

Code Review doesn't have to Suck!



Dr. Dobb's
THE WORLD OF SOFTWARE DEVELOPMENT

When Quality, Security Count

Static code analysis can make a big difference

By Sid Sidner
April 24, 2010
URL: <http://www.dj.com/tools/224600102>

Sid Sidner is director of security engineering for ACI Worldwide.

ACI Worldwide is a provider of payments software to banks and merchants around the world. With more than 800 software engineers working in development centers in seven time zones, issues such as software quality and security are critical to ACI's success. As director of security engineering, it's my job to ensure that our code base is bug-free and intruder resistant, while continually improving the software's overall quality. These concerns aren't new. They've been our mantra since the company was founded more than three decades ago. We decided several years ago that the best way to ensure quality and security was to introduce static source code analysis into our development processes.

Static code analysis is the process of examining and evaluating software without actually executing the code. Analyzing software when executing software is known as dynamic analysis. Static code analysis is all about moving the detection of critical security and quality problems upstream, ensuring they're identified and fixed early in the development process.

This approach yields significant productivity gains across the entire process and leads to cleaner, more stable builds, more efficient testing, and of a course, a higher quality product. Besides helping us find bugs that we've missed in unit testing, static code analysis has made all our engineers aware of security issues and helped us teach junior staff better coding techniques.

What's Involved

Static source code analysis tools are almost entirely automated. They're like compilers, but instead of generating machine-executable code, they simply find bugs and issue warnings about security vulnerabilities, logic errors, implementation defects, concurrency violations, boundary conditions, and other glitches in the code. The tools provide a list of problems, each tied to a specific location in the source code. Detailed context is usually provided to explain how the tool arrived at the conclusion.

Static analysis tools use very sophisticated process flow and data flow analysis. The quality and security issues they identify are often complex and involve obscure logic problems, which is why these tools can be so valuable.

Static source code analysis tools analyze 100% of the source code, far more than any external test tools. For organizations that must comply with the Payment Card Industry Data Security Standard (PCI DSS) or Payment Application Data Security Standard (PCI PADSS), these tools fulfill code review requirement. They also produce valuable metrics, including kilo-lines of code (KLoCs), file counts, and "churn" -- that is, the number of files that have changed between two regular builds.

Introducing static code analysis and the requisite tools into the development process isn't always painless, however. At ACI Worldwide, we found many subtle pitfalls in our efforts to roll out this approach company-wide. The tool changes the way many people work and must become a part of the organization and its culture.

For instance, static code analysis tools usually require careful integration into the project build process. For large software products, these builds are often somewhat of a black art, involving the use of Make and Ant. There are many options and dependencies. All static code analysis tools offer powerful utilities to analyze the build process and insert themselves into the right places, but some manual tuning is usually required.

5 Queries for Choosing the Right Code Analysis Tool

1. Do you need a static or dynamic analysis tool? 2. What languages and platforms does it support? 3. How flexible is the reporting component? 4. How easy is it to add or update rules? 5. Does it integrate with your IDE?

These tools also must be integrated into developers' daily work. Again, tool makers offer both command-line versions of the tools as well as plug-ins for many of the popular integrated development environments such as Eclipse and Visual Studio.

Most importantly, the tools require that the code base have a subject matter expert (SME) who can also provide the same service for the tools. That person will answer questions not just about how the tool operates but also about the issues that the tool is finding -- including identifying when the tool is generating a false positive. The SME will provide training and support to other developers, a fairly heavy workload for the first few weeks, until everyone is familiar with the static analysis tool. After that, that part of the workload should settle down to several hours a week.

Initial Analysis: Panic Time

The biggest challenges with static code analysis tools are problems in existing code. There's an old programmer's joke that says God made the world in six days because he had no installed base. This is certainly not the case for most businesses, which often have millions of lines of code.

The first time an existing codebase is analyzed, tens of thousands of issues will be found. Don't panic. Remember, these issues have been there for awhile, and the software continues to function and provide users with what they need.

At ACI Worldwide, all the issues from an initial build on existing code are immediately deferred and hidden from sight. That way developers don't get overwhelmed and can stay focused on ensuring that new problems aren't introduced into the code. At some point in the future, product planners and the senior development staff review the deferred issues, prioritize and group them, and decide when remediation can be factored into the planning for a future release. There's no perfect approach, and businesses must always make hard decisions about whether to counter a vulnerability or assume the risk.

Tips For Success

How can you ensure a successful deployment? Here are some hard-won tips from our experience:

- Define an initial issue policy. You may decide to only deal with the most severe issues for the first project cycle.
- Get the global mechanics working. Many of the tools require license managers and centralized result servers.
- Attack one product at a time. Get it working with one group and then move on to the next.
- Identify SMEs. Every product needs at least one subject matter expert. Large products that are broken into major components will naturally need a SME for each one. Be sure that the SME and his or her manager understand the ongoing responsibilities and time commitment.
- Train SMEs. Make them designated experts.
- Work with SMEs. Help them to do build and tool integration for their product or component.
- Train developers. The SME should guide how the tool is integrated into the team's development process.
- Perform initial analysis on existing code and defer all issues. Don't discuss the large quantity of issues with the developers. If any ask, explain to them that they've been set aside and will be considered in a future product cycle.
- Deliver help from SMEs to developers as required. During the first days of the roll-out, the SME should monitor the developers' work. Developers should be analyzing the code often, at least before they submit a completed unit of work into the product build. Just as a developer wouldn't check in a unit of code that doesn't compile, they won't want to check in a unit that still has static code analysis issues.
- Run the build analysis often. If the developers are doing their job and addressing issues as they come up then no issues should be found at this stage.
- Review deferred issues. After the process is running smoothly and the tool is a routine part of work, review deferred issues and plan whatever remediation is needed for future releases.

The Right Tool For You

There are numerous open source and commercially available static code analysis tools on the market. When choosing one, the place to start is with language support. Some tools like AdaCore's CodePeer and Green Hills' DoubleCheck support a single language. Other static code analysis tools support multiple languages.

But language support isn't the only consideration. When ACI Worldwide was in the market for a static code analysis tool two and a half years ago, we identified five vendors -- Coverity, Fortify Software, Klocwork, Ounce Labs, and Veracode. Veracode was eliminated immediately because it only offered code analysis as a service, and we wanted a tool that could be used in-house and provide developer training. Each of the other four vendors performed an in-house proof-of-concept on a large C++ program (2,500 KLoCs) and a large Java program (600 KLoCs).

Coverity was eliminated because, at the time, the tool provided excellent quality checking but had limited security checking. Conversely, we eliminated Ounce Labs because it focused almost exclusively on security, assuming that the prospect already had quality checkers, which wasn't the case for us.

Fortify Software and Klocwork were comparable in their ability to find important quality and security issues. However, Klocwork's licensing model made it less expensive for us. Klocwork used the FlexLM license manager with floating licenses, whereas Fortify Software had a dedicated code contributor model. Since we have development centers spread around the globe in different time zones, we're able to share the licenses very effectively around the clock, so Klocwork was the right fit for us.

Final Analysis

Overall, static code analysis has proven to be a valuable tool for ACI Worldwide. For a reasonable cost per developer, we're finding serious bugs more comprehensively and earlier in the development process. In addition, the Klocwork suite we chose provides a way to connect experienced senior developers with junior developers. The tools include extensive help files that refer developers having difficulty with an issue to a more experienced developer to get advice -- always a valuable interaction.

Bottom line: Static code analysis tools help incorporate security and quality awareness into the fabric of the entire development organization. Finding bugs earlier and avoiding security breaches is invaluable to any software development effort.

[Source Code Analysis](#)

Find critical bugs: C/C++, Java, C# Get Started Today. Free Trial!
www.coverity.com/trial

Ads by Google