A Test a Day Keeps the Boss Away

When polled, most developers say they would prefer to test themselves if it were easy enough. This is now becoming increasingly possible, with the introduction of script-free test tools that are easier to use and faster to implement.

Donald Doane is President and CEO of OpenDemand Systems

Corporate developers building enterprise Web applications and services have long shunned introducing test tools into the overall software development process. If such tools were used at all, they were relegated to the quality assurance process at the end of the line. This is largely because legacy test tools, spawned in the age of client/server computing, from the likes of Mercury Interactive, Segue Software, Compuware and Rational, carry significant overhead, requiring excessive scripting and overly complicated data analysis; after all most developers have enough on their plate without having to code and debug test scripts too.

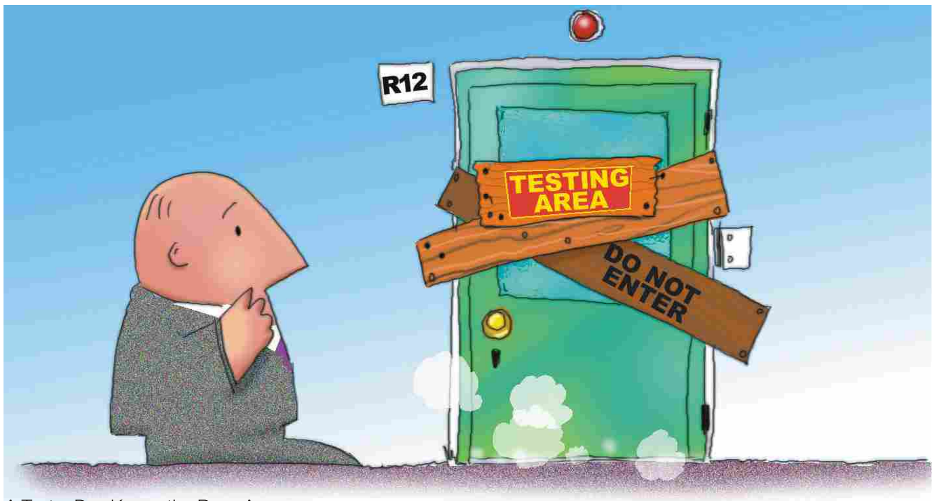
These legacy tools have been used in some cases well, in some cases half-heartedly, and in many other cases were relegated to shelf ware. Though such tools have long had a place in the pre-deployment process, due to the high cost of implementation their practical application at the early stages of development has never quite caught on, but all of that is changing with the introduction of script-free test tools that are much easier to use and faster to implement. In fact, when polled most developers say they would prefer to test themselves if it were easy enough.

As more applications and services move to the Web it becomes even more important to be able to gauge the performance and load capability of an application. The increasing complexity of deploying new technologies, combined with the growing expectations of business users, is once again forcing development managers to put the squeeze on their charges to make testing a key piece of each phase of the application life cycle. As a result, a growing number of development organizations are seeking out next generation automated test tools that facilitate iterative testing throughout the development lifecycle without the added overhead of scripting and complex data analysis.

Why Test?

Performance testing is a part of the overall quality assurance policy that software architects who are in the know institute to protect their firm's IT investment. It ensures that after all of the time, effort and money spent to implement a business critical application, IT actually delivers a final result that meets the business users' expectations. To most of us this probably sounds like good common sense; after all how many of us would keep spending money on goods or services that were slow, error-prone or worst of all not available when we need them most. Yet it is amazing how many senior-level managers view such testing as a less than critical budgetary expense, when in fact performance testing stands as the true ‘unsung hero’ behind successful implementations that no application development lifecycle should be without. For those of us who have been paged in the wee hours of the night because a system unexpectedly crashed, or even worse, called into an emergency meeting to face the public scrutiny of management and the finger pointing of our peers, we already know that testing prior to deployment provides a peace of mind that is, well, priceless.

How else would you know how many users your Web applications can adequately support and still meet your business' expected performance and availability requirements, or accurately predict how much to invest in IT infra-structure in order to support both short and long term customer growth goals for your organisation? The problem is that these types of scalability questions cannot be adequately addressed by simply buying more hardware than you think you need or taking a cookie cutter approach to your deployments, based on vendor specifications. There are simply just too many variables involved. Consider the multitude of ways real users can push your Web application and supporting infra-structure to its limits. The bottom line is that the only way to realistically know the performance and load capabilities of your Web system prior to going live is to closely approximate the live environment, which means actually simulating this real-world activity to identify potential problems before your users do.



A Test a Day Keeps the Boss Away

By simulating real-world transactions generated by hundreds or even thousands of users, you can test your web server performance under normal and stress conditions to ensure that critical information and services are available at speeds your end-users expect. This will enable you to maximize uptime, performance, return on investment, value and the company's long-term success, while minimizing efforts and cost.

Integrating Testing into Your Development Lifecycle

The fact is that over 70 per cent of performance bottlenecks are code-related. So catching problems in the early stages of development before your users find them in production can pay big dividends.

The benefits are fairly obvious, including shorter development cycles due to less code re-work and debugging as well as improved quality leading to increased customer satisfaction.

The challenge for most organizations is finding the time and resources to fit testing into an already overly aggressive project schedule. Experience has shown that that the real key to integrating testing into your development lifecycle is selecting a test tool that minimizes your time and resource requirements. That means the tool you select needs to be easy to learn and use, otherwise there's a good chance your team won't use it and it will become yet another piece of shelf ware.

Avoid tools that call for learning proprietary scripting languages. The perceived power that scripting offers is often times outweighed by the overhead it introduces when you consider that new changes to your application could also require a corresponding change in your test script. This ripple effect means double duty coding and debugging for your development team and is a real deterrent to getting an organisation to adopt an iterative testing model; especially if you're doing fairly frequent product releases. Script-based tools can also increase your development costs because their use is limited to the more technical members of your team, whose time is usually already at a premium.

Keep in mind that testing is a team sport. With today's complex, multi-tier Web environments, troubleshooting performance bottlenecks often means enlisting the help of other developers and administrators to pin-point problems. That means you'll need a tool that allows you to readily share test information with your peers, possibly across departmental lines and sometimes-geographic boundaries. Having a tool that offers browser-based access along with a flexible licensing model allows you to easily centralize your test data and share a common testing resource across multiple teams and office locations.

Finally, make sure the results of your tests are worth the effort. If the reports provided by the tool don't provide sufficient detail, are difficult to understand or worst of all inaccurate, you'll have a hard time convincing anyone that the process is worth repeating. Steer away from tools that are based on immature or proprietary frameworks, as the results are often unverified, leaving you at risk to unreliable or misleading data. Look for tools with technology based on open industry standards, as they are likely to be tried, proven and accepted by the IT community at large, ensuring both accuracy and interoperability with your existing systems. Don't cut corners by putting in place a tool that's inadequate for the job, and then make the mistake of deploying your application with a false sense of confidence. After all, your job may one day depend on it.

To conclude, now that the Web has become a natural extension of how organisations do business, the real focus from both an IT and business perspective must be on how the entire system performs. This takes all aspects of the system into consideration, including end-user access, application functionality, data management, hardware, networks, and service providers. Ultimately, developers must take more care than ever before to ensure the quality of the applications they implement because the costs of errors in a highly automated and integrated environment can be devastating ■

ddoane@opendemand.com

Subscribe to
Professional Tester Magazine