INCREDIBUILD

CASE STUDY

LUSAS

Accelerating QA Cycles

Industry

| Computer Software

Process

| Testing

Results

| Reduced QA Cycle Time from 758 mins to 88 mins

LUSAS is the trading name of Finite Element Analysis Ltd., a UK-based specialist engineering analysis software developer that develops a range of software products, based on the LUSAS finite element system. The software is used for all types of linear and nonlinear stress, dynamic, and thermal / field analysis problems in all branches of the engineering industry. For more information, please visit www.lusas.com.

About the Author

Jason Barnaby is a Project Manager at LUSAS, responsible for the development of LUSAS Modeller, the interactive part of the LUSAS suite.

Using Incredibuild, the development team for the LUSAS Modeller application has significantly reduced the time taken to QA, and therefore release their product. LUSAS Modeller is part of the LUSAS suite – an analysis tool used in several different engineering industries. It is a large, MFC-based graphical user interface application. Previously, a typical QA cycle for LUSAS Modeller was an overnight job. With Incredibuild, it now can be done over lunch. This article describes the manner in which this was achieved.

The Challenge

Being in a safety critical industry, LUSAS have very strict QA procedures. The LUSAS Modeller QA suite consists of nearly 900 test-cases, produces over 11000 output files and takes up over 1.5GB of disk space. A management program was written some years ago to run the tests, compare the output, and prepare a report. On a typical development PC, the complete job (running the tests and comparing the output) would typically take around 10 hours.

During the build-up to a release, each developer runs the QA procedure on his own code, merges his work

with that of other developers, runs the QA procedure again, and finally commits the changes. Then the release candidate is built, and the candidate goes through the QA cycle again. Since a single QA run took longer than a working day, each effectively took a whole 24 hour period, and consequently it could easily take a couple of weeks to cycle through just a few developers, even if every single QA run was successful. A single bug would write off another 24 hours, as the bug would need to be fixed, and the QA run again. This had a significant impact on the company's ability to rapidly respond to a customer's request for a new version.

The Alternatives

Before using Incredibuild, we had tried and considered several alternatives. Most obviously, we spent a lot of money on fast computers. This solution was never satisfactory – it always seemed that as quickly as hardware speeds increased, developers would be expanding our QA suite more and more.

Any attempt to shortcut the QA process by running only part of the QA process or even none on each developer's machine prior to commit, always led to disaster. Inevitably bugs would be found only at the last minute, wasting everyone's time, and with the cause being that much harder to track and fix.

We also spent considerable time profiling both our software and the differencing tools, in an attempt to make the whole process take less time, and with great success – but we never achieved the order of magnitude change required to really make a difference.

The Solution

A project was conceived to modify our existing QA management program. The program now creates an Incredibuild Submission Interface script file, detailing all the tasks to be undertaken. Each task consists of running a tiny newly created executable with the name of a single test-case. Each instance of the executable is responsible for running one test-case, and comparing the output against that expected. At the end, the management program gathers all the success/failure information and prepares the same report as before.

As described, this was a fairly simple operation. We were able to restructure our management program and get the whole thing going in just a few days. The difficulties that we subsequently encountered were all based around those parts of our program that assumed there was some sort of Windows user interface currently running. Of course, when running on a remote machine, any and all window creation must be completely suppressed.

Additionally, we had some concurrency problems when multiple instances of the program would be trying to access the same files on disk – e.g. for licensing and logging purposes.

Both of these problems were overcome by improving the code design of our program, and we were eventually able to distribute all of our QA cycle.

The following is a screen snapshot of the management program taken after a full QA cycle. As can be seen, the testing process itself took nearly 11 hours, another 100 minutes were required to compare the output files, and all of this was achieved in just 90 minutes of elapsed time – a speed up by a factor of 8.54. This factor varies according to the power and speed of the host PC – developers with slower machines get the most benefit! We have, however, found that a factor of 8 to 9 is typical when distributed to 12-15 CPUs.

Benefits

Being able to quickly and confidently release software is obviously vital. We now find that it is quite possible for up to three developers to update, build, QA, and commit in one day without cutting any corners.

Summary

Using Incredibuild to distribute our QA has greatly decreased the amount of time needed to verify that no new regression bugs have been introduced. This in turn means that we can respond to a customer's request for a new version much more quickly, whilst still having total confidence that the new version has passed all our rigorous QA procedures.