
Controlling Risk Through Software Code Governance

July 2011

Catastrophic Consequences

Today's headlines are filled with stories about catastrophic software failures and security breaches; medical devices being recalled, gaming systems getting hacked, and credit card information becoming compromised. These events cost companies millions of dollars in brand equity, lost revenue and result in the erosion of customer loyalty and in the most extreme cases; deaths. Recently, an automated external defibrillator (AED) product line was recalled because of software defects that could have resulted in "a failure to resuscitate" also known as patient death. With the higher visibility of field failures, companies are becoming increasingly aware of the critical link between software risk and business risk. And that risk is increasing exponentially as more companies turn to software to differentiate their products. For example, in the automotive industry software and electronics are expected to account for 90% of automobile innovation. Industry experts predict that the average car will contain over 300 million lines of code in the next decade, up from 10 million lines of code in today's cars.

Compounding risk is the increasing reliance companies have placed on third-party code. Almost all companies include code from out-sourcers, the open source or commercial vendors in their products. In some industries such as mobile, it's not uncommon to include code from more than ten different third-party sources. According to Forrester Consulting, third-party code is not tested for quality, safety or security with the same level of rigor as in-house developed code. In a recent Forrester study, more than 40 percent of respondents cited that problems from third-party code resulting in product delays or recalls, security vulnerabilities, an increase in development time, and revenue impact, have caused them to seek greater visibility into code integrity. Given this trend, it's imperative for companies to protect themselves and their customers from this risk. To do so, they need an effective and automated software code governance solution to establish and enforce the highest quality and security standards across internally distributed teams and the software supply chain.

Software Code Governance

Governance is a well established concept in many industries and was born out of necessity. The Sarbanes Oxley Act of 2002 was enacted as a reaction to a number of major corporate scandals including those affecting Enron, Tyco and WorldCom. These scandals cost investors billions of dollars when the stock prices of affected companies collapsed and shook investors' confidence in the stock market. In 2004, Payment Card Industry (PCI) compliance was introduced to protect credit card issuers from credit card fraud by ensuring that merchants meet minimum levels of security when they store, process, and transmit cardholder data.

Software code governance has similar regulatory roots. The concept started in industries subject to regulation such as medical devices and avionics where issues with software quality can have catastrophic consequences. The initial focus of software code governance was to assure software quality and security of in-house developed code by establishing clear guidelines and procedures such as the FDA's recommendation that infusion devices be tested with static analysis and DO-178B, Software Considerations in Airborne Systems and Equipment Certification for the avionics industry. Today, we see software code governance gaining momentum in a wide variety of industries as organizations seek to drive greater accountability and efficiency within distributed development teams and to achieve better visibility and control over third-party code.

Managing the quality and security of code across distributed development teams can be challenging. Companies must contend with language barriers, cultural differences and time zone incompatibility. For example, it is extremely difficult for teams from the United States, Russia, and China to find the right time and mechanisms for effective collaboration. Poor communication can lead to product release delays and missing critical time-to-market windows. Junior team members working in satellite development offices could have skill gaps which could be difficult to identify remotely and could lead to the introduction of excess complexity in the product or technical debt. Technical debt accumulates when teams fail to perform appropriate actions such as refactoring, removing duplication and redundancy. Failure to perform these kinds of activities at the right time can hamper future innovation. To deal with these challenges, companies are looking for code governance to help them establish and enforce clear standards and policies for the quality and security of the code that is produced by the teams and to ensure that unnecessary complexity is not introduced.

In addition to utilizing software code governance to help drive accountability within internal teams, companies are looking to governance to help them establish more control over third-party suppliers. When an issue arises in the code which is comprised of many third-party components, it can be extremely difficult to pinpoint the root cause of the error. Yet according to the Forrester Software Integrity Risk Report, in nearly one out of every two cases, the buyer side is held 100 percent responsible for quality and security issues found in third party code, compared to one in every ten cases where the third party supplier is held 100 percent accountable. The buyer assumes the responsibility and associated risk when software issues arise. To protect themselves from this risk, companies are demanding increased visibility into the quality and security of third-party code. Software code governance enables organizations to establish clear prescriptive policies with their suppliers and helps create a greater level of transparency into areas of risk.

A Multi-Step Process

Software code governance cannot be achieved with the click of a button. It's a process which needs to be embraced by the organization and enforced across the internal and external supply chain. The process will vary by organization based upon whether you are trying to establish governance across internal teams, with outsourcers, offshore development teams, or partners. However, there are some basic tenants which apply to all organizations:

Prior to implementing a governance process:

Embrace the need for change: Often, changing human behavior can be the biggest obstacle in any new process. Today's headline news about product recalls, security breaches, and deaths caused by faulty software clearly communicate the need for change for some organizations. For others, they will need to consider the potential headlines or implications that could result if they were to have a major defect found in the field, missed a critical product window, or experienced a product recall. They need to carefully consider the potential risk that a lack of effective software code governance poses to their brands, revenues, and customers.

Articulate the business problem:

Companies must be able to articulate the business problem they are trying to address whether it is maintaining high quality, security, and efficiency standards across the distributed organization, meeting a critical time-to-market window, or controlling risk across third party suppliers.

Implementing a software governance process:

Step 1: Define policies and thresholds

Organizations need to consider the appropriate policies or thresholds which should be put in place. Establishing realistic thresholds and service level agreements requires an understanding of today's current state. Companies need to understand the current levels of quality, security, and efficiency to establish appropriate targets. In some industries, that could mean they must not have any defects in their code, for other organizations, they may implement a phased approach to quality and security improvement.

Step 2: Test

Organizations need to test against established policies to ensure targets are being met.

Step 3: Control

Finally, companies need to consider how to effectively measure and report compliance with the established policies. They need improved visibility across the organization and supply chain to ensure violations are addressed and the appropriate actions are taken to bring the teams and projects back into compliance.

To effectively implement software code governance, organizations must embrace a multi-step process and establish clear and measureable objectives.

Coverity® Integrity Control

Coverity Integrity Control is part of a three step process for software code governance. It enables software development organizations to set policies for code quality and security, and then manage, monitor and report on these policies as code is tested. With Coverity Integrity Control, companies can automatically manage and enforce standard code testing policies across in-house development teams, outsourced development teams, and software provided by third-party suppliers, gaining deep visibility into development risk across the software supply chain.

Defining Software Policies and Thresholds

Implementing an automated approach to code governance begins with establishing clear, specific and actionable policies for quality and security. Coverity enables users to establish and enforce policies based on what's most critical to the business. Once defined, the policies are centrally published and then shared with geographically dispersed teams and with software suppliers. Specific policies which can be defined include:

Quality policies: Coverity Integrity Control enables organizations to establish policies for defect density, critical defects and uninspected defects. Defect density represents the number of outstanding high or medium-risk defects per 1000 lines of code. Critical defect policies can be established as well as thresholds for uninspected defects. If a company is close to their release date then they would likely establish a policy for zero uninspected defects since these defects could represent a risk to the overall code quality.

Security policies: Coverity Integrity Control enables organizations to establish and enforce policies for defects identified as security risks by the industry standard Common Weakness Enumeration, establish policies for security defect density and web security defects.

R&D Efficiency: Organizations can also establish policies for their internal teams and third-party suppliers for critical metrics tied to R&D efficiency. Organizations can establish thresholds for cyclomatic complexity which measures the number of linearly independent paths through a program's source code. Overly complex code is a leading contributor to technical debt and can lead to maintenance issues and difficulty with future innovation. Organizations can also establish thresholds for comment density. Comment density measures the amount of comments per line of code. Comments in the code help people understand what the code is intended to do and how it works. These comments can be particularly important when taking delivery of code from a third-party or for employees in large development organizations with high attrition rates. Without code comments, developers can waste valuable time and energy trying to assess what the code is designed to do and how it functions.

Usage and Savings: Coverity Integrity Control enables organizations to establish policies around the usage of Coverity® Static Analysis and Coverity® Dynamic Analysis. Policies can be established for the number of active users, projects and lines of code being scanned. This can be a critical component of enforcing code quality and security across the organization and supply chain.

The screenshot displays the Coverity Integrity Control web interface. On the left is a navigation tree for 'Mobile Android Based' with sub-items like Applications, Game A, Game B, Browser, Phone, Frameworks, 3rd Party, OpenGL, SQLite, Gaming, Kernel, Bluetooth Wi-Fi 3.0, and 3G GSM. The main content area has tabs for Executive Summary, Details, Quality, Security and Risk, Efficiency, Usage, and Other. The 'Quality' tab is active, showing a 'Policy Status' section with metrics for Defect Density, High Impact Defects (51), Security Defect Density (0.14), Web Security Defect Density (0.00), High Impact Defect Age (0), Comment Density (54), and Active Streams (100.00). Below this is a 'Node Contents' table listing CIM URLs, Stream, Component, and Lines of Code for various system components.

CIM URL	Stream	Component	Lines of Code
Overall	-	-	269,252
http://c-linux64-01.sf.coverity.com:4480	main	SessionMgr	7,422
http://c-linux64-01.sf.coverity.com:4480	main	NodeMgr	12,634
http://c-linux64-01.sf.coverity.com:4480	main	SysMgr	13,750
http://c-linux64-01.sf.coverity.com:4480	main	Vendor A	28,034
http://c-linux64-01.sf.coverity.com:4480	main	Ext Drivers	796
http://c-linux64-01.sf.coverity.com:4480	main	Autoprov	2,390
http://c-linux64-01.sf.coverity.com:4480	main	Browser	6,352
http://c-linux64-01.sf.coverity.com:4480	main	Phone	942
http://c-linux64-01.sf.coverity.com:4480	main	Netconfig	49,788

Automated Policy Management

Testing Often and Early

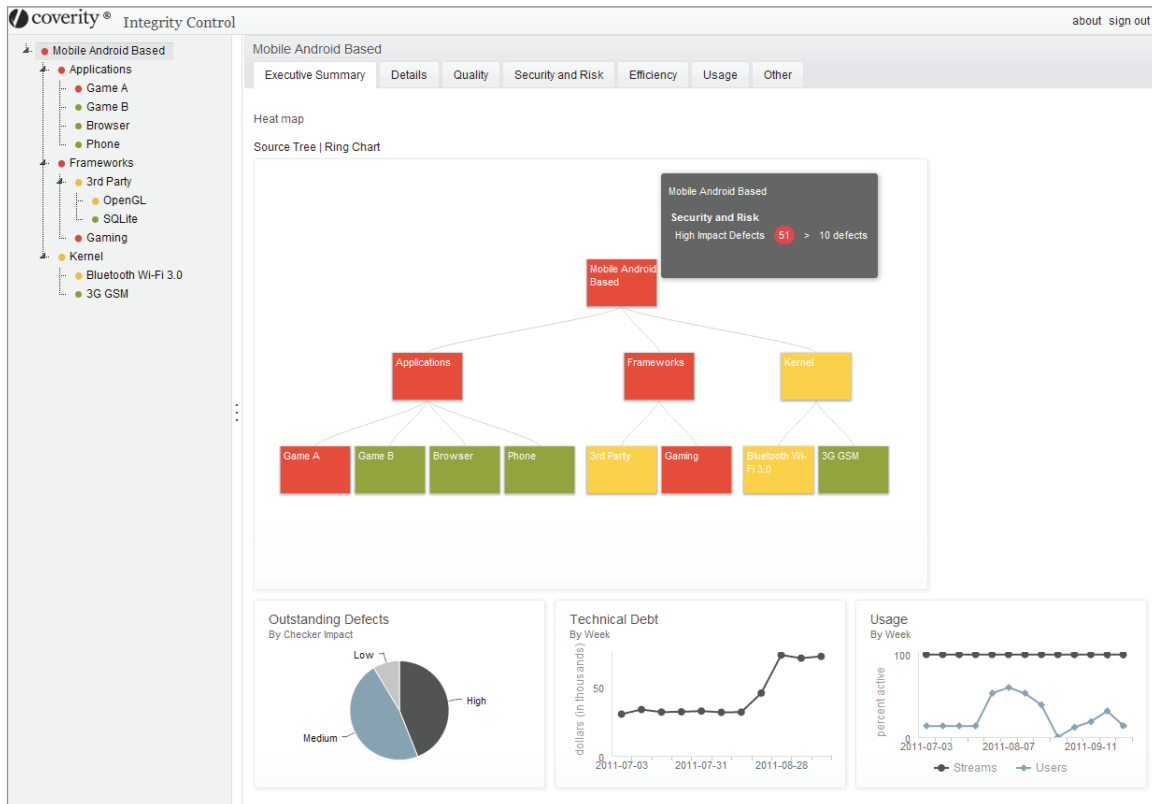
Once the policies have been established, the next step is to test against those policies. Coverity’s developer testing solutions enable teams to test against the established policies while the code is still in development, where issues are least expensive and time consuming to fix. Coverity Static Analysis and Dynamic Analysis use sophisticated algorithms to identify and triage high risk defects that could result in software crashes, security breaches, or safety issues. It enables users to find hard-to-spot issues such as null reference pointers, memory leaks, and potentially exploitable security flaws in the largest, most complex code bases. Once defects are found, developers are automatically notified of quality or security policy violations within their existing workflow, prioritized by risk and impact, so they know what problems to fix first, and report on progress towards compliance with policies. And Coverity fits within customers’ existing workflow so that a continuous approach to quality and security is applied to each build or iteration.

Coverity® Integrity Manager										
Dashboard Projects Integrity Control										
Defects Source Reports History Dashboard										
Impact	CID	Checkers	Status	First Detected Stream	Last Detected Stream	Count	Function	File		
▶ Type:	20334	UNINIT	New	VS SDL 1.12.14	DosBox	2	SDL_CreateRGBSurface()	/sdlnwsrcrtaudio/sdl_surface.c		
▶ Category:	20333	UNINIT	New	VS SDL 1.12.14	DosBox	2	SDL_LoadWAV_RW()	/sdlnwsrcrtaudio/sdl_wave.c		
▼ Streams & Snapshots:	20332	TAINTED_SCALAR	New	VS SDL 1.12.14	DosBox	17	SDL_LoadBMP_RW()	/sdlnwsrcrtaudio/sdl_bmp.c		
○ Ever detected in project	20331	TAINTED_SCALAR	New	VS SDL 1.12.14	DosBox	4	ReadChunk()	/sdlnwsrcrtaudio/sdl_wave.c		
○ Currently detected in project	20329	REVERSE_NULL	New	VS SDL 1.12.14	DosBox	2	DX5_DeleteDevice()	/sdlnwsrcrtaudio/windx5/sdl_		
○ Newly detected in project	20328	REVERSE_NULL	New	VS SDL 1.12.14	DosBox	2	DX5_DeleteDevice()	/sdlnwsrcrtaudio/windx5/sdl_		
○ Newly eliminated in project	20327	REVERSE_NULL	New	VS SDL 1.12.14	DosBox	2	RunThread()	/sdlnwsrcrtaudio/windx5/sdl_		
● Detected in all streams	20326	RESOURCE_LEAK	New	VS SDL 1.12.14	DosBox	2	DX5_SetVideoMode()	/sdlnwsrcrtaudio/windx5/sdl_		
○ Currently detected in all streams	20324	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	18		/sdlnwsrcrtaudio/sdl_riacc		
○ Completely eliminated from all streams	20323	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	2		/sdlnwsrcrtaudio/sdl_riacc		
○ Partially eliminated	20322	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	2		/sdlnwsrcrtaudio/sdl_riacc		
○ Custom edit...	20321	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	2		/sdlnwsrcrtaudio/sdl_riacc		
▶ Checker:	20320	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	4		/sdlnwsrcrtaudio/sdl_riacc		
▼ Status:	20319	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	36		/sdlnwsrcrtaudio/sdl_riacc		
☑ New	20318	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	4		/sdlnwsrcrtaudio/sdl_riacc		
☑ Triaged	20317	PW.PARAMETER_HIDDEN	New	VS SDL 1.12.14	DosBox	4		/sdlnwsrcrtaudio/sdl_riacc		
☐ Dismissed	20316	NO_EFFECT	New	VS SDL 1.12.14	DosBox	2	DX5_CheckInput()	/sdlnwsrcrtaudio/windx5/sdl_		
☐ Fixed	20313	FORWARD_NULL	New	VS SDL 1.12.14	DosBox	2	SDL_SetGammaRamp()	/sdlnwsrcrtaudio/sdl_gamm		
▶ Count:	20312	FORWARD_NULL	New	VS SDL 1.12.14	DosBox	2	UnEscapeQuotes()	/sdlnwsrcrtaudio/windx5/sdl_		
▶ Function:	20311	FORWARD_NULL	New	VS SDL 1.12.14	DosBox	2	ParseCommandLine()	/sdlnwsrcrtaudio/windx5/sdl_		
▶ File:	20310	DEADCODE	New	VS SDL 1.12.14	DosBox	2	WIN_GL_GetAttribute()	/sdlnwsrcrtaudio/windx5/sdl_		
	20309	DEADCODE	New	VS SDL 1.12.14	DosBox	2	WinMessage()	/sdlnwsrcrtaudio/windx5/sdl_		
	20308	CONSTANT_EXPRESSION_RESULT	New	VS SDL 1.12.14	DosBox	2	DX5_CheckInput()	/sdlnwsrcrtaudio/windx5/sdl_		
	20307	BAD_SIZEOF	New	VS SDL 1.12.14	DosBox	2	SDL_InvstickClose()	/sdlnwsrcrtaudio/windx5/sdl_		

Automatically view defects across branches of development for faster resolution time


Controlling Risk across Internal Teams, Offshore Resources and Third Party Suppliers

Once organizations have established their policies and tested against them, it's critical that they have visibility into the risk across the organization and the software supply chain. Coverity Integrity Control provides customers with a visual representation of the areas of risk across their projects, organization and teams. Customers can view a hierarchical heat map that's tailored specifically to the needs of their organization. Some may want to view distributed teams to ensure that the various teams are executing consistently and to surface any potential areas of risk or skills gaps. Other organizations may want to track a portfolio of products while others may want to view a particular project with the drill down to the components that are produced by internal and external teams. Coverity Integrity Control provides the visibility and control needed to consistently measure internal teams and suppliers against standard quality and security SLAs, and automatically audit for SLA violations on-demand. Users can drill down into each policy to pinpoint the full context of the code problem, the specific policy in violation, and where it originated. In addition, this console rides on top of the developer testing platform that the teams are using to find and fix defects. An updated risk profile is produced with every code iteration and test.



Coverity Integrity Control provides executive level visibility of risk

Once areas of risk are identified, users can automatically notify teams or third-party suppliers of code governance violations by automatically producing and sending a Coverity Software Integrity Control Report that summarizes the high risk defects that exist in their software and components. Suppliers can also take a proactive approach and build their own policies aligned to established SLAs and self-certify their code upon delivery to their supply chain partners.

			
<h1>Software Integrity Report</h1>			
Project Name: Mobile Android Based (FOO)			
Version: Unknown			
Project Description: Mobile Android Based (FOO)			
Lines of Code Inspected: 14,935			
		Integrity Level RED	
Customer Name:	Coverity	Coverity Product:	Coverity® Static Analysis
Point of Contact:	Customer Name	Product Version:	5.3.0
Customer email:	customer@company.com	Coverity Point of Contact:	Coverity Contact Name
Report Date:	Jun 21, 2011 4:47:27 PM	Coverity email:	contact@coverity.com
Report ID:	05f56b00-61f4-4e22-9e57-a109ee46abfb		
<p>The Coverity® Software Integrity Report provides a summary of the compliance status of a particular project, component or team based upon the policies established in Coverity Integrity Control™ and the results of the testing completed with Coverity® Static Analysis and/or Coverity® Dynamic Analysis. In addition to the summary, this report also provides a detailed breakdown of the areas in violation of established policies, at risk of violating established policies and in compliance with established policies. Organizations can use this report to facilitate a discussion with internal teams and third party suppliers.</p> <p>Copyright © 2004-2011 Coverity, Inc. All rights reserved worldwide.</p> <p>COVERITY CONFIDENTIAL. The information contained in this document is the proprietary and confidential information of Coverity and its licensors, and is supplied subject to, and may be used only by Customer in accordance with the terms and conditions of a license agreement previously accepted by Coverity and Customer.</p>			

Automatically communicate with suppliers when they are out of compliance

Summary

Every day we place our lives in the hands of software, from the brakes in our automobiles, to the air traffic controller systems, to medical devices. And companies, often unwittingly, place their reputations and their revenues in the hands of their third-party suppliers. Automated software code governance is critical to protecting consumers and companies alike. It provides companies with a means to establish and enforce prescriptive requirements and policies around software quality, security, and safety for the code they produce and that which they receive from their software supply chain partners. Coverity has 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 3.5 billion product shipments. And today, it is at the forefront in providing solutions for companies to implement software code governance across their internal development teams and third-party suppliers.

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

For More Information:
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

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.

