Software Performance in the Real World: Personal Lessons from the Performance Trauma Team

Jayshankar Sankarasetty
Systems Engineer,
Fidelity Information Services
MEPZ, Covansys
Chennai 600045, INDIA
91 (2262 8080)
Jayshankar.Sankarasetty@fnf.com

Kevin Mobley
Director of Software Performance Engineering,
Fidelity Information Services
14 Piedmont Center,
Suite 800
Atlanta, GA 30305
01 (404) 439–8434
kevinmobley@gmail.com

Libby Foster
Sr. Systems Engineer,
Fidelity Information Services
14 Piedmont Center,
Suite 800
Atlanta, GA 30305
01 (248) 792–9474
Libby.Foster@fnf.com

Tad Hammer
Sr. Systems Engineer,
Fidelity Information Services
14 Piedmont Center,
Suite 800
Atlanta, GA 30305
01 (404) 439–8338
Tad.Hammer@fnf.com

Terri Calderone
Systems Engineer,
Fidelity Information Services
14 Piedmont Center,
Suite 800
Atlanta, GA 30305
01 (404) 442–4235
Terri.Calderone@fnf.com

ABSTRACT

In the nine years that we have been involved in software performance engineering (SPE) and performance testing engagements we have learned several things. Across numerous eCommerce applications and an enterprise CRM product suite, our knowledge base about the field of Software Performance Engineering is constantly evolving. The focus of this paper is what we have learned in the areas of SPE project management, performance testing, defining the scope of SPE projects, ITIL, post production performance support, and exploration of the boundaries of applied SPE. Is it really just about performance?

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Software Performance Engineering best practices – *systems integration, project management, business process selection, performance testing, performance management, and case studies.*

General Terms

Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Standardization, Theory, Verification.

Keywords

Software Performance Engineering, Software Performance Management, Performance Testing, ITIL, Six Sigma, and Project Management

1. Real World SPE

This paper serves to provide examples of how SPE can, and has, been accomplished successfully in the real world within our organization. These vignettes show different views within our SPE group, and how the theories have been applied. The first section comprises the project management and overall guiding principles our SPE team uses. A case study involving one of our customers follows, then our scripting and testing system. Furthermore, an overview of ITIL related to production is introduced. The last section details the most pervasive and destructive assumptions within the software engineering discipline related to SPE.

WOSP'07, February 5–8, 2007, Buenos Aires, Argentina.

2. Software Performance Engineering: Project Management

The realm of Software Performance Engineering is riddled with misunderstanding from those outside of operations or those who have never worked on bringing software into production. Some of the common misconception it that performance testing is "just another functional test" to load test the system. Another primary misconception is that we'll just "get some numbers" to prove the system performs. If these misunderstandings are not managed and corrected, they can be detrimental to client relationships. Issues related to performance must be anticipated, thought through and planned accordingly. This makes the job of a project manager in the performance engineering team even more uncertain as this area is full of unknown pitfalls.

Contrary to the beliefs of the Project Management Institute®, it is very difficult - if not impossible - to project manage performance engineering projects without a base understanding of what the goals of the project are, and how the software performance engineering cycle comes into play. As with most software development firms, project resources are at a premium. The role of SPE project manager expands to fill the gaps where there is a great need to communicate and educate client teams. At the very least, the project manager needs to be able to convey a high level understanding of the ins and outs of what to expect from a performance engineering effort

2.1 The SPE Project Manager - Educator, SPE Public Relations Representative and Planner

Educating executives and project teams, both inside your own organization and outside, can be daunting. If done successfully, it can forge cross-disciplined, supportive proponents for the often overlooked role of the performance team. Executives have little to no time to understand lengthy white papers and technical reports on performance engineering, so boiling down the complex process of Performance Engineering is imperative. Herein lies the challenge: succinctly explaining complex and multi-faceted concepts to business savvy, harried executives and business,

operations, and support staff at the same time. As in the other disciplines of project management, communication is key. The technical roles inside the performance engineering team are complex and inter-disciplined, with few parallels in the software industry. With tighter deadlines and ever increasing demands for "performance insurance" - the base motivation for performance work - these roles are stretched thin. The time required for many in technical roles to translate what they do grows, while the amount of output expected expands. In order to bridge this gap and assist the SPE team, the project manager is a key asset.

While most executives tend to want shorter, higher level information, other groups want more detailed information about what is expected. The term "want" may be subjective, but it is important that these teams get the necessary information so that they may fully appreciate their important role in the business process selection process which drives positive post "go-live" performance.

Here is where the role of the project manager as "Public Relations Manager" comes into play. It is through this process that the project manager adds great value to the performance team. Generating this interest and desire to prioritize performance engineering efforts is generally an uphill battle, particularly if the staff is new to bringing software into production. If a buy-in from executives and line management is not achieved, the likelihood that the project will succeed, or be completed on schedule, is severely reduced. Again, this is contrary to the belief of many in project management, that understanding the effort in any amount of detail is not required to successfully manage projects. If the public relations tools are applied to create interest, the role of the performance team is significantly enhanced. The performance engineering team is definitely not an island, and cannot be successful without input into the priorities of the business and the understanding of the day-to-day operations of the business.

The planning effort for performance engineering is not an easy task. The plan must be flexible enough to allow for impromptu - and notoriously last minute - projects and the management of these issues and risks that arise takes dogged determination in resolving. In many cases escalating project roadblocks that are encountered along the way is necessary. In high level form, the process flow looks similar to the following:

2.2 Project Initiation and Education

This allows for acceptance of performance engineering into the implementation software development life cycle.

2.3 Environment Setup

Whether the subject is host, client/server, or web based there are a multitude of details involved with this process. Not only are typical requests made - firewall, network access, etc. - there are special requests that often require following more lengthy processes to achieve. These may include items such as owner access to database, web, and host servers. DO NOT underestimate this stage in planning the performance engineering effort. Oftentimes, the processes for getting this work accomplished are not clearly documented, nor are internal service level agreements established, to quickly resolve issues for a misconstrued "testing" environment. Misconstrued, because the performance environment is a completely different animal altogether, in that production scaling occurs along with database scaling which means the creation of the environment is much more demanding and complex. It is important to remember, more complex requests = longer delivery times. Red tape is not the only hurdle that must be overcome, but it certainly takes its toll on the SPE project schedule.

2.4 Business Process Selection

This is where key business and operations staff are educated in the method for business process selection. The members of this group are key contributors to the success of this effort and convincing these often over-taxed team members for participation sometimes takes a little more effort. The more acceptance and diligence that the staff exerts, the more precise is the focus of the performance engineering team's effort, and the better the chance of accurate performance results. At this juncture, the executive, operations, and performance engineering teams MUST agree. It is best to have a formal sign-off of the accepted terms of the performance testing. This includes agreement and acceptance of business process selection, utilization (how often the mix of business processes are invoked), and project performance targets. This has on more than one occasion come back to bite the performance team when not "closed" properly.

2.5 Performance Testing and Optimization Cycles

The majority of pitfalls exist at this point. As in standard quality assurance testing, the performance effort is the last car on the train. Furthermore, the estimates produced are based on finite assumptions. Even though the assumptions are documented, they are often ignored, or at best severely discounted, by the project team. This situation can create tension between the development project team and the performance team. Again, continued or repeated education is in order. As go-live approaches, the urgency of the project team to accomplish goals increases. Unfortunately, if buy-in was not achieved and the realization that performance IS a key contributor to the success of the "go-live" arrives late in the process; you have a "house on fire" scenario. This is never conducive to comprehensive analysis and engineering, but is often the state that the performance team finds themselves in.

2.6 What If Project Management Is Not Involved In The Performance Engineering Effort?

The results vary, but generally, it is not a pretty picture. As with any other area of software development, managing and controlling activities, if left to their own devices they do not always get accomplished. Part of the project manager's job is to dot I's and cross T's. This is most important in documenting the agreement of all parties of the objectives, targets, and processes of performance engineering. This goes hand in hand with education, along with accomplishment of buy-in to the performance effort.

In conclusion, the role of project manager is essential to the SPE team. Their expertise in understanding projects, educating new project implementation teams and gaining both buy-in, as well as terms acceptance, are all vital to the success of the performance engineering effort. Bringing the worlds of project management and performance engineering together allows for greater chances of project success in the timeframes outlined in the overall implementation schedule.

2.7 Software Performance Engineering: Using Six Sigma Define SPE Focus

The failure modes and effects analysis (FMEA) process is a Six Sigma concept used to determine the software modules to be tested for software performance engineering considerations. The Pareto principal (also known as the 80/20 rule) states that for many phenomena 80% of the consequences stem from 20% of the causes. The FMEA tool is an approach to define the 20% of software pathways that will create 80% of the performance problems in a working software system. By utilizing the FMEA tool early in the development process one can decrease the amount of time for performance testing - limiting the number of scenarios to test and increasing stability of the deployed software by testing the correct scenarios.

The FMEA process was originally created by the US military in 1949 to classify failures. Used in the Apollo space missions of the 1960's, it was also used by Ford to reduce future risks after the Pinto fuel tank ruptures. FMEA begins with brainstorming sessions of a cross-section of individuals that have knowledge of the product and have a stake in the product's success. The first step is to list all of the critical parts of the product. The next step is to list all of the possible failures in the product and to list the possible causes of the failures. These parts must then be ranked based on the criticality of a failure. The ranking of a failure is in the range of 1-9 and it is based on the severity, occurrence, and detection of the failure. For SPE, the FMEA process can include a 'willingness to wait' factor; this means the amount of time the user can afford to wait for a failure to be addressed. Each part of the product being analyzed must be given a number for each of these four factors. The final step is to calculate the Risk Priority Number (RPN) of each part of the product by multiplying together all of the factors.

RPN = Severity * Occurrence * Detection * Willingness to Wait

The higher the RPN of a component, the more risk that component will be to the overall success of the project, and the more attention that piece should be given.

Below is an illustration of applying the FMEA process to a live business situation. In 2005 Prospect Bank, located in Michigan, was deploying a substantial new release of their banking sales software developed by Fidelity Information Services (FIS). The FIS software performance engineering group was enlisted to test the software performance before deployment to ferret out any issues that could cause a postponement of the "go-live" date. The case study will illustrate the step by step process that was followed and how this process ensured the reliability of the newly released software.

3. The Real Deal - Working With Customers

The following case study implements the methodology described above, only the names have been changed to protect the guilty.

3.1 A Case Study:

Prospect Bank had originally deployed FIS' TouchPoint software product in 2002 but were moving on with a new major release, as well as customized functionality above and beyond what had been performance tested during the development cycle of this major release. The bank was not accustomed to working with a true SPE department or the FMEA process. The bank had a staff of two to run performance tests, with knowledge of performance software test tools, and that was all they thought they would need. The FMEA process was viewed by the bank as merely a revenue generating step for FIS, rather than a valid process to determine what scenarios to performance test. The political atmosphere was a major challenge to overcome.

Below are the steps that were taken to determine which scenarios should be tested to meet the 80/20 rule.

1. A group consisting of all stakeholders, including business analysts, software developers, and performance engineers, worked together to come up with a list of business processes (BP's) that would be performed with the new version of TouchPoint. The list also included the following data:
    a. The busiest hour of the day.
    b. The amount of BP's that would be used during that hour.
    c. The number of TouchPoint users.
    d. The time the users would log into the system as it was a system used nationwide.
2. The customer also worked with the business analysts to obtain the busy hour data for each of the BP's which is used for the occurrence factor of the FMEA RPN.
3. When the software was ready; the steps to perform each BP were documented.
4. A data capturing tool, embedded into the FIS banking software, was installed to monitor the amount of data being transferred between client and server. The BP's were performed and the data was stored to the user's hard drive. The following data, which pertains to the severity or process risks, was captured:
    a. The content to message size ratio of the XML data that was transferred.
    b. The number of parsing cycles that took place when the XML was received and at what tier the parsing took place.
    c. The number of XSL transformations.
    d. The XML message size.
    e. The tier at which the data was sorted.
    f. Database interaction. (The number of reads, inserts, and adds to the database).
    g. The number of round trips from the client to the server.
5. As the data capture was taking place, the customer formulated a 'Willingness to Wait' listing. Each BP was put in one of the three groups listed below.
    a. Customer is willing to wait up to 6 months for optimization. (1)
    b. Customer is willing to wait up to 2 months for optimization. (5)
    c. Customer is not willing to wait at all. (9)
6. The level of detection for errors in each BP was also ranked by the performance engineers. Each BP was placed in one of these 3 groups:
    a. Fully covered by a previous performance testing effort. (1)
    b. There was a substantial increase in usage of this BP from previous performance testing. (7)

c. The infrastructure differed from previous performance testing. (8)
d. This BP had never been tested by the performance group. (9)

7. Finally, this data was tabulated in a spreadsheet and given a ranking. The factor of all of the components; severity, occurrence, detection, and willingness to wait.
8. The final step was to create a performance test plan utilizing the top 5 BP's that were surmised from the FMEA process.

The FMEA process ensured the customer and the developers of a strong performance test plan that would exercise the most likely targets for failure. The software was successfully deployed, is currently in production now, and has not had any live performance production issues. Some optimization efforts were finished in the six month time period and some were addressed before the go-live date. Now that the bank has experienced and understands the FMEA process that the SPE department employs they are much more amenable to applying this process again. In fact, the bank has readily embraced the process for our next release of software.

In conclusion, FIS' SPE department has found that using the Pareto Principle through the FMEA process is an effective way to define the parameters of what to test for a successful performance cycle. When applied to real world business scenarios this process has also been a tool to gain credibility with the customer as well as to pave a smoother path for future performance cycles with the customer.

4. How We Test

The following section provides insight to the exact methods our team employs for validating and benchmarking our software performance. We have a rather complex application to test, and these steps provide a glimpse into the way we approach testing this "monster". This new world of SOA – where everything talks to everything – and increasingly diverse architectures provides a world of required learning to be effective. What follows is a nuts and bolts definition of how we do exactly that.

4.1 Software Performance Engineering - Performance Testing

Our team uses a variety of tools during performance testing efforts, but the two tools we rely upon heavily are the industry's 800lbs. gorillas: Mercury Interactive LoadRunner and WinRunner.

LoadRunner is a versatile tool that can be used to drive load across a multitude of systems with a wide range of communication protocols supported. Although the tool can interface with a vast amount of technology and protocols, the Web (HTTP/HTML) protocol is where we focus the majority of our testing. LoadRunner provides an extremely accurate way to simulate extensive amounts of load onto a test system to determine capacity planning and sizing guides for our core software products. It is also used to verify planned capacity compared to expected system load on customized client deployments.

One downside is that LoadRunner scripts can only account for response times from web server receipt to reply. Network transmission time, from the web server back to the client machine across a WAN, and GUI rendering times on the client machine, are not captured in LoadRunner testing. To bridge that gap, our team also uses WinRunner, which we run in conjunction with LoadRunner testing to get an end to end (key to glass) response time for the system under load. WinRunner is often used for automating quality assurance testing scripts in software applications, but for us it is imperative to gather front end GUI response times during performance testing. The combined use of these two tools allows the team to focus on detection and optimization of performance bottlenecks across the entire end to end system – database, network, middle tier, and front end tier.

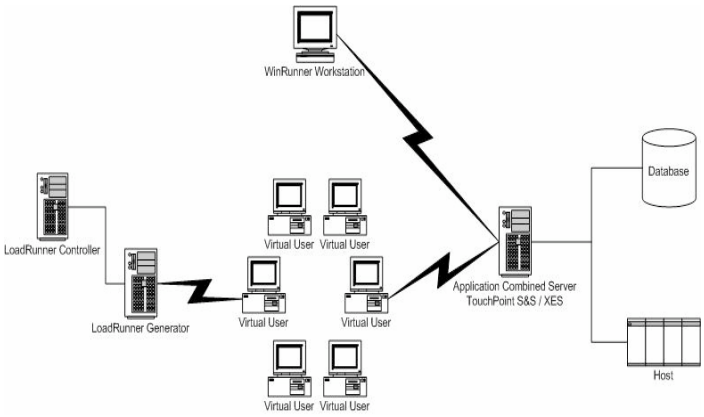

Diagram 1: Traffic flow diagram for WinRunner and LoadRunner against the TouchPoint application

Our standards document outlines each step required to move a performance test script through a full development cycle. The standards document also outlines required entrance criteria that must be met before any accurate performance scripting and testing can begin.

An outline of the basic steps to create LoadRunner and WinRunner scripts are as follows, with detailed descriptions below:

- Functional Walkthrough
- Baseline Recording
- Parameterization / Correlation / Synchronization
- Configuration Audit
- Initial Baseline

4.2 Functional Walkthrough

The areas within the application that have been identified for testing must be defect free. A defect free walkthrough ensures that the team can focus full attention to performance when all pieces of application code are working as expected. The Business Analysts work closely with the SPE team to ensure that the steps defined for a given script accurately reflect common usage within the system. This leads us to the first steps of creating a LoadRunner script - a functional walkthrough of all steps must be concluded before any scripting can take place. If any area is not completely functional, it is necessary work with development to get the system to a point where the script steps can be completed without error. Once the walkthrough is completed, an initial script recording can be performed.

4.3 Baseline Recording of TouchPoint with LoadRunner/WinRunner

The first LoadRunner recording, labeled as a baseline, is saved and used as a backup if the completed script has any issues or is lost. A baseline recording consists of two complete passes through the script. This is necessary for our applications as many transactions are cached within the front end during the first pass. By accounting for cached information, the scripts are a realistic representation of how the system will be used. Not accounting for cached items can greatly affect the performance on the system with unnecessary load and will not represent real world usage of the system. A baseline script contains transaction timers around predetermined steps, these timers are added during the recording process. Once a baseline script has been recorded and backed up, the parameterization phase begins.

The initial WinRunner recording is built upon with the addition of transaction timers, think times, and parameterization of a few key data items. Note that transaction timers in WinRunner should match the timers in the LoadRunner scripts, this allows for easy comparison of WinRunner to LoadRunner times. Once all of those items are taken care of, synchronization points must be added.

4.4 Parameterization, Correlation and Synchronization

Each LoadRunner script must go through this phase to ensure real world use of the system when testing and proper interaction with the application. Parameterization is replacing known data values in the script with variables that come from data files generated from a seeded database. These files are called parameter files, and are generated by the SPE DBA from the TouchPoint database, for each specific test case. Examples of data that should be parameterized for TouchPoint include: logon id, workstation, password, customer name, domain and/or IP address. This step allows adaptability for testing on different environments with different host and database platforms. It also assures real world usage of the system by allowing unique users/workstations to be used for each virtual user. Correlation is used to capture information returned by the application for use later in the script. Correlation is the most important, and complicated part, of scripting the application with LoadRunner. An example of using correlation would be capturing the session id and using it throughout the remainder of the script. Correlation is used to capture information about the user, customer, and the work folder or case for future use in the script. Once all configuration and parameterization is complete, and the script runs successfully with one virtual user, the configuration audit begins.

WinRunner simulates a real user by navigating through the front end of the application. Since our application uses a combination of ActiveX, JavaScript, and an AJAX implementation, a page may appear to have been loaded completely to the browser while data is still being loaded to the screen by services in the background. WinRunner does not possess the technology to determine when a page has finished loading completely in our application. For that reason, synchronization points must be added in conjunction with the transaction timers. Synchronization points may vary from page to page, but often consist of waiting for a page/objects existence, waiting for data to display in a list box, or checking to see if a window exists or has closed.

4.5 Configuration Audit

The configuration audit process verifies that a script can run with multiple virtual users, the system is stable under load, and the number of virtual users assigned to each script are sufficient to hit the throughput goals. We perform a configuration audit on each script individually. Once a script passes successfully, another script is added to the mix until all of the scripts can be run together successfully. The number of virtual users assigned to each script is validated during configuration audit. Each test script has a pre-determined execution time and business process per hour throughput goal. The application execution times must stay constant, and are calculated by measuring the average time a user takes to go through the script in the front end. The number of virtual users needed to hit the throughput goal is calculated before configuration audit, based on the execution time, but that number may need to be adjusted once all of the scripts have been added. System stability is the last piece of the configuration audit. The entire system (network, database, application server) must be stable under load for response times and testing results to be valid. After any instability issues are worked out, the first initial baseline is taken.

4.6 Initial Baseline – The First Benchmark

The initial baseline is the first test run with all of the LoadRunner and WinRunner scripts together. Our goal with the TouchPoint application is to keep the CPU under 80% on the application server and 60% on the database server. This is the first gauge of where the code is before any optimization efforts take place. Quite often the throughput goals are not met, but that is not the expectation of this test. The baseline is a comparison point for all changes to gauge whether a code optimization or configuration change has made an improvement in performance. Once the initial baseline is complete, optimization areas are identified based on the results of the baseline and analysis of the system. Detailed reports are created for the baseline and as optimizations are introduced and tested in the system. These reports detail test setup, changes made from the previous tests, operating system utilization statistics, application response time, memory growth, and process consumption. A spreadsheet is also used to track each test run and is useful for comparing the effects of optimization efforts on the system performance. This method of performance testing, in an iterative fashion, finds the “hotspots” within our applications. We can then focus on the work of optimizing, through software or configuration changes.

5. ITIL – From the Lab to Production

ITIL has developed as a guiding set of principles within the world of operations. Many of these ITIL components are comprised within the SPE realm. In our organization, anything outside of functional defects falls into the category of performance, once in production. To better serve our clients as we move from the lab to the production environment, we have adopted certain ITIL modalities, at least in principle. The following sections provide a glimpse into the basics of ITIL, and how they can be adapted to suit any organization.

5.1 Software Performance Engineering: Performance Optimization and ITIL

Utilizing components of ITIL in performance optimization is a two phased process. Pre-deployment optimization is the proactive phase and the post deployment phase which is the reactive stage.

To achieve the required level of optimization, we use agreed reference points, which are covered in SLM (Service level management) which is within the ITIL Service Delivery volume. Post deployment phase uses procedures in the Service support volume. SLM guidelines are framed to define what to measure, how to measure and the tools to be used to measure it.

5.2 Application performance optimization

Today's applications are highly interconnected and integrate many existing back-end systems and third-party service providers. This type of architecture has created a complex environment for building applications.

Apart from the SPE team, other teams which contribute to performance optimization of an application are developers, application manager, database architect, and specialist's who deal with the hardware and network devices. Each understands the requirements and targets agreed upon by the management in the SLM document.

After the application development process is complete, the SPE team puts a measurement process in place referring to the agreements defined by SLM which are Service-Level Agreements (SLAs), which measure the responsiveness and availability of an application as seen by the end user and Operating Level Agreements (OLAs), which measure the service that back-end systems provide.

The performance testing process follows the ITIL's "Deming wheel" or "Shewhart cycle" commonly referred to as the plan-do-check-act (PDCA) refer Diagram 2.The tests are continued with SLA and OLA as the final goal.

Incident Management and Problem Management components in the Service Support category are used in countering the post-deployment issues.

5.3 ITIL and Performance Optimization

ITIL gives us guidelines which tell us ways to do and how not to do things. There are seven primary components within ITIL, the IT Infrastructure Library, whose main focus is on Service Management (SM/ITSM). The purpose of this is to serve as guidelines for provision and management of effective IT services; classifying ITSM further gives us Service Delivery and Service Management

5.3.1 Service Delivery consists of 5 disciplines.

- Service level management
- Capacity management
- Continuity management
- Availability management
- IT financial management

5.3.2 Service Support has 6 disciplines.

- Configuration management
- Incident management
- Problem management
- Change management
- Service/Help Desk
- Release management

Where does performance and optimization of an application fit into ITIL? Performance optimization lies within the application development cycle, and in ITIL, service management covers the discipline of application management. The Application Services Library provides a framework for structuring application management.

5.4 Guidelines On How to Measure and What to Measure

The performance optimization tools used to accurately gauge performance, as with all the software tools used by the organization, have to be in the organization's CMDB (Configuration Management Database). Configuration management is the implementation of a database that contains details of the organization's elements that are used in the provision and management of its IT services. This also is known as an asset register.

Now, that we have the tool, the basis of usage is guided by the SLA. The application should comply with the service level agreements (SLA) before it passes out of the hands of the performance analysis team. An SLA is framed and finalized after discussions with the intended client, and it shall contain the kind of results expected in specific terms and the level of severity.

The performance team can raise the bar to eliminate unexpected results since ITIL consists of guidelines and are not fixed rules to abide by.

Diagram 2: ITIL in Action

ITIL includes planning what to test, performing the actual test, checking the results, acting on these results and repeating these steps until the SLA is met.

There are many more guidelines in ITIL which can be customized to an organization's individual environment. The above is only a mere glimpse of the wider topic.

6. The Broken Software Model

ITIL provides a generic methodology for practicing SPE, among other important disciplines, within a production environment. But is the issue much deeper than this? Do we have a fundamental misconception of how software should be developed and implemented? The following first person narrative seeks to close the gap in the increasing identity crisis within the current development cycle. Man has been developing software for many years now, but performance issues only continue to increase as the years pass.

6.1 Is SPE Only Performance?

For the past 20 years, Dr. Connie U. Smith [SMIT02] has evangelized that developers must "build performance into systems rather than try to add it later." With fourteen years of systems integration behind me – and nine of those years as a practitioner of SPE - my life has been filled with fire drills, project rescues, denial about the importance of SPE, calls for performance early and often, and migration from focusing on response time and throughput back to systems integration. What is the problem with SPE?

The assertion here is that historically, software companies have focused on building functionality. The left side of diagram 3 depicts the typical Software Engineering focus – a functional development environment with a stack of functional requirements, and a basic network and server infrastructure.

To software company executives, functionality is what customers purchase, and therefore rationalizes the investment in a software sales team, product management, business analysts, functional developers and quality assurance. The right side of the picture depicts the world of SPE: a mesh of internal and external applications, Local Area Networks (LAN) and Wide Area Networks (WAN), distributed servers and heterogeneous customer behavior. Customers do not explicitly purchase the right side of diagram 3; most casually assume it comes with the software. Because of the lack of an explicit and lucid demand from customers for systems integration from software vendors, most software executives under-invest in the right side of diagram 3. Veteran IT customers, who have been past victims of vendor performance engineering ineptness, attempt to protect themselves with late stage performance testing, and post implementation software acceleration solutions and monitoring products.

More than 50 software development projects and implementations have convinced me that SPE is more than modeling, load testing, response times and capacity planning. SPE is the integration arm of software development. It is where all things come together and effective SPE teams require knowledge of each and every element of the greater system, plus the specific methodologies of SPE (queuing theory, software modeling, anti-patterns, statistics, workload characterization, load testing and capacity planning).

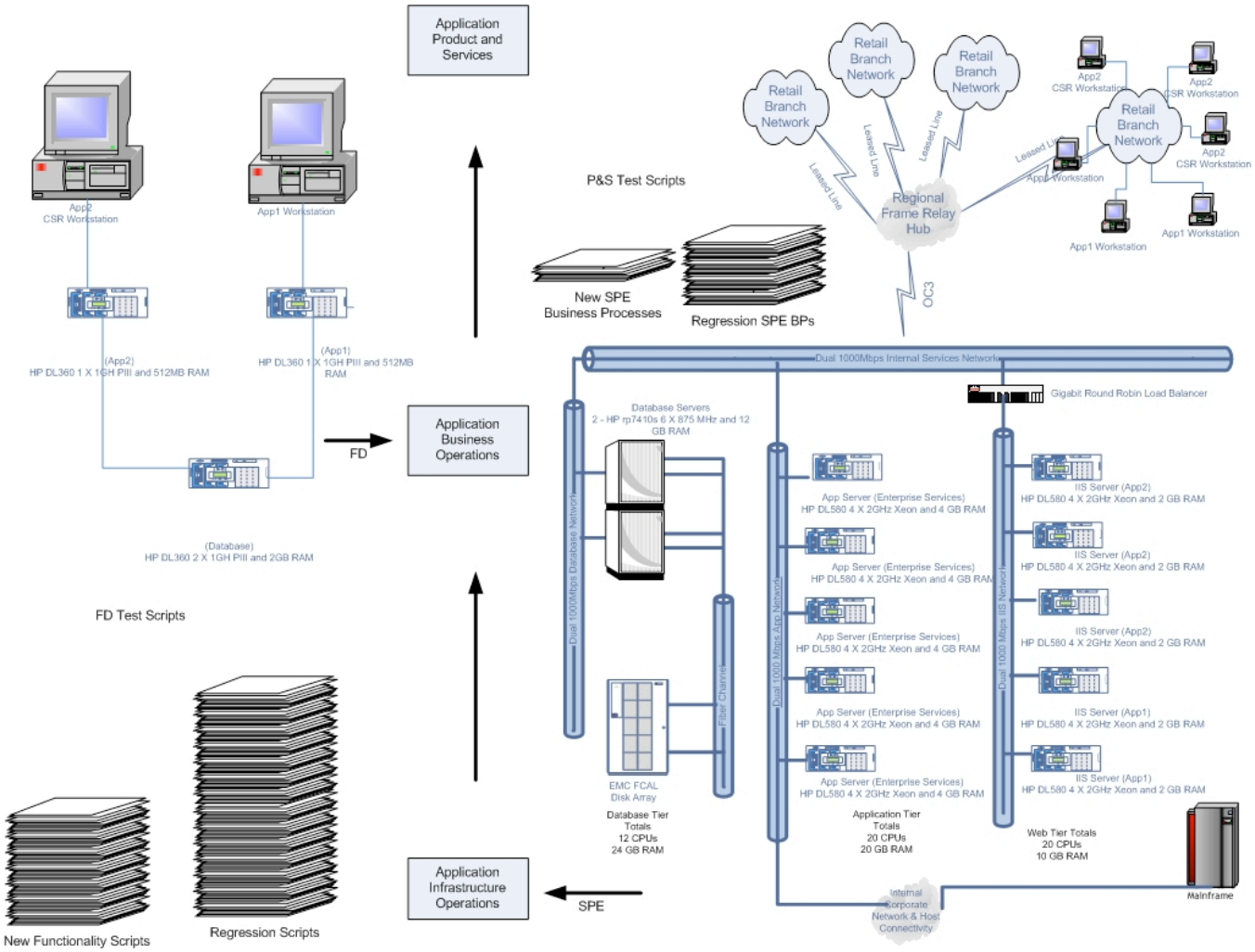

Diagram 3: Software Performance Engineering vs. Functional Development

The experience that I have had with SPE has forced me to question the reasons given for SPE's continued struggle for acceptance within the software development community. Is it really because of the "lack of scientific principles and models" [MENA02]? Is a business case for including performance in the software development process [SMITH04] really required? The experience of the author has stirred the following question: is the real issue with SPE's acceptance that fact that most software companies are really in the business of building functionality and not in the business of systems integration? The first prerequisite to an answer requires the audience to accept the assertions that the left side of diagram 3 drives software companies? The second prerequisite requires acceptance that the complexity of the right side of diagram 3 reflects the day-to-day world of SPE? Once the preceding two principals are accepted then the next question is "Does SPE suffer from an identify crisis, a poor toolset, or a lack of commitment from company executives?

This author suggests that all three issues are holding SPE back from becoming a mainstream practice in software companies, however there is a critical order of priority. This author's repertoire of projects has left me with the conclusion that the true identity of SPE is not well understood. Most people that I have encountered in the field of software development have limited software performance engineering to testing (specially load and stress testing), tightly associating the work of SPE engineers to that of functional testers (and widely referring to SPE groups as quality assurance). Most in the software community do not understand the amount of systems integration that SPE encompasses.

It is the author's experience that the reason for the easy association with SPE and QA has been the focus of most projects, and the associated companies, on functionality and not systems integration. The following is what the author has observed as a common evolution of SPE within companies.

- Projects blew up because of integration issues that manifested as slow response times or system unavailability
- The companies and clients of these projects concluded that an SPE team was required – to them, they needed someone to run a load test
- This new group required a home
- Most of these companies and customers only had functionality departments (product management, business analysis, application development, quality assurance and a systems group whose role was to supply servers for functional development environments).
- Since the decision makers in these companies associated SPE with testing, the logical home for SPE to them was QA.
- The expected output of SPE was test results, however when the systems did not perform in production, SPE was held accountable.
- To survive, SPE was forced to become more than testers, quickly evolving into a rag tag team of system integrators:
    - capable of profiling, instrumenting and changing application code
    - investigating and resolving network bottlenecks
    - analyzing and rewriting long running database queries
    - optimally configuring operations systems
    - wielding best practice knowledge of multiple COTS software packages
    - setting up complex simulations to analyze the impact of changes to any tier of the system

What could a have been done differently? If these software companies had already invested in systems integration in addition to functionality development, then the seeds of SPE would have existed -- i.e. systems architecture and engineering staff, systems engineering methodologies, and systems integration environments that emulate production. When each of these organizations realized that SPE was required to successfully deliver functionality to customers, it would have only been a matter of adopting the formal tools, methods and processes of SPE into their existing system integration effort in order to properly formalize the existing of a SPE team.

Once SPE has a proper day-to-day identity of systems integration, Smith's business case for SPE takes on a new level of weight, because the fixing performance issues later vs. earlier become the cost of designing, developing and deploying poor systems – the very thing that was purchased. The SPE ROI's becomes the ROI of proper systems integration. Which leads us back the assertion of Menascé and expands on his concern about scientific principals and models; in order to do a better job at systems integration, better SPE tools are required.

7. The Sum Total

With many years experience in the SPE field, as it continues to grow from seed within the distributed computing world, our team has witnessed a multitude of changes. We have provided a glimpse into our methodology, what we have seen, and where we are going. Looking forward, ITIL provides a sense of order in the chaos of the production world for our application. Our scripting, testing and optimization processes continue to grow with our experience, and scars from continuous SPE warfare form anew. The end game for us consists in a world where the education has taken place, and SPE becomes part of the tree that grows from the software engineering seed.

8. References

[RATH02] Six Sigma Pocket Guide, Rath & Strong, 2002, pg 26.

[SMITH01] Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software (1st Edition), Connie Smith and Lloyd Williams, 2001

[MENA03] Performance by Design: Computer Capacity Planning by Example, Daniel Menasce, Virgilio A. F. Almeida and Lawrence W. Dowdy, 2003

[MOBLEY05] SFMEA; Applying a Six Sigma Method to Software Performance Engineering, Kevin Mobley, 31st Annual International Conference of The Computer Measurement Group, Inc., December 4 - 2005

[SMIT02] Smith, C. U., notes from Software Performance Engineering seminar held April 28 through May 2, 2002, in Santa Fe, New Mexico, page 1-4.

[MENA02] "Software, Performance, or Engineering?", Daniel A. Menascé , Workshop on Software Performance Engineering, 2002

[SMITH04] "Making the Business Case for Software Performance Engineering", Connie Smith and Lloyd Williams, 30th Annual International Conference of The Computer Measurement Group, Inc., December 2004