# Refueling a Stalled Software Design Project

Designing original software for medical device technology can be challenging enough, but overcoming the additional obstacles that are encountered when a software project has developed into a hopelessly lagging burden is a completely different ordeal. This article provides an overview of how a company can reinvigorate a stalled software design project and what steps need to be followed in order to bring the task to completion.

By Andrew Dallas

All engineers have seen a software project that is already hopelessly stalled and which is many months, or even years, behind schedule. They are in a seemingly endless cycle of test and de-bug mode and the software is still not stable enough to even consider releasing it. The team is stressed out, management is applying pressure, and sales people are explaining to customers why the product is delayed. Everyone in the business of building commercial software has either been there or seen this happen. What follows are some practical, hands-on techniques that can be used immediately to turn things around and start making forward progress.

A large part of any software rescue mission is called code hardening. Generally, the soft-

Andrew Dallas is the president and chief technical officer of Full Spectrum Software, a software consulting firm that provides software development and testing services to the medical device industry. He can be reached at 508-620-6400 or adallas@fullspectrumsoftware.com.
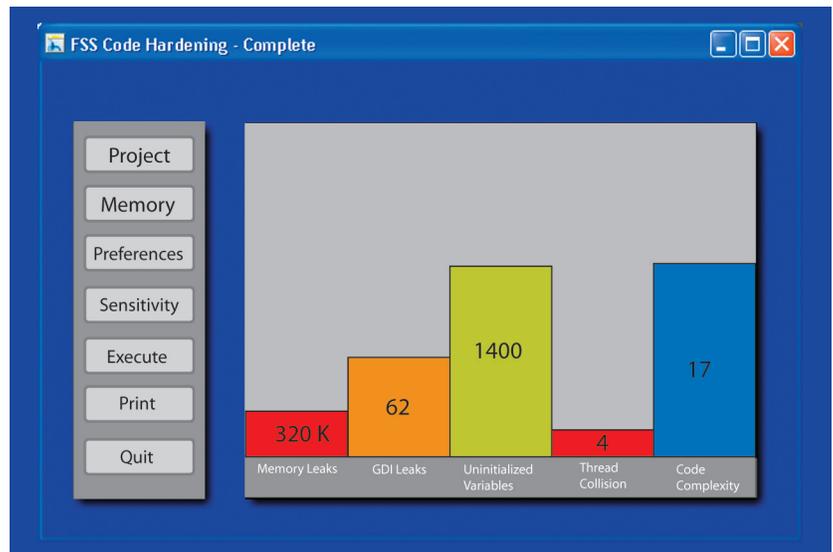
This screen shows the results page of a Static Analysis tool called FSS CodeHardening-Complete. This software tool does not "run" the code, it produces these results by algorithmically examining the source code and shows the results of the five parameters it was tuned to look at. This type of tool is used by a software developer to direct the focus of testing efforts.

ware being worked on is either an older product that the team is trying to upgrade and add new features and capabilities to, or the software is more complex than the software team is prepared to manage. For newer offerings, the product may have expanded in scope to require complex threading, may require expertise that the development team simply doesn't possess, or, in the worst case, the product may have been designed and implemented by engineers who did not understand the technology. Whatever the case may be, there are commercial software tools available, including both static analysis and run time analysis, that can help immediately.

Emphasis On Software Testing

Analysis Tools

Static analysis tools "read" the source code and identify certain classes of errors without actually running the code. Static analysis tools have evolved from simple syntax checkers to powerful tools that

▶ It should also be noted that static analysis tools are really software development test tools, not SQA test tools . . .

algorithmically examine code for errors and defects even in large code bases.

Runtime analysis tools are incorporated into the build process and identify errors while the code is running. The two tool types are complementary in that each is more effective at finding errors that the other can't, but runtime analysis has historically been more widely used by software developers. This is because it is only recently that static analysis tools have become sophisticated enough to give developers highly accurate and meaningful information about the state of their code. Runtime analysis tools will track and report problems such as unhandled exceptions or failures in the code, out of bounds parameters that are being passed to functions within the code, and memory errors such as freeing the same block of memory twice. While not impossible, every one of these errors is extremely difficult to detect using only manual techniques such as formal code reviews.

It should also be noted that static analysis tools are really software development test tools, not SQA (software quality assurance) test tools per se. The software team leader can use the results

or output of the tool to focus SQA efforts, but the error correction warnings are meant to be interpreted by developers and need to be corrected by them as well. They are one of the few automated test tools that are not generally run by the SQA team.

Altering the Mindset

The most important change that needs to be implemented immediately is a fundamental shift in mindset. The goal at this stage is to stop the test and de-bug cycle and begin a controlled process called code hardening. The goals, objectives, processes, and methodologies for code hardening are different from simply trying to de-bug the software and release it.

Everyone needs to be pulled off the project except for the software lead and the SQA lead, because their domain expertise and intimate knowledge of the code will be required. The next step is to acquire the appropriate commercial grade static analysis tools and one or more runtime analysis tools. Both of these toolsets are challenging to configure correctly so a reasonably large amount of time should be allocated for configuration. This is especially true if a team is not already using these types of tools. Once the tools are configured correctly and applied to the product, it is possible to see an immediate picture of the state of the code.

Concurrently, the most experienced software systems architect and a team of the most experienced software quality assurance engineers must be assigned to the project. The systems architect should have experience in doing formal code reviews and should be tasked with con-

ducting a thorough analysis of the code.

Their focus should be on evaluating the architecture, craftsmanship of the code, and areas of the code where there are known defects and problems. Their report should identify where the major problems exist within the code base. This allows the software team leader to help prioritize where resources should be allocated when they begin fixing defects.

The new SQA team should be tasked with developing formal test plans based on the results of the code review and output from the automated tools. It's important to emphasize that the new SQA team must be composed of the most highly experienced people in the company.

The SQA team's ability to test other people's code, uncover difficult to find errors, and clearly document difficult to reproduce defects is crucial in this code hardening process. This is an absolute necessity and is not the same as software engineers testing the code they have written. Under no circumstances should the software engineers be allowed to approve their own code. Experienced SQA engineers have an entirely different mindset about testing other people's code. While software developers look for elegant ways to make things work, SQA engineers have a profound fascination with looking for ways to make things break. Only they should be allowed to approve code.

Plan for Completion

Once the systems architect has finished his or her report, the static and runtime analysis tools have been correctly configured and the tests have been conducted,

▶ The SQA team's ability to test other people's code, uncover difficult to find errors, and clearly document difficult to reproduce defects is crucial in this code hardening process.

and the software quality assurance team has completed their detailed test plans and test protocols and completed at least one round of testing, the next step is to begin developing a plan for project completion. This phase will take approximately eight to 12 weeks to complete.

There are two issues which will arise and expectations must be set appropriately. One is that when this process is completed, the bug count will skyrocket. If the bug count does not jump substantially, it is a sign that something has been done incorrectly. Either the test tools have not been configured correctly or the new teams don't have the maturity or experience to execute the process described here.

The key point is that the results of doing this analysis will reveal many more bugs than were originally thought to exist. The second issue is that the "life expectancy" of the product will become obvious. It should be possible to make

an informed judgment that perhaps no further upgrades will be feasible for the product and that management should expect that a complete re-design and re-write will be required within a certain time frame.

At this point, the original development team should be brought back onto the project. However, the new SQA team should largely be kept in place until product release. At minimum, the most experienced SQA lead and the software architect should be kept on the project until completion.

With a detailed view of the code base, it is now possible to prioritize the order in which each defect should be addressed. Start with the low hanging fruit. Fix the easy bugs first so that additional defects are not introduced during the fix process. Address each bug methodically and release the fix to the SQA team for formal testing.

Do not begin bug fixing at a frenzied pace, even though there will be considerable pressure to do so. Wait until the SQA team completes its testing and reviews the results of each fix. From this point to final release, it is imperative that no additional bugs are generated as a result of a fix. Follow the now documented test procedures and protocols. Review the defect tracking system daily. Continue to run both the static and runtime analysis tools on a regular basis.

Conclusion

Talk to each team member at least once per day regarding their progress and any problems. Plan to hold formal weekly review meetings to be absolutely certain forward progress is being made. If this highly structured process is followed consistently, there is a very good chance of successfully releasing the product on the revised schedule.

Full Spectrum Software
Software Engineering & Testing

Full Spectrum Software
225 Turnpike Road
Southborough, MA 01772
508-620-6400
www.FullSpectrumSoftware.com

Full Spectrum Software is a medical device software engineering and software testing firm. The company's Quality Systems follow FDA guidelines and Full Spectrum Software has developed software for a variety of Class II and III medical devices. The company offers a wide range of software engineering, design, development, V&V and testing services. The company also offers unique services such as Code Hardening, porting and rescue missions. Full Spectrum Software has been in business for more than 12 years and their clients include both industry leaders and cutting edge start-ups. Andrew Dallas, the company's founder is widely considered one of the leading experts in software development for the medical device industry.