28TH ANNUAL
PACIFIC NW
SOFTWARE
QUALITY
CONFERENCE

OCTOBER 18TH – 19TH, 2010

ACHIEVING QUALITY IN A COMPLEX ENVIRONMENT

Conference Paper Excerpt from the

CONFERENCE PROCEEDINGS

Affecting Printer Installation Success in the Consumer Market

Kathleen Naughton
kathleen.e.naughton@hp.com

Abstract

This paper will describe the design and development of a test lab that has been affecting change in the R&D requirements and development processes to reduce the install call rates while supporting the adoption of wireless technology advances in the consumer customer segment. The paper will:

- Describe the process of aligning on a definition for install success
- Identify factors that contribute to install success and failures
- Explore how unactionable, field-found issues influenced test lab design
- Describe how test lab design objectives drove executive sponsorship, development, and implementation of a whole new testing service
- Show how the whole process has driven changes into R&D requirements and deliverables
- Examine recent call-center data to validate affects of these changes

Biography

Kathleen Naughton is a software test architect for Hewlett-Packard. Kathleen has over 20 years experience in the technology industry implementing a parts ordering system on one of the first nation-wide WANs for Tektronix, verifying graphical software development kits for CA, designing testing for streaming media cell phone software with Packet Video, and most recently, printer and scanner test design and lead for Hewlett-Packard's Imaging and Printing Group. She has been the printer installer test lead for over 5 years, co-chaired the Install Success Taskforce, and directed multiple programs and studies performed to reduce the install call rates. She is an active member of the Anita Borg Institute's organizing committee for the Grace Hopper Celebration of Women in Computing conference as well as a regular presenter at that conference.

Introduction

I just bought a new printer—needing to get my [term paper, resume, wedding invitations, soccer team flier] printed like yesterday. I unbox the device, follow the single-sheet "how to" instructions and I'm good to go. But wait! My PC can't find the printer on my wireless network. My security software keeps giving me pop-up notifications and warnings. And now I have gotten a fatal error from the installer! On the third attempt and with assistance from the "dreaded" call-center technician, I am finally ready to print that essential document that initiated this adventure into technology setup and configuration frustration and aggravation.

This scenario is more common than we as technology companies want to admit. It is an especially challenging experience in the consumer and small business marketplace where our PC and network environments are not managed by IT department policies and processes. At HP, we have categorized call-center calls to help us monetize the cost of failures in the field. We knew the data – at one point 70-75% of calls were categorized as "install" calls. We developed tools to help us mine what the failures were. But we could not seem to effectively change the install call-center rates in the consumer marketplace. That has been changing!

This paper will describe the design and development of a test lab that has been affecting change in the R&D requirements and development processes to reduce the install call rates while supporting the adoption of wireless technology advances in the customer segment. The paper will:

- Describe the process of aligning on a definition for install success
- Identify factors that contribute to install success and failures
- Explore how unactionable, field-found issues influenced test lab design
- Describe how test lab design objectives drove executive sponsorship, development, and implementation of a whole new testing service
- Show how the whole process has driven changes into R&D requirements and deliverables
- Examine recent call-center data to validate affects of these changes

Aligning on definitions

It seems obvious what it means to have a printer successfully installed, but install success has as many meanings as there are perspectives of looking at it.

We use our call center data as a direct conduit to customer experiences. Our call center "install" bucket was filled with calls regarding network failures, printer hardware power issues, ink cartridge installation difficulties, unpacking confusions, paper loading location misunderstandings, failures to print or scan after software drivers were seemingly successfully installed, and errors that occurred as the software was installing. From the user's perspective, they are all install failures. The software installer development team became the "owner" of all these issues because they are the only team with "install" in their name.

As a group assigned to look at the call center install failure bucket began to dive into the call data, it was quickly realized that the team needed to be more than just the software installer team. We created a cross-functional team and began re-defining the call center data. Since the "install" bucket actually encompassed failures that occur in the "1st day of ownership", we called the large bucket the "1st day experience" issues. We then further divided the issues into software, hardware, product design, and "other" buckets. The team then divided back into their respective functional areas and began working on attempting to address the different types of issues. The remainder of this paper will look at the software 1st day failures and how—through a series of tools, requirements, and, customer test programs—a new testing service was designed.

With the narrowing of the view of install failures to the software portion, we still needed to align on what install success really means. By definition of existing release standards and criteria, we were having 100% install success through defined testing methods and lab execution programs. These methods and execution programs were defined by program requirements generated by our Marketing and Customer Satisfaction groups, and by technical requirements generated by the R&D community. Our lab methods and executions used the principle of isolation such that we used clean operating systems with nothing else installed on the system. We also used relatively fast hardware and larger memory to allow for higher throughput of tests. However, our customer install success rate was still not improving as measured by the call center data, so we needed to better define what install success meant to the software lab to help focus the requirements and prioritization.

We developed a multi-part install success definition:

- Technical Install success: there are no errors generated by OS or installer, and communication between printer and PC is verified
- Install time success: a specified time limit that installs are expected to be complete
- User experience success: the customer can successfully navigate through the required steps to complete the installation with a functioning printer

In-depth examination of the User experience success will not be addressed in the remainder of this paper; it is still an evolving challenge. The other two definitions are the core driving definitions for R&D driven requirements and the focus of a new testing service.

Contributing factors to success and failure

Now that there was alignment on what install success was, we needed to see if we could figure out what contributes to install success and especially to install failures. Since we were experiencing 100% install success in our lab test environments, this required some out-of-the-lab-box approaches. The backdrop for all this work was the continuous product release cycles that we needed to deliver. On average, the product release cycle had major software product releases occurring every six months. These product releases needed to not only support new printer device features, but also major technologic advances and operating system releases. The technologic advances include moving from primarily USB connected devices to network connections and then to wireless connected PCs and devices. The operating system releases included support for 64-bit architecture and major OS changes from Windows 98 through Windows 7.

One of the first things we needed to do was to figure out how to analyze the failures occurring at the customer locations. We needed a way to do this remotely and preferably at the time of failure. The development team created an error recovery tool that would generate a hash code based on an algorithm that included the last executed line of code in the installer and other factors that would help isolate the state of the installation at the time of failure. Then, with customer approval, we would collect this hash code and system configuration information into a package and post it to secure HP servers. This took some refining and evolved to also include a download to the customer's PC of any fixes for matching hash codes. The main problem with this tool is that even with this information, we could not consistently reproduce the errors in a lab setting. So the number of fixes that actually got developed and deployed was very limited. We were still releasing products with similar software install failures so the call rates remained statistically unchanged.

The next step in the analysis was to try to figure out how to be there when customers experienced failures. We had an existing customer Beta program that took our products with HP observers to customers' homes. The program entailed having customers take a boxed printer and go through the entire setup from unboxing, setting up the printer (installing cartridges and paper, plugging it in), installing the printer software and drivers, and then using the printer. The HP observer's task was to document what the customers were experiencing—any missteps they took in the process, any technical issues they may have experienced—as well as documenting the customer's environment—their PC configuration, operating system, networking, and any installed software. The program left the device with the customers

for 2-4 weeks to gather in-depth usability feedback. This was a good source of information. However, there were three major flaws to the program: first, it was too late in the program to allow for issue fixing unless we delayed the product release; second, the sampling pool was too small—usually about 15-20 customers; third, we had very few technical install failures in the program.

R&D needed the information earlier in the project to be able to make any code changes to correct any issues. They also needed a larger sampling size so that we could be able to generate statistical information. A failure in a sample size of 20 (5%) is significantly different from a failure in a sample size of 50 or 100 (2% or 1% respectively). So the Iota program was developed. Essentially, it is an install-only beta program. We leveraged the Beta program support structure: data collection, customer search and matching, program management, etc. The program was modified to expand the sample size to 50 customers. We had R&D and Test engineers attend as observers. We took the printers to the sites all set up: ink installed, paper loaded, initial power up cycle completed, and had the customers start with the software install. After the install completed, the customer did a quick print and scan functionality check, then we uninstalled and cleaned the customer's system completely of our prototype software. The Iota program was designed to be executed before release of the software with hopes of being able to fix any issues that were found.

The first Iota was an eye-opening experience for everyone involved. It was executed late in the program life cycle, when our existing indicators and measures told us we were ready to ship. Our planned tests were passing at 100% pass rate and our normalized-defect-arrival-rate (nDAR) indicated that we were ready to ship. However, in the live customer environments, our install success rate was only about 50%. The Test lab owned reproduction and characterization of the failures. We only had success in reproducing about 20% of the issues. There were only a few code changes that resulted in directly fixing found issues. In the end, the product shipped with known defects with unknown resolutions.

This was an advantage over earlier programs. Previously, we shipped with unknown defects as evidenced by the continued call center install failure data. With the known defects, work continued even after shipping to help us analyze and characterize the install success and install failure environments. We were able to develop a list of characteristics of customer environments that contribute to the install success or failure. These factors were categorized as:

> PC Hardware factors: computer processor speed, amount of RAM, type of network (if any), and for older system, the size of the hard drive. In recent years, the costs of processors, memory and storage have come down to the point that these are not defining factors like they used to be in the install success space. However, the consumer hardware factors include PCs ranging from circa 1990's hardware to hot-off-the-line netbooks.
>
> PC Environment factors: this includes what type of security applications (anti-virus, firewalls) are installed, OS versioning and maintenance (i.e. system updates maintained), other third party applications and their state of maintenance, and partially installed software that puts the system into an unstable state, to name a few of the things that were verified issues in our customer visits
>
> Customer knowledge and understanding: there are customers who admit that they don't know or understand their wireless network or settings – and they are just fine with a USB connected device. However there are customers that think they know and understand their wireless network and really don't. They will attempt to connect their printer to their neighbor's open wireless access point, or not know what their SSID or passphrase is to their wireless access point. We discovered wireless install success is a major challenge for this group. With the move to enable all our devices to be wirelessly connected, these are major things that we need to address to make sure we can have install success with all our customers.

Test Lab design influences

We had been collecting and analyzing data from the field and customer test programs for several years. We then began looking at ways that the test lab could systematically incorporate the information. The goal was to be able to execute tests in more real-world environments so that we could have multiple iterations of execution. Other goals included facilitating defect reproduction and characterization, and giving development staff access to the environments to assist in debugging and analyzing root cause. These were things that we could not readily do in existing customer test programs. The other driving factor was the prohibitive cost of customer-facing programs. We contract out the customer identification and matching service, we offer monetary compensation to the customers who participate in the programs, and the productivity thrash due to not being able to reproduce and characterize issues in the lab quickly adds up.

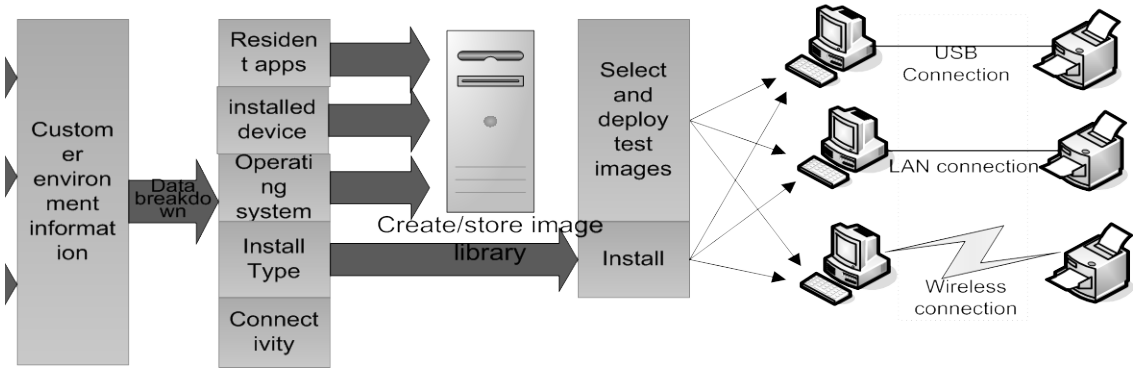
I took the initiative to collate and synthesize all the information collected to propose a new test lab. The over arching lab design is influenced by the desire to have “real customer” environments. This meant securing a variety of PC hardware, creating software environments that were based on real customer combinations versus lab-contrived combinations, securing a variety of wireless access points, and having enough variety in the inventory to be a meaningful contributor to test outcomes. All of this needed to be done with budgetary constraints.

Right Design Objectives

Since I needed budget support, I needed to secure management sponsorship. I made sure that my presentations to management included some core philosophies: the test lab would be customer centric—using “real customer” information to create the lab; targeting replacement of expensive customer visit programs—my goal is to replace the need for the Iota program; and modeling the test execution to include random selection of environments to match the “luck of the draw” aspect of established customer Beta and Iota programs. All of these things resonated with my management and I was given conditional support for a trial run after just one presentation.

To meet these design objectives, I designed a lab with approximately 20 PCs that could have about 10 unique software environments each for a total sample pool of 200 unique PC environments. I also needed an infrastructure to support managing PC images, multiple stations, and multiple wireless access points. The budgetary objectives were facilitated by some good timing and generous sponsorship.

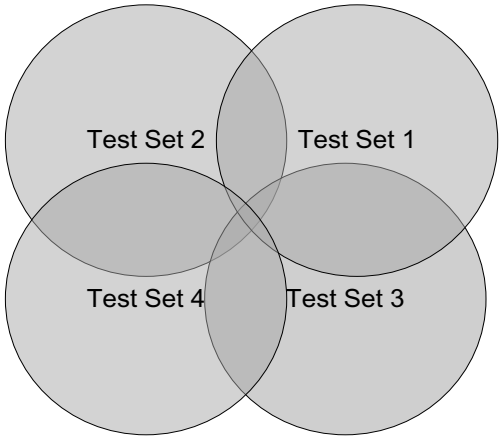
Conceptual diagram of lab design:



Another division of the company was dismantling a hardware lab. I approached the lab manager and secured a variety of PC hardware that was destined to the recycle bin and augmented the collection with standard test lab PCs. The cost for getting my 20 PCs was the cost of shipping them from one lab location to the other. The lab facilities were reconfiguring, which freed up several carts that could hold 4 PCs, mouse, keyboard, monitor, as well as small network and KVM switches. All the peripherals were test lab stock items. So I had the work stations for no cost. Using an existing more powerful test lab PC box, I had a server system built for the cost of added memory and hard disk space. I did some data mining through the data that was collected from customer test program visits (Beta and Iota) and secured system profiles for 200 real customer profiles. We then engaged in the replication of the real customer environments, creating Ghost images stored on the server. Labor cost to do this was the largest expense but, due to timing, was covered by existing budget surplus that needed to be used.

The next phase of the test lab design was to look at how to use the environments such that they could be efficiently used to make meaningful contribution to the overall testing—not a duplication of testing. I looked to the customer test programs for inspiration. I saw that the programs would target new customer pools. We may have a few repeat customer testers, but mostly the customer pools were a random selection of qualified volunteers. I had roughly 200 images to draw on so I chose to define a test run as a set of randomly selected images from the whole pool. This means that no two test cycles are identical. Over the course of the program with multiple test cycles, we would eventually get coverage of all images. The test cycles could be represented as Venn diagrams.

Venn diagram representing test cycles



Once I figured out that I wanted to randomize the environment selection, I needed to define the test set size, the test activities, and what metrics to report. Again, I turned to the customer test programs for inspiration. The Iota customer program was set at 50 installs. This was big enough to get meaningful statistics and about the right size that we could complete a test cycle in a 1-2 week period with one test execution resource. The other thing the Iota program did was to perform a simple print and scan after the installation to verify that everything was as it should be. So my test cycle tasks are relatively simple: install, print, scan. The final thing I wanted to collect and report was the install time—especially since this is a major customer satisfaction goal.

One last test lab design objective was to have a simple report that gave an aggregate picture of the install success in the new lab setting. The report began very simplistically with color coded success percentage across the test cycle set for the separate tasks: install, print, scan from device, scan from software, and average install times. The set was defined at 50 installs, so the averages and percentages reported are across 50 installs. The color coding is based on existing test lab standards: green >90%, yellow 70-90%, red for <70%. This report is referred to as the dashboard.

Sample dashboard (in grayscale):

	Device	Connection Type	Install	Print	Front Panel Scan	Software Scan	Avg.Time	Overall Percentage
Build 1	Device A	Wireless	84% (yellow)	84%	64% (red)	80%	0:27:20	78%
Build 2	Device B	Wireless	100% (green)	100%	86%	96%	0:34:27	96%
Build 3	Device A	Wireless	90%	94%	84%	86%	0:15:38	89%

I have created a web site where all the test data resides. The home page of the site has updated dashboard results for all projects under test. All levels of management have access to the site and email is sent to the program teams when results are posted.

From concept to first test run, the process took about 6 months. The process steps included not only the concept and design work, but also the presentations to management for support, an unplanned office

relocation just as we were finishing design implementation and setup, and securing buy-in for the first program to use the lab.

New Testing Service

The first test was executed for a program that was a newly-architected solution and was the first program to be using agile-type methods for development. They were a guinea pig for new things, so it seemed natural to be part of this new testing service. I was nervous about whether the theoretical lab design principles would actually give results that were meaningful. I waited not-so-patiently as the test execution was done and sifted through the results. It was wonderful when the first failure occurred! It was in an environment that had a "known issue" that our previous installer had addressed but was not included in the requirements for the new installer. It was rewarding to know that the lab design worked as expected.

This led to a core tenet for lab maintenance: if it is a known failure, keep it as part of the lab environment—even for old, unsupported things because our customer install base will have this old technology still out there.

The second test of the testing service came when I was approached by the previous generation software project manager, who wanted to see how the released software performed in the new lab. This was an interesting case because this program had gone through the customer-based Iota test program and there was real customer data available to compare the test lab results against. Again, I was nervous about the outcome. How would the lab results compare to real customer results? In the lab, we could not replicate the customer knowledge aspect, only the technical hardware-software environment. Once again, when the results were reported, the lab and customer tests were statistically identical! Success! The Install Success Test Lab (ISTL) was validated.

There were issues that still needed to be worked out. The program teams pressured me to use the same 50 environments for every test run. Under pressure from management, the program teams set up release criteria that were measured by the ISTL results. The program teams resisted being held accountable for something that seemed like changing environments – they felt that having different environments for each test run was exposing them to an apples-to-oranges comparison. I stood firm on the design principle of randomized image selection for each test run. I pointed out that we do not go to the exact same customers in our customer-based test programs. The outcome of those programs is greatly influenced by "the luck of the draw." I stressed that I was doing what I could to replicate real customers so the randomization was an essential element to the ISTL results. During program execution, ISTL test runs were scheduled at the end of each sprint – roughly every four weeks. This resulted in multiple test runs of 50 installs for the program. R&D saw the effects of their feature and code changes across the multiple test iterations. The changes generally resulted in improvements in ISTL test results, despite the randomized environments. The development staff became more comfortable with the idea and stopped asking for the same 50 environments.

Another surprising adoption resistance is the pressure to change the environments when issues are discovered to increase install success. This actually comes from my test lab colleagues. For example, I have several systems that are of minimum memory configurations. The test lab generally does little testing in minimum requirement settings due to the productivity hit when using these configurations. When there was an issue with an install failing because of this system configuration, my test lab colleague told me I had to change the system. The system was to specification, the installer was not behaving correctly. It was counted as an install failure in the test run. A defect was filed and the installer code was changed. Another example is when we encounter conflicts with an older application that is not on our official application compatibility list, I was told that I had to remove it from that particular environment because the defect is resolved as will-never-fix so we shouldn't be testing with this particular application in the environment. I refused because there are customers out there that still have these older applications and we need to know that we are not compatible with them and have support in place to address the issue for our customer.

This has helped solidify another core philosophy for the Install Success Test Lab—sour the pot as much as possible. By this I mean, make sure that I have the known failure environments: 3rd party applications, wireless access points that have caused problems in the past, maximum length passphrases, etc., in the lab at all times. This leads to things like refreshing the stock of wireless access points every three to six months and adding new environments after customer test programs have completed. After a year of lab operation, our environment library is almost 400 images and growing.

As programs worked their way through the ISTL execution, program requirements started being generated from the results. This is one the few test lab programs to generate program requirements. An example of the requirements include install time criteria based on ISTL installs versus the previously standardized "clean OS" install time measurement. This led to a requirement that 80% of installs must complete in x-amount of time as opposed to an average install time. This is because install time is influenced not only by hardware – processor speed, memory, disk space – but also by what other software is installed on the system. In the consumer customer base, there is a vast variety of hardware/software configurations. The previously standardized install measurements did not reflect this reality. But ISTL variety is reflective of customer base, so allowed for a realistic measure. Another example of a requirement driven by ISTL testing is the requirement to be compatible with a wide variety of security application suites.

When ISTL was first conceived, it was targeted for support of the consumer inkjet printers that my test lab supported. Once the results began to be published, there were discussions of the test service in other product labs. To my surprise, I was approached by the laserjet printer R&D and asked if I could support their programs. They wanted to see how their installer products fared in the replicated customer environments. We found several issues that they hadn't encountered in their lab-based testing that increased the quality of these products. The ISTL now executes tests for all the inkjet and laser jet products—even the ones targeted for enterprise markets.

Call Center data examination

Unfortunately, the printers that have been through the ISTL testing service are newly released and have not generated enough call center data to give statistically reliable comparisons at the time of publication. However, the preliminary indications show a significant reduction of install related calls for the ISTL "graduates."

	% Install calls (of total calls)	% Wireless calls (of total calls)
Pre-ISTL	~42%	~30%
ISTL	~28%	*no data

*There is no wireless data because wireless devices have not been released to market at time of publication.

Summary

In conclusion, software test labs can influence product requirements if the test lab offers unique test objectives. In the case of the Install Success Test Lab, there was a very specific and persistent issue that the test lab is addressing. By stepping back and looking at all the attempts to address the issue, then analyzing the faults or failures in the attempts, a different test approach and service was created and accepted by R&D. With a little imagination, creativity, simplistic design and some luck, the testing service is a success.

Another contributing aspect to acceptance and product requirement generation is having well reasoned, data driven philosophies that are adhered to regardless of pressure and influence to change. The consistency in adhering to the philosophies has driven broad organizational changes. This is evidenced by the solicitation across the printer divisions for use of the testing service.

I am waiting for more product introductions to customers and the call center data to be analyzed to see how significantly the test service has affected the install call rates. All the program teams and management feel better about the ISTL "graduates" than we have about any previously released programs. But even then, the backdrop of ever changing environments – operating system upgrades, ever-changing 3rd party applications, web-based computing, etc – are always there to create install success land mines for our device installs. This only means that the Install Success Test Lab will be making continuous adjustments to continue to contribute to the product test cycles.