Combinations Testing Case Study

Version 11.0

Table of Contents

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 were not qualified to manage. Such issues were forwarded to SecNet's support department as they arose.

IV. Internal Situation at Greene Software Systems

To give further context to the case study, the internal situation at GSS must be discussed. Before the crash of the technology industry in 2000-2001, GSS employed 300 people on a variety of software projects. After the crash, most of the products being made by GSS were cancelled. Only a few were profitable enough to survive, such as SecNet. GSS's staff of 300 was reduced to a staff of 50.

Many experienced employees at GSS were let go, and the company suffered as a result. The employees who left included those who had designed the unit testing harness (see: unit testing). The unit testing harness was excellent, but no one remaining at the company after the staff reduction had a complete understanding of why it worked. The employees at GSS in 2004 knew how to operate the harness, but not how to diagnose or repair any problems. If a test caused errors, it was commented out because no one knew how to debug it. Had anything serious gone wrong with the harness, there would have been no one at the company who could have fixed the problem. This was a significant risk for GSS.

There were between 50 and 80 employees at GSS by the time the project to develop SecNet 5.0 began in January 2004. Most of the staff worked in one large office in Ottawa, with a smattering of salespeople working in the UK and various parts of the US. The division of roles and responsibilities between the staff was sometimes poorly defined and illogical. The only person who knew how to test the campaign manager was the QA lead, so whenever there was a need to test the campaign manager, the QA lead had to stop everything else he was doing to test the campaign manager. This disorganization impacted productivity.

There were three teams working on the SecNet software in 2004. One team was working on the development of the client software, another team was working on the back-end software, and a third team was working on the integration of the software with the ISP, including branding. Each team was composed of approximately 10 developers and testers, a few technicians, a couple of technical writers, and one project manager.

Employee turnover at GSS was high. The president of GSS was not concerned about the rate of employee turnover; the prevailing attitude was that if an employee wished to leave, he was welcome to. GSS was not interested in engaging a bidding war with other companies over valuable employees, so critical staff departed the company often enough to make turnover a significant project risk. For example, the build master left the company several days after the project with GSS began. A new build master needed to be hired very quickly to create new builds. GSS was fortunate enough to find a qualified person, but there was a steep learning curve for the new build master to get settled into the position, which decreased productivity for a significant amount of time.

In short, while GSS had managed to successfully release previous versions of SecNet, there was some dysfunction in the company from management to the employee level.

V. 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 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 Adelphia (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.

In January 2004, a stakeholder at Lepton who was against the business arrangements with GSS found a testing lab called Requiem and hired the lab to test

SecNet. Requiem did not have SecNet 5.0 because this testing was being done without the permission or knowledge of GSS. Instead, Requiem obtained the currently deployed version of SecNet, version 4.2, likely by signing up for an account with one of GSS's Canadian ISP customers. 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 involved 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 testing commenced, it was done unprofessionally, with the same bug often being 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.x had more bugs than there were in actuality. Furthermore, Requiem lacked the software from the Canadian ISPs with which SecNet 4.2.x was made to work with, which contributed to a number of errors.

After the completion of testing, the list of bugs contrived by Requiem was presented to stakeholders at Lepton for the purpose of frightening them out of their business arrangement with GSS. This event caused a great deal of bad feelings 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. The business proposal was jeopardized, but both sides still managed to move forward.

VI. 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 platform testing (see: unit testing, system testing, acceptance testing, beta testing, platform testing).

These tests were being 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 the combinations testing aspect of the project, which includes both beta testing and platform testing (see: beta testing, platform testing).

VII. Rationale for Combinations Testing

Combinations testing was required because unit testing was only discovering some of the platform-related bugs. Despite the quality of the systematic testing done by the unit testing harness, unit testing could only cover a few of the millions of possible platform combinations without choking the throughput of the build system.

Beta testing finds bugs that are likely to be encountered by users in real world. Platform testing finds bugs that may not occur often in a typical user environment, but that may be very harmful in a small number of cases. The two techniques are complementary.

VIII. Rationale for the Outsourcing of Beta Testing

Before Mr. Sabourin was on contract at GSS, the idea of performing beta testing was already entrenched in GSS's management. The goal of management was to find 400 testers to beta test the product. They attempted to accomplish this through several failed initiatives lead by the product manager at the time. He was experienced and competent at his job, but had no experience with beta testing.

One initiative was to get employees to beta test the product at home. Management did not consider the fact that these testers would not fit the profile of a typical end user of SecNet. They were concerned only with finding 400 testers in total. Only twenty employees volunteered as beta testers, which was nowhere near the number that management was hoping for.

Another mistaken initiative was to use Google advertising services to display an ad for beta testers when a Google user searched for antivirus software, security software, or one of several other key terms. The cost of the contract with Google was fair and reasonable, but failed to attract many beta customers.

These customers were interested in obtaining free antivirus software, not in participating in a beta test. If SecNet did not install properly, they usually did not contact GSS to resolve the problem. Instead, they did not install the software at all. If SecNet did not work to their satisfaction after installation, most would uninstall it. Approximately forty people signed up for the beta test in this way, but the feedback they gave was poor.

Beta testing was being approached from the perspective that the most important consideration was the number of people who beta tested the product. There was no regard

for the motivation or demography of the testers. When Mr. Sabourin came on to the project, he focused on finding testers who would give reliable feedback and who fit the demographic model of a typical SecNet user.

It was clear to Mr. Sabourin that the knowledge and experience in the field required to run the beta testing project was not present at GSS. The most timely and cost-effective way of executing quality beta testing was to outsource it.

IX. Rationale for the Outsourcing of Platform Testing

Platform testing could not be performed in-house with available resources. GSS did not have an in-house testing facility in which to do platform testing, and the time it would have taken to set up such a lab was prohibitive. The testers at GSS were occupied by the other testing projects taking place at the same time, and the timeframe for the project was so limited that it would not have been possible to wait until the in-house testers were available.

The lack of a strong attachment to the employees by the executive management at GSS had the positive effect of leaving management open to the possibility of outsourcing the platform testing, so long as a business case was made for it, the core development was kept in-house, and GSS was still be capable of changing testing partners if their partner were to go bankrupt.

There was resistance to the idea of outsourcing by employees at GSS. Some had the impression that outsourcing would deny jobs to local testers. However, outsourcing was necessary to get the testing task completed successfully and professionally.

Outsourcing was not being chosen to reduce cost by hiring cheaper labor, but to complete the testing project in the proper time frame and with the proper quality.

X. Selecting a Partner for Beta Testing

Mr. Sabourin outsourced beta testing to a testing company called Lore. At the time, Lore was the largest outsourcing company in the world, with tens of thousands of testers and a particularly well-renowned beta testing department. Mr. Sabourin had worked with other localization and testing departments at Lore on previous testing projects, and had heard about their supply of 30,000 beta testers stationed all around the world. He was very impressed by their beta testing operation. Lore's costs were fair; they charged less than $100 for each tester, and only charged for those who completed the testing and feedback. They had the ability to quickly create a base of beta testers to fit almost any demographic requirements. No other company had a comparable operation. Mr. Sabourin sent Lore a statement of work, and a contract was signed for Lore to manage the beta testing of SecNet (see: Beta Testing Statement of Work).

XI. Selecting Categories for Platform Testing

In the first few weeks of January, Mr. Sabourin found a great deal of customer support data from previous problems reported to the GSS support department. Whenever the helpdesk at GSS was contacted by a user with a problem pertaining to SecNet, the helpdesk representative had instructions to ask the user to click a button in the software that said "Diag". When the user clicked this button, a large amount of diagnostic information about the configuration of the user's computer was sent to the helpdesk in the

form of a text file (see: Sample Diag File). Little was done with this information at the helpdesk because the helpdesk representatives didn't know what to do with it; the Diag instructions were part of the system set up by employees who had left GSS long before. There were tens of thousands of collected and unused Diag files, with dozens to hundreds more being forwarded to Mr. Sabourin on a daily basis on his request during the project. These files were an invaluable resource for creating a model of the types of applications running on an end user's computer and determining which of these had the highest risk of conflicting with SecNet.

A program was written to analyze the correlation between applications running at the time each Diag file was sent and to store the results in a Microsoft Office Excel spreadsheet (see: Diag File Analysis). Applications that seemed to be running often when SecNet users experienced errors included multimedia applications, file-sharing applications, web-based applications, and games.

This data, combined with common sense, experience, and information provided by the product manager, gave Mr. Sabourin enough knowledge to model the categories of software and the selections of software in those categories that would be useful for platform testing. This was used to define the Statement of Work (SOW) given in the Request for Proposals (RFP) for the outsourced platform testing project (see: Platform Testing Statement of Work, statement of work, request for proposals).

XII. Selecting a Partner for Platform Testing

There were several criteria that Mr. Sabourin used to define whether a company was well-suited to be a partner for the platform testing project: track record on prior

platform testing projects, ability to vary the parameters specified in the SOW, availability, flexibility, understanding of the project, and values sync (see: values sync).

The partner had to conform to the testing project's budget. Executive management at GSS budgeted $50,000 USD for platform testing, excluding the costs of software for running the tests. This budget needed to cover three passes of testing: one to find bugs in the first pass, another to find any bugs uncovered by fixing bugs found in the first pass, and another pass to fix those bugs. While Mr. Sabourin wished to finish testing in two passes, he needed to ensure that he had the budget for three in the event that a third pass was required. This $50,000 USD figure was not revealed to the companies bidding on the project since it would have influenced their bids. Making a bid below the hourly rate of other prospective partners was an advantage in a bidder's favor, but it was not the most important aspect of the selection process.

The way in which a prospective partner proposed to vary the configurations was important in the selection process. Since many different platforms were to be tested, the varying of configuration parameters needed to be done by reconfiguring computers to operate many different platforms. This reconfiguration could be done either by using virtual machines or by ghosting (see: virtual machine, ghosting). For this project, ghosting was preferred by Mr. Sabourin.

Mr. Sabourin initially identified between five and ten companies that he had heard of from trade shows, software engineering journals, and his own and others' past experiences that he thought might be acceptable platform testing partners. Each of these companies was sent the exact same Request for Proposals containing the same Statement of Work.

It was important to Mr. Sabourin as part of ensuring customer-centeredness for a prospective partner to look at his testing project individually even though most of the prospective partners were responding to many RFPs at the same time that they were responding to his. Form letters did not impress him as much as custom replies. One particular problem he encountered which showcased a refusal to be flexible or customer-centered was a refusal to sign GSS's own non-disclosure agreement, or NDA (see: NDA). This problem generally arose only when Mr. Sabourin was dealing with junior salespeople who didn't understand the rules of doing business. Such salespeople would only agree to the signing of their own company's NDA. Since GSS's intellectual property was being dealt with, and not the partner company's, this was unacceptable.

After an initial round of research to determine which of the initial group of companies had a strong positive history with platform testing, Mr. Sabourin narrowed down the list of potential partners to three companies. These companies were Lore, Perceptor, and ThreeBears. Having at least three companies involved with bidding was important for two reasons. If only one company had been bidding and something had gone badly, the bidding process would have had to start again from square one. Problems that could have arisen included the partner going bankrupt, the partner starting another testing contract that used up too much of its resources for it to take on GSS's testing, or a change in GSS's platform testing schedule to a timeframe when the partner was unavailable. It was also important to keep the bidding process competitive by making each potential partner aware that other companies were bidding on the testing project. No company was told it would not be GSS's partner until GSS signed a contract with another partner.

Lore wanted to do the testing in Ireland because Lore's hourly costs in Ireland were so low that they thought they could make the lowest bid by working from there. Lore's initial bid was 20-30% less than any competing bids, the amount of work they were proposing to do was consistent with what Mr. Sabourin expected the project to require, and Lore was available for the timeframe during which Mr. Sabourin wanted to do the platform testing.

However, the type of testing that Lore wanted to do was not what Mr. Sabourin wanted. The list of platforms Lore proposed was adequate, but Lore did not give any indication of how they had come up with the platform list, which did not make Mr. Sabourin confident in the comprehensiveness of the platforms being tested. Furthermore, Lore refused to test with cable modem connectivity on any platforms since their Ireland facilities did not have the ability to test using a cable modem. Lore claimed that they could test using something similar to a cable modem, but since Lepton was a cable ISP, this was unsatisfactory.

When Mr. Sabourin made it clear to Lore that they would not get the business if they could not test configurations with cable modem access, Lore determined that they could do half of the work in Ireland and half in the United States. The logistical problems Lore had with this scenario caused Lore's bid to raise 30% above the competition.

Mr. Sabourin was also not pleased with the entirely script-based testing that Lore was proposing. He thought that much of the testing would be done better with exploratory tests, but Lore was insistent upon doing all of the testing using scripts.

Lore did not demonstrate flexibility in the bidding process. Based upon Lore's inability to provide a rationale for its selected platforms and its firmness on scripted tests,

Lore also didn't appear to understand the testing project very well. The problems Lore had with performing cable modem testing revealed internal political issues and lack of flexibility in Lore's platform testing division that did not add to Mr. Sabourin's confidence. GSS was already using Lore as a partner for beta testing, but the quality of Lore's testing departments was inconsistent because they were acquired all over the world through mergers. Lore was too concerned with offering the lowest possible price and not concerned enough with testing all the necessary constraints or with getting into values sync.

Although Perceptor was a very well-renowned software testing lab and seemed to have a good grasp of the testing project, Perceptor would only guarantee a certain number of testers to work at a certain hourly rate for a certain period of time. Mr. Sabourin wanted a fixed price bid on the project, not a bid on time and materials. Perceptor would not give a fixed price bid because they claimed they couldn't know how long testing would take. Although Mr. Sabourin proposed a way to timebox the project, Perceptor was too inflexible to agree to it (see: timeboxing). Perceptor was so involved with the details of people and testing methods that they neglected to pay attention to what the customer wanted, which was a fixed price bid, regardless of the skill of the testers or the testing methods being proposed.

ThreeBears had a good track record in platform testing, understood the approach to testing that Mr. Sabourin wanted to use, and had a good proposal for bug workflow (see: bug workflow). ThreeBears understood that exploratory testing was needed to find conflicts between programs running concurrently with SecNet, and understood how to timebox the testing using session-based exploratory testing (see: session-based

exploratory testing). ThreeBears were also willing to be flexible. They did not have the ability to do testing with a cable modem in their lab facilities, so they offered to perform cable modem testing in the homes of testers with cable modem access. ThreeBears were also very interested in discussing values and philosophy to get into values sync with Mr. Sabourin.

Mr. Sabourin was particular on price since he had a pre-established testing budget, and he knew what kind of testing he wanted to have done. ThreeBears tried to be competitive, but they knew that they tended to be more expensive than other testing labs, so they had to provide more value than their competitors and ensure that they were providing the right services to their customer to make the extra cost worthwhile. ThreeBears was made aware that Mr. Sabourin was looking at other labs, and although he wouldn't say which other labs he was talking to, he was honest and fair about what he would say. That was appreciated by the test lead, John Bee, who was in charge of the bidding and negotiations process on ThreeBears' side.

ThreeBears had qualified testers, could vary all the parameters, were flexible, gave a fixed-price bid, understood the project, made the effort to get in values sync, were available to do the project, and wanted to build a long-term relationship with GSS. For all of these reasons, ThreeBears was chosen to be the partner for the platform testing. Their bid was not the lowest, but Mr. Sabourin was very happy with it.

XIII. Beta Testing Strategy

Beta testing is difficult to initiate and to keep organized. It includes the recruiting of beta testers, finding an appropriate way to compensate these testers, making sure the

testers do the job, collecting feedback from the testers, and handling the feedback. This takes very specialized knowledge and attention to detail.

The statement of work sent to Lore addressed several key issues on which Lore and GSS needed to collaborate to initiate the beta testing. From its myriad of previous initiatives, GSS had obtained feedback from 250 beta testers, though the feedback was not of very high quality. Since their goal was to have 400 beta testers, it was agreed between GSS and Lore that 150 beta testers would be used. This number was divided between two beta testing sessions, each a week long. The second session was done approximately one week after the first with an improved build.

The beta testers needed to be non-technical home users, with a specific percentage of the test group using each of the operating systems (Windows 98 Second Edition, Windows Millennium Edition, Windows 2000, Windows XP Home Service Pack 1, Windows XP Professional Service Pack 1, and Windows XP Professional Service Pack 2 Beta; Windows XP Home Service Pack 2 was not released at the time of testing), each of the locales (Canadian English, Canadian French, and US English), and each of the internet connection options (DSL, cable, and dial-up) relevant to the project. Lore found testers from its pool that matched these specifications.

The information GSS wanted to gather from beta testers included whether SecNet 5.0 interfered with the operation of other software on the end users' computers, whether the performance of the end users' computers were impacted while SecNet was running, and what the CPU and memory usage were. There were two ways in which this data was collected: through immediate user feedback for severe problems such as installation

difficulty, and through a questionnaire to be filled out during each week-long testing period by each distinct group of testers.

Writing a beta testing questionnaire is a challenging task since it must frame the questions in such a way that obtains the most useful information possible from the testers, who are not highly computer-literate users. It was Lore's responsibility to write the questionnaire and to distribute and collect it from the testers. One section of the questionnaire was filled out immediately during and after installation, while the other was submitted at the end of the testing period.

The management tasks related to the beta testing were left to Lore alone. Lore was responsible for making sure that appropriate testers were found and that these testers did the testing. GSS was not concerned with what incentives these testers were offered to test SecNet so long as the testing was done well.

XIV. Platform Testing Strategy

Mr. Sabourin and Mr. Bee decided that a two-pass testing strategy would be sufficient. The first pass included a scripted test to ensure that the basic functions of SecNet were working and then exploratory testing to find bugs in SecNet's interaction with other software and third-party drivers. It is important to note that this is not a case of deciding between scripted and exploratory testing. Each type of testing was chosen for a specific purpose; scripted testing was used to confirm the operation of a specific set of functionalities common to all platforms, while exploratory testing was used to find any bugs due to interactions specific to a platform. Neither approach would have been suited for the job of the other.

The second pass was run after the bugs found in the first pass had been dealt with. It made sure that the discovered bugs were no longer present and that new bugs had not surfaced. Only the scripted test performed in the first pass was included in the second pass (see: Diagram of Platform Testing).

Ten configuration items were deemed most important to include in platform testing: the operating system, the web browser, the email, the web-based email, a few select applications, file sharing modes (downloading, uploading, or acting as a server), chat applications, some select online applications, the type of internet connection (dial-up, cable modem, or DSL), and the other security software installed. These could create over a million different platform configurations. It would have been impossible to test each of them, so they needed to find a reasonable subset of platforms to test. This subset of platforms still needed to add to Mr. Sabourin's level of confidence in the system, so all the parameters needed to be tested.

It was decided to select a subset of test cases to test each pair of items at least once. This ensured that all of the parameters were varied and that they all encountered each other at least once, improving the chance that a critical bug would be found. A subset of tests that fit this condition was found by ThreeBears using a tool called AllPairs (see: AllPairs, AllPairs Results).

This narrowed the number of tests down to forty-eight. Since this was still too many tests, the amount of platforms to test was further decreased to thirty-eight by eliminating test cases that were similar to other selected test cases; only a few extra pairs would be left untested this way. The remaining thirty-eight were prioritized according to

their operating system, with those that were most popular at the time being giving a higher priority.

One platform specified by an executive at Lepton was added along the way. This executive believed that he had sufficient knowledge of software engineering and testing to volunteer a particular platform that he had come up with, and demanded that it be included in testing (see: president's testing). The test involved a lot of OS migration, such as installing Windows ME on top of Windows 98. Such tests are generally handled in migration testing, but other specifications that were made pertaining to programs to be installed and/or running on the machine made the configuration part of platform testing (see: migration testing). What the executive shareholder initially contrived was not physically possible, since one of the platform upgrades specified was not possible in the software. This resulted in the only direct meeting Mr. Sabourin ever had with any Lepton representative. The project manager was also present at the meeting and assisted in communicating to Lepton the necessary changes to the test configuration to make it feasible.

These platform tests could conceivably be done in the ten business days allotted to the first pass of the project. The schedule was aggressive, but ThreeBears was used to being a last resort for testing, so they often worked under tight time constraints.

Almost as soon as the contract was signed, testing preparation began. This included designing the platform configurations, which was done by the test lead; building the ghost image of each system from the base images stored at ThreeBears using the paper configurations made by the test lead, which was done by the testing team; reserving the machines to use for testing, which was also done by the test team; and filling the

purchase order associated with each platform (see: purchase order). The configuration templates made by Mr. Bee included the configuration number, the name of the machine used to test it, the name of the ghost image, and other configuration information and comments (see: Example Configuration Template).

The scripted test was provided by GSS and was converted to spreadsheet format by ThreeBears (see: Scripted Test for SecNet). The charters of the exploratory tests to be performed after the scripted test were created by Mr. Bee in the first week of the project. They were sent to Mr. Sabourin for approval, which resulted in a battery of eleven exploratory tests to be run on each platform (see: Testing Charter Summary).

The testers needed to familiarize themselves with SecNet. ThreeBears was given a build by GSS on which they were to run the tests, so the testing team spent about an hour looking over the build together to learn about the program.

XV. Beta Testing Workflow

The beta testing of SecNet took place during the second quarter of 2004. Communication between Lore and GSS about the status of testing took place between the product lead at GSS and the project lead assigned by Lore. When GSS received feedback, their testers contacted the beta testers who had encountered issues by phone or by email to isolate bugs. A phone call between a tester at GSS and a beta tester took a maximum of twenty minutes. All other communication took place through email only. Communication between GSS and the beta testers was kept as minimal as possible, and the lines of communication were only open for a short time. The beta testers were under no obligation to GSS after the conclusion of the testing period.

XVI. Platform Testing Workflow

A lot of information was communicated between ThreeBears and GSS during platform testing. Most of it 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 on a particular platform and any questions raised during the testing of that platform.

Testing at ThreeBears was intense since the timebox was small. The first week of testing was spent setting up the configurations to test and making sure they were correctly installed; this process was supervised by Mr. Bee. Although this took up about half the testing time, it ensured that testing ran smoothly during the second week and that, were GSS to choose ThreeBears for later testing projects, the ghosted platform images would be available for use within minutes instead of hours, making ThreeBears more competitive in the future and cutting costs for GSS if they chose ThreeBears as a partner for later projects. During this process, questions arose, such as whether it mattered if software was run over a network or from CD, and whether a typical or custom installation of Microsoft Office should be used. These questions were answered by Mr. Sabourin.

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. Mr. Bee created a template for the testers at ThreeBears to use to report the results of each session (see: Combinations Testing Session Template).

The test lead took any questions about whether a bug was relevant to the project to Mr. Sabourin. For example, there was an issue with the software activation found during platform testing, and Mr. Bee called to ask whether this was within the scope of the project. Mr. Sabourin responded that it was not, and Mr. Bee was instructed to ignore it.

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. If Mr. Bee did not receive a purchase order on schedule, if another purchase order needed to be made, if an existing purchase order became more expensive, or if a tester was in the process of finding a bug, he told Mr. Sabourin during their daily conversations. Mr. Sabourin also told Mr. Bee anything that Mr. Bee wanted to know about particular areas of the program to examine carefully.

This communication process was extremely valuable to both ThreeBears and GSS. It meant that GSS was not receiving false bug reports and wasting their time sifting through them, so the credibility of the ThreeBears testers as perceived by the employees at GSS was not undermined by the annoyance of false bug reports. Also, the need for extra software purchases was dealt with as soon as possible through this fast line of communication; since anything that could have changed the project could have changed the contract, such as extra money being spent on purchase orders, it was critical for purchase orders to be denied or approved quickly to prevent project delays for contractual issues.

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. Status reports were made through a simple spreadsheet with a list of bug numbers provided through the database on an almost daily basis.

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. This workflow was effective most of the time, but there was one instance in which a bug was found on a computer at ThreeBears and the computer had to be physically transported to GSS for the GSS developers to reproduce the bug.

The project manager at GSS needed to know if the platform testing project was achieving its goals and whether a third cycle of testing would be needed. He received this information from Mr. Sabourin. The important consideration for the project manager was whether or not the project was moving onto the critical path (see: critical path). Mr. Sabourin had structured the platform testing to be off the critical path, and he and the project manager wanted to keep it that way.

The executive management at GSS only wanted to know whether or not the platform testing project was on schedule and on budget. Mr. Sabourin did not try to project the number of bugs. He did project the dollar amount that would be spent, and whether an unacceptable amount of support calls would be generated by SecNet. The stakeholders at Lepton saw only the final results of testing because there were many non-technical stakeholders among their executive management who would have considered finding bugs in SecNet to be bad news instead of seeing it as part of a gradual process to improve the software. The Lepton stakeholders received a detailed summary of every

platform tested and the exact tests performed on each platform, but only from the end results of the second pass of testing, in which the bugs were shown as having already been fixed.

XVII. Bug Tracking and Triage at Greene Software Systems

When information on a new bug was made available to GSS either from beta testing or platform testing, it was added to a list of bugs on the internal bug tracking system at GSS, an implementation of Bugzilla. Combinations testing bugs were triaged along with bugs found through all other forms of testing being conducted at GSS. The triage team, consisting of the GSS test lead, the product lead, and the development lead, decided for each bug whether to fix it, avoid it, or ignore it.

One consideration that would almost always get a bug fixed was if it was expected to generate support calls, since the purpose of the project was to decrease the number of support calls to avoid an SLA violation. The triage team was able to get a sense of the number of support calls that a bug would generate in several ways. They knew the OS numbers for their target market, so if a bug affected a particular platform, they were able to determine what percentage of their customers could be affected.

They also knew what sort of problems had led to support calls in the past, so if a similar bug was found they could approximate whether the number of support calls would be high or low. To do this, the triage team would speak with their technical support department; this sort of investigation was only necessary in a small number of cases. Another important consideration was whether a bug would cause a high volume of support calls at once or a low volume of support calls over a longer period of time. Since

a large number of support calls in a small time frame would cause more stress on an ISP's technical support lines than if the calls were distributed over a larger time frame, such bugs were considered to be of higher severity.

XVIII. Results of Beta Testing

The beta testers in each of the two testing phases completed their testing and returned the results to GSS. In one of these phases, seventy-two testers out of seventy-five returned the full questionnaire (see: Beta Testing Questionnaire Results).

XIX. Results of the First Pass of Platform Testing

The results of the first pass were submitted on April 4th, 2004 (see: Platform Testing Pass 1 Summary). Forty-two bugs were found during the week of testing. Most had not been discovered previously at GSS through other testing methods, so platform testing was very successful at finding bugs that were not found through other methods. Some of these bugs involved critical stability issues on Windows ME. In particular, installation through a dial-up connection consistently produced a blue screen. Without platform testing, such a serious issue may not have been discovered.

XX. Results of the Second Pass of Platform Testing

The summary of the second pass was submitted to GSS by ThreeBears on May 10th, 2004 (see: Platform Testing Pass 2 Summary, Platform Testing Pass 2 Appendices). The scripted test was passed on all thirty-nine configurations, indicating that no new

platform-related bugs had been unearthed through fixing the bugs discovered in the first pass.