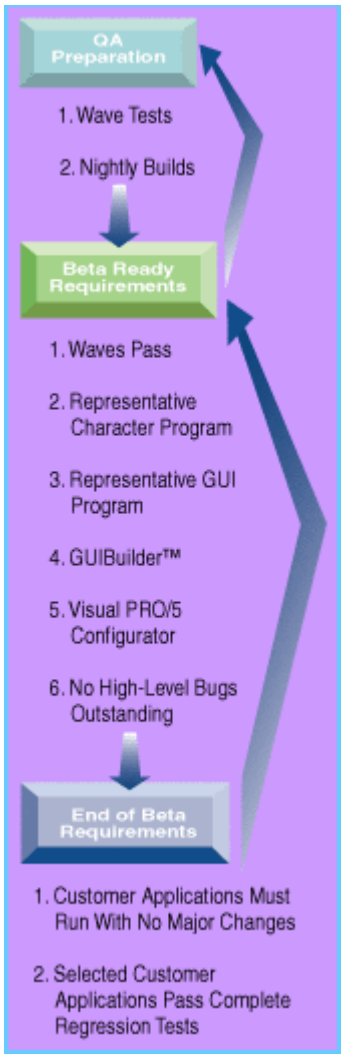
THE BBj™ BETA PROGRAM

By John Schroeder

When BASIS embarked on the development of BBj™ almost two years ago, one of our major concerns was testing the new language. For the first time in 15 years, we were completely rewriting BBx®. Not one line of code could be reused because we were writing the new product in Java, a language that wasn't even dreamed of when BBx was first written. The advantages inherent in providing the Java environment for the time-tested, enterprise-level Business Basic solutions were many. But our overriding goal was to provide a new Business Basic that would run applications written for PRO/5® and Visual PRO/5® with no major changes.

One of the first questions then was, how could we test BBj? How could we assure ourselves and our Customers that BBj would run their applications out of the box?



QA PREPARATION

Along with the development of BBj, we began a parallel effort to develop test suites that would allow us to quickly and easily test the behavior of every verb and function in PRO/5 and Visual PRO/5. Once developed, these tests would serve as a basis for testing BBj. This is no small task. Over the years, a lot of functionality had been put into BBx. Nuances in behavior, such as how BBx rounded numbers in a calculation or how MODE=strings work on I/O verbs, had to be taken into account.

Obviously, it could take a lot of programmers a very long time to write tests for every possible function of BBx. And on top of that, somehow, the combinations of behaviors that under some circumstances might produce erroneous results had to be taken into account also. Add to this all the variations inherent in user interfaces, both character and GUI, and you have quite a testing job. 100% test coverage is not possible. So we needed to adopt a methodology that would allow comprehensive testing of functions, automated data entry simulations and application testing that would give us the confidence that BBj does, in fact, run PRO/5 and Visual PRO/5 applications with no major changes to the applications.

The method we adopted is basically a layered approach. We wrote a series of tests that tested increasingly more and more detailed functions of BBx. The first layer, or wave, checks more obvious things like assignments, simple file access, various functions, etc. Succeeding layers test I/O in more detail, with more depth; for example, in pushing file size limits, or exercising different types of keys with multiple segments with various options.

These are followed by the GUI control tests, which are the recorded sessions of an operator

interacting with a program that has GUI components. To build these tests, the operator goes through a script that exercises every feature of a control. The entire session is recorded, including the operator's keyboard and mouse activity as well as the changes to the screen and contents of data elements. This result set is saved and then played back as changes are made. A new result set is generated and compared to the original and any differences are identified.

All results are kept in a database and we link this database to our internal QA Web page. Test results are grouped by category with an indicator displayed for each category that shows success or failure. Anyone can drill down through the results to see the exact tests that failed and report the failures to the engineering staff.

This process has been incorporated into our nightly build. In this process, all code changes generated during the day are merged into our code libraries, and then the entire code library is passed to various machines to be compiled and linked on the various operating system platforms where these products are ported. After a build is successfully completed, the appropriate wave tests are run, and the results generated and compared to the standards. Any deviations are recorded and linked to the Web page for analysis the next morning.

As BBj has progressed through development, it has been phased into the nightly build. The database of results from the PRO/5 tests becomes the standard against which the results from the BBj tests are measured. The criterion for passing a test is that the test results for the BBj build match the results from the latest PRO/5 build.

BETA-READY REQUIREMENTS

The nightly build and QA process described above does a wonderful job of testing basic functionality. But that still leaves the problem of various combinations of events that could lead to an error. The only realistic way to test for these is to do full regression tests on representative applications. For the beta process, we decided to approach this in two steps. The first step, or the beta criterion step, would use representative real applications and give us the confidence that BBj was ready for beta. The second step, the actual beta test, would begin once we were confident that beta testers could have a high probability of success in making their current application run in the BBj environment.

The beta process would involve the rigorous application regression testing that would allow us to say BBj was ready for the Business Basic market.

The tests that make up the beta criterion step consist of the following groups of programs.

- PRO/5 wave tests
- TAOS-generated database application program
- Chile Company Master File Maintenance Program from the Advantage CD
- GUIBuilderÈ, the Visual PRO/5 GUI Development Environment
- The Visual PRO/5 GUI Configurator Utility

Before the beta test could begin, BBj had to pass all of these tests with no major problems.

The PRO/5 tests were already automated. To get the testing program ready to handle the beta criterion tests, we had to automate the various applications and build a result database based on PRO/5 results, so that we could measure the behavior of BBj. The same process that we used in automating the GUI portion of the PRO/5 tests was used here. We created scripts that a test engineer would follow to enter data, use the mouse where appropriate and

go through every step in each of the applications as they ran under PRO/5 or Visual PRO/5. The results of these tests were saved as the standards for comparison of the BBj results. As of this writing, the beta criterion tests had all passed, and BBj was ready for the beta test.

RELEASE REQUIREMENTS

The final step in getting BBj to market is the beta test. As with all phases of testing BBj, the beta had to be the most rigorous testing process ever done on a BASIS product since our first release of BBx. With this in mind, we established the following release criteria:

1. Customer applications must run with no major changes
2. Selected Customer applications pass complete regression tests
3. BASIS must be running its internal accounting on BBj
4. BASIS' e-Commerce application must be up and running under BBj

The last two tests show BASIS' commitment to the compatibility of BBj and PRO/5. Our internal applications must run before we will ship BBj. In addition, the selected customer applications must pass their authors' regression tests.

As you can see, we are devoting significant resources to ensure that BBj is compatible with the PRO/5 family. This is a primary requirement of BBj. However, if that were all BBj did, why would our Customers buy it? Obviously, BBj has to offer more benefits to our Customer base and to end users in order to be a market success. The availability of a thin client, which can run stand-alone or within a Web browser, the capability of true three-tier client/server architecture, external database access, a very robust enterprise-level file system along with compatibility with PRO/5 make BBj a very attractive product for the Business Basic community.