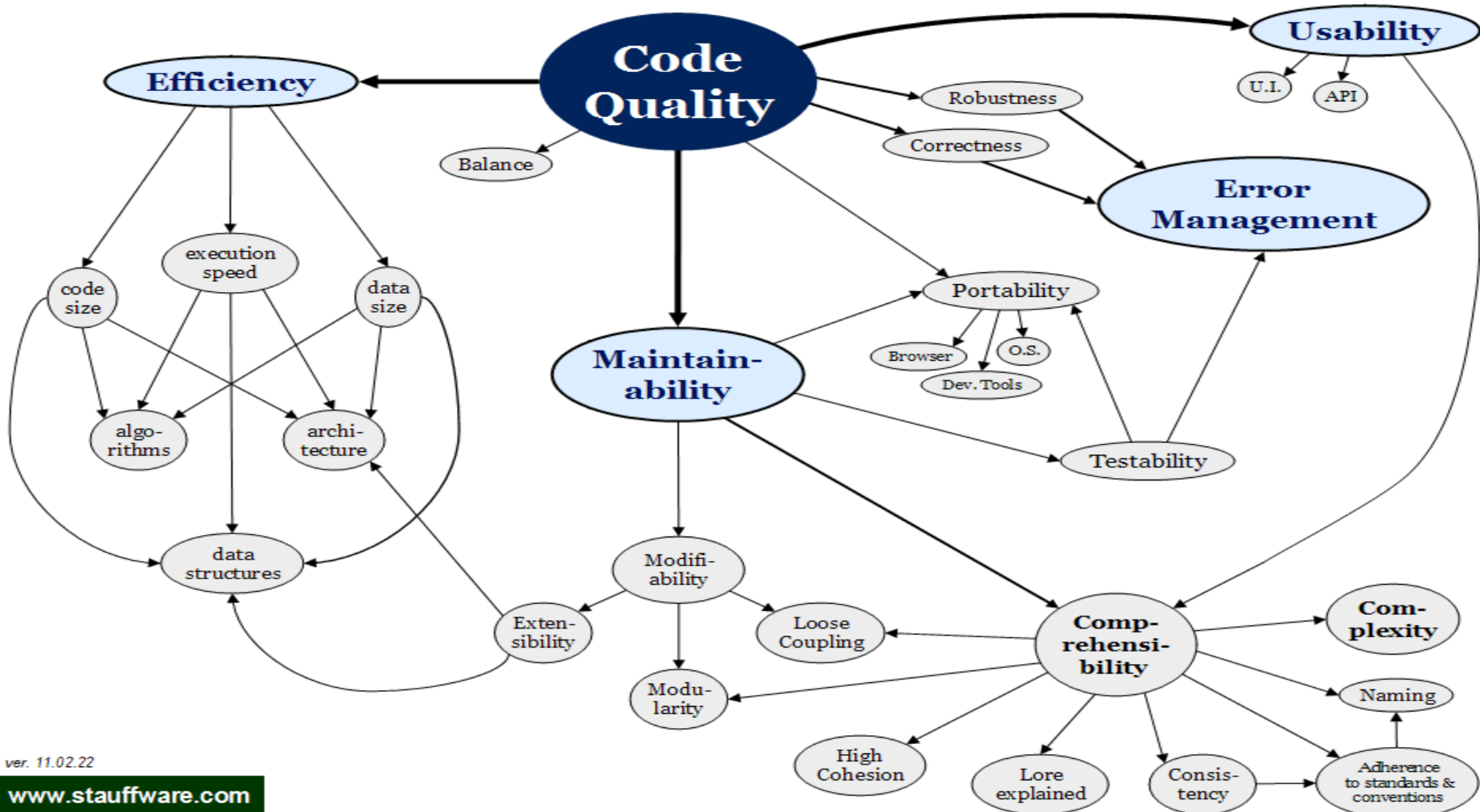


SonarQube

# Overview

- Code Quality (What, Why and When)
- The 7 Axes of Quality and then Technical debt
- SonarQube Introduction
- Demo

# What is Code Quality?




# What is Code Quality?

- In Nutshell – It's a indicator about how quickly developers can add business value to a software system

# Why measure?



- Source code is the heart of each system 
- Developers don't write new software. They maintain “legacy” systems
- A system is (almost) never “finished”
- You can't improve if you don't measure
- Broken window theory



# When you should measure?

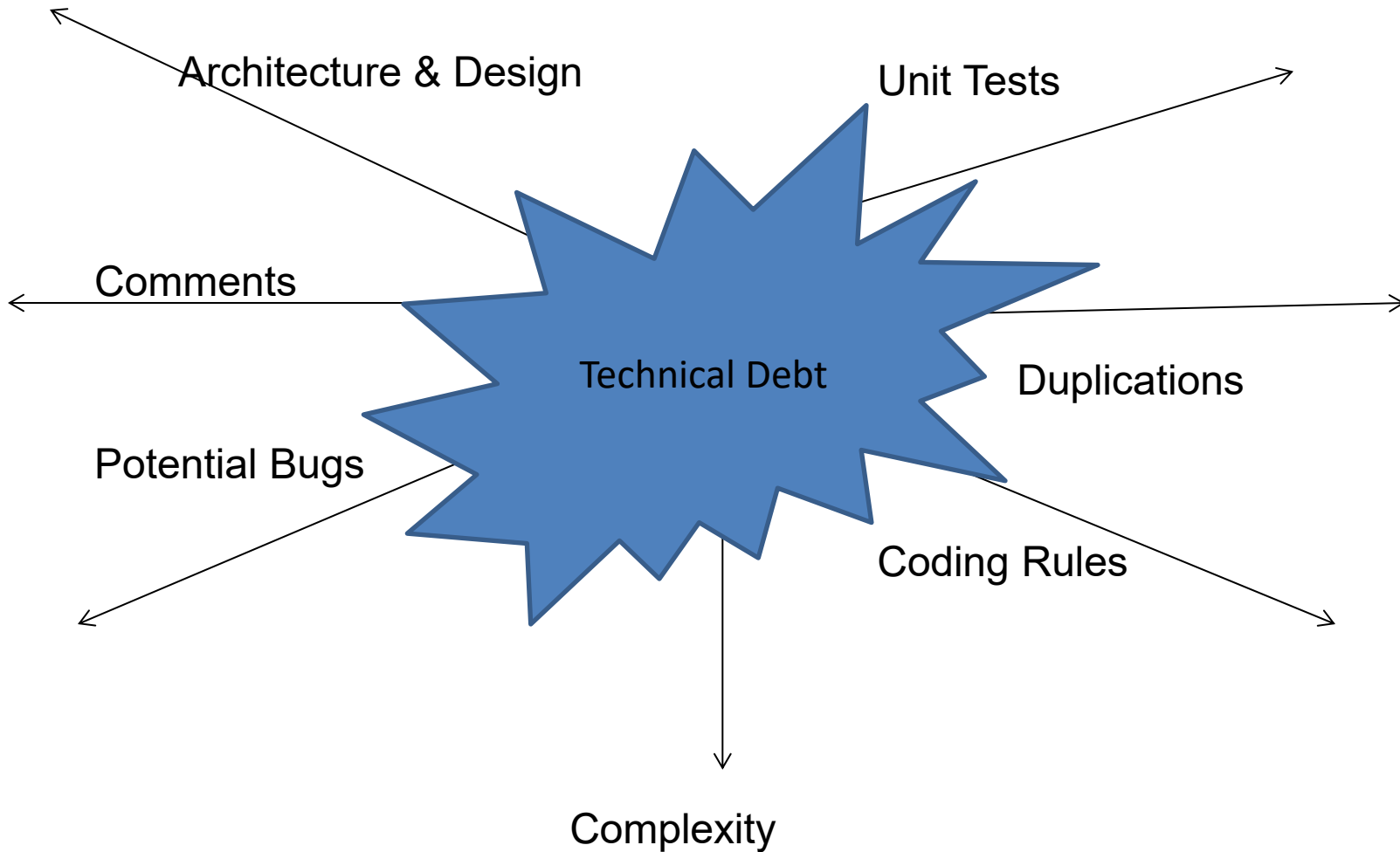
- From project day #0
- Continuously
- Prevent vs post-actions
- Prioritize and plan



# What you should measure?

- Not just abstract numbers
- Evolution through time
- Metrics?
- Welcome to the 7 deadly sins of devs

# The Seven Axes of Quality



# Technical Debt



*“If the debt grows large enough, eventually the company will spend more on servicing its debt than it invests in increasing the value of its other assets”*



**Steve McConnell**

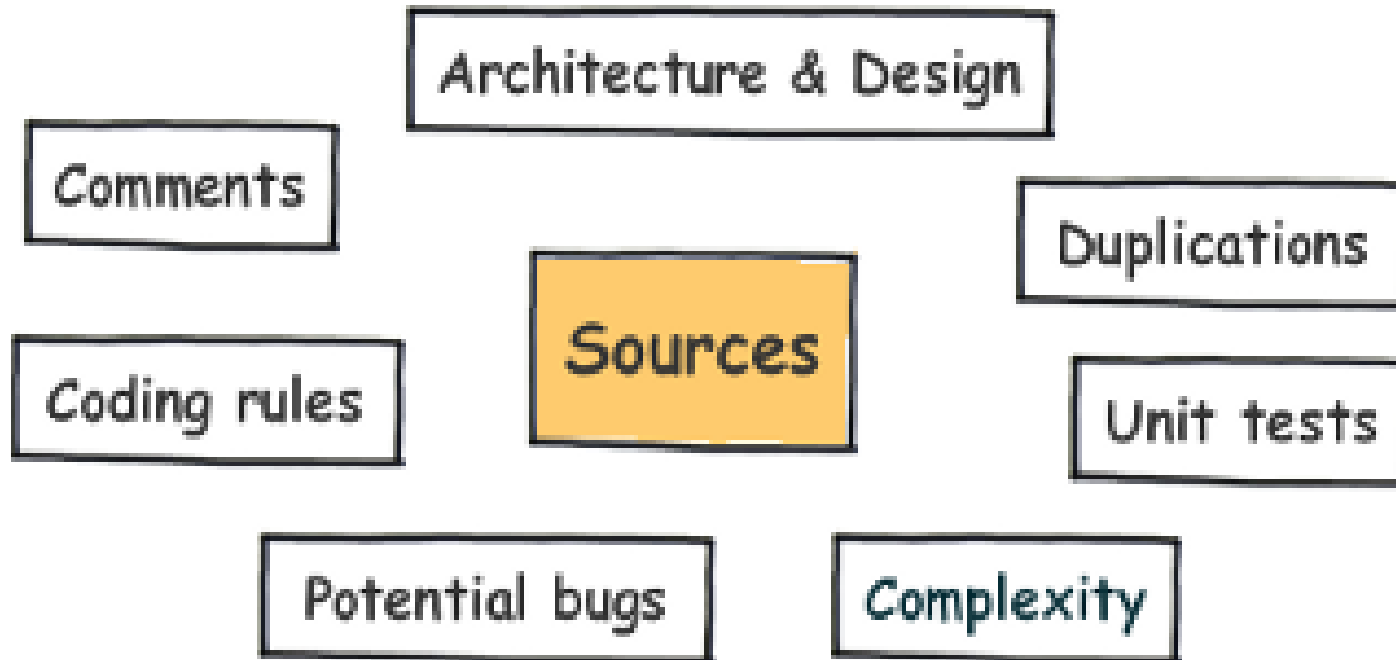
(author of code complete)



This is your source code  
when you don't pay your  
technical debt

# What is SonarQube?

It is a open platform to Manage Code Quality!



# In Another Word.

- SonarQube is a web-based application.
- Rules, alerts, thresholds, exclusions, settings... can be configured online.
- By leveraging its database, SonarQube not only allows to combine metrics altogether but also to mix them with historical measures.
- Got a very efficient way of navigating, a balance between high-level view, dashboard, TimeMachine and defect hunting tools.



# What SonarQube is / does

- Free & open source “Code Quality Platform”
- Provides moment-in-time quality snapshots
- Gives trends of lagging and leading indicators
- Tracks developers’ seven deadly sins (seven axes of quality )

# SonarQube Features

- Continuous Inspection
- Detect Tricky Issues
- Centralize Quality
- DevOps Integration

# Continuous Inspection

SonarQube provides the capability to not only show health of an application but also to highlight issues newly introduced. With a Quality Gate in place, you can fix the leak and therefore improve code quality systematically.

# Detect Tricky Issues

SonarQube code analyzers are equipped with powerful path sensitive dataflow engines to detect tricky issues such as null-pointers, dereferences, logic errors, resource leaks...

# Centralize Quality

One place to provide a shared vision of code quality for developers, tech leads, managers and executives in charge of a few to a few thousands projects and also to act as a toll gate for application promotion or release.

# DevOps Integration

SonarQube integrates with the entire DevOps toolchain including build systems, CI engines, promotion pipelines... using webhooks and its comprehensive RestAPI.

# Multi-Language

With SonarQube comes a code analyzer for each major programming language. Each analyzer provides numerous rules to spot general and language-specific quality issues.

C/C++	PHP	Objective-C
JavaScript	ABAP	Swift
C#	VB.NET	Web
Java	VB6	XML
COBOL	Python	
PL/SQL	RPG	
PL/I	Flex	

# How does it work?



- Analyzes source code and byte code
- Computes hundreds of metrics
- Associates metrics with analysis snapshots
- Shows the results in dashboards and widgets accessible by any browser

# SonarQube for everything



- Initially developed only for Java projects
- Today supports over twenty languages

Commercial : ABAP, C, C++, Cobol, Natural, PL/SQL,  
Visual Basic

Open Source : C++, C#, Flex, Groovy, Android,  
Javascript, PHP, Python, XML,  
Web(xhtml, jsp , jsf, )

# ... and for everyone



For developers. Is my code “good”?  
How can I improve it?



For testers / QA staff. Which parts of  
the system lack unit testing?



For architects. Is the initial design  
“broken”? How about complexity?



For managers. Give me the numbers!!  
Are we going up or down?

# Managing code quality

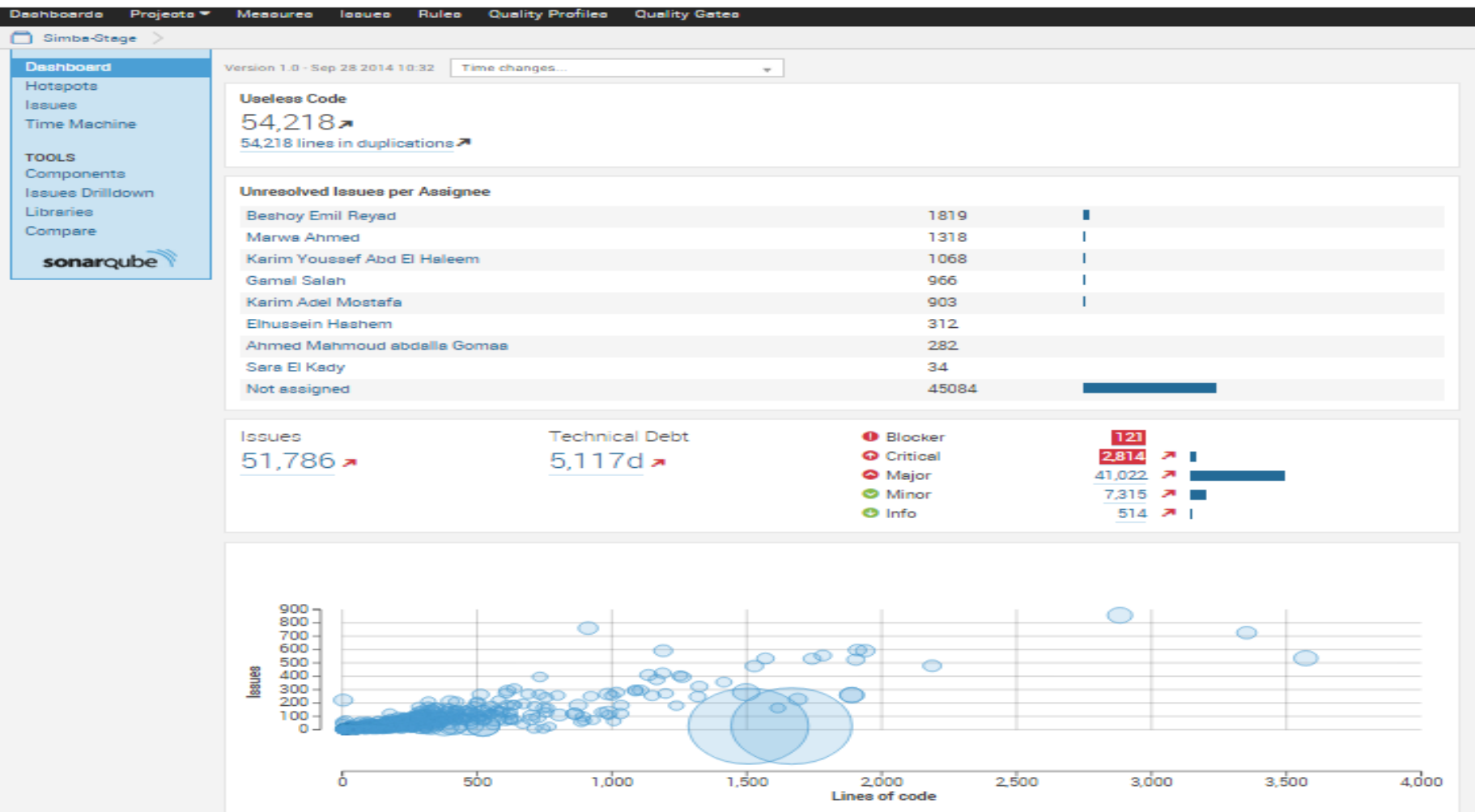


- Dashboards
- Historical data
- Differential views
- Compare service
- Code reviews
- Action plans

# Extend with Plugins

Covering new languages, adding rules engines, computing advanced metrics can be done through a powerful extension mechanism. More than 50 plugins are already available.

# Managing code quality

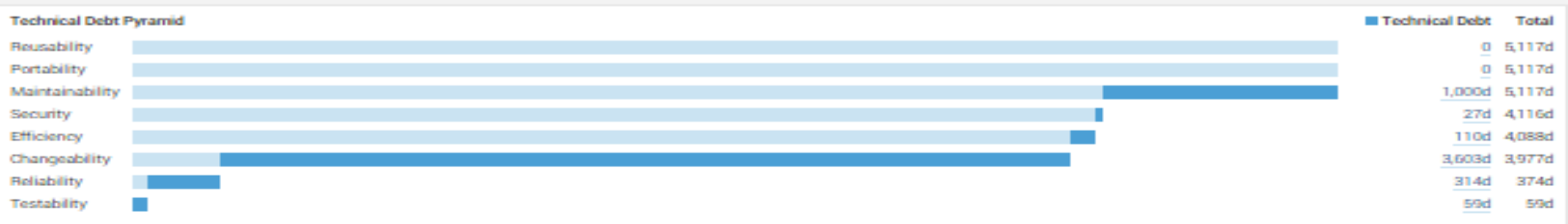


# Sonar – Basic statistics



**Documentation**  
 36.4% docu. API  
 12,673 public API  
 8,054 undocu. API

**Comments**  
 17.7%  
 49,797 lines



**Most Violated Components**

File	Issue 1	Issue 2	Issue 3	Issue 4	Issue 5
ElementDescriptor.java	0	0	757	3	0
SimbaSearch.java	0	41	610	204	1
SimbaWizard.java	0	19	567	131	9
RepairResellers.java	6	85	409	81	10
HTMLExtended.java	0	3	580	8	0

**Line of Code**  
 230,818

**File**  
 1,358

**Functions**  
 11,960

Java

Directories	342	Lines	358,399
Classes	1,460	Statements	132,599
Accessors	5,657		

**Duplications**  
 26.4%

Lines	94,654	Blocks	28,382	Files	635
-------	--------	--------	--------	-------	-----

# Sonar – Project Drill Down

ISSUES New Search Save As

Project: Simba-Stage Severity: Blocker, Critical, M... Status: Open Assignee: All Resolution: Unresolved Language: Java Rule: All + More Criteria Search

Found: 10,000

Only the first 10,000 issues matching the search criteria have been retrieved. Add some additional criteria to get fewer results to be able to sort this list.

Simba-Stage

WEB-INF/classes/ChangeSpeedWS/ChangeSpeedWSSoapBindingStub.java

128 Lines of code 3d 3h 22 Debt 22 Issues 87.5% Duplicated lines (%) SCM

Major Open about a month  
Rename this local variable name to match the ...  
WEB-INF/classes/ChangeSpeedWS/ChangeSpeedWSS...

Major Open about a month  
Move this "else" on the same line that the prev...  
WEB-INF/classes/ChangeSpeedWS/ChangeSpeedWSS...

Major Open about a month  
Rename this local variable name to match the ...  
WEB-INF/classes/ChangeSpeedWS/ChangeSpeedWSS...

Major Open about a month  
Replace the synchronized class "Vector" by an...  
WEB-INF/classes/ChangeSpeedWS/ChangeSpeedWSS...

Blocker Open about a month  
Catch Exception instead of Throwable.  
WEB-INF/classes/ChangeSpeedWS/ChangeSpeedWSS...

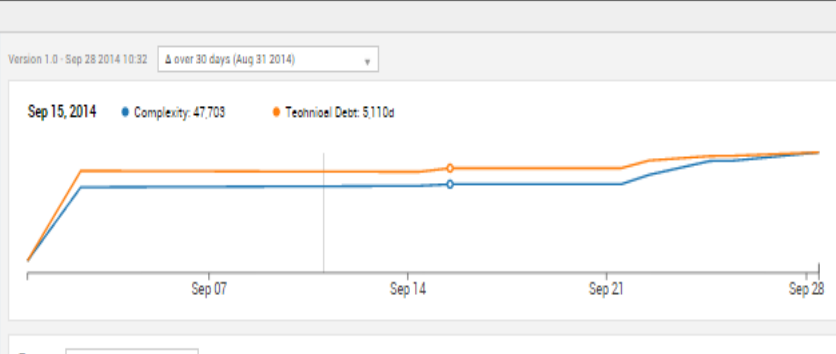
```
104     _call.setOperationName(new javax.xml.namespace.QName("urn:ChangeSpeedWS", "returnHello"));
105
106     setRequestHeaders(_call);
107     setAttachments(_call);
108     java.lang.Object _resp = _call.invoke(new java.lang.Object[] {helloWorld});
109
110     if (_resp instanceof java.rmi.RemoteException) {
111         throw (java.rmi.RemoteException)_resp;
112     }
```

Rename this local variable name to match the regular expression "[a-z][a-zA-Z0-9]\*\$".  
Comment Open Confirm Resolve False Positive Assign (to me) Plan Change Severity Debt: 20min

# Sonar – Time Machine



- Dashboard
- Hotspots
- Issues
- Time Machine**
- TOOLS
- Components
- Issues Drilldown
- Libraries
- Compare



Events All

Sep 28 2014	Version	1.0
Aug 21 2014	Quality Gate	Red
Aug 21 2014	Quality Profile	Changes in 'Sonar way with Findbugs' (Java)
Aug 19 2014	Quality Profile	Sonar way with Findbugs version 1

Duplications **26.4%** (+0.2)

Lines	Blocks	Files
94,654 (+984)	28,382 (+217)	635 (+6)

Complexity **4.0** (+0.0) /function

**32.8** (+0.0) /class

**35.2** (+0.1) /file

Total: **47,865** (+551)

	Aug 31 2014	Sep 28 2014
Issues	51,108	51,786
B blocker issues	121	121
C critical issues	2,729	2,814
M major issues	40,561	41,022
Minor issues	7,198	7,315
I info issues	499	514
Weighted issues	143,736	145,661

Simba-Stage Simba

TEData Billing System

Profiles: [Sonar way with Findbugs \(Java\)](#)

Quality Gate: [SonarQube way \(Default\)](#)

Documentation <b>36.4%</b> docu. API (+0.2)	Comments <b>17.7%</b> (+0.0)
12,673 public API (+125)	49,797 lines (+846)
8,054 undocu. API (+54)	

	Aug 31 2014	Sep 28 2014
Coverage		1.0
Line coverage		
Branch coverage		
Unit tests success (%)		
Unit tests failure		
Unit tests error		
Unit tests		
Unit tests duration		

# The big picture



- Track and reduce Technical Debt on an ongoing basis. (Clean up kitchen every day)
- Engage all developers from project day #1 (Not only mums wash the dishes)
- Get alerted when Technical debt is beyond a threshold (when someone is leaving the kitchen in a mess)

# Terminology

- Analyzer
- Database
- Server
- Bug
- Coding Rule
- Code Smell
- Component
- Remediation Cost
- Technical Debt
- Issue
- Leak Period
- Measure
- Metric
- Quality Profile
- Vulnerability

<https://docs.sonarqube.org/display/SONAR/Concepts>

# Pre Requisites

JDK 1.7 +

<http://docs.sonarqube.org/display/SONAR/Requirements>

# Download

- <http://www.sonarqube.org/downloads/>

# Install

1. Download and unzip the SonarQube distribution (let's say in "C:\sonarqube" or "/etc/sonarqube")

2. Start the SonarQube server:

# On Windows, execute:

```
C:\sonarqube\bin\windows-x86-xx\StartSonar.bat
```

# On other operating system, execute:

```
/etc/sonarqube/bin/[OS]/sonar.sh console
```

3. Download and unzip the SonarQube Scanner (let's say in "C:\sonar-runner" or "/etc/sonar-runner")

4. Download and unzip some project samples (let's say in "C:\sonar-examples" or "/etc/sonar-examples")

5. Analyze a project:

# On Windows:

```
cd C:\sonar-examples\projects\languages\java\sonar-runner\java-sonar-runner-simple
```

```
C:\sonar-runner\bin\sonar-runner.bat
```

# On other operating system:

```
cd /etc/sonar-examples/projects/languages/java/sonar-runner/java-sonar-runner-simple
```

```
/etc/sonar-runner/bin/sonar-runner
```

# Dashboard Time

Browse the results at <http://localhost:9000>  
(default System administrator credentials are  
admin/admin)

# Integration with MySql

- <http://docs.sonarqube.org/display/SONAR/Installing+the+Server>

# Integration with Maven

<http://docs.sonarqube.org/display/SONAR/Analyzing+with+SonarQube+Scanner+for+Maven>

# Integration with Jenkins

- <http://docs.sonarqube.org/display/SONAR/Analyzing+with+SonarQube+Scanner+for+Jenkins>

# Issue Lifecycle

- Issues are detected automatically.

# Statuses

- After creation, issues flow through a lifecycle, taking on one of five possible statuses:
- **Open** - set by SonarQube on new issues
- **Confirmed** - set manually to indicate that the issue is valid
- **Resolved** - set manually to indicate that the next analysis should *Close* the issue
- **Reopened** - set automatically by SonarQube when a *Resolved* issue hasn't actually been corrected
- **Closed** - set automatically by SonarQube for automatically created issues.

# Resolutions

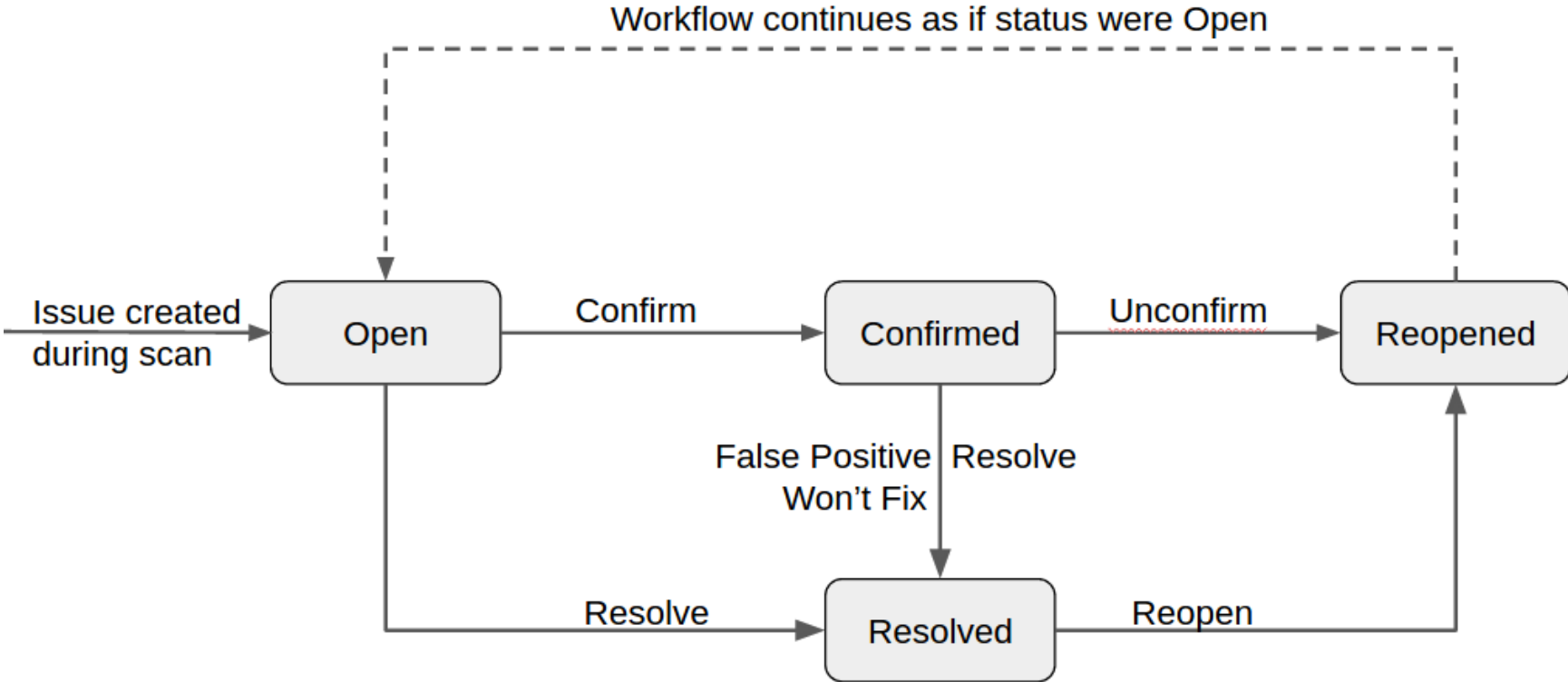
- Closed issues will have one of two resolutions:
  - **Fixed** - set automatically when a subsequent analysis shows that the issue has been corrected
  - **Removed** - set automatically when either the related coding rule or the file is no longer available. The rule may not be available either because it has been removed from the profile or because the underlying plugin has been uninstalled. The file could be unavailable because it has been removed from the project, moved to a different location or renamed.
- Resolved issues will have one of two resolutions:
  - *False Positive* - set manually
  - *Won't Fix* - set manually

# Issues severities

Each issue has one of five severities:

- **BLOCKER**  
Bug with a high probability to impact the behavior of the application in production: memory leak, unclosed JDBC connection, .... The code **MUST** be immediately fixed.
- **CRITICAL**  
Either a bug with a low probability to impact the behavior of the application in production or an issue which represents a security flaw: empty catch block, SQL injection, ... The code **MUST** be immediately reviewed.
- **MAJOR**  
Quality flaw which can highly impact the developer productivity: uncovered piece of code, duplicated blocks, unused parameters, ...
- **MINOR**  
Quality flaw which can slightly impact the developer productivity: lines should not be too long, "switch" statements should have at least 3 cases, ...
- **INFO**  
Neither a bug nor a quality flaw, just a finding.

# Issue workflow



- Issues are automatically closed (status: *Closed*) when:
  - an issue (of any status) has been properly fixed => Resolution: *Fixed*
  - an issue no longer exists because the related coding rule has been deactivated or is no longer available (ie: plugin has been removed) => Resolution: *Removed*
- Issues are automatically reopened (status: *Reopened*) when:
  - an issue that was *Resolved* (but Resolution is not *False positive*) is shown by a subsequent analysis to still exist

# You want to check for new issues before committing your code

- [SonarLint](#) is the answer for this. It raises issues in your IDE *as you type*.

# You want to check for new issues before a Pull Request is merged

- Pull request (PR) analysis allows you to configure your CI engine to perform "preview" analysis (i.e. analysis without updating SonarQube) on PR's. Any new issues are annotated in the PR itself. If none are found, an "all clear" message is added instead. PR analysis is available for GitHub projects with the [GitHub plugin](#), and on some other providers with [Other Plugins](#).

# You want to analyze a long-lived branch in SonarQube

- In this case, you want to maintain analysis of the canonical project in SonarQube, say the "master" branch, *and* also publish in SonarQube ongoing analysis of a secondary branch. The typical case is a release branch. In this scenario, you add the `sonar.branch=[branch key]` analysis property to the release branch to create a second, independent project in SonarQube.

-

# SonarQube Using Docker

```
> docker pull  
library/sonarqube@sha256:e03b6c1f5195676dad7f  
3b5c02b818bf64a213fe34e04e3fb60103034fb50d5  
b  
>
```

Reference

[https://hub.docker.com/\\_/sonarqube/](https://hub.docker.com/_/sonarqube/)

# Installing a Plugin

- There is two options to install a plugin into SonarQube :
  - Automatically, from the SonarQube UI using the Update Center
  - Manually

# Plugins Location

- <https://docs.sonarqube.org/display/PLUG/Plugin+Library>

# Installing a Plugins- Automatically

- If you have access to the internet and you are connected with a SonarQube user having the Global Permission "Administer System", you can go in Settings > Update Center.
  - Locate the "Available Plugins" tab
  - Find the plugin you want to install
  - Click on Install and wait for the download to be processed
- Once done, you will need to restart your SonarQube Server.

# Installing a Plugins- Manual

- Upload the downloaded jar file in your SonarQube Server and put it in the directory : **`$SONARQUBE_HOME/extensions/plugins`**.
- If another version of the same plugin is already there, you need to remove it, since only one version of a given plugin may be available in the extensions/plugins directory.
- Once done, you will need to restart your SonarQube Server.

# Demo Time

# Reference

- <https://docs.sonarqube.org>
- <http://www.devopsschool.com/tutorial/sonarqube/>
- <http://www.scmgalaxy.com/tutorials/content/code-analysis>

# Questions

