

Search and Destroy

New defect-tracking tools, practices help keep code free of bugs

BY ALEX HANDY

Bugs are inevitable. They're going to exist, no matter how many hours you spend staring at a collaboratively written snippet of code, no matter how many Roach Motels you put down. They're a fact, so you'd better have a system in place for making sure they're taken care of.

In this modern age of Web 2.47672.rev. b, online offices and instant messaging, it's remarkable how easily some information can still vanish into the ether. E-mail and AIM can swallow up messages about bugs, and without an easily accessible system, some of your bugs are probably being swept under the rug by a lazy coder here or there.

If your organization is still using an antiquated bug-tracking system, such as an in-house FileMaker database or a massive Excel spreadsheet, it's time to upgrade.

Matt Hargett, a Silicon Valley-based testing and QA consultant, said that the bare minimum any organization should have is Bugzilla. Hargett went on to say that this open-source tool is far from perfect, but if an organization has no defect-tracking system, and doesn't plan on extensively customizing an imple-

mented system, then Bugzilla is the first stop on the way to a decision.

"Bugzilla works if you don't do anything to it," said Hargett. "With Bugzilla, it's a giant duct tape and bailing wire tangle of Perl scripts, but it's solid duct tape and bailing wire. I've never heard of anyone losing the database or it crashing on them. Bugzilla is free, but it's not in a way. I'm all for open source—I run Linux on my laptop—but you need a Bugzilla specialist on staff if you're going to use it full-time."

Hargett said that at one of his previous employers this is exactly what happened. "A security patch came out, and since our defect-tracking system was customer-facing, it was a security exposure and we weren't able to resolve it in a timely fashion. The diffs were horrible because the code was horrible. If you customize it, there is a long-term maintenance cost."

But despite Bugzilla's shortcomings, Hargett maintains that a company looking to implement its first bug-tracking system should consider it as a first move. He said that the tool itself is solid and gives users a great understanding of what running a Web-based defect-tracking system is all about.

Dan Nobuto, a QA engineer at San Francisco-based Cloudmark, said that Bugzilla is a great tool. "Every other alternative I've seen requires a lot more resources. It's great to have a proper bug-tracking system that really works instead of relying on e-mail. And because it's open-source, you can modify it to do whatever you need."

But, agreed Nobuto, Bugzilla does have its problems. "For one thing, modification is a pain. The source has been going through clean-up, so it's getting better, but it has a much longer



Ratcliffe said that his company's coders and testers are spread out across the globe, which makes an accessible bug-tracking tool a high priority. "We do a development build of Solaris every night," he said. "Once that build is done for multiple architectures, it's actually delivered onto a number of systems here at Sun. Some of those are desktops, some are servers, some host databases. We actually get real-world people using the very latest build of the process. The builds get shipped off to a lab in Ireland where they get regression tests: thousands of separate tests—everything from very small one-CPU boxes, all the way up to large enterprise systems. This all happens automatically. What it means is every morning when I come into work, I can go to a Web site and check up on last night's build of Solaris."

And that, said Ratcliffe, comes thanks to the company's distributed bug-tracking system. Each morning, with just a few mouse clicks, Ratcliffe can see a detailed report on how many bugs were discovered overnight. He can watch these bugs as they are tracked down, resolved and then removed from the database of current issues.

But a system that's able to stand up to this type of use is not going to imple-

development cycle than most other open-source software. Also, in most places I've been (where Bugzilla is used), I've noticed search has been less than reliable once bugs go up above a certain count. I'm not sure if that's Bugzilla's fault, or the fault of the back-end database, though."

What Bugzilla does offer, however, is a reliable Web-based defect-tracking system. And it is against this system that all others should be measured, said Hargett.

SCALABILITY'S KEY

Chris Ratcliffe, director of marketing for Solaris at Sun Microsystems, said that his company sees its defect-tracking system as a far-reaching tool, similar to its customer service Web sites.



Test/QA consultant Hargett says Bugzilla is a good first move when deciding on a system.

ment itself overnight. It requires careful planning and a strong Web host.

When Hargett deployed a defect-tracking system at Network Associates, he quickly discovered what made some systems work and others fail.

"One of the things that tend to get overlooked is scalability," said Hargett. "How many users can be signed on at once?"

That's an issue that may be hard to quantify on a proprietary system that's not based on a Web server. In general, said Hargett, Web-based defect-tracking tools are much more flexible than other sorts, and with AJAX-enabled applications expanding quickly, it's a good bet that some nifty new tools will be available for Web-based tracking systems over the course of the coming year.

Hargett also said that the back-end database upon which a defect-tracking system is based can lead to a lot of headaches down the road. "A lot of these bug-tracking systems have really crappy back-end databases that corrupt themselves. You want to see if you can export the data into another system later," he advised. "Some try to lock you in with proprietary data formats." Exporting the database itself to a standard file format can save you in a pinch if the system crashes or gets corrupted. Also, it means that a broken defect-tracking system won't take down your information along with it if it fails.

But the biggest challenge when choosing a defect-tracking system, said Hargett, is probably your fellow employees. This is the same sort of problem a team can encounter while attempting to design any database. "One of the things I learned was that everyone wanted their own little field in the defect-tracking system database," said Hargett. "You have to avoid that. A lot of times the field they wanted was there; they just didn't like the name of the field. Their objections were purely nomenclature-based."

Adding too many fields to each entry in the database can lead to a cluttered interface and a confused team of testers. When designing a widely accessible defect-tracking system, it's important to

remember that some of the users you'll likely be enlisting to add bugs won't be programmers or testers. As such, requiring a dozen different strings of information will probably result in confused information, or worse still, fields left blank.

Another problem that an overly elaborate database can create is slower searches. This is a notorious Achilles' heel for Bugzilla, and other defect-tracking systems could be slowed to a crawl when searching if your database is filled with too many fields of information.

REDUCED TO TIERS

One way to avoid this issue is to break up the defect listings into tiers. This means that end users can input a few fields' worth of information, say circumstances, symptoms and machine configuration. The rest is then input by the testing team as they recreate the bug. This is how Sun optimizes its massive defect-tracking system, said Ratcliffe.

He said that a bug "essentially goes through the same process whether it's found internally or externally. The bug gets reported and placed in a bug-tracking system we host internally at Sun. We then try to recreate it. In some cases, in order to reproduce the problem, we have to get access to customer systems, but that's rare nowadays. Dependent on the type of bug, we generally have a code fragment that indicates what's causing the problem. We then look at the scope of the problem and prioritize the bug. The rare bugs that we come across nowadays are ones that will cause an entire system crash. These are rated highly. Bugs that affect performance are rated highly as well."

Hargett also pointed out that managers have different requirements than developers. The folks in the suits want to have pretty pictures and charts printed out by the defect-tracking system, and easily accessible metrics to show progress over time. These fellows are after raw numbers, and a good defect-tracking system should be able to provide them quickly and in a legible form.

But if your system cannot output graphs, be sure that the information can be transferred to some other program that can, such as Excel, he advised.

REPRODUCING BUGS

Once a bug is listed in the defect-tracking system, your first action should probably be to replicate it. Once that's been done, the replication should be written into an automated test that can reliably



Agitar has 'bug week,' when everyone in the organization looks for defects, says Savoia.

bring the defect back to the surface. Once this test has been created, your developers will have a one-click way to figure out if they've successfully squished the issue or not. You may even want to go as far as to list a "yes/no" toggle within each defect entry to indicate the presence of an automatic test in your arsenal, said Alberto Savoia, CTO of Agitar.

"The practice I found to be most useful is to link every bug to an actual automated test," Savoia said. "You can have a bug that describes something in the abstract. If you put yourself through the discipline of making the bug appear through an automated test, you give yourself two advantages. First, you have a way of telling that the bug has been fixed. Second you have another test in your arsenal, and you can never have too many tests."

And there's another benefit, said Savoia. "I've seen it happen a ton of times. You fix a bug and think it'll never come back, then three months later it's back. Every bug we have gets a number, and then a test is named after that number. It's not trivial to do, but it's a huge advantage."

What if a salesman is pushing your wares in the field, but is rebuked by an irate customer plagued by a hyperspecific bug in the last revision of your software? Does your sales team even know what defect tracking is?

Everyone in your organization should be able to submit a bug quickly and easily. Your customers should be able to add them too. Freelance developers, sales associates, contractors, even your

interns should all be able to submit bugs to your defect-tracking system. Whether this means you actually allow direct access via the Web, or you allow all interested parties to contact QA over the phone for support is up to you. But if you don't build a simple path toward bug submissions, you'll probably end up hearing about the showstoppers at the most inopportune moments.

Sun employs a full-time staff of around 150 phone workers who not only provide customers with technical support, but also track and catalog bugs. This means that the first interface a customer has with Sun after a deployment can lead to a properly formatted bug report being submitted to Sun's systems. Thus, the moment the customer reports the issue to a representative, it is known to the software development team.

"What'll typically happen is [the customer] calls a support line," said Ratcliffe. "There's a guy on the end of the phone and he talks to the customer about the problem. The more information we can get the better. The person on our end of the phone will compare the data against the bugs in the database, and if they don't find a match, they'll enter that bug in the database."

JBoss uses a Web portal to track its bugs, and has opened it to both customers and partners so that defects can be tracked in an open process. Bob Bickel, JBoss' vice president of strategy and corporate development, said that his company has "a customer support portal that essentially tracks all of the issues we have from customers, and as part of that, there's a bug-tracking and patch-management system, and that weaves into the back end."

Savoia said that no one in his company is safe from bug detection. "Finding, fixing and verifying bugs should be the responsibility of everyone in the organization," said Savoia. "We have bug week. During bug week, we get all the developers to go bang on the code and find bugs. We have developers verify and close existing bugs. This includes all the managers, the QA people, the developers, all of our people. Different people approach the software in different ways. If you just have the QA people and customers finding bugs, the developers are never exposed to the cost of bugs in the first place."

In other words, just because you write the code does not excuse you from having to test the code. And just because they wrote the code, does not mean they're excused from writing up bugs in the defect-tracking system. ■