My Great Performance Testing Project

Tom Wilson
Lockheed Martin

1 Introduction

I like many of Apple's products (e.g. iMovie, iDVD), but am I really going to use the default filename “My Great Movie" or “My Great DVD"? Lacking a title for this paper, I couldn't get this file naming convention out of my mind.

It has been a while since I have written something for *MeasureIT*, even though I have had a lot to write about. I have spent the last two years working on a project where my team was performance testing the Autonomic Logistics Information System (ALIS) for the Joint Strike Fighter (JSF). While I was pleased to be invited to speak at the St. Louis Regional CMG in 2012, getting something through the review system where I work was too much trouble. My first obstacle was convincing some reviewers that, while PowerPoint was the medium that the presentation template is defined in, it was not the critical factor in approving material for public release. That was just the first stumbling block; someone else found a better reason to suppress the material that actually had to do with the technical content. While you will find more information on the World Wide Web about JSF and ALIS than I would ever write about, I will keep my discussions away from the program's subject matter and stick to the interesting stuff!

I spent five years on the ALIS program in the recent past with a brief assignment on another project that I frequently wrote about in as vague a way as I could find. Then I came back for another three years with the last two focused on my great performance testing project.

2 My Environment

My recent paper on performance perspectives ([WIL11]) is so appropriate for this discussion. I work in a Department of Defense (DoD) environment and not a commercial one; business competition has a different meaning here. The government often cannot take its business elsewhere. I do not work in an IT organization. There is one and IT work is being done, but little of it is focused on performance. I am in an engineering organization, but I would not claim to be an engineer. I have computer science degrees and am more interested in studying performance than in engineering solutions that do not perform. We definitely do not do performance engineering at my location. That would entail performance being a focus from the beginning of a program or project until the end, with a team that specializes in that area. I know we do not have such people and, therefore, no such organization.

We do have an engineering organization that specializes mostly in the integration of solutions and sometimes in the development of those solutions. For example, ALIS is mostly a collection of commercial products with the addition of a few custom products, all glued together with commercial and custom interface/communication products. The result is one monstrous management nightmare. But do not get me wrong: we produce useful things that keep the corporation alive. But the point that I am driving towards is that my project did not align with the typical processes and organizations that exist where I work.

3 Engineering Overhead

Typical engineering development and integration start with requirements. These define what is being engineered and define tests to confirm that the requirements have been met. People estimate work and develop a schedule. Parts of the team might handle stages of the project such as requirements analysis, design, implementation, integration, and testing. Usually there are clean boundaries where the hand-over of artifacts are well defined. While the concept of agile confuses the whole idea of knowing what you are getting in advance, there is still some connection between the requirements at the front and the testing at the rear.

Endless processes control how the work is executed and statused. My program has so much process, it is amazing that much work is accomplished. It is often apparent why processes exist, but they never seem to be efficient (in spite of the repeated refinement).

So, my project came along where there were no (performance) requirements to start with. The project's statement of work simply said that the system under test should be stressed to the breaking point to understand what loads cause this condition and what the effects are when such points are reached. Our first task was to better define the scope of work given the budget and time frame. This was actually a good thing, although we were likely to expend quite a bit of both trying to get agreement on our approach. One agreement we made was that we would only slightly exceed realistic workloads in our attempt to overload the system. If overload was not achieved, then the conclusion was that the system is sufficient in performance.

One of the proposed ideas was that we would allow sensitivity analyses to be performed where we would allow modifications to be done on the workload or the configuration of the system under test in order to learn more. That will have important meaning in a moment.

Next, we encountered inappropriate development processes. First, after we have developed the solution and integrated it, we would have “dry runs" before the formal test period. Of course, these dry runs would just be running the same tests as the formal test period, so that we were confident that the tests were

working. Of course, before entering dry runs, we would have to ensure that these tests were working before exiting integration. The result: several months learning nothing while only demonstrating that we could repeat the tests.

After convincing the stakeholders that dry runs were a waste of schedule and budget, we removed it. We replaced it with “not dry runs" and eventually renamed this “stability runs" because the former name was so stupid. By the way, “stability runs" is the same as “dry runs" in practice. Why is this so? The formal test period was going to allow the stakeholders to witness the test. Granted, there was nothing to see, but they wanted to make sure that we did what we said we would. Also, *Quality Assurance* (QA) will be verifying that you followed a documented engineering process with a *Configuration Managed* (CM'd) system. During the stability run period, QA did the same thing in the absence of the stakeholders.

Having a CM'd system had some benefit (i.e., knowing what the configuration of the system is that you're testing), but that incurred some overbearing processes to review and implement changes to the system that was not ready to be CM'd yet.

Now, consider implementing the idea of sensitivity analysis. If we wanted to change the system, the changes had to be reviewed. They have to be CM'd and QA audited. While it sounded feasible to do this, it required dedicated resources to do it in a timely manner. In my environment, it was not happening!

I was told repeatedly that schedule and cost were paramount on the project. My response always was that the technical content was the priority. I understand that the schedule and cost bound the work, but delivering a solution that fell short of the objective was not success.

4 Guaranteed Success . . . or Guaranteed Failure?

As I viewed this project, I felt it could only fail in one way -- fail to implement the test. We structured a sequence of loads to be heavier with the understanding that if a breaking point was reached, heavier loads need not be run. Whatever the performance was, it was acceptable because all that we were doing was reporting what the performance was. The stakeholders had the responsibility of understanding the measured performance and judging it in light of the unspecified needs (i.e., no requirements). So, as long as the tests ran and we reported the performance, we could not fail.

However, I said that the stakeholders needed to understand the measured performance. This meant that they needed to understand (1) measurement, (2) operational analysis, (3) statistics, (4) workloads, (5) performance testing objectives as a minimum. The case of understanding performance testing objectives highlighted the difficulty. Performance testing did not need to exercise

all functionality in a system. It needed to highlight the resource usage of common functionality. That right there was a show stopper. The stakeholder's jobs on the main program were typically to understand how the functionality met the functional requirements of the program and how it enabled the user to do his job. These were smart, experienced, and responsible individuals who were just not used to the performance domain. They did understand when a delivered system was not performing acceptably when put into operations.

So, we were going to write a report with lots of numbers and graphs to tell a story about the performance of the system. If the material could not be received, then how could there be success? Also, there was a preconception that if there was something that the project did not do, then the project was a waste of time.

5 What Is to Come

So, now that I have gotten all of that business out of the way, we can look forward to the interesting material that has nothing to do with the project: measurement concepts, workload modeling, operational analysis, data presentation techniques, etc. We will look at integrating measurement data from several sources, automating reports with quick turn around when you are running multiple tests each day, and ways to develop multiple-run reports in which to compare results.

Bibliography

[WIL11] Tom Wilson. “What Is Performance? Part 1: Perspectives". CMG MeasureIT, Issue 8, 2011.