WOPR3
Experience Report Notes
by Dawn Haynes

TOPIC:
Engaging the development team when running first performance tests.

SCENARIO:
Large software company releasing new corporate Web site. Performance metrics were required before going live with the new system.

DEVELOPMENT TEAM:
80% of the development was outsourced to a single Web development company. The outsourced team was co-located at the corporation's local office, and worked closely with their internal Web team (requirements managers, Web developers and testers). The outsourced team reported to a project manager who in turn directly reported to the corporation's Web site manager.

TECHNOLOGIES:
Original Web site: Primarily HTML and .ASP.
New Web site: HTML and.JSP pages, EJBs and servlets. Believe the database was DB2.

GOALS OF PERFORMANCE TESTING:
To ensure that new Web site's response times for target transactions were within allowable ranges (based on current Web site experience).

User load goal: 100
User breakdown and transactions:

- (40%) Browse site (hit 5 pages)
- (10%) Download a whitepaper (requires log in)
- (20%) Request information via a Web form
- (25%) Search
- (5%) Buy product (requires log in)

TEST TOOL:
Rational PerformanceStudio.

PERSONAL INVOLVEMENT:
As the go-live date became imminent, the team scrambled to gather some cursory performance metrics on the new system. But they were not successful with their 100-user test. As a matter of fact, they were unable to get 10 users running successfully. After several weeks of failed testing, they determined their problems were tool related and at three working days prior to go live, they asked for a tool expert to assist. I joined the party. ;-)

SPECIAL CONSIDERATION – LOGISTICS:
The outsourced team was crammed into small spaces so they could be co-located at the corporation's site. There were two to three people per cube or office, and the less lucky were jammed into conference rooms. The testing "lab" (about 4 feet of a table) was in a conference room with six developers from the outsourced team. The staging and live servers were located in another building at the corporation's site.

MY CONTACTS:
My initial briefing came from the corporation's Web site manager who laid out the objectives and timeframes for testing. After this meeting, only status reports and critical needs were elevated to this person.

My primary contact was the outsourced team's project manager who was responsible for the performance testing. I worked with the project manager "knee-to-knee" until the system went live.

INITIAL ASSESSMENT & STRATEGY:
The project manager had a lot of knowledge about the new Web site and limited knowledge about the existing Web site. However, the corporation's Web team provided a very detailed analysis of the target transactions and user rates. This enabled the project manager to develop solid load tests and design an accurate workload. The major stumbling block with the tool was that for any size test, only six concurrent users would run. I trusted the players and bought into the idea that the tool was responsible for the limited load and planned to focus on that instead of doing a broader assessment (given the time constraints).

DAY ONE – WEDNESDAY'S CHILD IS FULL OF WOE:
I bought into the panic of the situation and began to apply each and every tool tweak I could think of that would allow more than six users to run concurrently. By the end of the day one of my involvement, we had re-recorded tests, re-created the workload, tweaked the ramp-up scenario for starting the users in the test, tried other workload designs, and ran tests with full logging and debugging options. Analyzing the detailed log data resulted in the discovery that the failed users were all dying in the same place in the test.

DAY TWO – THURSDAY'S CHILD HAS FAR TO GO:
On day two, frustration set in. Time was running out and the project manager was starting to get desperate. After running the first failed test of the day, he asked the developers in the room if anyone could think of why our test wouldn't run with more than six concurrent users. We explained to the team that we were unable to run more than six concurrent users doing any activity on the site. One of the servlet developers sat bolt upright, turned to us and said "six?"

After about ten minutes of quiet muttering, furious pounding on the keyboard, and several trips in and out of the "test lab" to visit other developers, he said "try it now." The project manager pointedly asked what had been changed. The developer said that he found a hard-coded limit in the session manager servlet,

and he pushed a change to the staging server. This servlet was the gatekeeper for all users accessing the site, whether they actually logged into the site or not. Hmpf.

We ran the same test that had always failed … and we got a different result! We now started failing at 10 concurrent users.

BEEP … BEEP … BEEP … THE TRUCK IS REVERSING:
After this minor victory, the project manager was not about to waste any time. He assembled the entire dev team into the “test lab” and asked everyone if they could think of a reason we couldn’t run 10 concurrent users. Two folks thought it might be a database configuration or licensing issue. Turns out it was. About 30 minutes later, the changes were made on the staging servers and we re-ran the test. And there was another short-lived victory! We got past the 10-user load level this time, but then started to get users randomly failing during the ramp up as the load approached 50 users. They were victims of timeouts on HTTP requests.

I then enquired about the network configuration of the room. I had not previously thought to check if it would support our testing activity … and we inadvertently proved it would not. When we unearthed the hub from behind piles of books and binders and boxes, we could observe the lights indicating errors, collisions, or dropped packets. We got the Web site manager to procure an IT resource for us to analyze the situation and provide a solution. Turns out all the network traffic in the room was being routed through one hub and one network connection. The IT department worked late to outfit us with a higher-speed hub and network connection, and isolated us from the rest of the nodes in the room.

DAY 3 – GETTING TO THE HEART OF THE MATTER:
Finally, on the morning of day three, we were able to run tests with 100 users and get somewhat trustworthy response data. Many of the target transactions were performing VERY poorly. And even though the testing environment wasn’t perfect, the response times were so bad … it was clear there were problems to fix.

With only a few hours left to the workweek, and a glorious weekend inside the cramped office to look forward to, the project manager triaged the performance problems with the entire team (corporate staff and outsourced folks, developers and managers too – no one was left out). They worked through the issues one by one and developed an action plan together. Then they set priorities to each action item and resolved to push changes to the staging server and retest at 3pm.

At 4pm the team reassembled to evaluate the results. There was still a major performance bottleneck outstanding – the Web site log in operation was taking too long. Since the Web site log in was a new feature that was important to the rollout, they resolved to find a solution. By the end of the day, load balancing

hardware was ordered to be delivered and installed on Saturday. Holy pizza Batman!

Due to a previous weekend engagement, I could not join the weekend fun. However, I did support the team via phone several times.

On Monday the new site went live. There will still some outstanding performance issues, but none of them severe. The team continued to work on the issues with their “testing and triage” method throughout the rest of the project.

TAKEAWAYS
This whole experience, while frustrating at first, was responsible for changing the way I wanted to approach testing projects.

First and foremost, I was absolutely dumbstruck by the developer who kept repeating the number six to himself until he fixed our first problem. It made me want to ask questions of future development project teams like:

“Can you think of any reason why we wouldn’t be able to perform this transaction with 50 (relatively) concurrent users?”

“What will happen if I attempt to log in 20 users at a time?”

“I’m planning on running 1000 users performing these 5 tasks randomly. Can anyone think of a configuration or architectural constraint that would not allow this to happen?”

And so on.

Secondly, I was supremely impressed with the collaboration efforts of the entire team when asked to triage and suggest fixes for each problem in priority order. This was a just-in-time risk/priority-based method of performance tuning. And given the time constraints, it’s unlikely another method would have been successful.

It’s now my goal to try and involve architects, developers, DBAs, IT folks, and anyone else I can wrangle into my test planning phase, as well as ask them to participate in the first test runs with the performance tool.