A Bug's Life and Death

by Elizabeth Barnett

The primary way BASIS eliminates bugs is not to create them in the first place. That sounds simplistic, but it's not as easy as it sounds.

Fortunately, the code that built BASIS' current product family, the 2.20 release of PRO/5®, Visual PRO/5®, the PRO/5 Data Server® and the BASIS ODBC Driver®, is mature enough to have most of the bugs worked out already. Now, it's a matter of ensuring that new capabilities and technologies are implemented in ways that do not break legacy applications. BASIS has a rigorous battery of tests that all new releases undergo to help with this.

The Design Phase

Eliminating bugs from a new product, one being written from scratch with the requirement that it be fully compatible with legacy BBx code, is a different animal. How is BASIS making sure that the first release of BBj™ acts the way we, and you, expect? We've formalized the design phase of BBj, mapping out specifications and designing quality assurance tests around those specifications.

With the development of BBj, we've defined requirements with an eye to the future, so that BBj will be easy to scale up, as developers need. And with these requirements, we can make sure that the first release of the product will conform to our expectations. To do this, we're using a software design product called Rational Rose. It's a visual design tool, tightly integrated with Java™, that allows us to design BBj, maintaining object-level relations and use cases. We're using other tools that allow us to extract fields and templates for our design documentation. We've also implemented periodic design and code reviews to ensure that requirements are being met.

Software Testing

But, despite the effort BASIS is putting into eliminating bugs in the design phase, it's impossible to anticipate the myriad possibilities of what might break in the field, so we've developed a series of automated tests that are run on each and every revision of all our products.

This test bed has been developed to first ensure that every single build meets minimum standards of functionality. But the suite is continuously augmented as new problems are discovered and resolved. When a bug is found, new tests are written and added to the test bed to make sure the bug never surfaces again in another revision of a BASIS product.

The test bed itself is structured in "waves," or groups, to pinpoint problems. The first group tests for general, commonly used functionality, such as the product starting up correctly, making connections to a data server, reading/writing files, etc. Subsequent groups of tests drill down deeper into the code to progressively test more specific portions of the product, such as file I/O and verb functions on different file types.

At present, BASIS has about 1,000 tests in the bed. Some of the tests, such as the ones that check the functionality of GUI controls, contain tests for literally hundreds of functions and proper configurations.

BASIS builds and tests our products on multiple operating systems and multiple versions of those operating systems. All told, BASIS supports our products on 37 ports, representing 18 different platforms. A year and a half ago, it took us over a week to build and assemble all our products for testing on these ports, and the testing wasn't nearly as comprehensive as it is today. Now, the build and general testing take place in a night.

Tests that check common functionality are set up to run each night on all of the products built from source code in the repository. That means that *every* modification *every* BASIS software engineer makes throughout the course of *every* day gets tested *every* night to make sure that those modifications haven't affected core product functions. Additional groups of tests are run sequentially on different nights, so by the time a new revision of product is released, we can be sure that it has passed all the tests in the entire test bed.

Sometimes things can appear to be bugs when the problem is really just a configuration issue involving a product, an operating system or a network. Or maybe the problem is in a piece of code that doesn't belong to BASIS. Often, a description of the problem over e-mail or a short telephone conversation can reveal to us which it is. Then we can begin to help you solve the problem.

Will The Real Bug Please Stand Up?

But sometimes, despite our Herculean efforts, bugs elude the formal processes we've established and make it into the release. What then? Well, sometimes we find them, sometimes you find them. If you find them, we hope you call BASIS Technical Support.

In Tech Support, the first thing we do is determine if it really is a bug in the BASIS product. Sometimes things can appear to be bugs when the problem is really just a configuration issue involving a product, an operating system or a network. Or maybe the problem is in a piece of code that doesn't belong to BASIS. Often, a description of the problem over e-mail or a short telephone conversation can reveal to us which it is. Then we can begin to help you solve the problem.

Every call to BASIS Tech Support is tracked in the Tech Support Call Log System, whether it results in the discovery of a bug or not.

But if it turns out that we in Tech Support can't resolve the problem right away, the single most important thing we must be able to do is reproduce the problem, here at BASIS on our equipment. It's critical. Sample code is immensely helpful and saves us (and you) a lot of time and money. Later in the process, it also helps us in writing a test to ensure we've fixed the problem and that it won't reoccur.

Once we can reproduce the problem in house, we write up a report, a Qualty Assurance (QA) memo, that is entered into our QA database.

Formally In QA

From this point forward, the bug you've reported is formally tracked through the BASIS QA process. In the QA group, we verify that the problem described by Tech Support is an unknown behavior and that it is not duplicated in any other entry in the QA system. Then, the problem is assigned to BASIS Engineering for resolution.

Back To The Drawing Board: Engineering

The engineer responsible for resolving the problem begins analyzing. Depending on the nature of the problem, this is where sample code can become invaluable. Our analysis begins with the most current copy of the product because sometimes problems get fixed in the course of routine code development modifications. We pinpoint where in the code the problem is occurring and why. We'll then modify the code to correct the problem and check the new code into the source code repository, which automatically records the time and date. Time stamping allows us to go back and rebuild any of our products at an exact time and date in the past, which can help with debugging. Because the new code is now in the repository, the fix will be included in the next night's build.

Tests that check common functionality are set up to run each night on all of the products built from source code in the repository. That means that *every* modification every BASIS software engineer makes throughout the course of *every* day gets tested *every* night to make sure that those modifications haven't affected core product functions

QA Verification

The engineer then reports the fix to QA. There, we download the latest build and verify that the fix actually resolves the initially reported bug. If the bug is not fixed, it goes back to the engineer. If it is fixed, we'll write an automated test and add it to the test repository to permanently check for the bug and the fix in all subsequent product revisions.

Notifying You

One of the many enhancements we're bringing you is a formal system for letting you know what's happening while a bug is being investigated and when it is fixed. We've begun redesigning the QA database to maintain a list of affected customers and their e-mail addresses. We will have an automated system that will send messages as soon as the fix is verified by QA. We're also looking into ways to allow customers to search the database for bugs under investigation and add themselves to the notification list.

So there you have it: the basics of the BASIS process of eliminating bugs. In the session, we'll show you how it actually works with reported bugs.

This session will be presented by a team of BASIS personnel directly involved in the hunt, capture, and killing of software bugs. Earl Box is the BASIS porting engineer; Scott Gamron, BASIS' QA supervisor and the primary creator of our software testing system, was recently awarded the Certified Software Quality Engineer certification by the American Society for Quality; Brian Hipple is a BASIS Software Engineer; and Janet Sheldon is the BASIS Technical Support Supervisor.