Rethinking software testing
David Rubinstein
Software Development Times; Oct 1, 2003; 87; ABI/INFORM Trade & Industry
pg. 29

Rethinking Software Testing

As organizations test for defects earlier in life cycle, developers are being trained to get involved

BY DAVID RUBINSTEIN

Writing tests for code in development can be a difficult process, dotted with potential pitfalls at almost every turn. The complexity of .NET and Java technologies, the use of Web services written outside the organization, language gaps between testers and developers as well as developers and business requirements writers, questions of source code versioning and even of when to test, all conspire to work against quality software.

Many development shops find testing to be such a problem that they put it off until the end, and often end up reducing the testing time to almost nothing when projects run longer than they should, according to vendors and development managers. The result—in far too many cases, they say—is software that doesn't meet functional requirements, code that is loaded with bugs that developers find next to impossible to recreate, and, at the bottom line, a hit to a company's profits.

In the early 1990s, testing tools vendors thought they had the problem licked when they came out with automated testing tools that could run unit and functional tests at regular times. But according to Dave Locke, program director in the Rational software division of IBM Corp., vendors pushing automated testing tools "shot ourselves in the foot a bit. There were promises of point-and-click ease and 'It'll be amazing.' It wasn't."

The director of quality assurance at a major cruise line company, who did not wish to be identified for this article, said, "We're not a shining example of the benefits of automated QA. Part of it is skill set. Our existing testing people are not strong in automated testing. And I'm still not convinced of the value of automated testing for a Web site or things that change quickly."

Automated testing does have its benefits, others argue. Regular builds and testing can help improve the development process, they say. "If you're building every night and running tests, you find out right away if the software does what it is supposed to," said Robert Leahey, director of developer relations for test tools vendor AutomatedQA Corp. "There's no 'I'll take care of that in a few weeks.' It's a dramatic change in the way people code."

PROCESS MAKES PERFECT

Indeed, many advocates of strong development processes believe testing earlier in the cycle is critical to creating high-quality software. One of the key principles of the Extreme Programming methodology, as detailed in the book "Extreme Programming Explained" by Kent Beck, for example, is that it relies "on automated tests written by programmers and customers to monitor the progress of development, to allow the system to evolve, and to catch defects early."

While vendors and industry experts maintain that more testing is being moved up in the development cycle, at least one development organization is going in the other direction. "We're moving away from writing [code] and testing at the same time," said Bob Armstrong, director of Internet and Information Systems at Delaware North Co., a holding company for several subsidiaries involved in the hospitality and food services industries. "If I'm doing QA and development on the same servers, it's hard to know where you're at. You can't do consistent regression testing if things are changing midstream."

The company's policy is to maintain separate development and QA areas, and it has been able to reduce the costs of hardware and software by using virtualization software from VMware Inc. to "clone" the development environment for testers. Armstrong said this reduces DLL conflicts and versioning problems.

There is wide agreement, though, that catching defects earlier in the development cycle can save countless hours of development time, lower the cost of producing the software and reduce time-to-market. Some say it is the business side of a company that's driving the push toward software testing.

"It's not so much about the code as it is about the business process," said Lori Gipp, vice president of marketing at automated testing tool vendor Solstice Software Inc., claiming that companies are spending more of their development budgets on integration projects rather than creating new code. "Very little is about changing code; it's putting pieces together," she said. "This is a testing exercise."

The Y2K problem at the turn of the century put more eyes on the problem, Gipp suggested. "Post-Y2K, more business people got involved in validating and testing systems. It brought a focus on QA earlier, and the business influence [of a company] has a lot to do with where they sit in the spectrum" of testing.

Joe Oddo, Solstice president and

▸ continued on page 30

TESTING

◂ continued from page 29

CEO, recalled that the stock markets almost melted down due to volume in 1987, and said the financial industry no longer could accept assurances that systems would not fail. "They had to be rigorously tested on a regular basis. In the financial services sector, quality of service is the only way to differentiate. Business people cannot accept long cycles" to deployment.

Businesses also are more concerned about software security than they have been in the past, as a growing number of business transactions are completed over the Internet.

Indeed, building security right into an application by reducing its vulnerability to outside attacks is becoming more of a developer issue as well, according to Diane Fraiman, vice president of marketing at security testing software provider Sanctum Inc. "Every single customer has pounded the table and said we have prioritized training our development organizations in producing quality software. That means secure. Security testing is being driven into development."

DOWN TO DEVELOPERS

To find and fix errors as early as possible in the life cycles, developers need to be on board for testing. "Foremost, you'll see the growth of unit testing with developers," AutomatedQA's Leahey said. "They should have been doing it all along. Far too few do it now. The vast majority of people who do it are in an Extreme Programming mindset. Unit testing is a part of their methodology."

Developers and testers will be able to come together only when tools are created to bridge their worlds, according to Compuware Corp.'s Peter Varhol, product manager for the DevPartner development and debugging tool.

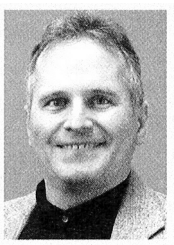
"Developers absolutely hate having a defect reported by testers and having to spend hours trying to recreate it because the language [of testers] is ambiguous. It's very frustrating," Varhol said. For developers to successfully embrace doing testing earlier, he believes companies "must make it as natural a part of the development process as possible. Asking developers to spend two weeks debugging is like asking them to spend two weeks in a dentist's chair to get all their teeth pulled."

The key, according to Varhol, is to get developers to accept doing static source-code analysis or unit tests before moving on to the next phase of coding.

This marks quite a cultural shift for developers, acknowledged IBM's Locke. "The issue is...coding seems to be the 'productive' part of the application. Developers think, 'If we're coding, we must be getting somewhere.'" Also, he said, developers tend to believe they haven't written any bugs. Locke said companies need to do testing at the requirements level and during development, and said reuse of software design patterns that are known to be good can

▸ continued on page 31

DEBUGGING MACHINE-GENERATED CODE CAN BE TRICKIER THAN IT APPEARS



These new errors won't be evident until load testing, says Compuware's Varhol.

With the introduction by Microsoft of the .NET Framework in early 2002, and the continued evolution of enterprise Java, came something that all developers thought was a huge benefit—automated memory management. But the benefits of automated code generation for what vendors call "all the housekeeping and plumbing" often can result in defects that are more difficult to find for developers and QA testers new to these technologies.

Prior to the use of these development and deployment platforms, programmers manually manipulated the memory of a system to implement the functionality of a program, according to Peter Varhol, product manager for the DevPartner development and debugging tool at Compuware Corp. As these new managed runtime platforms arrived, such things as pointer errors and all the old types of memory errors went away, he said.

The downside, of course, is that new types of errors are cropping up that developers and QA people are not experienced in detecting, resulting in the delay of finding these errors until later in the life cycle, resulting in higher costs to correct them and, often, delays in getting the application into production. "It's not like they're working with C++ code on Windows or AIX and manipulating memory allocation," he said. "The platforms are doing all the housekeeping.

"When you are writing code that's correct from the standpoint of a language specification, you could be creating large numbers of temporary objects that increase the memory footprint," he continued. The platform, he said, "actually keeps objects around too long, making [the application] less scalable."

Developers, he said, more likely than not will miss these types of errors in functional testing; they will become apparent only in load testing. And, in the case of proprietary platforms, developers and testers might not even be able to correct the problem, as they in many cases do not have access to the underlying source code.

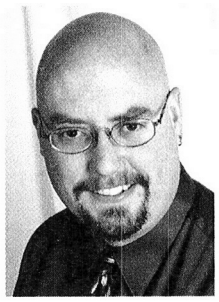
—David Rubinstein

TESTING

« continued from page 30

reduce costs of producing software.

One way to ease the communication between testers and developers is to put a tool in front of them that uses a common language. "Wouldn't it be more effective for QA that the same level of information gathered during development be recorded during the testing phase?" asked Compuware's Mark Eshelby, product manager for QACenter, an automated testing tool suite. "You could provide a detailed level of information for the developer along with the defect. Rather than running the whole regression test again, you can see the methods called and the response times. Communicating both ways is key."



More developers should be doing unit testing, says AutomatedQA's Leahey.

Eshelby indicated that Compuware will base the next iteration of its DevPartner tool on Microsoft's Visual Basic for Applications, a simple "standard" programming language that testers can benefit from as well.

DON'T LEAVE IT TILL THE END

But what happens when the project that had to be delivered in six months is not yet complete with only weeks remaining? Testing gets shoved aside, according to the vendors and developers alike.

"Most people believe testing is important, but the schedule is out of control so what do you cut?" asked IBM's Locke. "Not the features. That's the whole point of the project. So they cut testing."

Eshelby said companies can reduce the risks of failure with some test planning, which he defined as identifying the highest areas of risk and testing those when the time gets tight. But this risk-based testing does not preclude running automated tests, he said. "Automation still is a great part if you have a clear understanding of the risk to the business and the risk of failure."

By designing object-oriented test scripts, which allow changes to the object to ripple through the tests scripts that use it; and data-driven test automation, which allows the test automation tool to run through new data sets without scripting changes, Eshelby said last-minute changes to code can be made with minimal impact on the test environment. "It's not a shift away from automation in the quality assurance teams, but doing it more effectively" that is the goal, he said. ■