

Upping the game

Improving your software development process

John Ferguson Smart
Principle Consultant
Wakaleo Consulting

Email: john.smart@wakaleo.com
Web: <http://www.wakaleo.com>
Twitter: [wakaleo](#)

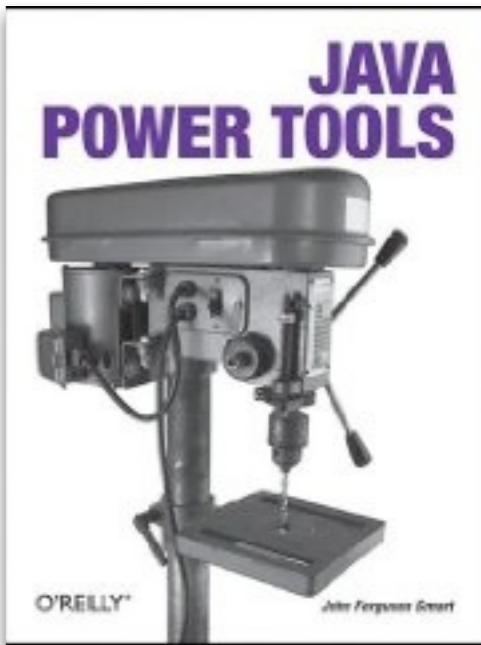


Presentation Goals

Learn how to improve, harmonize and automate your development process using tools like Maven, Hudson, and many others.

Speaker's qualifications

- ▶ John Ferguson Smart
- ▶ Consultant, Trainer, Mentor, Author,...
- ▶ Works with Enterprise Java, Web Development, and Open Source technologies
- ▶ Author of 'Java Power Tools' (O'Reilly)
- ▶ Writes articles for sites like JavaWorld, DevX and Java.net, and blogs on Java.net
- ▶ Speaks at conferences, Java User Groups etc.
- ▶ Likes to write about himself in the third person



Agenda

- ▶ What we will cover today:
 - ▶ Industrializing your build process
 - ▶ Automate your builds
 - ▶ Better testing practices
 - ▶ Reducing technical debt

Why improve?



HERE is Edward Bear, coming downstairs now, bump, bump, bump, bump, on the back of his head, behind Christopher Robin. It is, as far as he knows, the only way of coming downstairs, but sometimes he feels that there really is another way, if only he could stop bumping for a moment and think of it.

-- A. A. Milne

Why should we improve?

► Why should we improve our development process?

- Lower development costs
- Lower maintenance costs
- Less bugs
- Higher code quality
- Be flexible - adapt to change more easily
- Happier more productive users



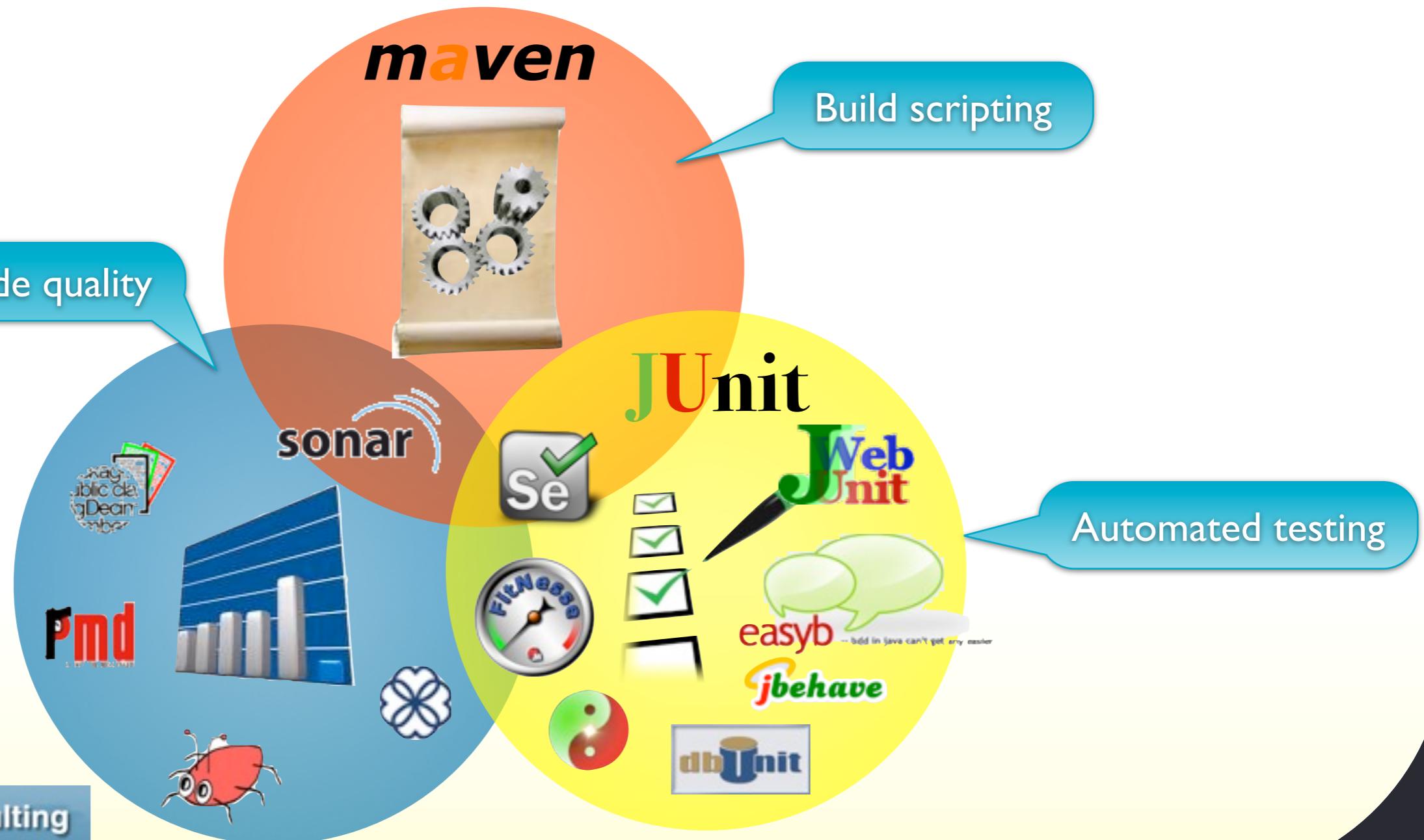
How can we improve?

► How can we improve our development process?

- Standardized build practices
- Better testing practices
- Better visibility
- Faster feedback
- Quality metrics
- Automate!

Tools for the job

- ▶ There are plenty of tools available - and most are free!



Towards a better build process

- ▶ Standardize your build process with Maven
- ▶ What is Maven, anyway?
 - A high-level open source build scripting framework
 - Extensively used in the Java world



Towards a better build process

► How does Maven help?

- Standards
- Conventions
- Lower Maintenance Costs
- Knowledge sharing
- Dependency Management
- Promoting good architecture



So how can Maven help me?

► Standards and Conventions

- A standard directory structure
- A standard, but extensible build lifecycle



So new developers understand the project structure and build process straight away!

...and the build scripts are much easier to understand and maintain



So how can Maven help me?

► Technical documentation

- Generate technical project documentation
- Easy to integrate code quality metrics



A screenshot of a web-based project information page. At the top, it says "Inland Revenue Tax Tax Task". Below that is a sidebar with links like "Main Page", "Project Overview", "Project Documentation", and "Build by Maven". The main content area is titled "Project Information" and contains sections for "Overview" and "Documents". Under "Documents", there are links for "Continuous Integration", "Dependencies", "Issue Tracking", "Mailing Lists", and "Project Licence". Each link has a brief description below it. At the bottom of the page, it says "Last Published: 2007/05/17 11:40:00 UTC Version: 3.0-SNAPSHOT".

TaxCalculatorImpl
TAX_RATES
getTaxRates()
setTaxRates()
getTaxRate()
calculateIncomeTax()
calculateGST()

TaxRate
getMinimumRevenue()
getMaximumRevenue()
getRate()
calculate()

I can even generate UML
diagrams in my Javadocs

And setting up code quality
metrics is a breeze!



So how can Maven help me?

► Project architecture

- Encourages developers to use modular design
- More flexible architecture
- Reduced complexity and maintenance costs



Breaking our application down
into clean modules is much
easier

...and the smaller modules are
easier to test and maintain



So how can Maven help me?

► Dependency Management

- Understand precisely what libraries your application needs
- Safer and more reproducible builds
- A standard way to share internal libraries



All our libraries are shared and safely stored on a central server

...we just have to name the ones we need in our build script



So how can Maven help me?

► Dependency Management before Maven

- Each project has its own set of JAR files
- Unnecessary duplication
- Hard to keep track of versions
- Errors due to incompatible JAR files
- Overloads the source code repository



...and you never know what
versions you are using

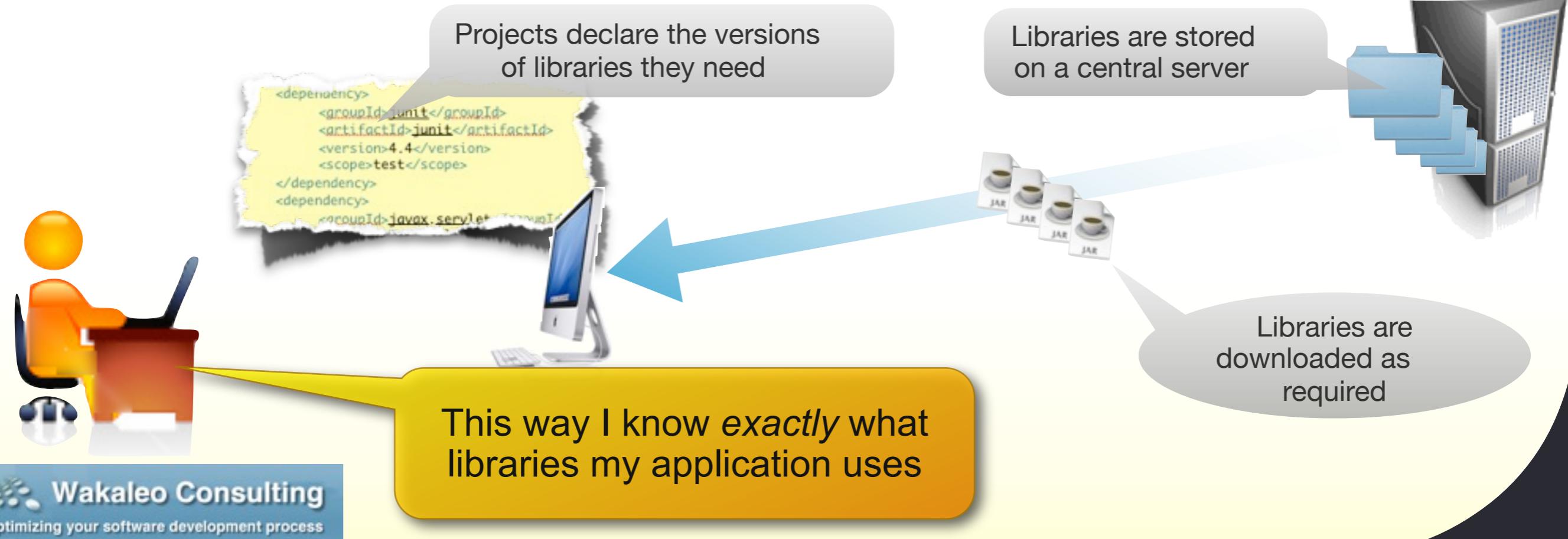


So how can Maven help me?

► Dependency Management using Maven



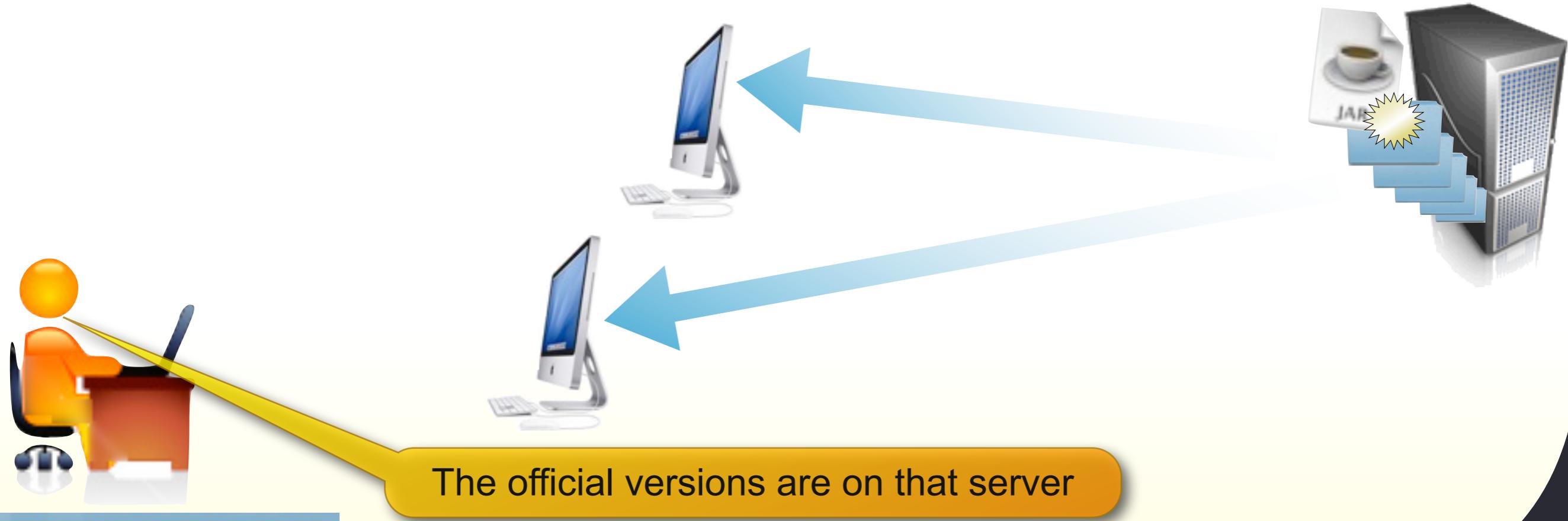
- Library versions are stored on a central server
- Each project “declares” what libraries and versions it needs
- All the required dependencies are automatically downloaded
- The server is called a ‘Maven Enterprise Repository Manager’



So how can Maven help me?

▶ Release Management

- A standard way to track and release versions
- Official versions stored on a central server
- Can be used to automate the deployment process



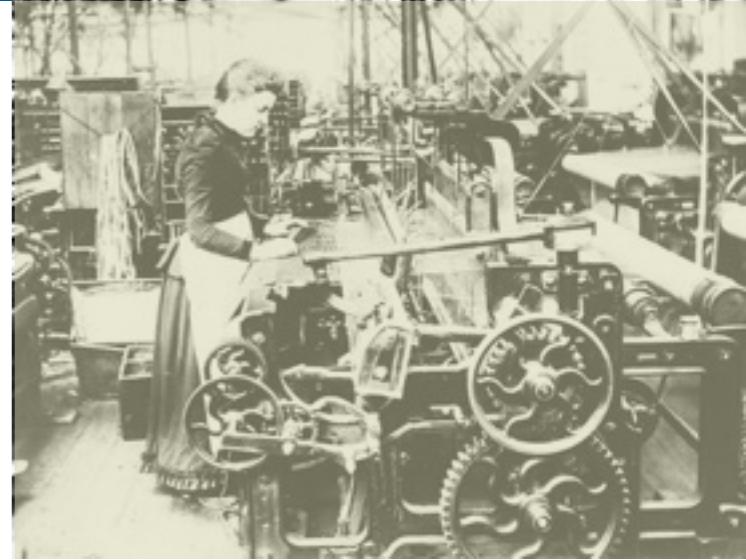
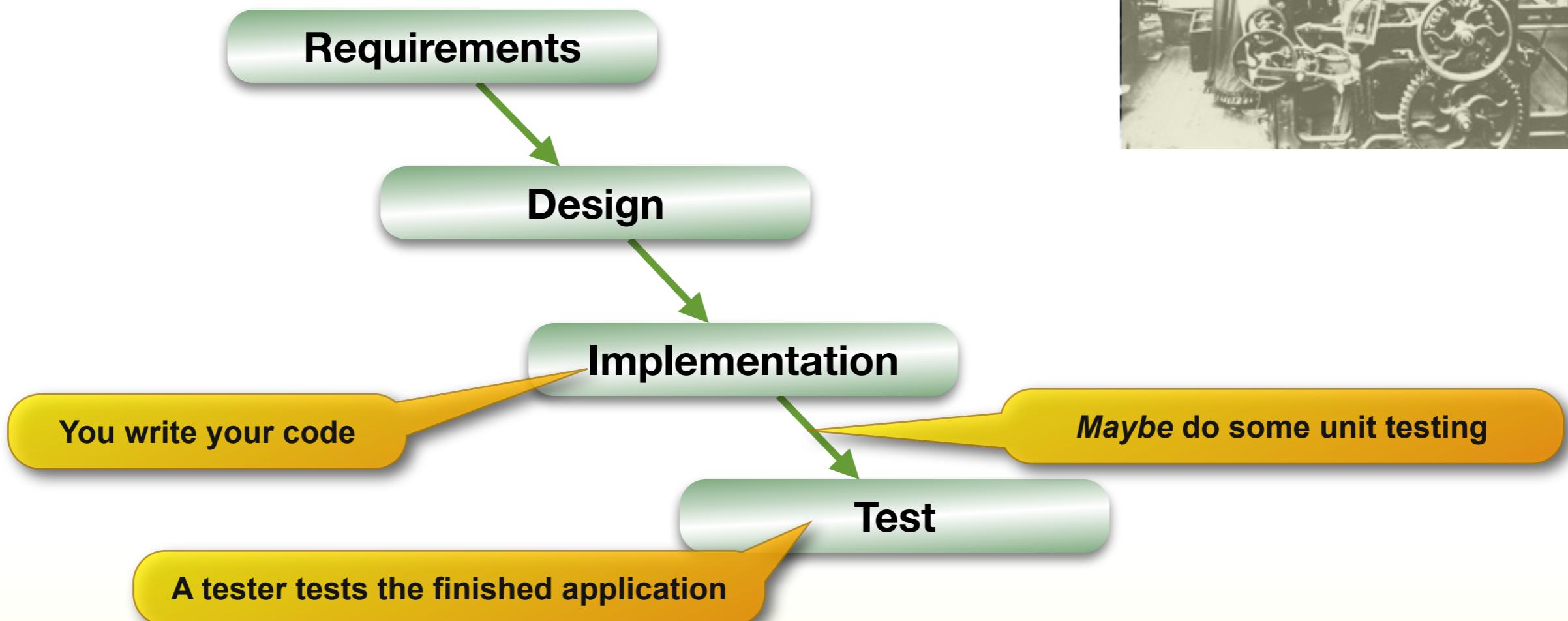
Towards better testing practices

► Why is good testing so important?

- Development costs
- Maintenance costs
- Visibility
- Flexibility
- Documentation

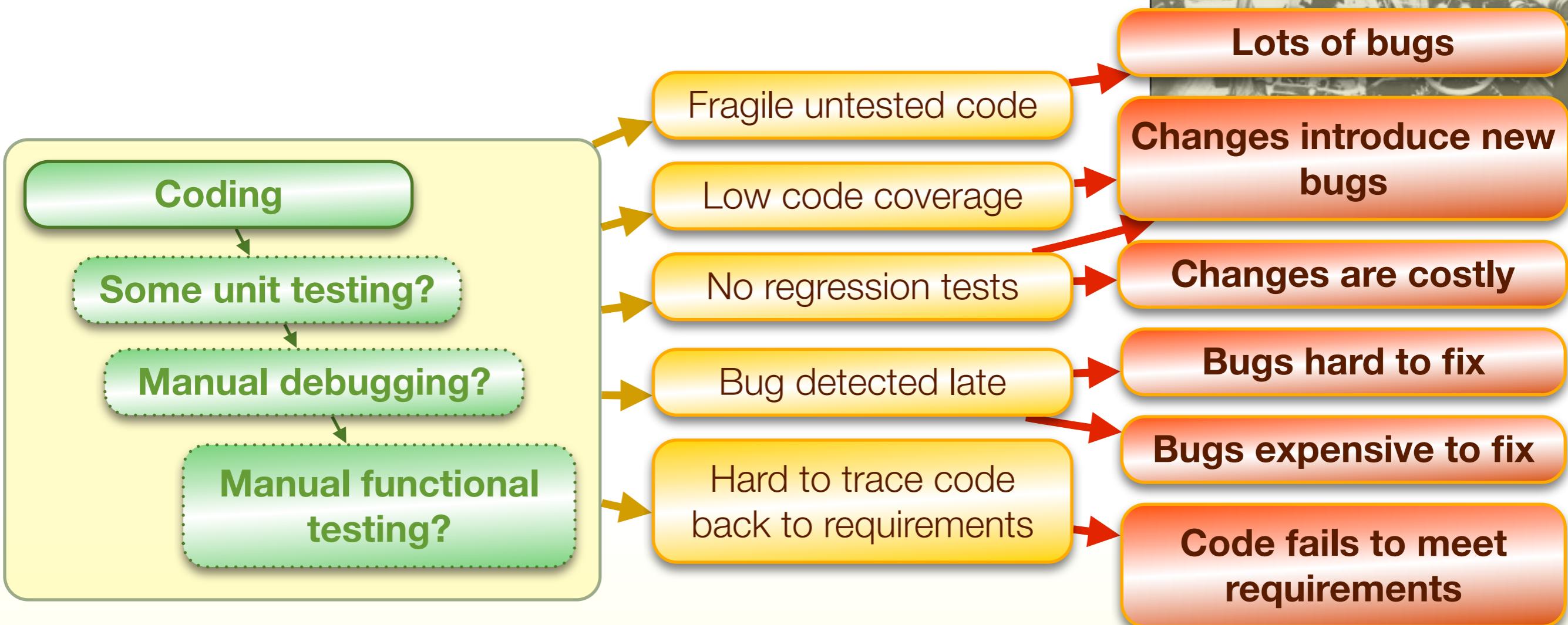
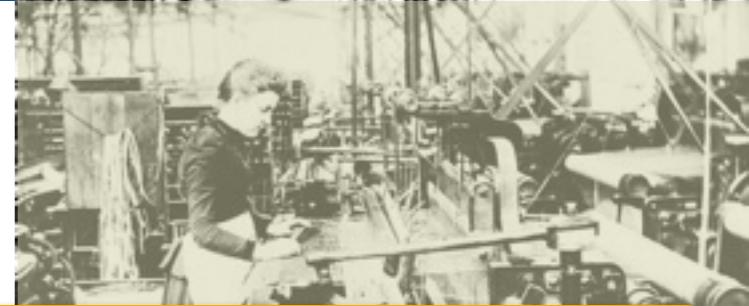
Towards better testing practices

- ▶ Coding the traditional way



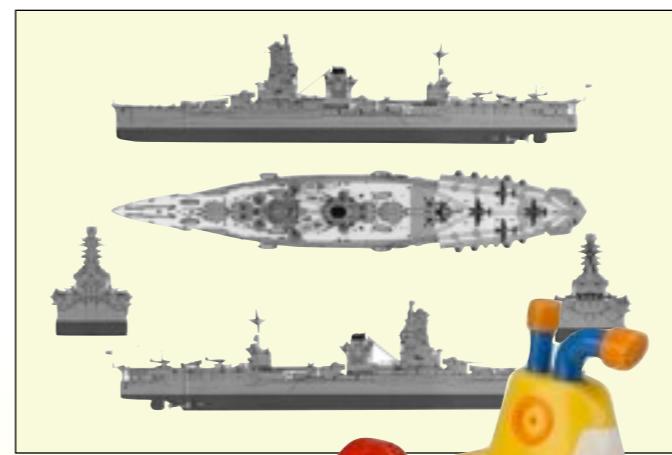
Towards better testing practices

► Coding the traditional way



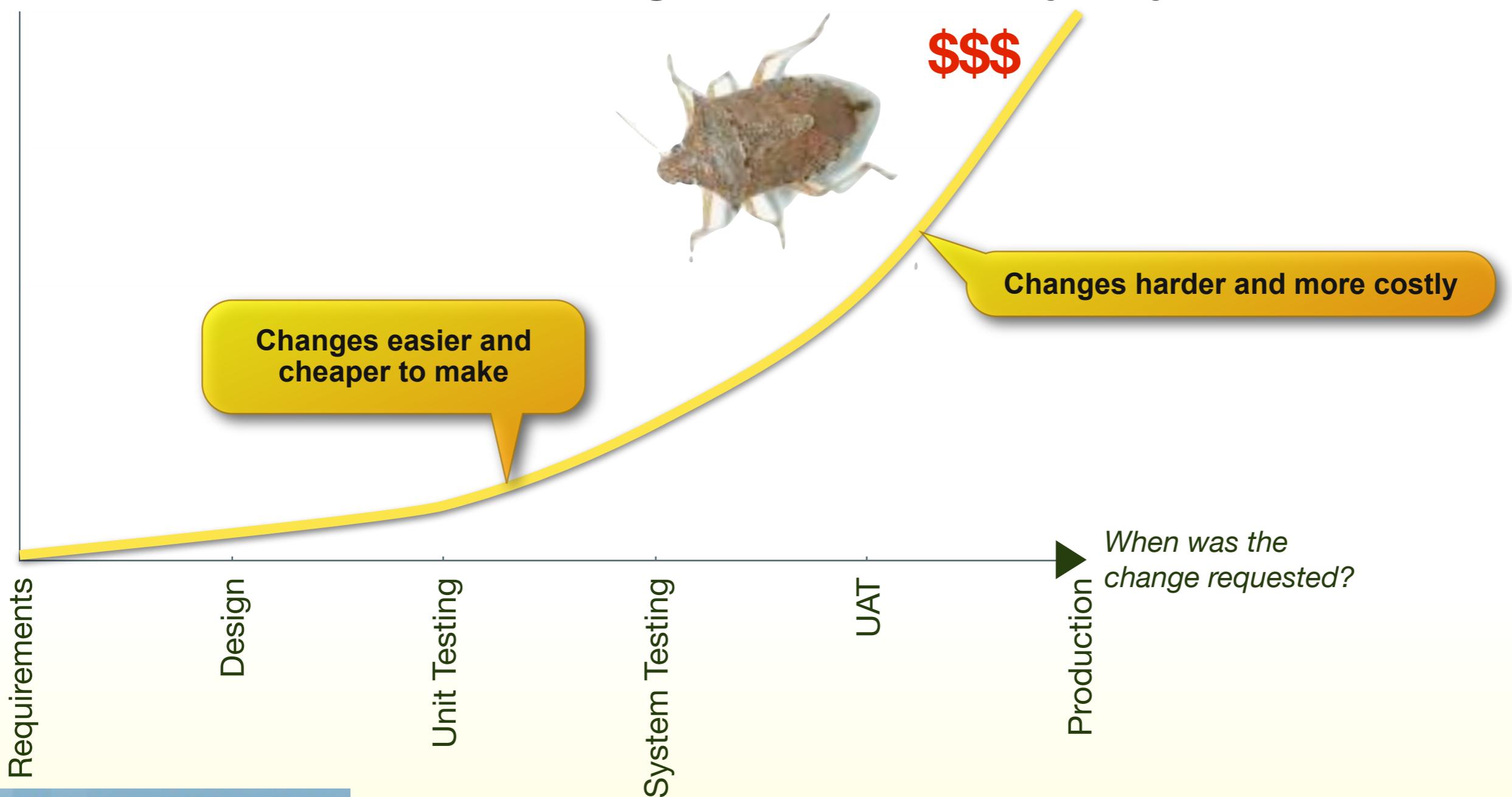
Towards better testing practices

- ▶ So what's wrong with the old way?
- ▶ Lots of defects. Really, lots.
- ▶ High maintenance costs
- ▶ Hard to introduce new features
- ▶ Doesn't meet the actual requirements
- ▶ Delayed deliveries
- ▶ Unhappy end-users



Towards better testing practices

- ▶ How much does a bug cost to fix, anyway?



Towards better testing practices

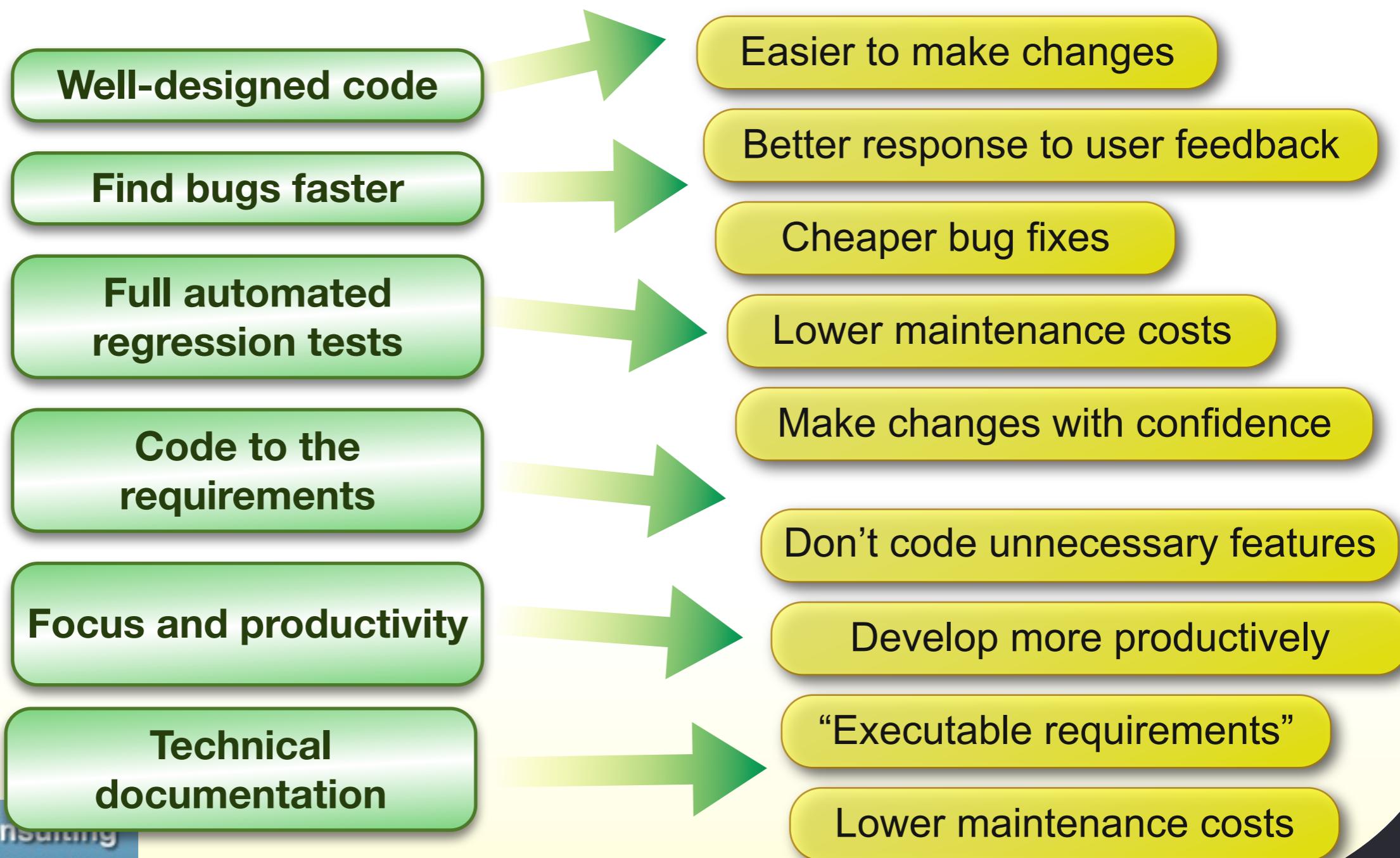
► How can good testing practices help?

- Reduce bugs
- Write better-designed code
- Have more confidence in our code
- Make changes more easily
- Meet user requirements more accurately
- Lower maintenance costs



Towards better testing practices

► How can good testing practices help?



Towards better testing practices

► More flexibility

- Testable code is easier to change
- Full regression tests avoid introducing errors



I'm not afraid to change the code
- the tests are my safety net



Towards better testing practices

► Better visibility

- Tests are “executable requirements”
- Automated acceptance tests measure progress



A feature can't be “90% finished”
- it either works or it doesn't



Summary				
Behaviors	Failed	Pending	Time (sec)	
4	1	1	0.738	
Stories Summary				
Stories	Scenarios	Failed	Pending	Time (sec)
2	4	1	1	0.738
Specifications Summary				
Specifications	Failed	Pending	Time (sec)	
0	0	0	0.0	

Towards better testing practices

► Documentation

- Tests are “living documentation” of your code
- Always accurate and up-to-date



I can understand how the code works by reading the tests



Towards better testing practices

▶ Lower maintenance costs

- Less bugs, found faster
- Changes are easier to make



Maintaining this sort of application is a real pleasure!



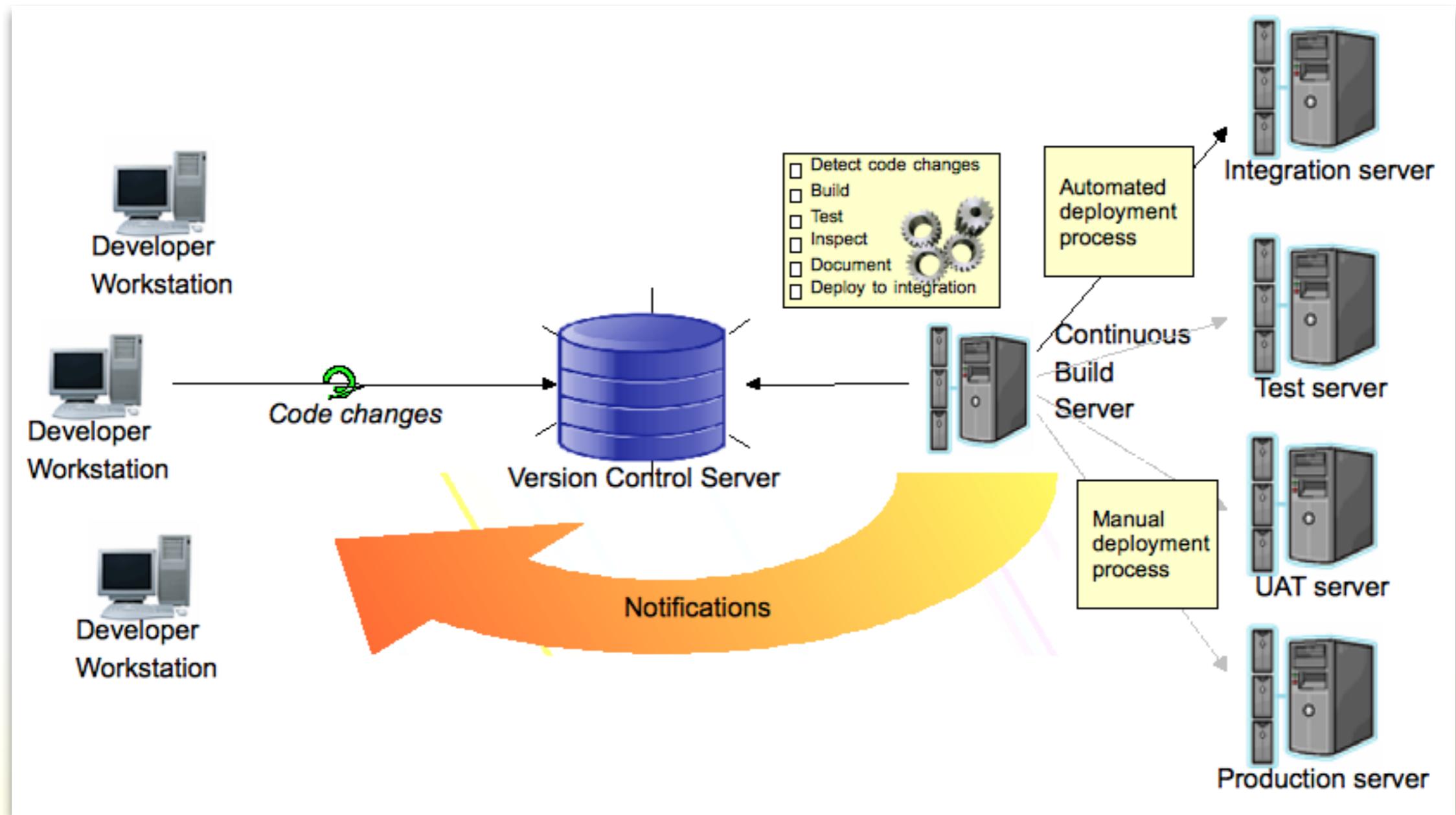
Automating the build process

- ▶ Continuous Integration - what's the issue?
- ▶ Traditional development cycles are bad for your health:
 - ▶ Integration is long and difficult
 - ▶ Poor visibility on development progress
 - ▶ Functional tests are done too late
 - ▶ Raised issues are harder to fix
 - ▶ The client gets a sub-optimal product



Automating the build process

► Continuous Integration - what's involved?



Automating the build process

► Continuous Integration - why bother?

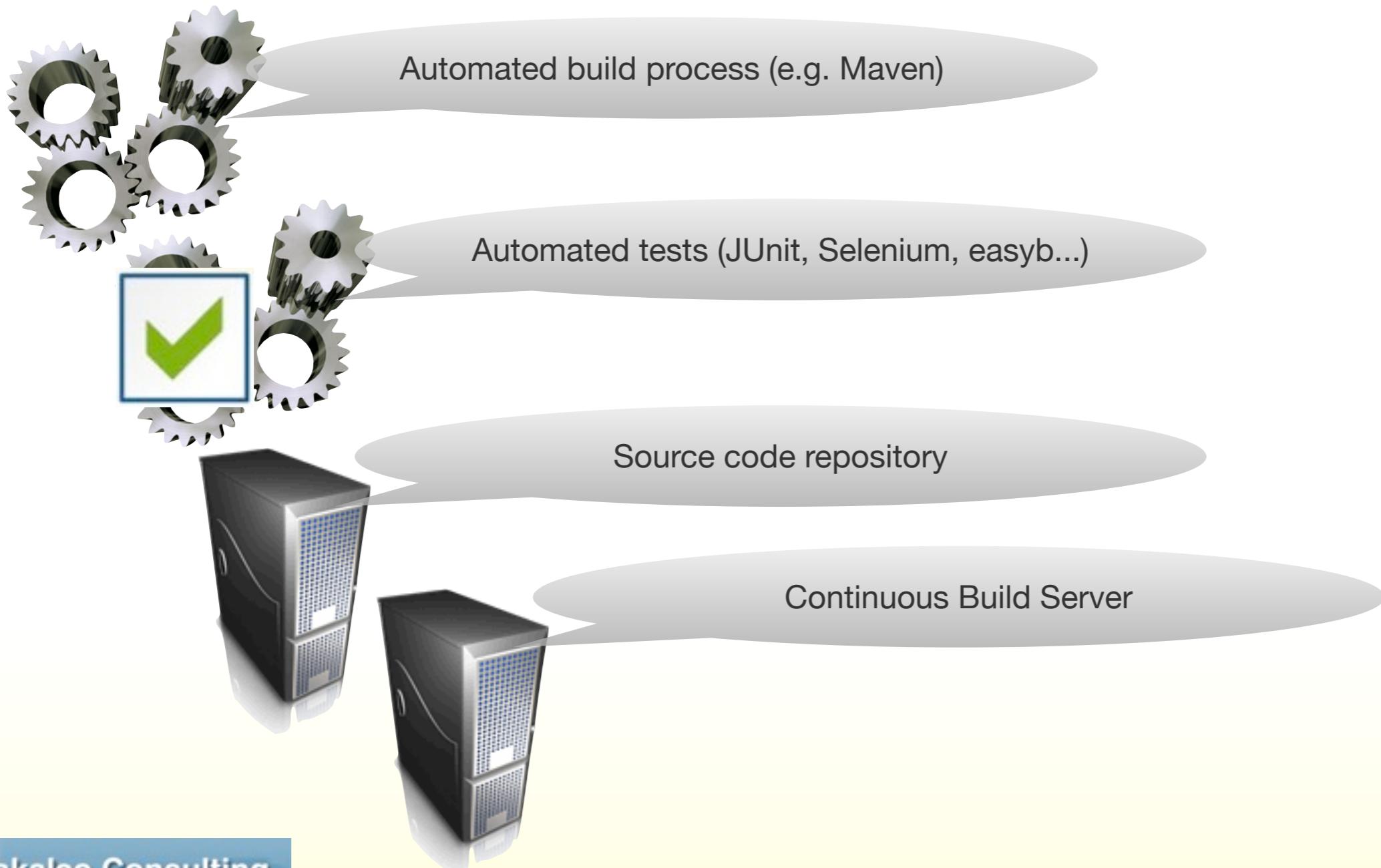
- Smoother integration process
- Automatic regression testing
- Regular working releases
- Earlier functional testing
- Faster and easier bug fixes
- Better visibility

No more “it works on my machine”



Automating the build process

► Continuous Integration - what you need



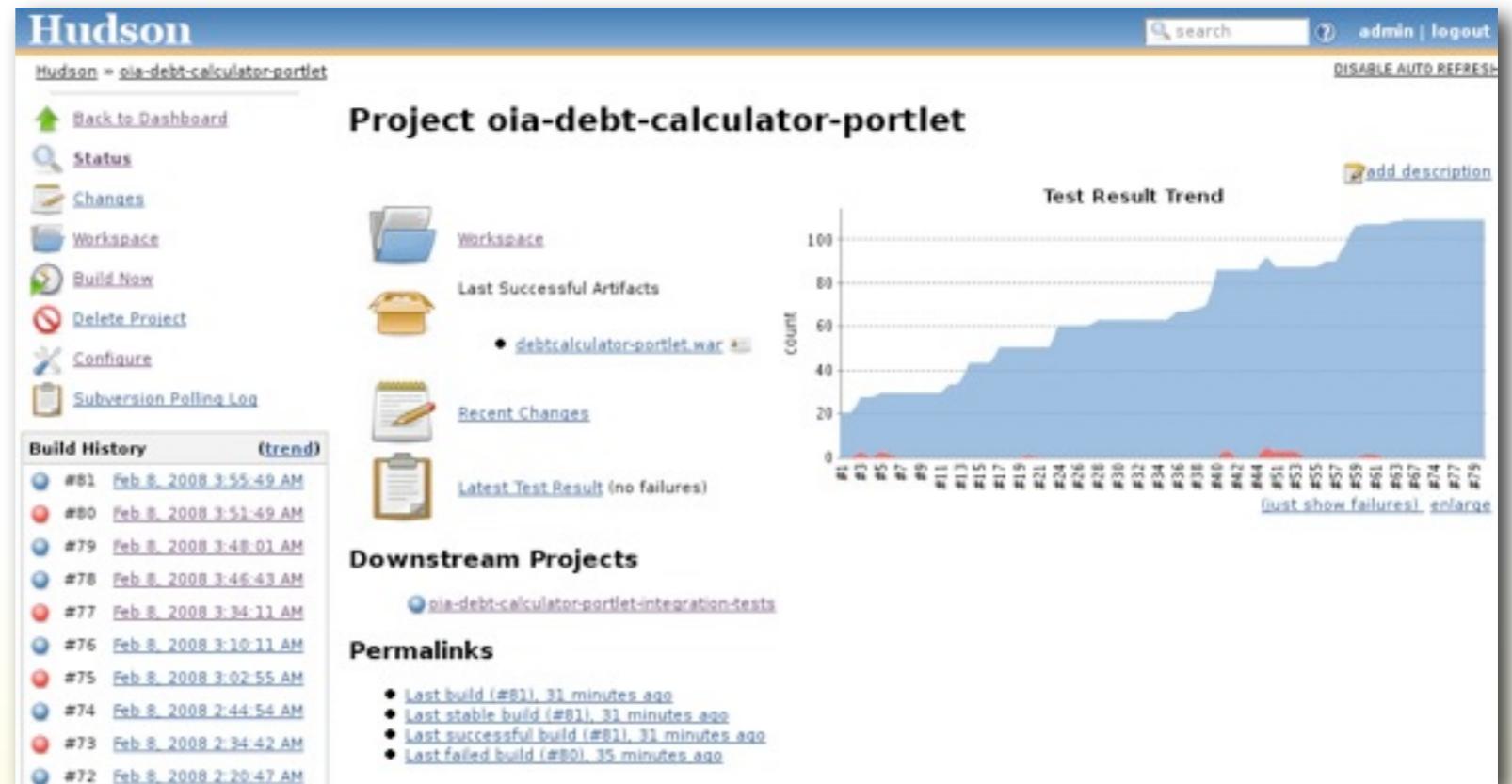
Automating the build process

► Continuous Integration - what can it do?

- Raise integration issues - fast!
- Monitor your build process
- Monitor and report on code quality and code coverage
- Build promotion and release management
- Automated deployments

Automating the build process

- ▶ Looking for a good O/S Continuous Integration tool?
- ▶ Try Hudson!
- ▶ Easy to set up and configure
- ▶ Good build and code quality metrics
- ▶ Lots of plugins



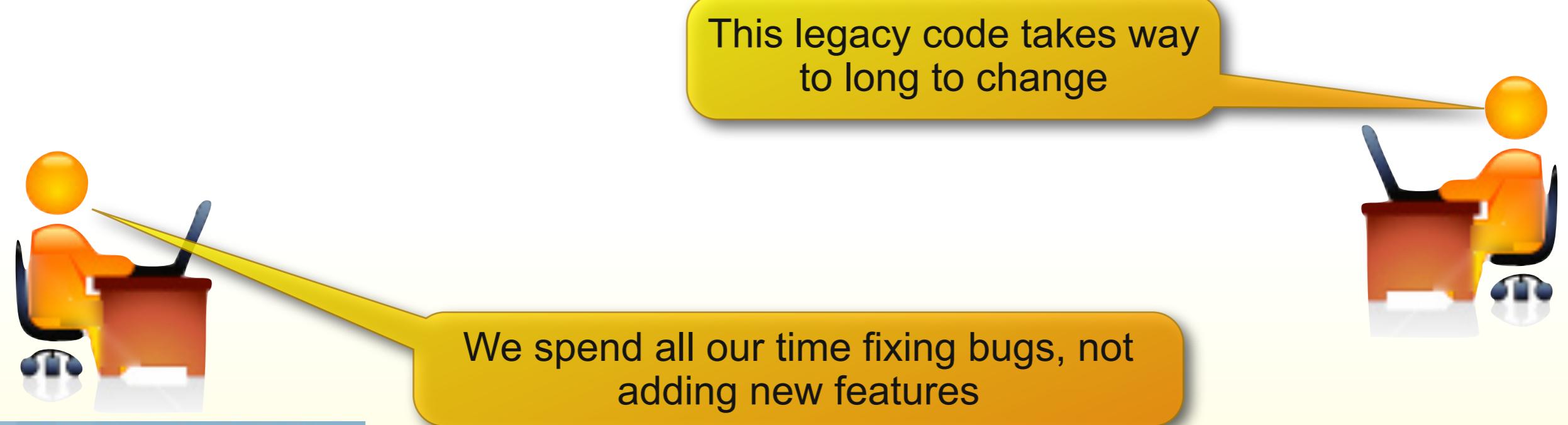
Automated Code Quality

► Why use code quality metrics

- Better quality code
- Enforce corporate coding standards
- Detect potential bugs
- Code is easier to maintain
- Train new staff
- Keep technical debt down

Automated Code Quality

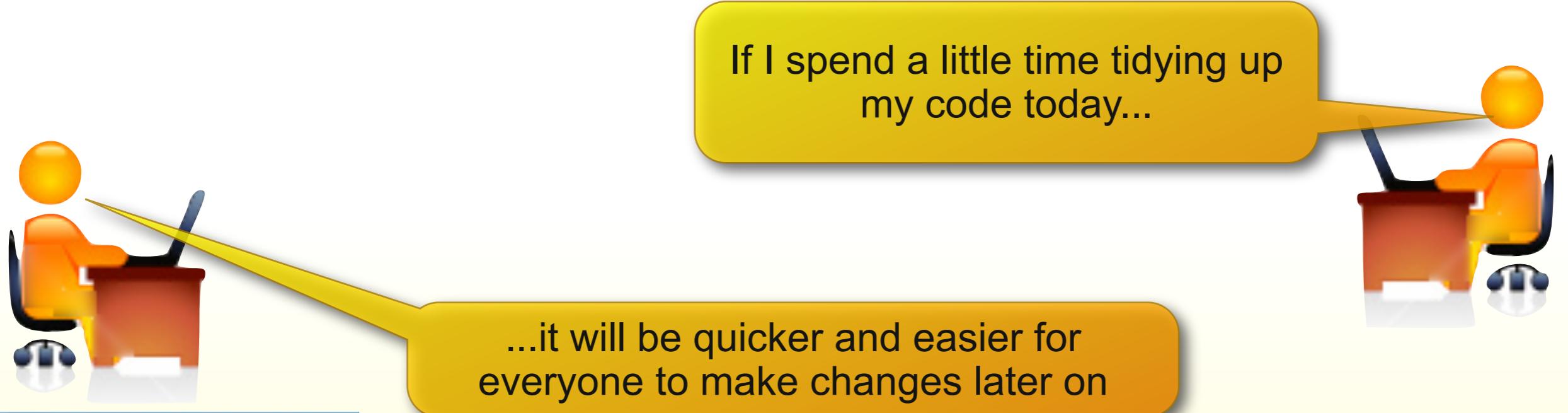
- ▶ What is technical debt?
- ▶ The cost of poor quality code:
 - ▶ Harder to make changes
 - ▶ Too much time spent fixing bugs
 - ▶ Takes too long to add competitive new features



Automated Code Quality

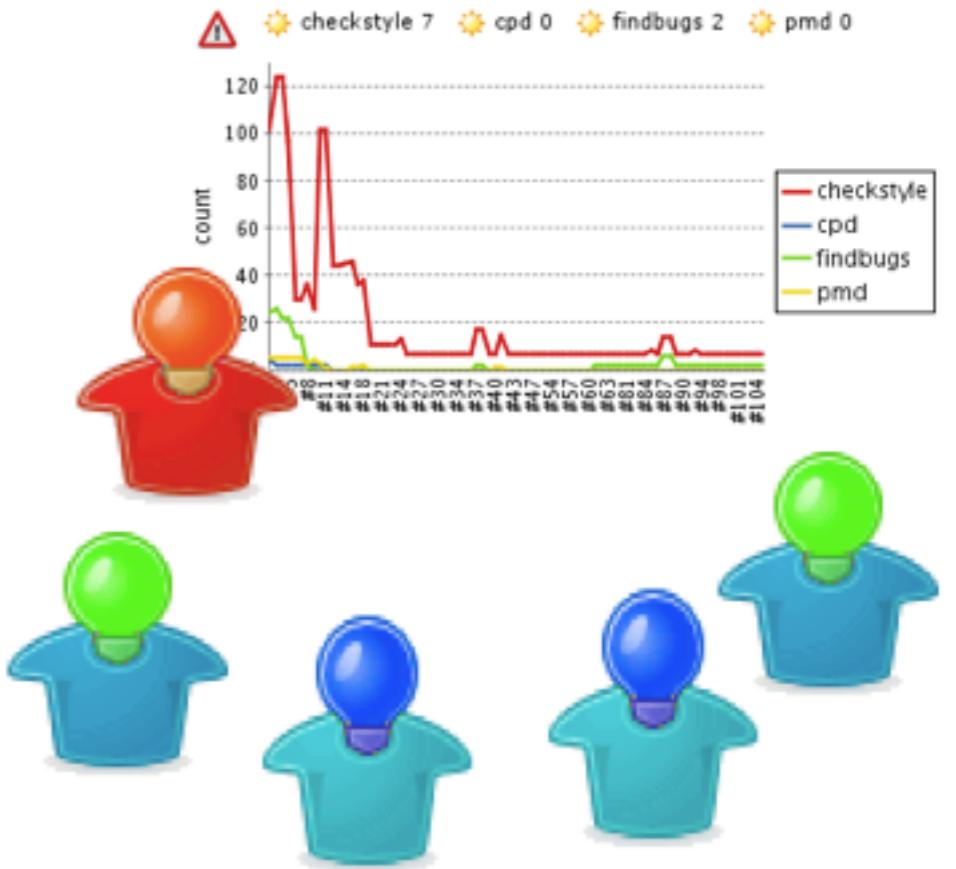
► How do we pay off technical debt?

- Enforce coding standards
- Teach developers good coding practices
- Spend time keeping the code clean (refactoring)



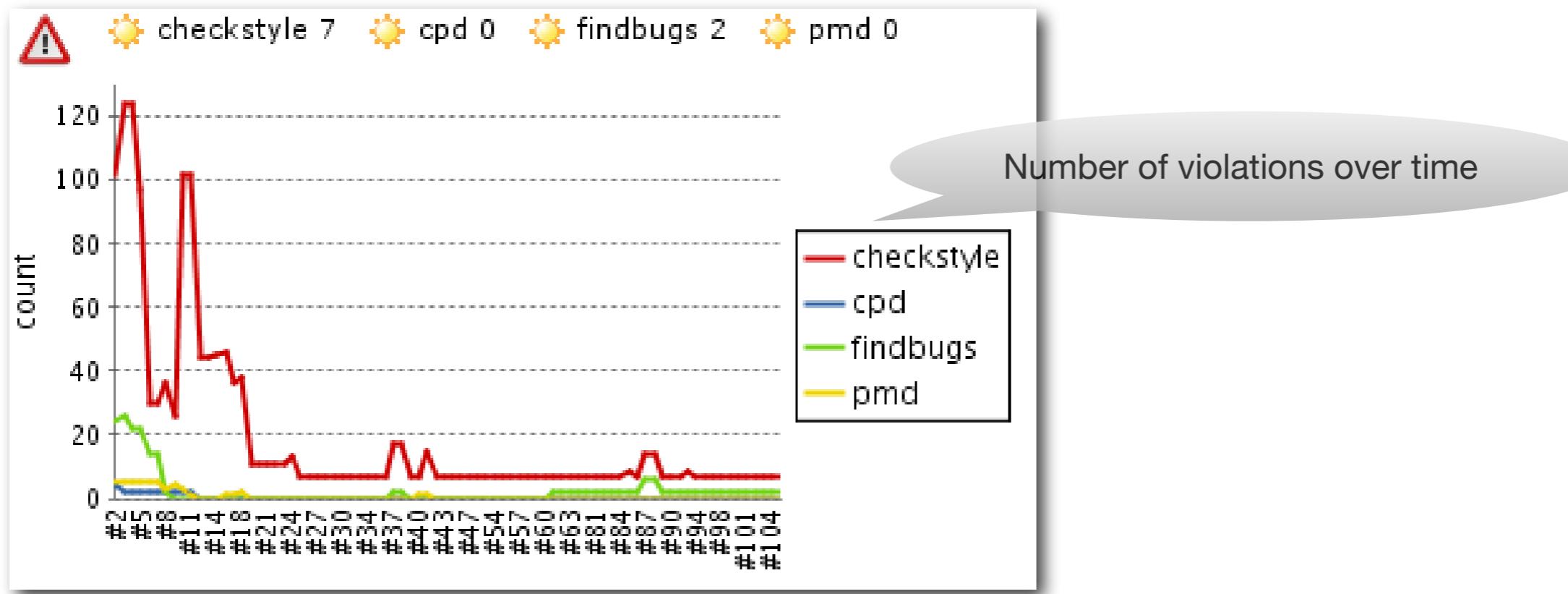
Automated Code Quality

- ▶ Team code reviews
 - ▶ Review code as a group
 - ▶ Long and slow if done manually
 - ▶ Benefits greatly from the use of tools



Automated Code Quality

► Enforcing coding standards with Hudson



Automated Code Quality

► Enforcing coding standards with Hudson

⚠️ Violations Report for build 105

Type	Violations	Files in violation
checkstyle	7	4
cpd	0	0
findbugs	2	1
pmd	0	0

checkstyle

Drilling down

filename
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle

Automated Code Quality

► Enforcing coding standards with Hudson

The screenshot shows a Hudson job results page for a Java project. At the top, there's a red warning icon with an exclamation mark. Below it, the URL is displayed: `data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet`. The main content area has a light gray background.

checkstyle 4 violations

21		Utility classes should not have a public or default constructor.
34		Avoid inline conditionals.
60		'32' is a magic number.
112		Missing a Javadoc comment.

File: HashCodeUtil.java Lines 12 to 44

```
12 *     int result = HashCodeUtil.SEED;
13 *     //collect the contributions of various fields
14 *     result = HashCodeUtil.hash(result, fPrimitive);
15 *     result = HashCodeUtil.hash(result, fObject);
16 *     result = HashCodeUtil.hash(result, fArray);
17 *     return result;
18 * }
19 * </pre>
20 */
21 public final class HashCodeUtil {
```

Details for a particular issue

Type	Class	Description
checkstyle	HideUtilityClassConstructorCheck	Utility classes should not have a public or default constructor.

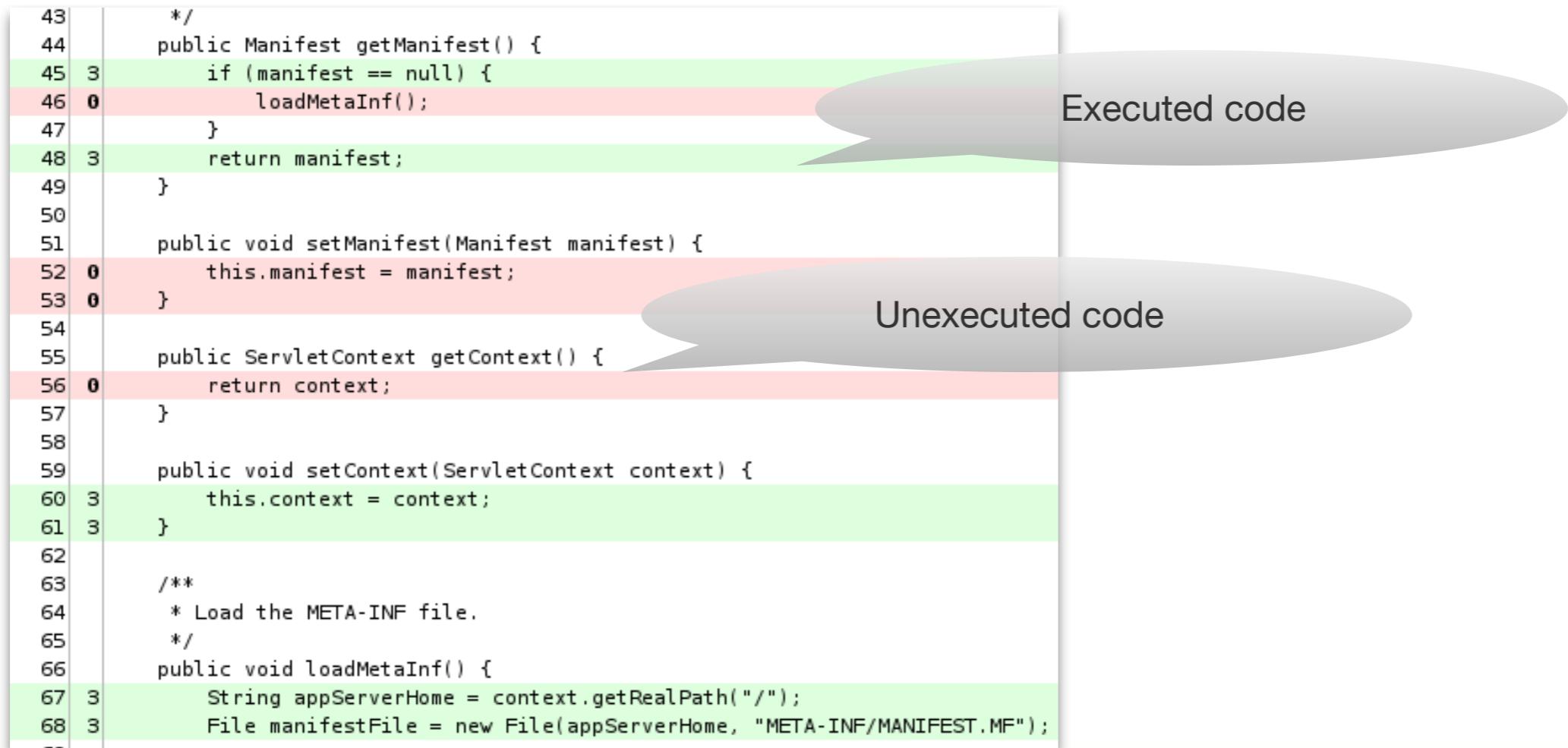
is added contributions
25 * from fields. Using a non-zero value decreases collisions of <code>hashCode</code>
26 * values.
27 */
28 public static final int SEED = 23;

Automated Code Quality

- ▶ Code Coverage
- ▶ See what code is being executed by your unit tests.
- ▶ Isolate untested code
- ▶ Can help to estimate if testing practices are being applied

Automated Code Quality

► Monitoring Code Coverage with Hudson



Automated Code Quality

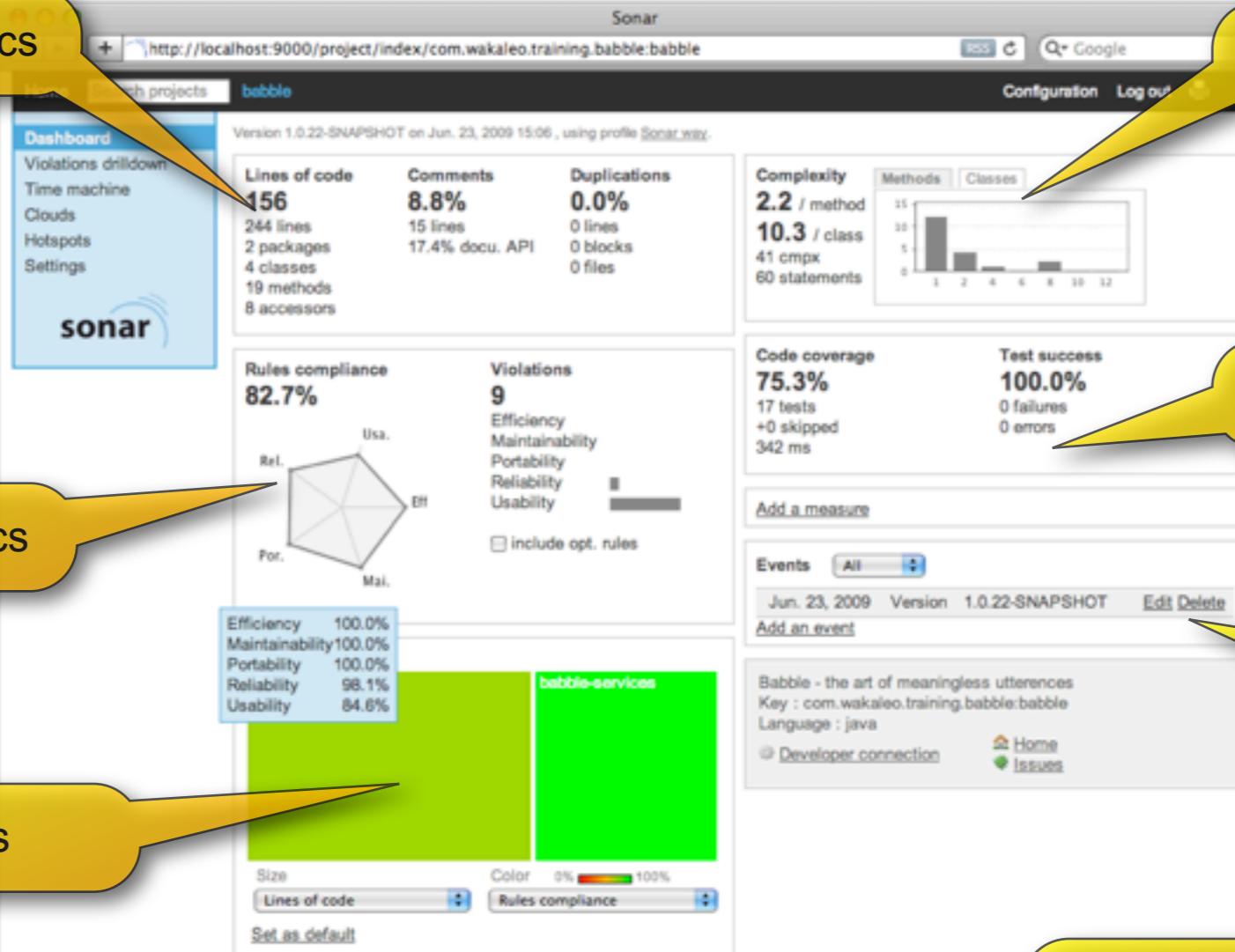
► Code Quality Governance with Sonar

- Centralized code quality management
- Works on any Maven project
- Store code quality metrics in a database
- Code quality metrics can be consulted on a web site

Automated Code Quality

- ▶ Code Quality Governance with Sonar
- ▶ Sonar centralizes many code quality metrics

Source code metrics



Code complexity metrics

Code quality metrics

Test results and code coverage

Modules

Build history

Click anywhere to drill down

Automated Code Quality

- ▶ Code Quality Governance with Sonar
- ▶ You can drill down to view the details for each type of issue

The screenshot illustrates the SonarQube interface for automated code quality governance. It features several yellow callout bubbles pointing to specific parts of the interface:

- A large yellow callout labeled "Overview" points to the top-level dashboard showing violation counts by category (Mandatory: 9, Optional: 18) and rule details.
- A yellow callout labeled "Different types of violations" points to a code editor window displaying multiple violations with error messages and line numbers.
- A yellow callout labeled "Violations in this class" points to a detailed view of violations for the `Babble` class, showing metrics like Lines: 69, Coverage: 74.1%, and a list of violations with their locations.
- A yellow callout labeled "Violation details" points to a detailed view of a specific violation in the code editor, showing the exact line of code and the associated error message.

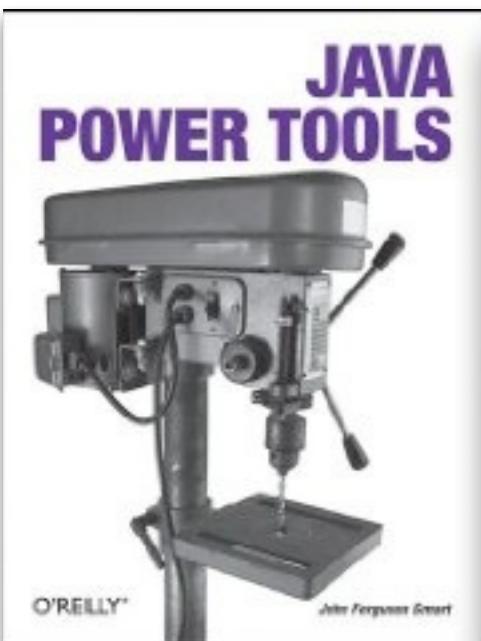
The SonarQube interface includes a sidebar with links to Home, Search projects, Dashboard, Violations drilldown, Time machine, Clouds, Hotspots, and Settings. The main dashboard shows a summary of code quality across various categories and provides a link to the full rule details.

Summary

- ▶ How can you improve the development process?
 - Standardize your build process
 - Improve testing practices
 - Use Continuous Integration
 - Reduce your technical debt
 - Automate, automate, automate!

John Ferguson Smart
Email: john.smart@wakaleo.com
Web: http://www.wakaleo.com
Twitter: wakaleo

Thanks for your attention!



Wakaleo Consulting - Building Software Better
<http://www.wakaleo.com/>

Wakaleo Consulting
Optimizing your software development process

HOME SERVICES TRAINING NEWS RESOURCES JAVA POWER TOOLS ABOUT US

Wakaleo Consulting is a consulting company that helps organizations refine their software development processes. We provide consulting, training and mentoring services in Enterprise Java Development, Software Development Life Cycle optimisation and Agile Methodologies.

Wakaleo Consulting is also behind Java Power Tools, a series of open source tools that can improve productivity across all Continuous Integration, Code Quality and more.

Hone your development skills with the Java Power Tools Bootcamp!

"Best development course I have been on in a very long time...Greatly enjoyed the course...A 'must' course for serious Java developers..." - Read what people are saying about the Java Power Tools Bootcamps.

The Java Power Tools Bootcamp is an intense 4-day hands-on workshop covering some of the best open source tools and state-of-the-art development practices for Java development on the market.

Coming up soon: workshops in Melbourne, Sydney, Brisbane and Wellington. Check out the full schedule here.

This one-of-a-kind course takes you on a in-depth guided tour of some of the best open source Java tools around, showing how you can use them individually and together to write code better, faster and smarter. We cover virtually all aspects of software development, from Build Scripts, SCM Best Practices, Unit and Integration Testing, Continuous Integration, Code Quality and more.

Optimize Your SOFTWARE DEVELOPMENT

<http://www.wakaleo.com/news-and-events>

java.net The Source for Java Technology Collaboration
http://weblogs.java.net/blog/johnsmart/archive/2008/11/managing_autom.html

John Ferguson Smart's Blog

Managing automated build dependencies with Maven and Hudson

One of the tricky parts of setting up a Continuous Integration system is managing dependencies. Many organisations have projects made up of tens or hundreds of different, interrelated modules, with complex dependencies between them. So when you change a module somewhere, you may need to rebuild and retest other modules that depend on this module.

One of the great features of Maven is the way it handles dependency management. Maven detectors may moan and groan about black magic and not being able to see what happens under the hood, but for me, declarative, transitive dependency management is a real life-saver.

Now, when you add Maven dependency management to Hudson automated builds, the result is nothing short of miraculous. Well, maybe I'm exaggerating slightly, but it's certainly very cool. In short, if you use "Maven" build jobs in Hudson instead of the more frequently-used "free-style" build jobs, Hudson will figure out the correct build order all by itself - and rebuild all the other projects directly or indirectly affected by your changes.

Hudson

Let's look at an example. Suppose my team is working on a Maven project called "cone-api". We've already released one version of the "cone-api" (version 1.0.0), which is available on the Maven enterprise repository for other teams. However, we are currently busy adding new features in the 1.0.1-SNAPSHOT version.