

Got bugs? New project lets real computer users gang up on software bugs



Ben Liblit, an assistant professor in the UW-Madison department of computer sciences, has developed a novel way to root out and eradicate software bugs by enlisting the power of real users. His approach, called Cooperative Bug Isolation, uses a program that generates feedback reports from thousands of software programs in use and helps identify the most common and troublesome software glitches. Liblit's project is helping bring statistical rigor to post-deployment software debugging techniques, allowing developers to improve their software more efficiently. Liblit is pictured with his laptop computer outdoors at the Memorial Union Terrace at sunset. Photo by: courtesy Bob Rashid.

Ben Liblit offers a bold prediction regarding all of the complicated software programs churning away in your computer: They have bugs. All of them. Guaranteed.

"Software bugs are part of the mathematical fabric of the universe," says Liblit, a University of Wisconsin-Madison computer scientist. "It is impossible with a capital 'I' to detect or anticipate all bugs."

The staggering complexity of software is only part of the issue, he explains. Software has so many different points of interaction - with hardware, with networks, with other software and mostly with humans - that opportunities for buggy behavior abound.

"That behavior is so dynamic that it becomes useful to look at [software programs] almost like they were some sort of organic system, whose complete behavior is unknowable to you," says Liblit. "But there are behavior trends you can observe in a statistically significant way."

Liblit takes that metaphor literally in his research, known as the "Cooperative Bug Isolation Project." Combining programming languages and software engineering with a dose of machine learning, Liblit has enlisted real software users to tell him where the bugs are. And he has a growing "kill board" as evidence the idea works.

Liblit has created lightweight instrumentation that is added into the binary language of software. The instrumentation creates a sparse, but statistically fair, random sample of software behavior, while also protecting user privacy. The system produces regular "feedback reports" across the thousands of software programs that are in use. All of those reports get fed into a powerful database that accumulates the data and starts to identify trends.

Then, through statistical modeling techniques, Liblit is able to pinpoint software bugs that are occurring with enough regularity to affect many users. The final step in the feedback loop is a bug report, prepared by Liblit and sent back to the software engineers capable of acting on the results.

The science behind this real-world debugging approach received national attention this year. The Association for Computing Machinery (ACM) named Liblit's doctoral dissertation on cooperative bug isolation, completed at the University of California-Berkeley, as the best in the world in 2005 among dissertations nominated from both computer science and engineering programs. And while the open-source software community has begun to adopt the program, Liblit also has attracted interest from IBM and Microsoft.

The real excitement of the project, Liblit says, is that it could dramatically improve the ability to enhance

software post-deployment. "Software developers deploy their programs and rarely hear directly from users, but the poor guy in tech support gets an earful," he says. "That's the only kind of feedback you get; you lob it over the wall and hope it works." This system provides direct information about real-world software problems that developers can act on with statistical certainty. Right now, the only way to gauge real-world performance is based on the "squeaky wheel" effect of those who file bug reports or call tech support. And real-world performance will always be an important variable in dynamic debugging techniques.

"It has been cynically observed that software developers use their consumers as beta testers," he says. "I think there's a lot of truth to that observation. The problem is consumers are not very good beta testers. They're not very disciplined, they don't keep good records, they never do the same thing twice. My solution is to make them better beta testers."

Right now, Liblit has a number of users in the open-source software environment, where the application has been added to popular programs such as Evolution (similar to Microsoft Outlook), Gnumeric (a type of spreadsheet), Rhythmbox (similar to iTunes) and Spin, a CPU simulator in heavy university use. He has posted 192 versions of eight different open-source applications during the three-year life of the project.

The system averages just under 3,000 new reports per month, and bug rates vary a great deal across applications, he says, with "crash rates" (where the program shuts down) as high as 8 percent in one application to a low of 0.4 percent in another. All in all, his "kill board" has recorded 546 outright program crashes and 11,369 lower-level errors as of April 2006.

Liblit says user privacy is absolutely essential to the program's future viability. The randomly sampled snippets of data have no identifiable information unto themselves, and only have meaning when aggregated in large numbers.

Will software development ever reach a level of sophistication that would render Liblit's bug machine obsolete? Liblit says that scenario is unlikely. Given that software becomes vastly more complicated with each generation, his better guess is that the industry is maintaining an even keel.

"We're trusting our software more than ever before," Liblit notes. "We're also hating our software more than ever before. And when the software fails, it's more damaging than it ever was before. But with the right technology, the users themselves can help make software better for everyone."

Source: University of Wisconsin-Madison

This document is subject to copyright. Apart from any fair dealing for the purpose of private study, research, no part may be reproduced without the written permission. The content is provided for information purposes only.