

---

# Coverity Development Testing Solution for Medical Device Software Validation

December 2011

---

## Introduction

Over the years, medical devices have become increasingly dependent on software, evolving from the use of a metronome circuit for early cardiac pacemakers to functions that include electrocardiogram analysis, laser surgery, and intravenous delivery systems that adjust dosages based on patient feedback. Software in Implantable Medical Devices (IMDs) performs life-sustaining functions such as cardiac pacing and defibrillation, drug delivery, and insulin administration. Similar to most other industries, the amount of software in medical devices is doubling every two years. Modern infusion pumps can contain more than one hundred thousand of lines of code, while proton beam therapy machines may contain in excess of one million lines of code.

## Challenges of Successful Static Analysis Defect Management

Medical device manufacturers at the forefront of innovation rely heavily on software to build devices that help patients lead better lives. Delivering innovations through software allows features to be built more quickly and enables competitive differentiation. However, with the benefits of software come the risk of defects and bugs. In the first half of 2010, the Food and Drug Administration (FDA), the regulatory body in United States, issued 23 recalls of defective devices, all of which were categorized as Class I - meaning, "there is reasonable probability that use of these products will cause serious adverse health consequences or death." At least six of the recalls were likely caused by software defects.

The challenge for Original Equipment Manufacturers (OEMs) is to find and fix as many defects as possible before releasing a device to the market, since even a single undetected defect could result in severe injury or even death. This places a strong emphasis on software validation as a critical part of the innovation process.

## Validation of Software during Development

OEMs are interested in the technologies and processes available to assist in implementing guidelines and best practices in software development. They must determine which development solutions can help in meeting regulatory requirements, and what best practices should be followed to ensure that the benefits of software are not undermined by risks of failures and unpredictable behavior due to defects as software gets more complex.

To assist device manufacturers, the FDA has published a guideline in the form of General Principles of Software Validation; Final Guidance for Industry and FDA Staff. Specifically, section 4 within the guideline lists the general principles that should be considered for software validation. Among other things, the section contains recommendations for defect prevention, software validation after a change, need for independent review, ensuring coding guidelines, and developer testing.

### Introduction to Coverity Development Testing Platform

The Coverity Development Testing Platform (Figure 1) contains numerous industry-leading components that can be adopted as a part of the software development and verification process and assist in meeting specific suggestions contained in the FDA guidelines.

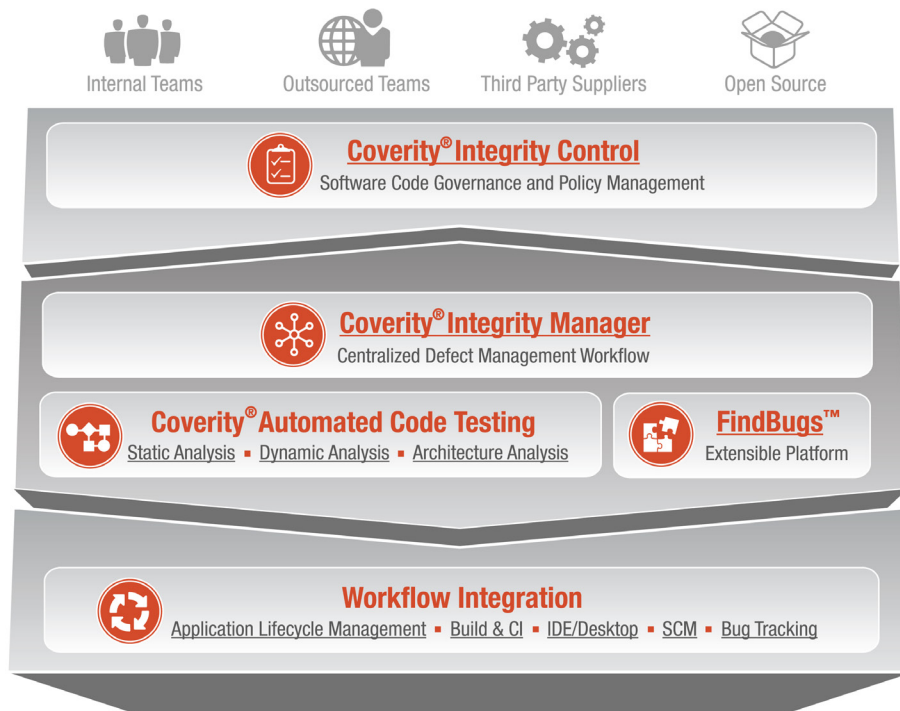


Figure 1: Coverity Development Testing Platform

## Coverity® Static Analysis

Coverity delivers the industry's most accurate static analysis solution – a method to verify the correctness of the software without executing the software. It is used by developers around the world to improve the quality of their code. It enables them to find and fix defects in C/C++, Java and C# early in development as the code is being written, when it's the least expensive and easiest to fix, instead of testing quality into the software after the fact. Though not specifically identified in the FDA guidelines, in 2009, The FDA recognized the strengths of static analysis and included it in the guidelines for infusion pump software safety testing for runtime errors.

Modern static analysis can discover complex defects in the code through symbolic path simulation, the process of analyzing every possible execution path of the program. Additionally, static analysis technology has evolved beyond simple pattern matching by focusing on path coverage, which allows it to uncover more defects with real run-time implications. By shifting focus from 'suspicious constructs' to 'run-time defects,' new static analysis technologies evaluate more of the intricate interactions within code bases (e.g., values of variables as they are manipulated down a path through the code, or the relationship between how parameters of functions are treated and the corresponding return values). To analyze code with this additional level of sophistication, mature analysis solutions combine path flow analysis with inter-procedural analysis to evaluate what happens when the flow of control passes from one function to another within a given software system.

Coverity Static Analysis uses sophisticated algorithms to identify and triage high risk defects that could result in runtime software crashes and unexpected behavior that affect the integrity of the device on which the software runs. It enables users to find hard to spot runtime defects such as null reference pointers, memory leaks and buffer overflows in the large, complex code bases.

## Coverity® Integrity Manager

This solution provides a centralized defect management workflow that enables developers and managers to quickly view defects, prioritized by risk and impact, in the source code and take the appropriate action to resolve them. Developers and managers can identify defects associated within a particular component of the code, and even identify cases where a defect exists across various code branches. Typically as development teams create various versions of the code base by branching, a large number of defects overlap between different branches due to code duplication.

## Coverity® Integrity Control

Coverity enables organizations to establish and enforce policies based on specific recommendations provided in General Principles of Software Validation document. Static analysis goes a long way to meet the challenge for development teams to ensure code defects do not slip into the field. However, to assist in the regulatory process and to build long-term best practices in the software development process, a Governance, Risk, and Compliance solution is required that builds on the strengths of Coverity Static Analysis and Coverity Integrity Manager. Once the policies have been established, organizations can test against them with the Coverity testing solutions, and quickly visualize areas of risk in the project by component and by specific FDA guidance. Managers and executives get a hierarchical view of risk, can understand the relative effort required to address the defect, and can drill down to details to pinpoint the specific issues or verify that specific safety requirements have been satisfied.

## Applying Coverity Development Testing to FDA Software Validation Guidance

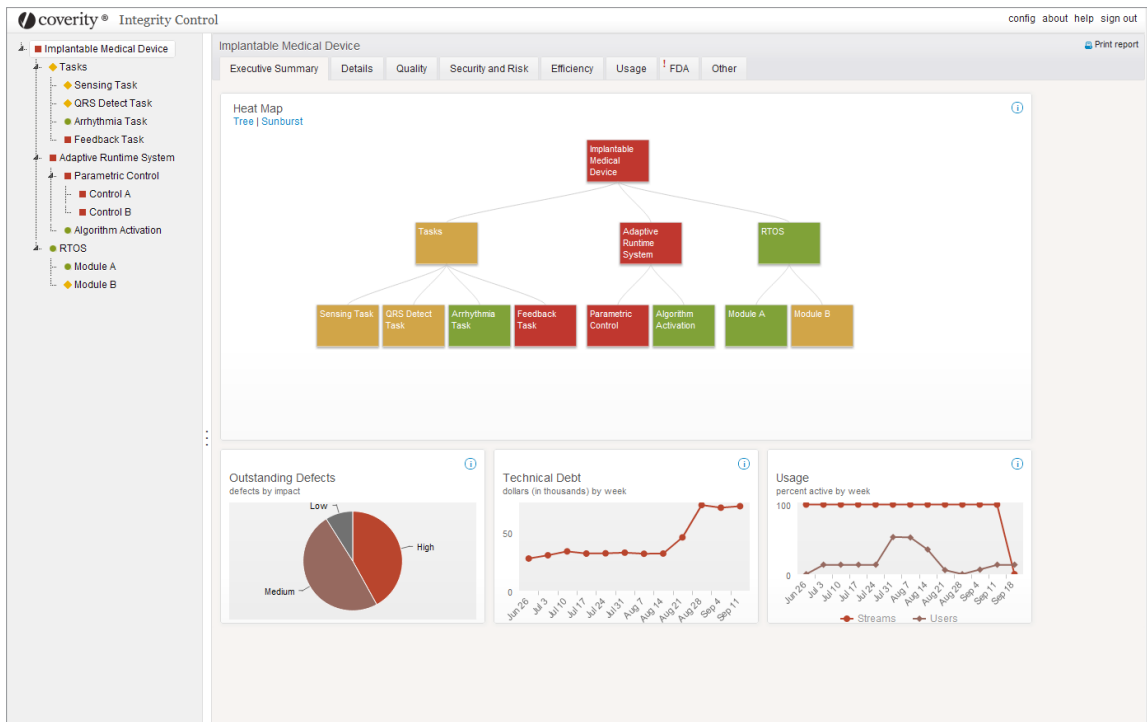


Figure 2: Coverity Integrity Control presents a hierarchical view of risk and policy violations

The latest release of Coverity Integrity Control contains a template that allows out-of-the-box policy management for meeting FDA guidelines for software validation with static analysis. With Coverity Static Analysis, teams can test against the policies from within their IDE or as part of the central build process. Coverity Integrity Control provides real-time visibility into the risk metrics and compliance violations through a visual representation of the areas of risk across their projects, organization and teams. Throughout the development process, the teams now have visibility into the areas of code that do not adhere to the defined policies, and can pinpoint the corresponding violations and the specific defects in code responsible for the violations. Table 1 lists the specific sections from General Principles of Software Validation document issued on January 11, 2002.

**Table 1: Coverity Integrity Control Policies & Metrics for FDA General Principles of Software Validation**

General Principles of Software Validation Section	Description in the Software Validation Guidance	Type of Security Risk
<a href="#">3.1.2 Verification and Validation</a>	Measures such as defects found in specifications documents, estimates of defects remaining, testing coverage, and other techniques are all used to develop an acceptable level of confidence before shipping the product.	FDA 3.1.2 Verification and Validation: Resolved Findings by impact and week
		FDA 3.1.2 Verification and Validation: Outstanding Findings by checker category
		FDA 3.1.2 Verification and Validation: Outstanding Findings Density by impact and node
		FDA 3.1.2 Verification and Validation: Uninspected Findings by impact and week
		FDA 3.1.2 Verification and Validation: Uninspected Findings Density by impact and node
<a href="#">4.2 Defect Prevention</a>	Software quality assurance needs to focus on preventing the introduction of defects into the software development process and not on trying to “test quality into” the software code after it is written. Software testing is a necessary activity. However, in most cases software testing by itself is not sufficient to establish confidence that the software is fit for its intended use.	FDA 4.2: Defect Prevention: Uninspected Findings Age Violations findings by week
<a href="#">4.7 Software Validation after a change</a>	Whenever software is changed, a validation analysis should be conducted not just for validation of the individual change, but also to determine the extent and impact of that change on the entire software system.	FDA 4.7 Software Validation After Change percentage of codebases with analysis by week
<a href="#">5.2.4 Construction or Coding</a>	Firms frequently adopt specific coding guidelines that establish quality policies and procedures related to the software coding process. Source code should be evaluated to verify its compliance with specified coding guidelines. Such guidelines should include coding conventions regarding clarity, style, complexity management, and commenting.	FDA 5.2.4 Construction or Coding: Complexity by node and week
		FDA 5.2.4 Construction or Coding: Comments Density by node and week
<a href="#">5.2.5 Testing by Developer (partial)</a>	Structural testing can identify “dead” code that is never executed when the program is run. Structural testing is accomplished primarily with unit (module) level testing, but can be extended to other levels of software testing.	FDA 5.2.5 Testing by Developer: No unused code found by checkers that detect deadcode

Individual policies can be enabled and disabled, and specific thresholds can be set so that the software development teams can identify which software modules and components are meeting the established policies, which ones are at risk, and which ones are in violation.

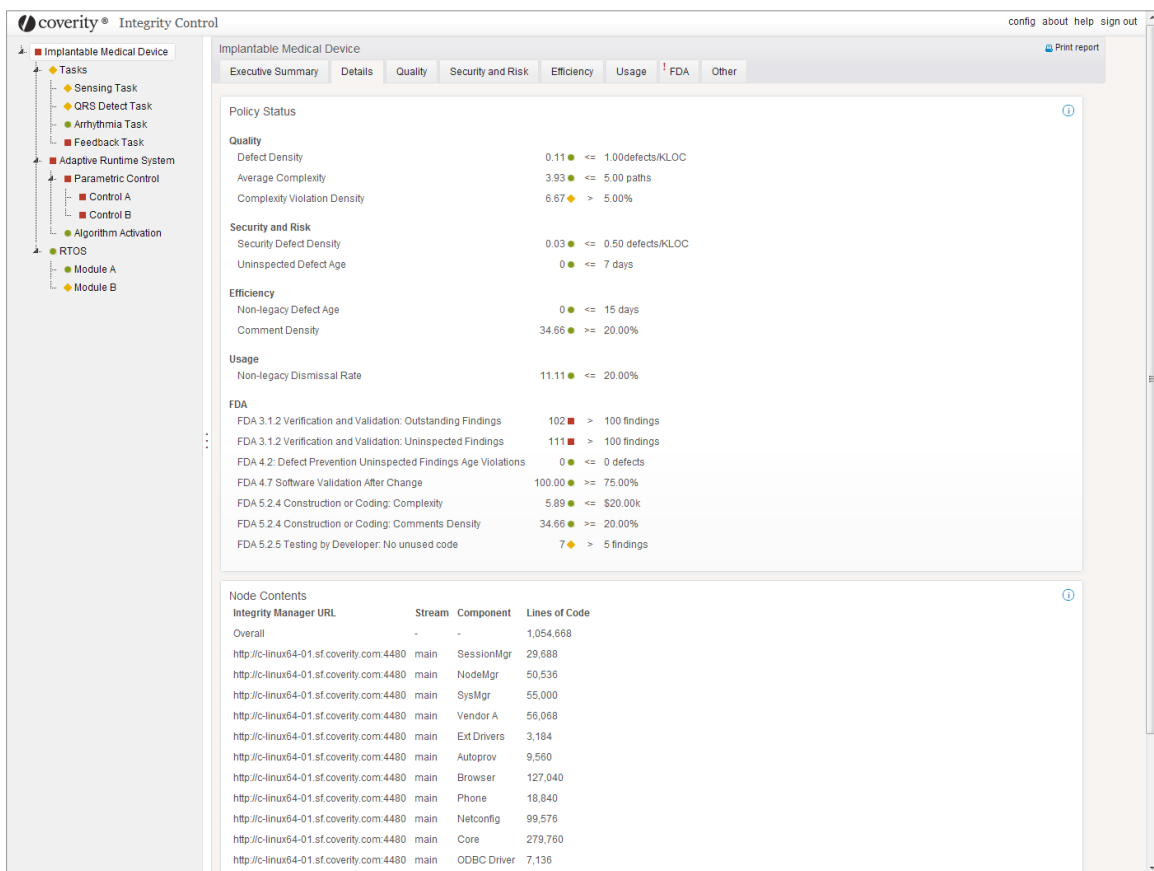


Figure 3: Define policies for specific sections in the FDA guidance for software validation

In addition managers can generate Software Integrity Reports that provide a snapshot into the state of the software. This can be used to provide a report on adherence to these policies as part the FDA pre-market submission process, or for ongoing validation as software is changed post-release. An updated risk profile is produced with every code iteration and test.

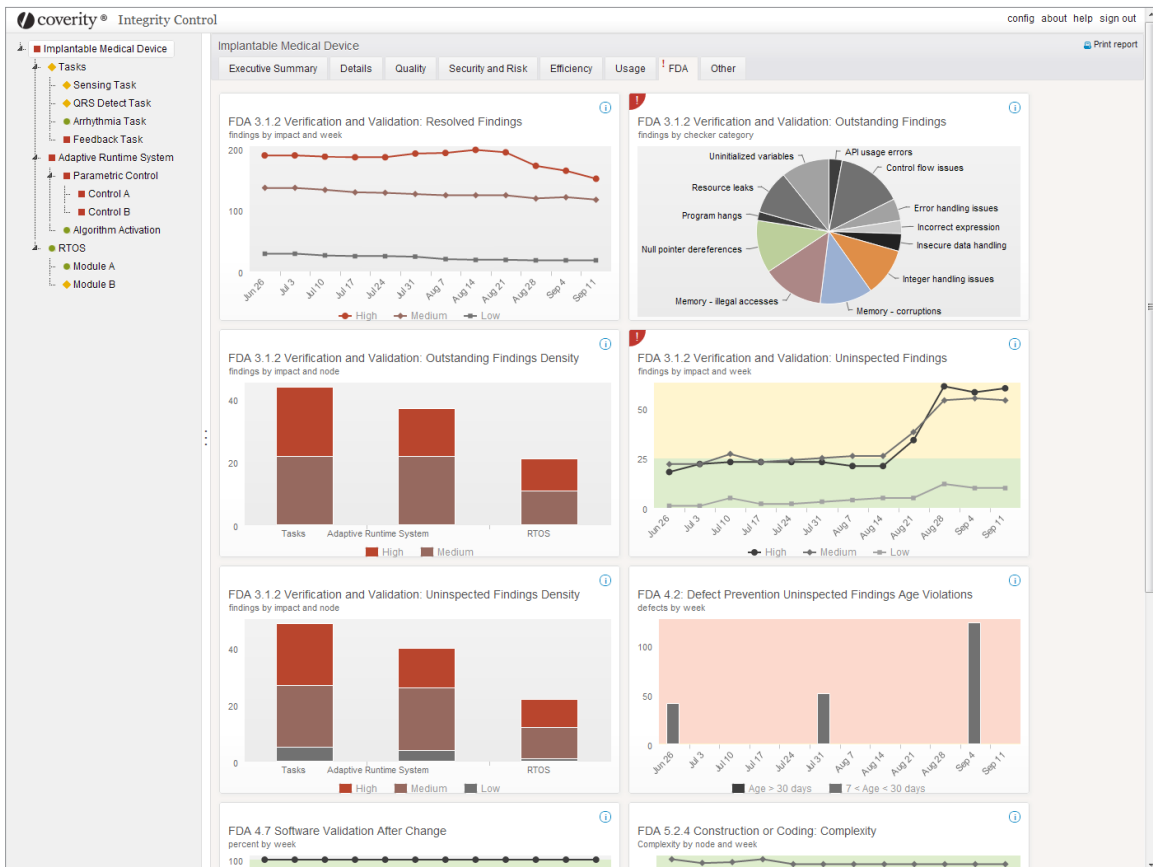


Figure 2: Coverity Integrity Control presents a hierarchical view of risk and policy violations

## Summary

Software can enable medical device manufacturers to deliver innovative devices that help patients lead better lives and create competitive differentiation. However, medical device manufacturers must overcome the challenge of managing the risk of failure and complexity inherent in software. As the amount of software increases in these devices, and as new software development methodologies like Agile are increasingly used in Class II and Class III devices, development testing offers a solution to aid with the software validation process. Coverity provides the industry's first development testing platform designed to meet the regulatory guidelines identified for software validation by the FDA and to control the quality, reliability, and safety of the software throughout development.

Coverity's solutions for code testing have been used by over 1,100 customers to protect the integrity of their software and their brands. It has been the gate to more than 11 billion product shipments in various industries including over 22 million medical devices from leading medical device companies such as Bayer, Hospira, Medtronic, Siemens Healthcare and Stryker. To find out how you can protect the integrity of your software and get started with software code governance, contact Coverity at [www.coverity.com](http://www.coverity.com).

## About Coverity

Coverity, Inc. ([www.coverity.com](http://www.coverity.com)), the development testing leader, is the trusted standard for companies that need to protect their brands and bottom lines from software failures. More than 1,100 Coverity customers use Coverity's development testing suite of products to automatically test source code for software defects that could lead to product crashes, unexpected behavior, security breaches, or catastrophic failure.

**For More Information:**  
[sales@coverity.com](mailto:sales@coverity.com)

**Coverity Inc. Headquarters**  
185 Berry Street, Suite 6500  
San Francisco, CA 94107 USA

U.S. Sales: (800) 873-8193  
International Sales: +1 (415) 321-5237  
[www.coverity.com](http://www.coverity.com)

Coverity and the Coverity logo are trademarks or registered trademarks of Coverity, Inc. in the U.S. and other countries. All other company and product names are the property of their respective owners. © 2008-2011 Coverity, Inc. All rights reserved.

