceptance testing, beta testing, and combinations testing (see: unit testing, system testing, beta testing).

These tests were being created and conducted simultaneously. The unit testing and system testing were done in-house; the acceptance testing and combinations testing were outsourced. This case study is primarily concerned with acceptance testing.

VI. Original Contract Negotiations

The first contract GSS signed with Lepton covered two separate issues: the licensing of the base product, SecNet 5.0, and the customization of SecNet 5.0 that Lepton required before the product could be deployed to its users. This customization included branding, the integration of the program with the billing software used by

Lepton, and making SecNet 5.0 compatible with other Lepton software end users' computers.

The contract between GSS and Lepton for the use of SecNet 5.0 granted Lepton a nonexclusive worldwide license to deploy SecNet 5.0 on its end users computers and to profit from it. GSS was also obligated to provide Lepton with any updates or patches to SecNet 5.0. Lepton was given the right to archive a reasonable number of copies of SecNet 5.0 to use for recovery from any disaster that might occur. Lepton also agreed to respect GSS' trade secrets and to not knowingly permit any of Lepton's employees to reverse engineer SecNet or to otherwise acquire any information about SecNet's source code. The contract also included the Service Level Agreement.

The sections of the contract related to the customization agreement stated what GSS' and Lepton's respective obligations were. GSS was obligated to customize API interfaces and other parts of the program to brand the product specifically for Lepton, and granted Lepton exclusive rights over its particular customization source code. Lepton also gave GSS the rights to use Lepton's trademarks for the purposes of the customization laid out in the contract.

The initial contract signed at Lepton was negotiated unusually. Normally before the delivery of a software product, acceptance testing is run on the product by either the party receiving the software or an independent third party agreed to by the customer and the supplier in order to ensure that there are no serious flaws in the program before its release. Once acceptance testing is passed, the product supplier is paid for the product, and the product is deployed. If a serious problem is discovered during acceptance testing that precludes the product from being accepted by the customer, the supplier does not get

paid, so acceptance testing creates a significant incentive on the part of the software supplier to ensure that the product conforms to mutually agreed-upon requirements. For this reason, acceptance testing has become a standard contractual issue in the software industry.

On Lepton's side, the contract was negotiated by employees whose jobs were to find new revenue streams. Lepton was engaged in a very serious search for new ways to profit from its ISP business, so the employees negotiating the contract may have felt pressured to get the contract signed quickly. While the team of negotiators at Lepton understood the need for a strict SLA, they were not knowledgeable enough about software business practices to realize the need for acceptance testing.

The GSS negotiators were inexperienced at dealing with such a high-profile client. Because of the inexperience and haste on the part of the Lepton and GSS negotiators, the contract was signed by both parties without acceptance testing being specified as a condition for product acceptance by Lepton. This meant that GSS could have submitted a product to Lepton that did not include all the functionality or polish expected by Lepton while still fulfilling all of its contractual agreements so long as any remaining bugs were fixed in accordance with the SLA.

After the contract was signed, the Lepton stakeholders changed as the project was shifted into the control of another group. This group was more knowledgeable about the terms and conditions of software contracts than the previous stakeholders had been, and they were surprised to see that no acceptance testing had been placed into the contract. These stakeholders at Lepton realized that even if GSS complied with the terms of the SLA and repaired any bugs found in the field after deployment in the timeframe

specified, having large numbers of bugs when the product was deployed that would otherwise have been caught in acceptance testing could damage Lepton's reputation.

However, it is important to note that GSS was not attempting to take advantage of Lepton by leaving acceptance testing out of the contract; GSS did not realize that it was necessary. Nevertheless, the stakeholders at Lepton were unhappy with the contract, and one decided to take action to force a renegotiation of the contract's terms.

VII. Requiem Acceptance Testing

A stakeholder in the new group in control of the project at Lepton found a testing lab called Requiem in January 2004 and hired Requiem to commence with acceptance testing of SecNet.

Requiem did not have access to SecNet 5.0 because the testing was being done without the permission or knowledge of GSS, so Requiem was instructed to perform acceptance testing on SecNet 4.2, the version of SecNet that was deployed to the Canadian ISP customers of GSS at the time, by signing up for an account with one of the Canadian ISPs and obtaining the software. Requiem also did not have a copy of the requirements and had no way of obtaining them without GSS's knowledge. Since the stakeholder at Lepton who had commissioned Requiem to test SecNet wanted the testing done secretly, Requiem instead invented the requirements of SecNet, some of which had nothing to do with SecNet's intended functionality, and had stakeholders at Lepton sign off on them, despite their inability to do so with any technical authority.

When Requiem performed the acceptance testing, it was done unprofessionally. This secretive testing was accomplished using invented requirements with an old version

of the software that wasn't even properly deployed, and for malicious reasons. The stakeholder at Lepton who was responsible for the testing wanted to make SecNet look like a poor piece of software because he was annoyed at the contract and the people at Lepton who had signed it.

VIII. Revised Contract

After the completion of the unauthorized acceptance testing of SecNet 4.2 conducted by Requiem, the list of bugs contrived by Requiem was presented to stakeholders at Lepton, and then to GSS. This was done at the behest of the Lepton stakeholder involved for the purpose of frightening Lepton out of its business arrangement with GSS by making SecNet look like a risky, poor-quality product.

This caused a significant amount of strife between Lepton and GSS, both on Lepton's side because of the perceived instability of SecNet and on GSS's side because of the unfairness and lack of ethics in the testing. GSS analyzed Requiem's results and found that the same bug was often listed multiple times on the bug list since Requiem did not bother to identify multiple errors as the same bug. This made it appear as though SecNet 4.2 had more bugs than it had in actuality. Furthermore, Requiem lacked the software from the Canadian ISPs with which SecNet 4.2 was made to work; this contributed to a number of errors, along with the inaccurate requirements (see: GSS Analysis of Requiem Requirements).

The business proposal was jeopardized, but both sides agreed to a renegotiation of some of the contract's terms and conditions in order to satisfy Lepton. If GSS had not agreed to the renegotiation, Lepton would have sued GSS for being in breach of contract,

and GSS would not have gotten the business. The new contract made Lepton pay less money for SecNet, but gave Lepton a longer-term commitment to the product. Acceptance testing was also included in the contract at this time.

IX. Creating a Requirements Document

Before the revised contract with Lepton could be signed, a list of requirements that had to undergo acceptance testing needed to be signed off on by all parties.

When Mr. Sabourin began his contract with GSS, GSS did not have a full requirements document. GSS had all of SecNet 5.0 well-specified, but the specifications were a combination of requirements and design documents, not a complete requirements specification. One of Mr. Sabourin's tasks at GSS was to combine the information presented in several different types of documents into a requirements document.

Over the course of two days, Mr. Sabourin and two others specified the requirements of SecNet 5.0. The requirements document specified exactly what the acceptance testing needed to test. Platform requirements stated which operating systems SecNet 5.0 had to operate on (Windows 98 SE, ME, 2000, XP Home SP1, and XP Home SP2) and which co-existent applications it had to be compatible with. Installation requirements specified how the program was to be installed and that it had to notify users of the presence of any redundant security software, such as McAfee and Norton.

Activation requirements gave the conditions under which the product was to be activated and made available for use by an end user. Uninstallation requirements stated that the program must come equipped with an uninstallation utility, and what the

uninstallation utility was required to uninstall. Performance requirements gave performance benchmarks that SecNet had to meet.

Other requirements were created for specific parts of the application: the antivirus, firewall, anti-spyware, popup blocker, privacy manager, and parental control (see: 20040506 SecNet Requirements Draft).

One portion of SecNet that was not included in the acceptance testing requirements and was not tested during acceptance testing was the branding of the product to suit Lepton's requirements. Branding, though required, was not one of Lepton's primary concerns with the product. However, even though it was not technically part of acceptance testing, testing confirming the proper application of branding was performed to add to confidence in the correctness of the product.

This testing was not something to outsource to a lab of specialized testers since it required only generalized testing skills. Instead, a tester at GSS developed a long, detailed test script to visually confirm all aspects of the branding. Because branding was not of great importance to Lepton, this was all the confirmation GSS did for the branding. Other projects can require acceptance testing of branding, but this one did not.

After GSS and Lepton agreed to the requirements for acceptance testing, only minor changes to them were made. These changes were exclusively qualifications to the existing requirements which became necessary as clarity issues between the requirements and the program's operation became evident. For example, if a requirement stated that a dialog must appear after a particular user action and that action resulted in several dialogs appearing, the dialog being referred to in the requirement would have been referred to more specifically.

X. Selecting a Partner for Acceptance Testing

Before performing acceptance testing there were several tasks to be done. The first was to create a working draft of the test plan document, with every acceptance testing test case corresponding to one or more requirements to be verified. When this was completed, it was then necessary to run these test cases to confirm that SecNet would pass acceptance testing. The test case document, the requirements document, and the testing were to be iterated over a period of several weeks until Mr. Sabourin and GSS could be confident that SecNet would pass acceptance testing.

Mr. Sabourin did not go through a long, rigorous process to select a partner. He considered two companies. One was ThreeBears, the testing company that had handled the combinations testing of SecNet 5.0 very well, that was already very knowledgeable about the product, and that Mr. Sabourin trusted because he was in values sync with the ThreeBears test lead, Mr. John Bee (see: values sync).

The other was an offshore company which was confident that it could undercut the bid of any of its competitors due to its low labor cost. Because of the offshore company's overconfidence, it ended up placing a bid for the project that was, in fact, higher than ThreeBears' bid. This was because it charged per person per week, even though Mr. Sabourin did not require all the time the offshore company was offering, but it refused to charge any other way.

Because Mr. Sabourin knew ThreeBears had the skills, availability, and values he was looking for in a testing partner and because ThreeBears' testing and documentation

services were competitively priced, Mr. Sabourin chose ThreeBears as the partner for the acceptance testing project.

XI. Test Plan Document

To write and to iterate the test plan, Mr. Sabourin enlisted Mr. John Bee, the test lead from ThreeBears who Mr. Sabourin had worked with closely during combinations testing of SecNet 5.0 and who was the test lead of the acceptance testing of SecNet 5.0, and Yvan Boulanger, an employee of Maxtest who was working on the test plan as an independent consultant.

The main portion of the test plan was devoted to the test cases. Each test case corresponded to one or more requirements, which were referenced in each test case according to their identifiers in the requirements document. After the initial draft of the test plan was made, updates to the overview and the test cases occurred on at least a weekly, and sometimes a daily, basis. This included adding overview information, filling in missing test cases, and modifying test cases added in previous versions of the test plan. Whenever a test case could not be passed, and the reason for the test not passing was determined to be the wording of a test case, the test case document was updated to make the test case comply more accurately with the requirement or requirements it was meant to test.

To ensure that only one current version of the test plan existed at any given time, only one of the three people hired to work on the test plan could modify it at any given time. This was accomplished through a method of passing an imaginary baton between Mr. Sabourin, Mr. Bee, and Mr. Boulanger. Whenever one had the baton, he was the one

allowed to modify the document. When he was finished, he would pass the baton to someone else, and that person would then be the only one allowed to make any changes to the test plan. If anyone was unsure of who had the right to modify the document at any given time, he would enquire through email as to who had the baton. If anyone else felt that he had something to add to the document but did not have the baton, he would pass on his comments to the person with the baton or wait until it was his turn (see: Final Acceptance Test Plan).

XII. Pre-Final Run Acceptance Testing Workflow

Before the official acceptance testing agreed to by GSS and Lepton, it was important for GSS to ensure that SecNet 5.0 would pass the acceptance tests so that GSS' contractual obligations would be fulfilled. This meant that each acceptance test had to be run and passed.

Most of the communication between GSS and ThreeBears during pre-acceptance testing took place via email and pertained to the status of testing, what had and had not yet been tested, and bugs or potential bugs. Status information was relayed regarding which tests had been passed or failed and any ambiguities encountered during testing.

Since ThreeBears had done the combinations testing of SecNet 5.0 earlier in the testing cycle, it already had ghosted images of platforms that could be used to run the acceptance tests as well, so there were no overhead costs associated with setting up the necessary platforms (see: ghosting).

When a bug was found on a platform by ThreeBears, the test lead made sure the bug was well-reported. A well-reported bug included all the information the developers

would need to repeat the bug, such as system status information acquired through Dr. Watson. The testers used a template to report the results of their testing from the second week of testing onward (see: Sample Bug Report).

The primary line of communication was between Mr. Sabourin and the test lead, Mr. Bee. They had phone calls scheduled daily and contacted each other at any time to handle emergencies. Once a bug was confirmed and reported, it was added to an internal bug database at ThreeBears on a TestTrack Pro database, a convenient internet-based application that could be used anywhere in the ThreeBears lab. This database made bug information available online to GSS in real-time as the bugs were reported. If more information was required in triage or by the developers, the developers and testers involved with the bug at GSS spoke directly to the tester at ThreeBears who had discovered the bug.

Weekly status reports were made through a simple spreadsheet. These status reports included the number and percentage of tests passed, failed, blocked, inconclusive, and not run in any given week (see: blocked, inconclusive, Status Report Week 1, Status Report Week 2, Status Report Week 3, Status Report Week 4, Status Report Week 5). There were a number of reasons that any given test would not pass. The requirement on which the test was based could be flawed, in which case the requirement would need to be clarified and the test case rewritten; the test case could need clarification, in which case only the test case would need to be modified; or there could be a bug in SecNet 5.0 that needed to be fixed. There were also weekly deliverables sent to GSS from ThreeBears with all the bug reports made during the week.

The project manager at GSS needed to manage the relationship between GSS and Lepton. He was very involved with the progress of the test development for the final run of acceptance testing since it was on the critical path of the project and needed to be completed before the final exchange of money for the product between GSS and Lepton could be made (see: critical path).

XIII. Selecting a Partner for Final Acceptance Testing

Although ThreeBears had been chosen by GSS to perform pre-acceptance testing to confirm that SecNet would pass acceptance testing, ThreeBears was not guaranteed the contract for the final, official round of acceptance testing. The acceptance testing company had to be agreed upon by both GSS and Lepton, as stated in the contract. GSS suggested using ThreeBears, and Lepton agreed to have ThreeBears perform the acceptance testing so long as its laboratory facilities passed an inspection by Lepton employees. ThreeBears and GSS agreed to this condition.

Lepton evaluated ThreeBears' testing site during the third week of pre-acceptance testing. Lepton was primarily concerned with whether ThreeBears had the facilities to perform all of the necessary tests; ThreeBears representatives showed Lepton's stakeholders around the site and explained their capabilities. Lepton was satisfied that ThreeBears could do the acceptance testing, so ThreeBears obtained the contract for the final run.

XIV. Acceptance Testing Protocol

The acceptance testing protocol document was created for the stakeholders and quality assurance representatives from GSS and Lepton and was signed by both parties before the final run of acceptance testing. Its original version was agreed upon prior to ThreeBears being given the acceptance testing contract after Lepton's inspection; it was later modified to reflect ThreeBears' involvement.

The protocol defined that each acceptance test case would explicitly match one or more of the product's requirements, and that ThreeBears would perform acceptance testing at its testing laboratory. It also dictated conditions that ThreeBears' laboratory had to adhere to during acceptance testing, and the specifications of the triage meeting to take place between Lepton and GSS after acceptance testing. Its appendices included an issue log template the ThreeBears testers were required to use whenever an issue was found, the test case template that all tests had to conform to, and agreed-upon scales for severity and priority (see: Acceptance Testing Protocol). Any P1 bugs, those of highest priority, had to be resolved before the initial deployment of the product, so Lepton would not accept the product if any P1 bugs were found during acceptance testing.

XV. Acceptance Testing and Results

The final run of acceptance testing was completed over a period of two days in May 2004 at ThreeBears' testing facilities. Since all of the tests had been passed during the fifth week of pre-acceptance testing, it was expected that all the tests would be passed on the last round of acceptance testing as well. SecNet 5.0 did indeed pass each acceptance test without exception (see: Acceptance Testing Final Summary).

The final conference with Lepton pertaining to acceptance testing was held soon after. Lepton was very satisfied with the results of the acceptance testing and was obligated by the results to accept the product.