Goldman Sachs

Software Engineering: Design & Code Quality

Mohammad Rezaei, PhD

March 2014

Goldman Sachs

Overview

- What is code quality?
- Relationship to project management.
- Picking the appropriate emphasis on quality.
- Architecture & design.
- Some object oriented design guidelines.
- Ensuring code quality.
- Social aspects of code quality.

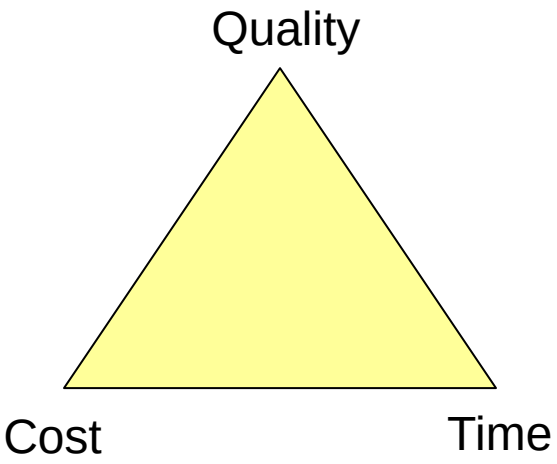
Goldman Sachs

What Is Code Quality?

- The simplest way to think about code quality is to consider it a measure of conformance to set of requirements.
- Functional requirements: how correct is the code?
  - Defects can be measured (in total or as a density)
- Non-functional requirements: sometimes called the “illities”
  - Performance / Capacity / Scalability
  - Testability
  - Reliability / Recoverability / Availability / Resilience
  - Readability / Maintainability
  - Reusability
  - Economy (Resource usage)
  - Extensibility / Agility
  - Manageability / Operability
  - Longevity
  - Security
  - Usability

Relationship to Project Management

- Three variables are recognized as critical to managing software development:
  - Cost
  - Quality
  - Time
- The usual rule is: you can set any two of these, but not all three.
- Which two depends on the particular problem at hand.



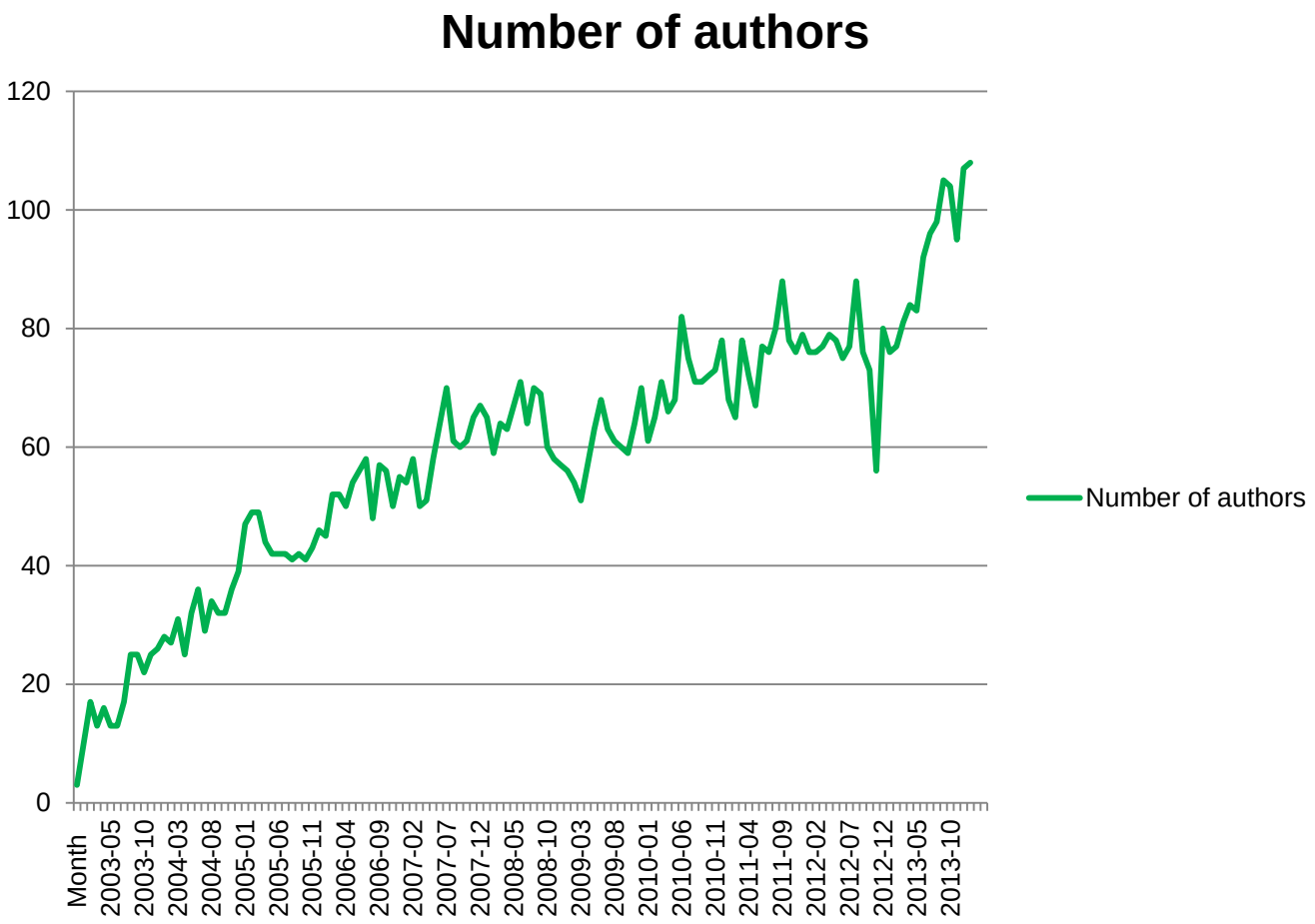
Goldman Sachs

Picking the Right Emphasis on Quality

- Highest possible quality is not always attainable, nor even necessarily desirable.
- The purpose of the code forces a different perspective on the emphasis on quality.
  - Software for an auto-pilot or a heart-lung machine would have to be judged against a much stricter standard than a chat app.
- The longer the expected lifetime of the code, the higher quality has to be.
  - Possibly the biggest difference between classroom and commercial setting.
- The risk of failure also plays a role. There are many types of risk:
  - Heath & safety risk
  - Compliance / legal risk
  - Financial risk
  - Reputational risk

Long Lived Code

- From a code base with 2.5 million lines of code.



Goldman Sachs

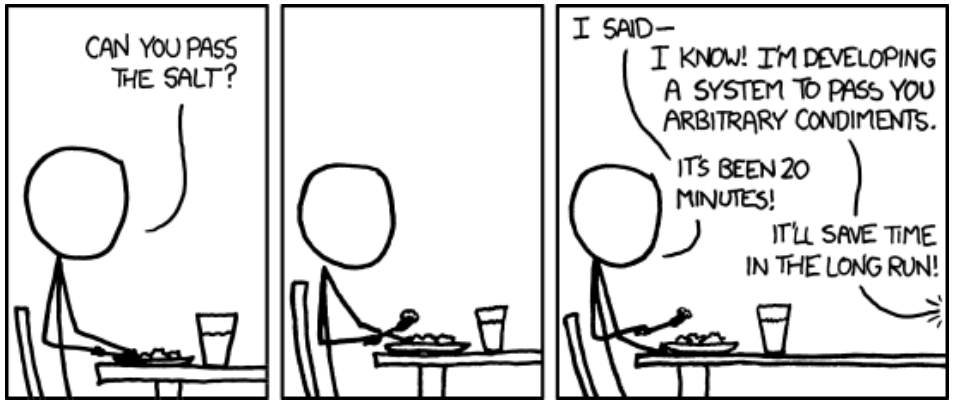
A Few Famous Bugs

- A bug in the code controlling the Therac-25 radiation therapy machine was directly responsible for at least five patient deaths in the 1980s when it administered excessive quantities of X-rays.
- The European Space Agency's Ariane 5 Flight 501 was destroyed 40 seconds after takeoff (June 4, 1996). The US$1 billion prototype rocket self-destructed due to a bug in the on-board guidance software.
- The 2003 North America blackout was triggered by a local outage that went undetected due to a race condition in General Electric Energy's XA/21 monitoring software.
- A Chinook crash on Mull of Kintyre in June 1994, killing 29. An investigation by Computer Weekly uncovered sufficient evidence to convince a House of Lords inquiry that it may have been caused by a software bug in the aircraft's engine control computer.

(from wikipedia)

Architecture & Design

- Architecture: aspects of the software (choices) that are hard to change.
- Design: day to day choices that are generally easier to change.
- Both influence the overall code quality greatly.
- They can also be the hardest thing to get right.
  - This is partly because requirements change and an architecture that was perfect for a given set of requirements might become a poor choice as the requirements evolve.
- "Don't reinvent the wheel" principle: if a given problem has a given solution, stick to it.
  - The naturally leads to cataloguing design and application patterns.
  - Most common/novice mistake: creating a complex solution from patterns where a simpler solution (without the patterns) would've been more serviceable.
- Don't over-engineer.



(from xkcd.com)

Goldman Sachs

Some Object Oriented Design Guidelines

- From http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
- The Interface Segregation Principle
  - Make fine grained interfaces that are client specific.
- The Open Closed Principle
  - You should be able to extend a class's behavior, without modifying it.
- The Liskov Substitution Principle
  - Derived classes must be substitutable for their base classes.
- The Release Reuse Equivalency Principle
  - The granule of reuse is the granule of release.
- The Common Closure Principle
  - Classes that change together are packaged together.
- The Common Reuse Principle
  - Classes that are used together are packaged together.

Goldman Sachs

More Object Oriented Design Guidelines (Dependency)

- The Dependency Inversion Principle
  - Depend on abstractions, not on concretions.
- The Acyclic Dependencies Principle
  - The dependency graph of packages must have no cycles
- The Stable Dependencies Principle
  - Depend in the direction of stability.
- The Stable Abstractions Principle
  - Abstractness increases with stability.

Goldman Sachs

Ensuring Code Quality

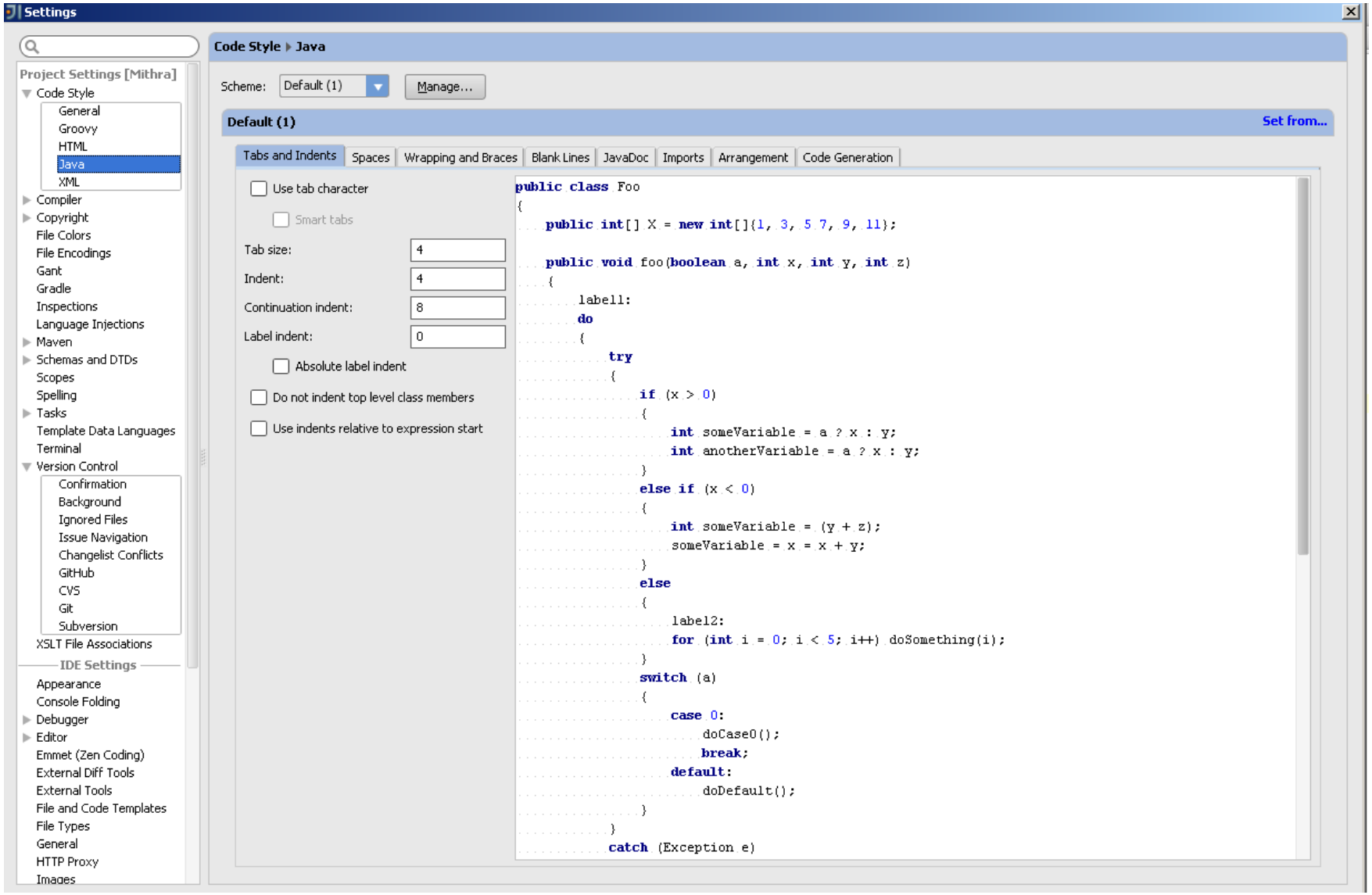
- Coding standards
- Tests (automated and manual)
- Continuous integration
- Refactoring
- Code reviews
- Managing technical debt
- Static code analysis

Goldman Sachs

Coding (Style) Standards

- It's not a matter of better or worse, but consistency.
- For example, naming standards:
  - Upper camel case for class name, lower camel case first character for methods and variables, get/set method prefix for properties, ...
- Enforce a uniform look & esthetic. This improves readability.
- Prevent edit wars.
- Reduce the commit noise.
- Makes long term maintenance easier.

Style Example

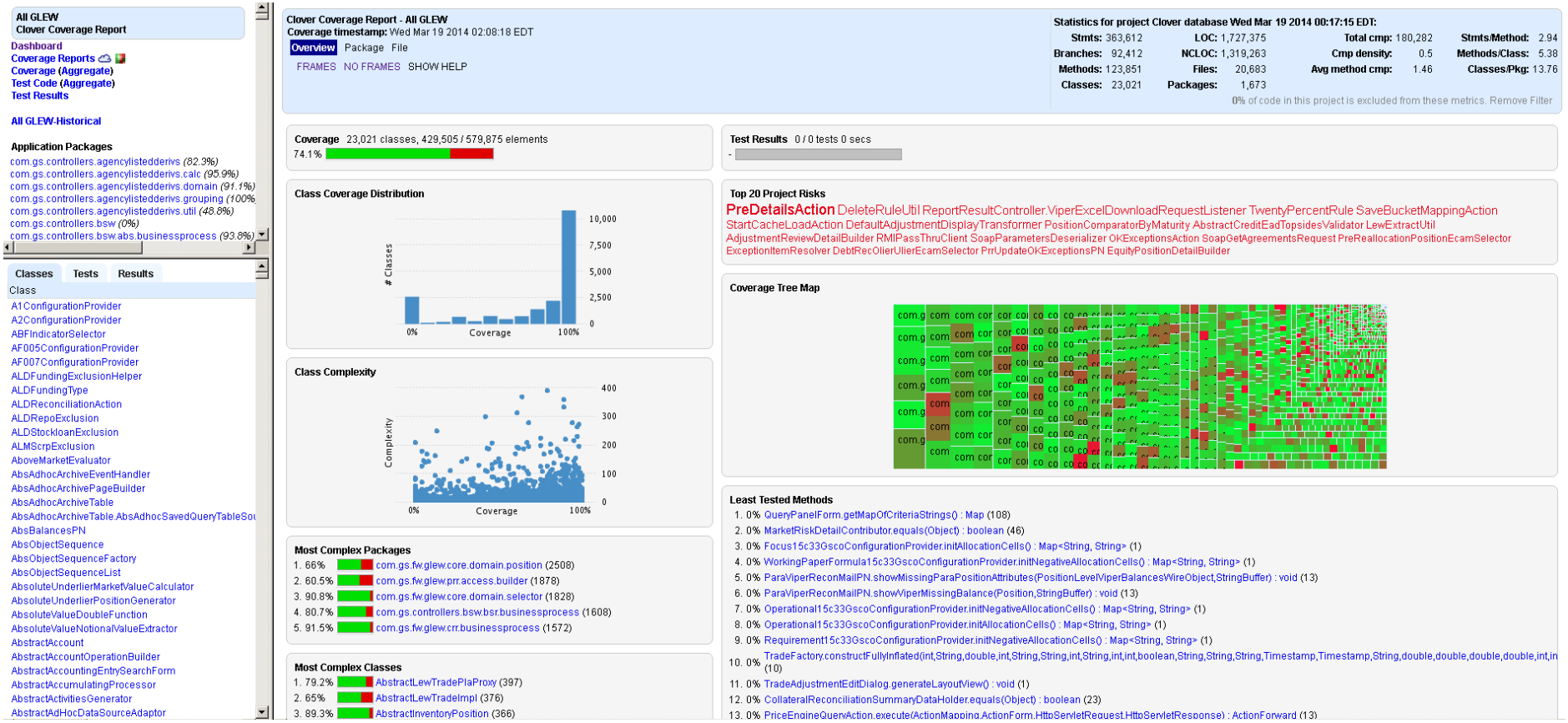


Goldman Sachs

Tests

- If quality is a measure of conformance to requirements, then tests are that measurement.
- Tests are often the most direct way to not only measure but enforce quality.
- Automated tests are preferred to manual ones. They provide higher value in the long term.
- The closer the test runs to development, the better.
  - Most valuable tests: commit time tests. No one is allowed to break these.
- Enabler of two more important code quality concepts: continuous integration & refactoring.
- Testability of a piece of code (how easy or hard is it to test) is a factor of quality in itself.
- Test Driven Development (TDD): a valuable process when applied appropriately. TDD makes testability an explicit goal.
- A lengthy subject that will be covered later in the course.

Automated Tests Coverage Example



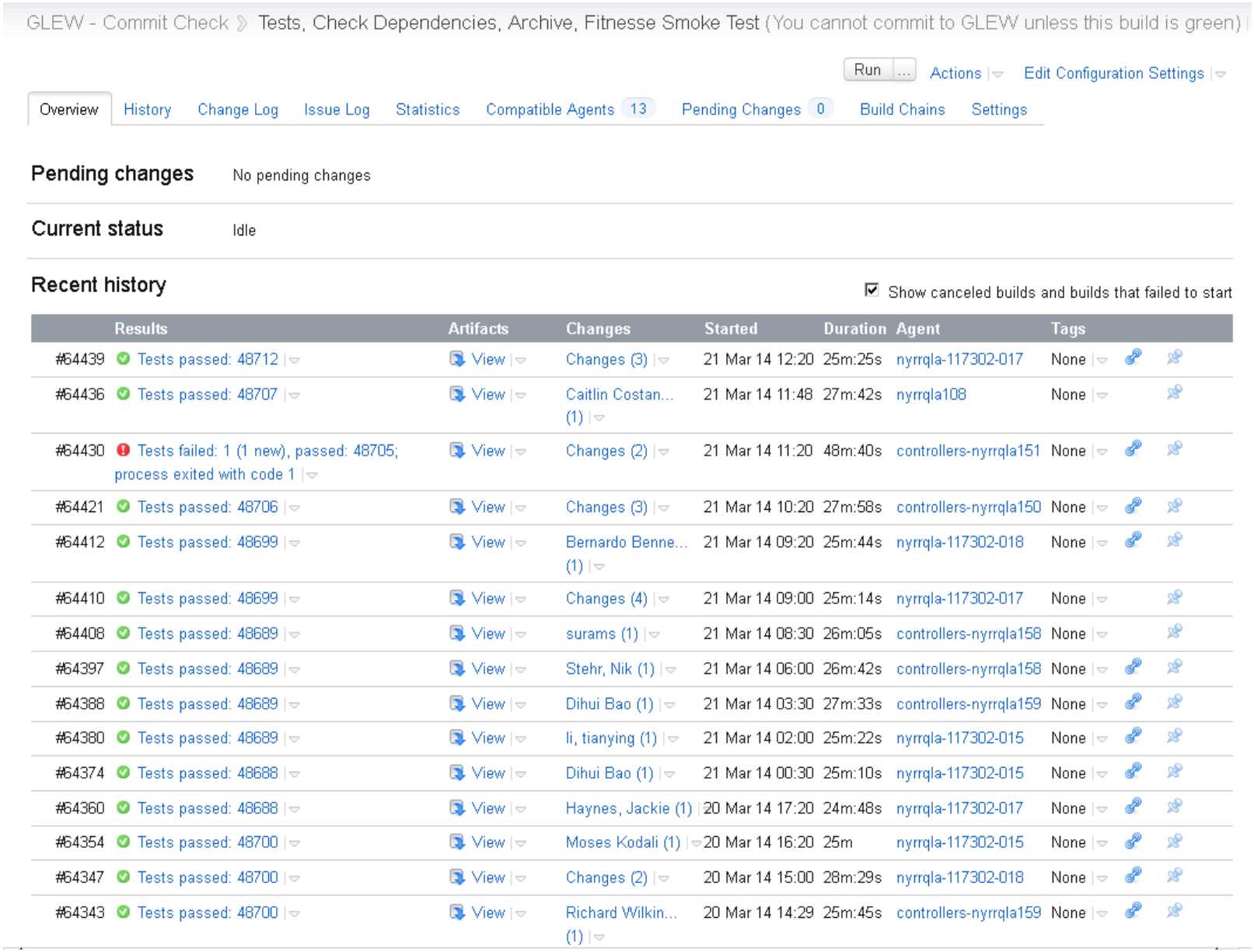
Automated Tests Coverage Example

Goldman Sachs

Continuous Integration

- Software development is sometimes like construction: it's easier to make a mess while you're constructing (clean it up later).
- That doesn't work when many people need to work on the same code base, which is facilitated through a shared repository (version control system).
- How we write code has to take that into account:
  - A commit to the shared repository is not allowed to make a mess.
  - Instead, a commit should take the code from a known good state to a new good state.
- That's the idea behind continuous integration: keep what's visible to the team in a relatively clean state.
  - Do this by using a build server: if an automated process can build the code, so can everyone else and the mess is kept to a minimum.
- Continuous integration usually includes a fresh checkout of the whole repository, compiling the code and running the automated tests.
- The build is either green or red. Partial success is not possible.
- For teams that pay attention to code quality, a red build is taken very seriously and addressed quickly (hours, not days).

Continuous Integration – Commit Check Example



GLEW - Commit Check » Tests, Check Dependencies, Archive, Fitnesse Smoke Test (You cannot commit to GLEW unless this build is green)

Run ... Actions Edit Configuration Settings

Overview | History | Change Log | Issue Log | Statistics | Compatible Agents 13 | Pending Changes 0 | Build Chains | Settings

Pending changes No pending changes

Current status Idle

Recent history

Show canceled builds and builds that failed to start

	Results	Artifacts	Changes	Started	Duration	Agent	Tags
#64439	Tests passed: 48712	View	Changes (3)	21 Mar 14 12:20	25m:25s	nyrrqla-117302-017	None
#64436	Tests passed: 48707	View	Caitlin Costan... (1)	21 Mar 14 11:48	27m:42s	nyrrqla108	None
#64430	Tests failed: 1 (1 new), passed: 48705; process exited with code 1	View	Changes (2)	21 Mar 14 11:20	48m:40s	controllers-nyrrqla151	None
#64421	Tests passed: 48706	View	Changes (3)	21 Mar 14 10:20	27m:58s	controllers-nyrrqla150	None
#64412	Tests passed: 48699	View	Bernardo Benne... (1)	21 Mar 14 09:20	25m:44s	nyrrqla-117302-018	None
#64410	Tests passed: 48699	View	Changes (4)	21 Mar 14 09:00	25m:14s	nyrrqla-117302-017	None
#64408	Tests passed: 48689	View	surams (1)	21 Mar 14 08:30	26m:05s	controllers-nyrrqla158	None
#64397	Tests passed: 48689	View	Stehr, Nik (1)	21 Mar 14 06:00	26m:42s	controllers-nyrrqla158	None
#64388	Tests passed: 48689	View	Dihui Bao (1)	21 Mar 14 03:30	27m:33s	controllers-nyrrqla159	None
#64380	Tests passed: 48689	View	li, tianying (1)	21 Mar 14 02:00	25m:22s	nyrrqla-117302-015	None
#64374	Tests passed: 48688	View	Dihui Bao (1)	21 Mar 14 00:30	25m:10s	nyrrqla-117302-015	None
#64360	Tests passed: 48688	View	Haynes, Jackie (1)	20 Mar 14 17:20	24m:48s	nyrrqla-117302-017	None
#64354	Tests passed: 48700	View	Moses Kodali (1)	20 Mar 14 16:20	25m	nyrrqla-117302-015	None
#64347	Tests passed: 48700	View	Changes (2)	20 Mar 14 15:00	28m:29s	nyrrqla-117302-018	None
#64343	Tests passed: 48700	View	Richard Wilkin... (1)	20 Mar 14 14:29	25m:45s	controllers-nyrrqla159	None

Continuous Integration – Build Farm

Controllers Technology pool — All agents are busy

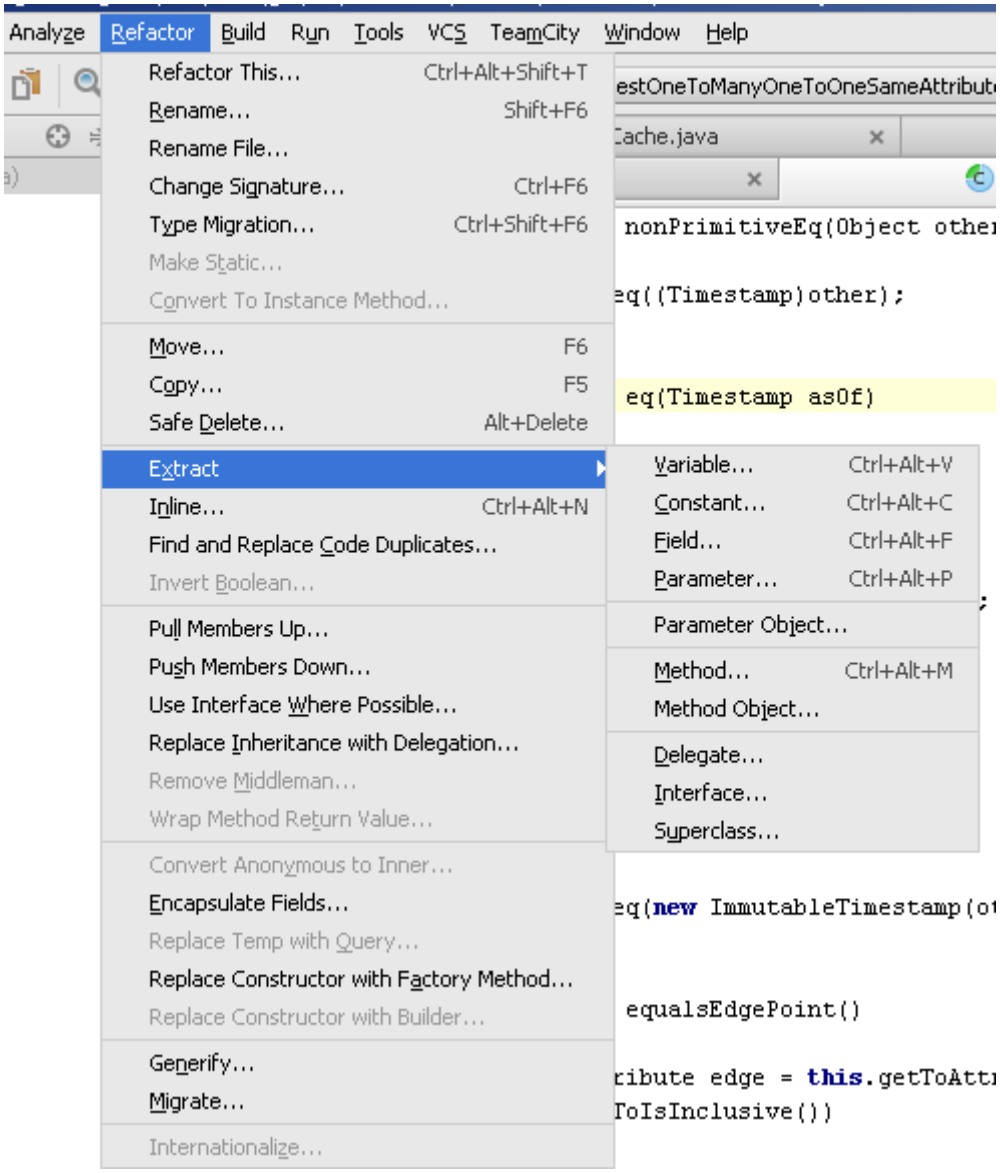
controllers-nyctdla109	377	Enabled	GLEW - Metrics :: GLEW Sonar (DO NOT REMOTE RUN)	sonar	18 Mar 14 18:36 (15h:29m)	Stop
controllers-nyrrqla102	332	Enabled	LEW3.0 :: Commit Check	Tests passed: 57; run-test-suite	1m:59s left	Stop
controllers-nyrrqla150	615	Enabled	GLEW - Commit Check :: Tests, Check Dependencies, Archive, Fitnesse Smoke Test	Tests passed: 9206	15m:28s left	Stop
controllers-nyrrqla151	615	Enabled	PARA - Core :: PARA Commit Check (SVN)	Tests passed: 15046	32s left	Stop
controllers-nyrrqla152	616	Enabled	PARA - Core :: PARA Commit Check (SVN)	Running	26m:52s left	Stop
controllers-nyrrqla153	616	Enabled	PARA :: PARA - Validate Dbobjects todo-entries Sync between QA & PROD	Running	3m:17s left	Stop
controllers-nyrrqla155	616	Enabled	GLEW - Commit Check :: Tests, Check Dependencies, Archive, Fitnesse Smoke Test	Tests passed: 60	24m:30s left	Stop
controllers-nyrrqla158	616	Enabled	PARA :: PARA Test P3 SDA Build	Running	19 Mar 14 09:43 (22m:01s)	Stop
controllers-nyrrqla159	617	Enabled	GLEW - Commit Check :: Tests, Check Dependencies, Archive, Fitnesse Smoke Test	Tests passed: 104	19m:39s left	Stop
nyrrqla-117302-015	468	Enabled	PARA - Conductor :: PARA Conductor Build (DONT REMOTE RUN)	Tests passed: 349	8m:49s left	Stop
nyrrqla-117302-016	468	Enabled	GLEW - QA :: Tests, Check Dependencies, Archive, Fitnesse Smoke Test (use for Remote run)	Running	26m:48s left	Stop
nyrrqla-117302-017	468	Enabled	GLEW - Production :: Interanl Audit - Test - Create Move to AREA Test Artifacts	Tests passed: 18	28m:21s left	Stop
nyrrqla-117302-018	468	Enabled	PARA - Conductor :: PARA Conductor Build (DONT REMOTE RUN)	Tests passed: 703	14m:22s left	Stop
nyrrqla-117302-020	468	Enabled	PARA - Conductor :: PARA Conductor Build (DONT REMOTE RUN)	Tests passed: 8285	12m:17s left	Stop

Goldman Sachs

Refactoring

- Defined as changing the code implementation without affecting its output.
  - In other words, it's about the non-functional requirements.
- Example: extract method.
- One of the agile practices that recognizes that:
  - Requirements are not always known ahead of time.
  - Code can and does stay around for a long time (decades), and it has to change during that time.
  - Small mistakes or oversights accumulates over time.
  - Build - refactor - build - refactor ... can provide incremental value compared to trying to get everything right from the start.
- Enabled by automated tests and refactoring tools (IDE)
  - If code can be changed without failing any tests, refactoring is successful.
- Critical for long term health of a code base: a large number of small improvements over time adds up.

Refactoring



Goldman Sachs

Code Reviews

- Typically, a senior developer (or architect) will review the code of more junior developers.
- Can be done pre or post commit. Pre-commit code reviews tend to get better overall quality enforcement.
- The review should be a dialogue between the reviewer and reviewee.
- The reviewers looks at both functional and non-functional aspects of the new code.
- The reviewer also ensures that the new code meets the architecture and design standards.
- Important secondary benefit: both the reviewer and reviewee learn during this process.
  - By improving the quality of the developers, the code quality naturally tends to go higher.

Code Review – Checklist

- Be a 'boy scout' and always leave things at least as good as you found it.
- If not familiar with the issue being addressed, have the developer demonstrate or explain the use case and what is being changed or added
- Get a "feeling" for the code -- does it have "funny smells"?
- Be aware of existing "broken windows" that can be fixed up
- Don't be afraid to not signing-off on a ticket or piece of work until it meets the department standards
- Try to explain why we would prefer to see something done one way or another by giving specific reasons about why we feel that way.
- Be prepared to really read the code being reviewed, and commit to spending time helping the reviewee discover simpler, better ways of doing the same thing.
- Treat the code as if it was your own, and be prepared to sign your name to it after you are done, as if you were the author, not the critic.

Goldman Sachs

Managing Technical Debt

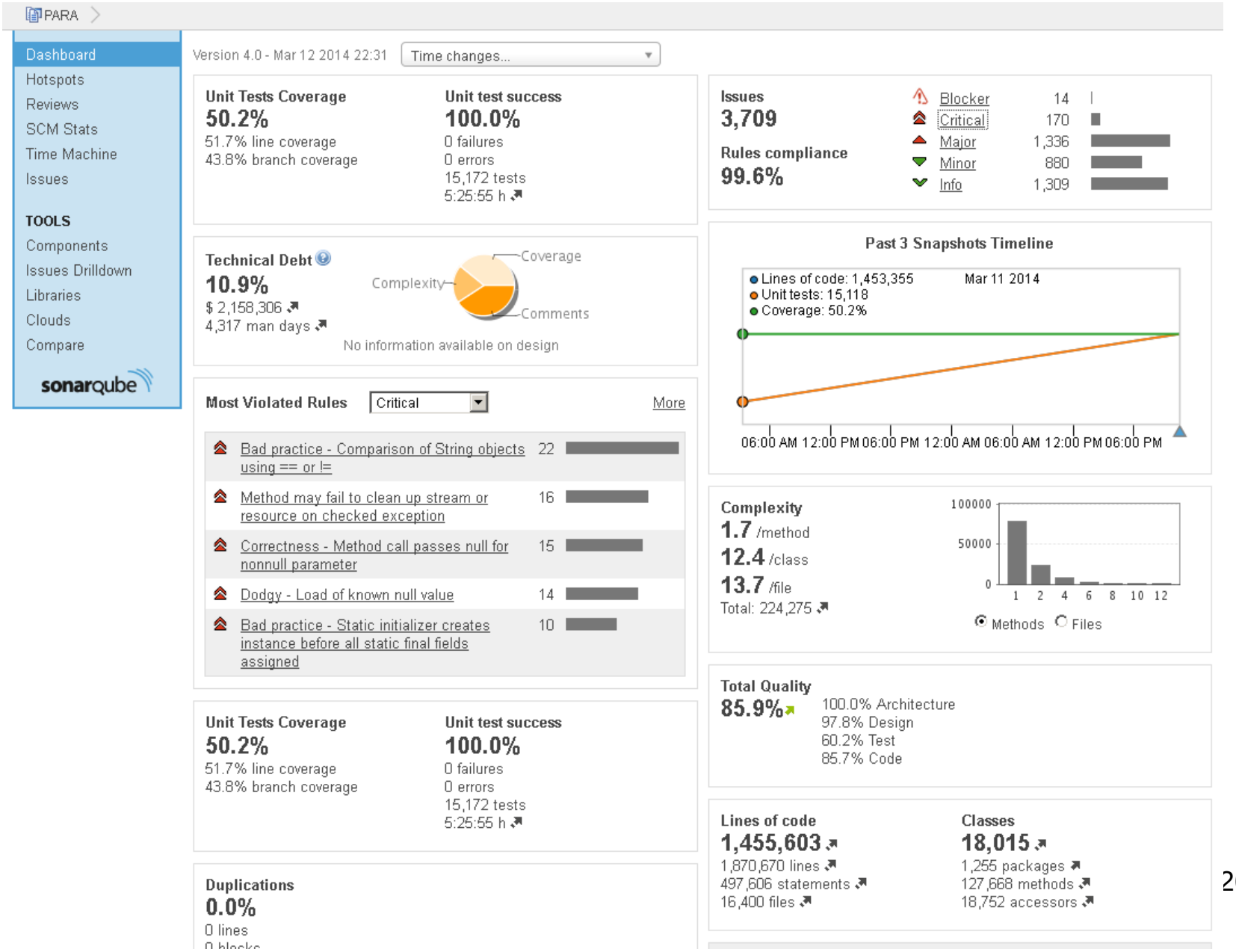
- There is often a choice between clean, maintainable code that takes more effort and dirty, hard to maintain code that's quick to write.
- It's occasionally ok to take the quick & dirty approach if the following criteria are met:
  - There is a significant reason outside the control of the team to deliver quickly.
  - There is a commitment to clean up the code in a planned fashion.
- Similar to a financial debt: taking on technical debt incurs interest:
  - Maintaining dirty code cost more.
  - Dirt has a tendency to spread.
  - The overall cost will be higher when taking on technical debt. Not only because of the interest, but also because the clean design may have to re-write large portions of the implementation.
- Occasionally, a debt is incurred accidentally.
  - These should be avoided as much as possible.
  - When they happen, they should be remedied with a planned commitment.

Goldman Sachs

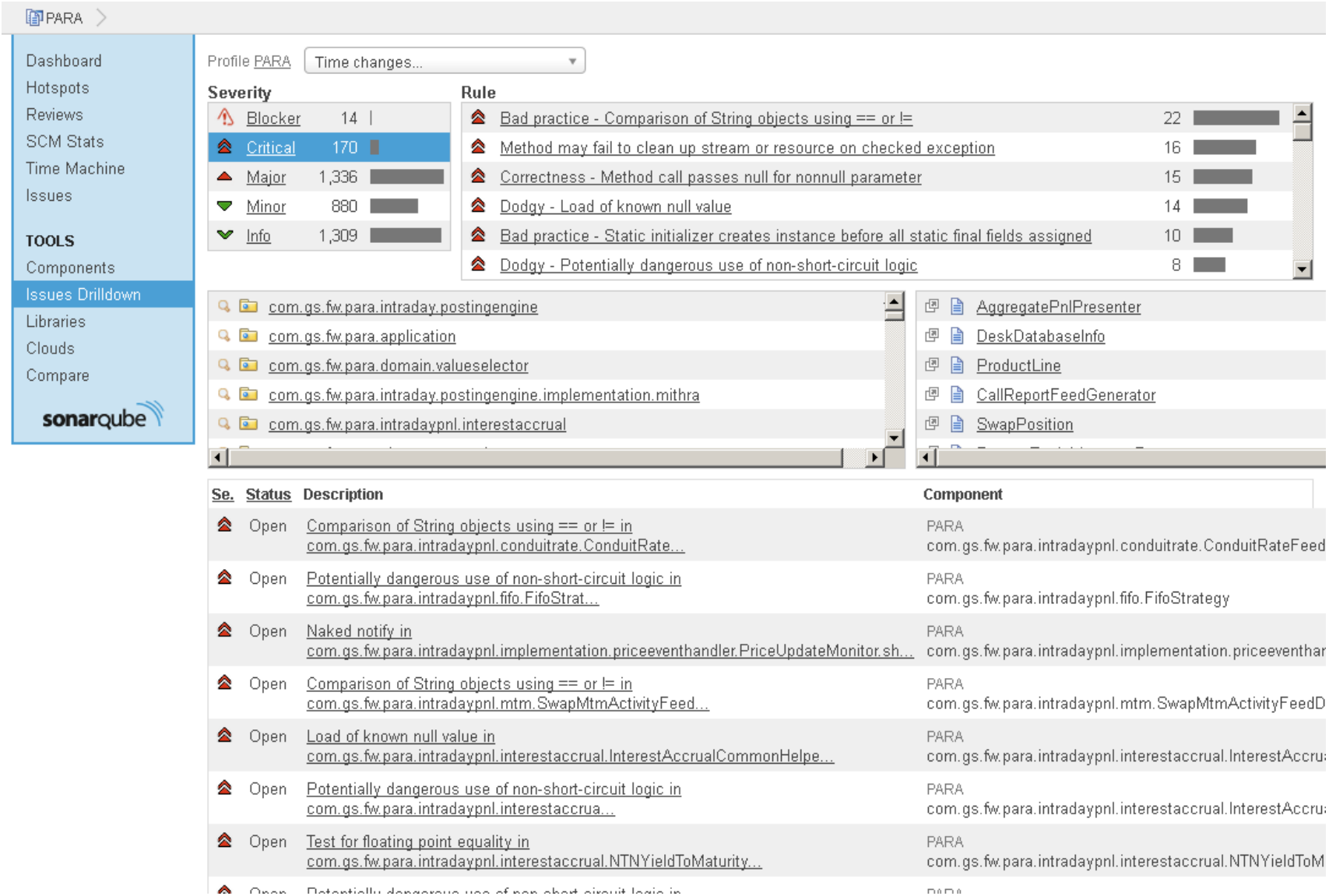
Static Code Analysis

- Various types of tools:
  - Code style: checkstyle http://checkstyle.sourceforge.net/
  - Rule based: Findbugs http://findbugs.sourceforge.net and PMD http://pmd.sourceforge.net/
  - Structural: Macker
  - Duplicate code detection: Simian
  - Test coverage: Emma, Clover
- The tools are often employed via continuous integration to ensure quality.
- A lot of these checks are built into good IDE's, which can be great help to getting more mundane aspects of code quality under control.
- Can be aggregated/summarized with tools like Sonar.
- The output is not 100% correct and needs further (human) analysis before taking action.

Sonar Summary Example



Sonar Detail Example



Goldman Sachs

Social Aspects of Code Quality

- Long lived code is edited by many people over a long period of time.
- Must avoid tragedy of the commons.
  - An individual developer may correctly reason about taking a shortcut for the code they are working on based on their requirements, but may miss the big picture.
- You can't fault people for code they didn't write.
  - But then, how do you incentivize fixing old code?
    - Baseline existing violations.
    - Highlight code quality issues for delta code changes.
    - Start with a lenient criteria to make the adoption easier.
    - Encourage the Boy Scout Rule: “Always leave the campground cleaner than you found it.”
- Using the above principles, we’ve customized some existing tools to give developers a report card for their code commits.
  - If the score is outside an acceptable range, action is usually taken to remediate.

Example Report

From: Para Dev. Tools Test [mailto:devtools@nyrrqla153.ny.fw.gs.com]
Subject: Sonar results for checkin: -42.3 (22 files, min score: -45.6)

Team City Changes

GLEW-5523 [] Change Fix Breaks button to call LEW3.0 service to refresh positions

TOTAL SCORE: -42.3

FILE BREAKDOWN:

32 com.gs.controllers.bsw.reconciliation.parabsw.ui.ParaBswFixBreaksButtonHandler
14 com.gs.controllers.bsw.reconciliation.parabsw.ui.ParaBswFixBreaksButtonHandlerTest
11.5 com.gs.controllers.bsw.reconciliation.parabsw.breakfix.ParaBswBreakFixServiceImpl
-10 com.gs.fw.glew.domain.ctinfra.ParaBswBreakFixPosition
-11.8 com.gs.fw.glew.domain.ctinfra.ParaBswBreakFixAccount
-32.7 com.gs.controllers.bsw.reconciliation.parabsw.breakfix.ParaBswBreakFixServiceImplTest
-45.6 com.gs.controllers.bsw.reconciliation.parabsw.breakfix.ParaBswBreakFixPositionFixer

* Some files with abs(score) < 5 may have been omitted from the results for brevity -
* You can see the full results in summary.txt in the teamcity build

METRICS BREAKDOWN:

	violations_density	class_complexity	function_complexity	duplicated_lines	uncovered_conditions	uncovered_lines	dit
ParaBswFixBreaksButtonHandler	-6.6	0	.6	0	20.0	18.0	0
ParaBswFixBreaksButtonHandlerTest	14.0	0	0	0	0	0	0
ParaBswBreakFixServiceImpl	-.9	1	1.4	0	0	10.0	0
ParaBswBreakFixPosition	0	0	0	0	0	0	-10.0
ParaBswBreakFixAccount	0	-1	-.8	0	0	0	-10.0
ParaBswBreakFixServiceImplTest	.3	0	0	-33.0	0	0	0
ParaBswBreakFixPositionFixer	-6.6	-3	-5.0	0	-20.0	-11.0	0

Report After Fixing the Bad Code

From: Para Dev. Tools Test [mailto:devtools@nyrrqla153.ny.fw.gs.com]
Subject: Sonar results for checkin: 57.0 (2 files, min score: 26.0)

Team City Changes

GLEW-5545 [] Sonar follow-up check-in

TOTAL SCORE: 57.0

FILE BREAKDOWN:

31 com.gs.controllers.bsw.reconciliation.parabsw.breakfix.ParaBswBreakFixPositionFixer

26 com.gs.controllers.bsw.reconciliation.parabsw.breakfix.ParaBswBreakFixServiceImplTest

* Some files with abs(score) < 5 may have been omitted from the results for brevity -
* You can see the full results in summary.txt in the teamcity build

METRICS BREAKDOWN:

	violations_density	class_complexity	function_complexity	duplicated_lines	uncovered_conditions	uncovered_lines	dit
ParaBswBreakFixPositionFixer	0	0	0	0	20.0	11.0	0
ParaBswBreakFixServiceImplTest	0	0	0	26.0	0	0	0

Goldman Sachs

Conclusion

- Good architecture and design is the first step toward high quality code.
- Code quality is an ongoing concern for the health of a code base.
- Code quality is extremely important in the real world.
  - It can often make the difference between a successful project and a failure.
  - The right level of quality for a given situation ensures the proper set of commercial trade-offs.
- Code quality is a team effort and needs appropriate social incentives.