Coding in Industry

David Berry

Director of Engineering

Qualcomm Cambridge Ltd

Agenda

- Potted history
- Basic Tools of the Trade
- Test Driven Development
- Code Quality
- Performance
- Open Source

Potted History

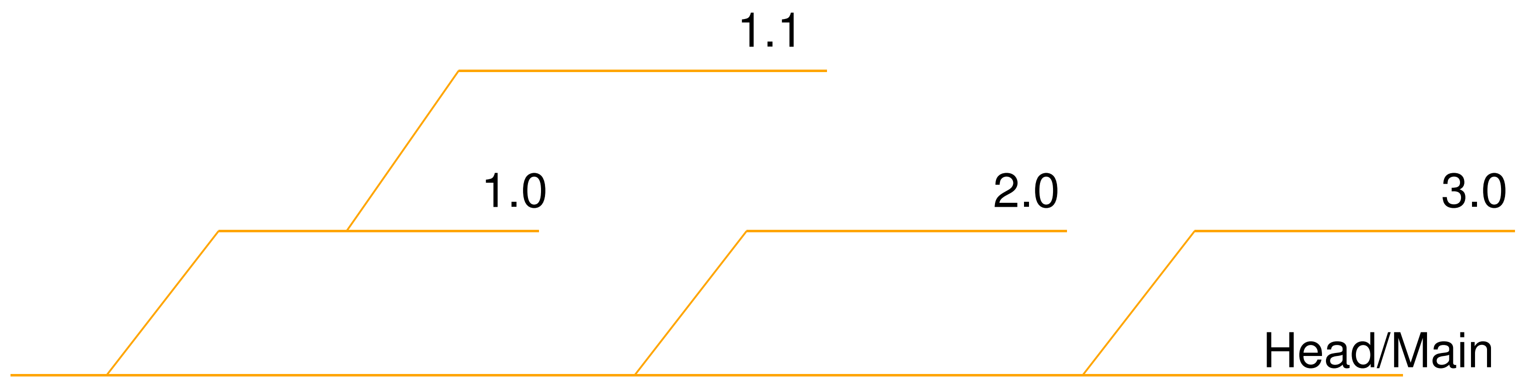
- PhD, Heriot-Watt University
  - Learned programming in C, Unix V7
- Sun Microsystems – Staff Engineer
  - Programming in C and C++
- Harlequin (Cambridge) – Group Manager
  - C, C++, PostScript, PDF
- Qualcomm (aqcuired Trigenix) – Director of Eng
  - C++, Python, Java

TOOLS OF THE TRADE

Source Code Control

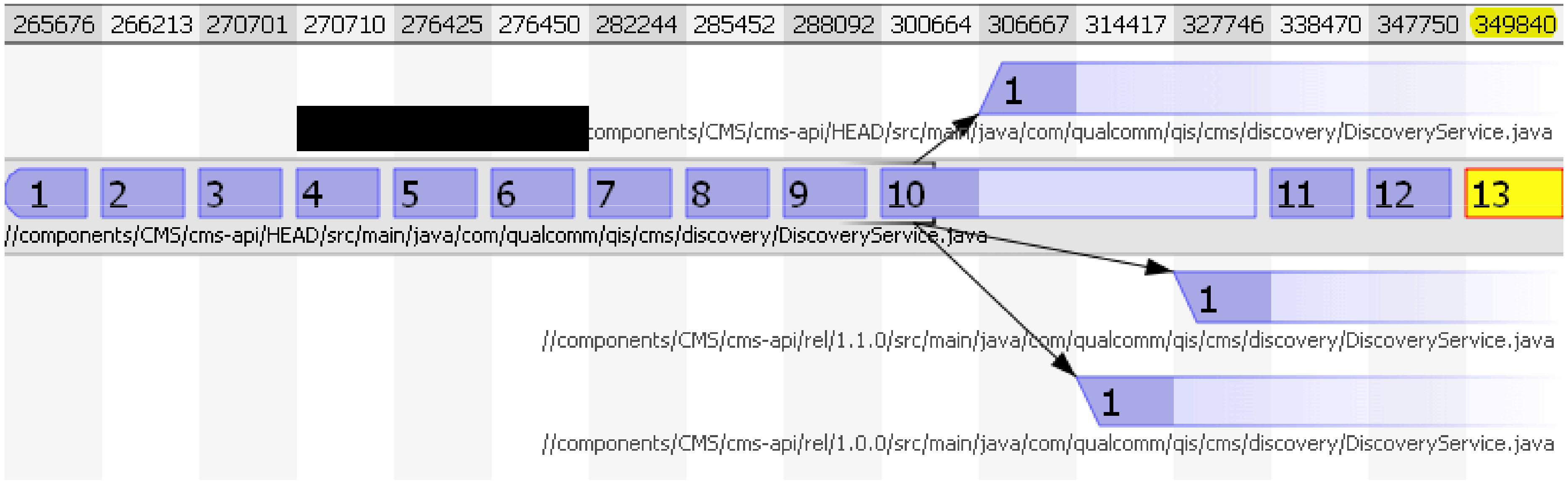
- Allows multiple developers to work in parallel
- Traceability provides a history of changes and why, when things change
- Must be able to re-build releases from scratch
  - Consideration of branches, labels
- Examples
  - CVS, SVN
    - Google Code uses SVN
    - SourceForge recommends SVN with legacy products on CVS
  - Perforce
  - ClearCase

Source Code Control - Branches



- Check in multiple file changes in one go
  - Makes it easier to merge sets of files
- Branches provide a means to develop different product code lines
- Protects released versions, allows you to re-build them from scratch needed for maintenance
- Requires developers to know how to merge code
  - Takes practice and skill to deal with conflicts
  - Tools from the source code control help
- Needs a house policy on which direction to merge from
  - Head or branch first

Single File Branch Example



265676
266213
270701
270710
276425
276450
282244
285452
288092
300664
306667
314417
327746
338470
347750
349840
1
[redacted]
components/CMS/cms-api/HEAD/src/main/java/com/qualcomm/qis/cms/discovery/DiscoveryService.java
1
2
3
4
5
6
7
8
9
10
11
12
13
//components/CMS/cms-api/HEAD/src/main/java/com/qualcomm/qis/cms/discovery/DiscoveryService.java
1
//components/CMS/cms-api/rel/1.1.0/src/main/java/com/qualcomm/qis/cms/discovery/DiscoveryService.java
1
//components/CMS/cms-api/rel/1.0.0/src/main/java/com/qualcomm/qis/cms/discovery/DiscoveryService.java

Writing Code - IDEs

- Essential tools to make development tasks easier
- Examples
  - Eclipse
  - NetBeans
  - Visual Studio
  - SunStudio (C, C++, Fortran)
  - Emacs, vi ☺
    - Gdb
    - dbx

Writing Code – Build Tools

- Make
- Make alternatives
  - Jam
  - Cook
- Home Grown
- Ant
- Maven
- Preference is to have command line driven
  - Allows automation, continuous integration
  - IDE Projects can be accommodated

Maven

- http://maven.apache.org/
- It’s better than ant ☺
- Standard directory layout for code/tests
- Allows you to manage your dependencies
  - Gives you control over open source being used
  - Versioned
- Maven servers provide a means to download dependencies
- IDE Integration (Eclipse, NetBeans)
- Plug in mechanism
- Wide community support
- Auto generation of a project web site

Example POM file

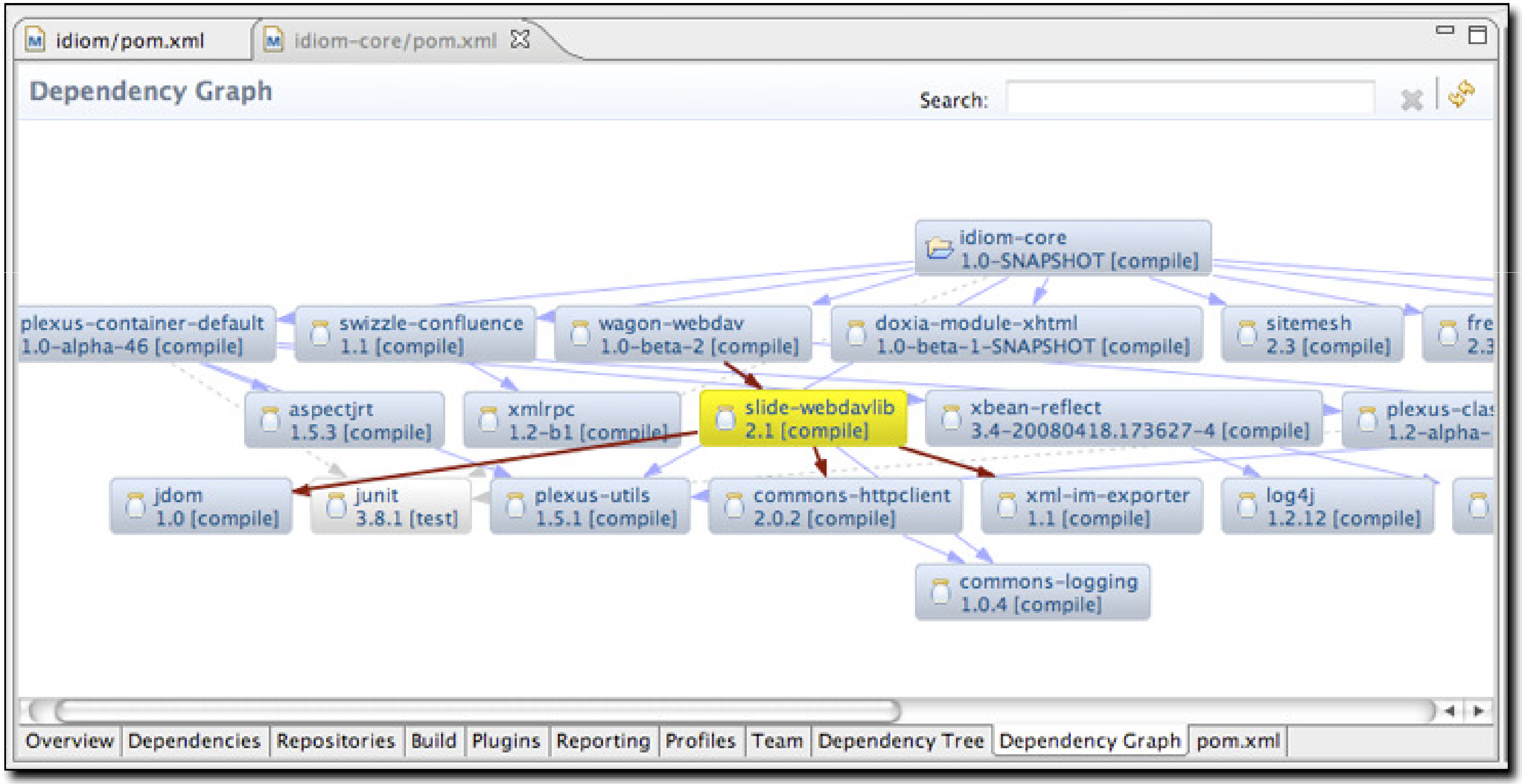
- POM = Project Object Model

```
<parent>
    <groupId>com.qualcomm.qis</groupId>
     <artifactId>oneCMS</artifactId>
           <version>2.0.0.06-SNAPSHOT</version>
</parent>
<groupId>com.qualcomm.qis.oneCMS</groupId>
<artifactId>cms-api</artifactId>
<version>2.0.0.06-SNAPSHOT</version>
<packaging>jar</packaging>

      <!-- Dependencies without version indicate they are inherited from parent pom -->
<dependencies>
    <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>2.5.5</version>
        <exclusions>
            <exclusion>
                <groupId>commons-logging</groupId>
                <artifactId>commons-logging</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
```

Dependency Graph

http://www.sonatype.com/books/m2eclipse-book/reference/eclipse-sect-analyze-depend.h

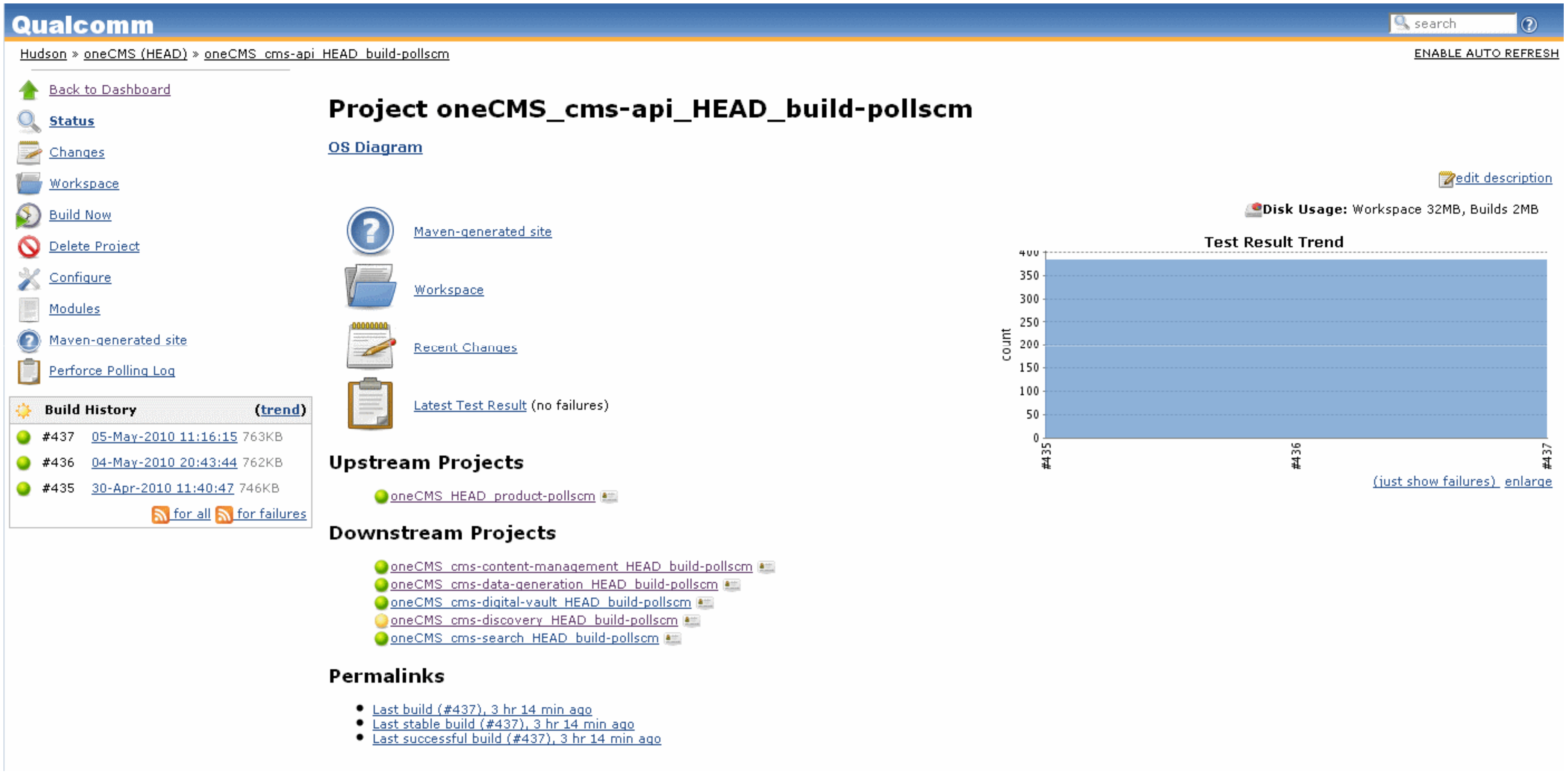


idiom/pom.xml
idiom-core/pom.xml
Dependency Graph
Search:
idiom-core 1.0-SNAPSHOT [compile]
plexus-container-default 1.0-alpha-46 [compile]
swizzle-confluence 1.1 [compile]
wagon-webdav 1.0-beta-2 [compile]
doxia-module-xhtml 1.0-beta-1-SNAPSHOT [compile]
sitemesh 2.3 [compile]
aspectjrt 1.5.3 [compile]
xmlrpc 1.2-b1 [compile]
slide-webdavlib 2.1 [compile]
xbean-reflect 3.4-20080418.173627-4 [compile]
jdom 1.0 [compile]
junit 3.8.1 [test]
plexus-utils 1.5.1 [compile]
commons-httpclient 2.0.2 [compile]
xml-im-exporter 1.1 [compile]
log4j 1.2.12 [compile]
commons-logging 1.0.4 [compile]
Overview
Dependencies
Repositories
Build
Plugins
Reporting
Profiles
Team
Dependency Tree
Dependency Graph
pom.xml

Continuous Integration

- Build code and run tests every time a check in is made
  - Tells you immediately that a build has failed
- Automated
- Essential part of Agile software development
  - It’s just good engineering so do it anyway
- Example Tools
  - Hudson
  - Cruise Control
  - Home grown (qpbuild Python based)
  - TinderBox
- See http://en.wikipedia.org/wiki/Continuous_integration

Hudson Example


Qualcomm

Hudson » oneCMS (HEAD) » oneCMS_cms-api_HEAD_build-pollscm

ENABLE AUTO REFRESH

- Back to Dashboard
- Status
- Changes
- Workspace
- Build Now
- Delete Project
- Configure
- Modules
- Maven-generated site
- Perforce Polling Log

Build History (trend)

#437	05-May-2010 11:16:15	763KB
#436	04-May-2010 20:43:44	762KB
#435	30-Apr-2010 11:40:47	746KB

for all for failures

Project oneCMS_cms-api_HEAD_build-pollscm

OS Diagram

edit description

Disk Usage: Workspace 32MB, Builds 2MB

- Maven-generated site
- Workspace
- Recent Changes
- Latest Test Result (no failures)

Test Result Trend

(just show failures) enlarge

Upstream Projects

- oneCMS_HEAD_product-pollscm

Downstream Projects

- oneCMS_cms-content-management_HEAD_build-pollscm
- oneCMS_cms-data-generation_HEAD_build-pollscm
- oneCMS_cms-digital-vault_HEAD_build-pollscm
- oneCMS_cms-discovery_HEAD_build-pollscm
- oneCMS_cms-search_HEAD_build-pollscm

Permalinks

- Last build (#437), 3 hr 14 min ago
- Last stable build (#437), 3 hr 14 min ago
- Last successful build (#437), 3 hr 14 min ago

Defect Tracking

- There will be bugs so we need to track them
- Used to track defects reported
  - Another measure of quality
  - Used in release notes to say what was fixed
- Tools
  - Bugzilla
  - TeamTrack
  - Quality Center
  - VersionOne
  - JIRA

TEST DRIVEN DEVELOPMENT

Test Driven Development

- Writing tests is often some piece of throw away code
  - You develop it, make sure the code you are writing works then move on
- Arrival of test frameworks like JUnit has changed this
  - Similar frameworks exist for other languages
- Write the tests before writing the code
  - Helps you think about the API by writing tests
  - Tests allow you to change the code more easily
  - http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd
- Measure the code coverage (%age lines executed) your tests give you
  - Use the debugger to single step code
  - Tools
    - Sonar based tools for CI, http://nemo.sonarsource.org
    - http://www.eclemma.org/index.html (Eclipse plug in)
    - Rational
    - gcov

Automated Test

- Repeatable the machine doesn’t get tired of doing the same thing
  - Provides a regression suite
- JUnit
  - Can be used to write pure unit tests and integration tests
  - Integration tests need some other service, eg an Oracle/MySQL database
  - Maven provides a standard place for these
  - Drives code coverage measurement
  - Other extensions of JUnit exist
- Python PyUnit
- C++ CPPUnit

- Selenium used for wider system test
  - GUI
  - Harder to get code coverage (requires an instrumented build deployed)
  - Other tools exist

- Quality of the test code is just as important as the code itself

- Opportunities
  - JavaScript
  - CSS (Validation available)

Manual Testing

- Some manual test will always be required
- Frequently for look and feel issues in Uis
- An experienced tester can flush out many edge cases that developers tend not to think about
  - For example on a web form filling the field with a large number of characters
    - The system will often not check and fail at trying to insert the data into the database

Sonar Code Coverage

- Sonar - Sonar.pdf

Frameworks to aid Unit Test

- A pure unit test only tests the code you are writing
  - Need to mock out underlying layers
  - Provide dummy code that implements an interface
- EasyMock – can generate mock objects on the fly

Spring Framework

- http://www.springsource.org/
- Uses Inversion of Control (IOC) and dependency injection
  - http://en.wikipedia.org/wiki/Inversion_of_control
  - http://en.wikipedia.org/wiki/Dependency_injection
- Code written to interfaces
  - Allows the implementation to be configured
  - Code can be unit tested key to our unit testing
  - Use of Plain Old Java Objects (POJOs)
    - Code does not know what environment it is being used in done by dependency injection
    - Dependencies usually specified in XML files
    - Solves problems of EJB2.0 which always required a container to run the code in
  - Hypersonic is an in memory database which can be used to mock Oracle/MySQL
- Other frameworks along these principles exist for other languages
  - Ruby
  - Python
  - Google-Juice (Java)

CODE QUALITY

Writing Clean Code

- To be maintainable code needs to be “Clean”
  - Projects, products fail when you own a mess
  - Messes happen over time as changes are made
  - Developers end up not wanting to change the code for fear of breaking it, test costs rise
  - Developers write the code not anyone else
  - Developers move around the same people that started the project usually aren’t there a few years later

- For example,
    - Naming matters
    - Smaller methods/functions
    - You don’t need lots of comments that get out of date as the code moves
  - Robert C Martin, Clean Code
  - http://wiki.java.net/bin/view/People/SmellsToRefactorings

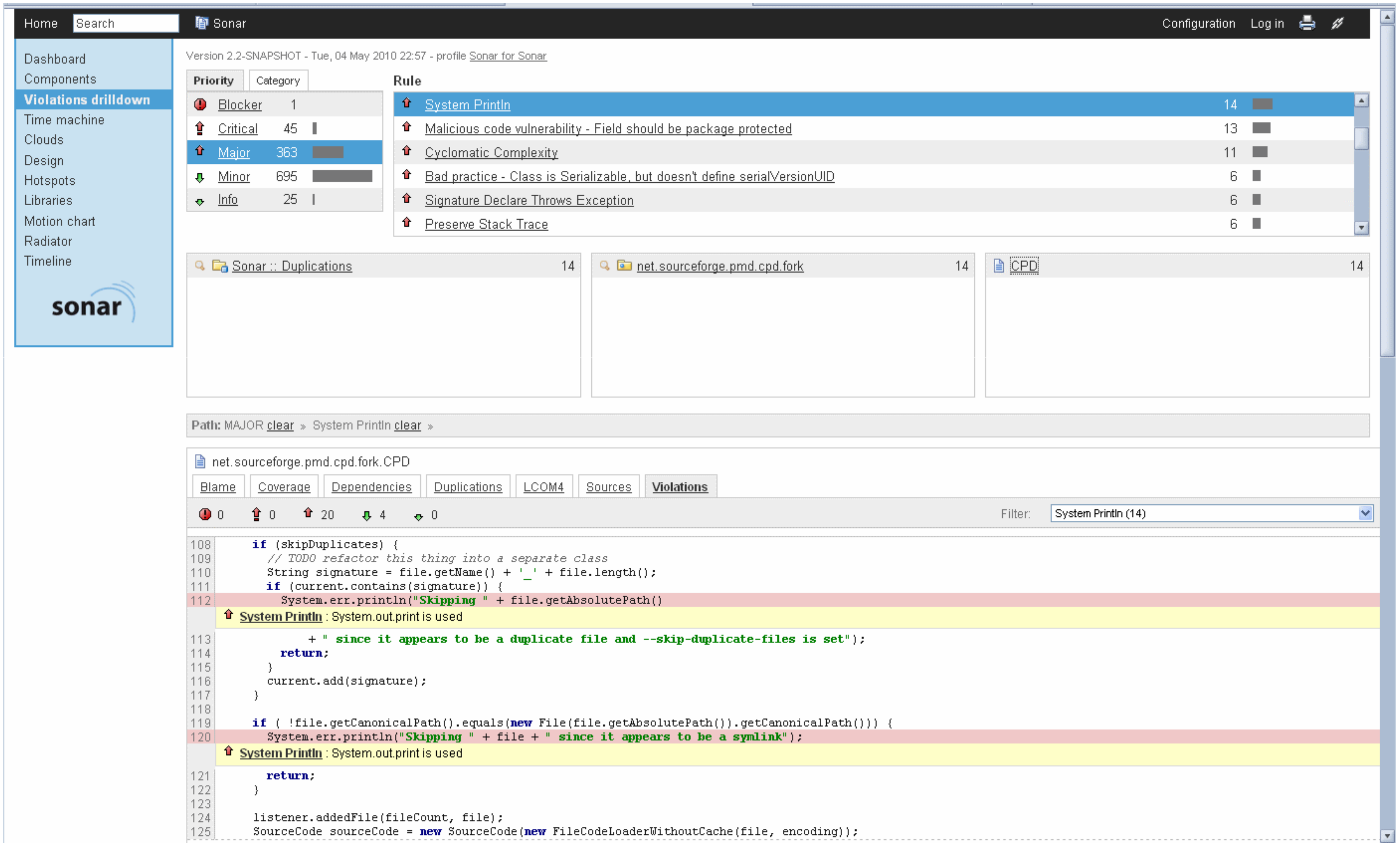
Sample Code Quality Rules

- All Code
  - Must follow the check in rules for the project
  - Check in comments should tell you why the change is being made and a description of the change the BI number is not enough or "code coverage" for example
  - Check in comments must include the BI, Defect task number
  - All code changes should be reviewed via Crucible or by review with a colleague
  - External APIs must have corresponding javadoc
  - If the build breaks (including test failures) due to a change, your first priority is to fix it
  - If the Selenium tests fail due to a change, your first priority is to fix it
- New code
  - All code will have a corresponding set of unit and integration tests where appropriate
  - Minimum of 75% code coverage, aiming for 90%+
  - 0 (zero) compliance warnings added
- Existing code (when changed)
  - At worst, no decrease in code coverage for the code in question, aim to raise it to new code levels
  - At worst, no increase in code compliance violations
  - Clean as you go – always leave the code better than you found it, eg
    - add tests
    - fix broken tests
    - remove code compliance issues
    - re-factor the code to improve it, make it more readable, cleaner, remove duplications

Static Code Analysis Tools

- Tools
  - Lint
  - Eclipse/Compiler warnings for example
    - Unused imports
  - PMD
  - findBugs
  - CheckStyle
- Combined with continuous integration give you a running measure of code quality

Sonar Code Compliance



Version 2.2-SNAPSHOT - Tue, 04 May 2010 22:57 - profile Sonar for Sonar

Priority	Category	
Blocker	1	
Critical	45	
Major	363	
Minor	695	
Info	25	

Rule	
System Println	14
Malicious code vulnerability - Field should be package protected	13
Cyclomatic Complexity	11
Bad practice - Class is Serializable, but doesn't define serialVersionUID	6
Signature Declare Throws Exception	6
Preserve Stack Trace	6

Sonar :: Duplications	14
net.sourceforge.pmd.cpd.fork	14
CPD	14

Path: MAJOR clear » System Println clear »

net.sourceforge.pmd.cpd.fork.CPD

Blame | Coverage | Dependencies | Duplications | LCOM4 | Sources | Violations

0 0 20 4 0 Filter: System Println (14)

```
    if (skipDuplicates) {
      // TODO refactor this thing into a separate class
      String signature = file.getName() + '_' + file.length();
      if (current.contains(signature)) {
        System.err.println("Skipping " + file.getAbsolutePath()
System Println : System.out.print is used
            + " since it appears to be a duplicate file and --skip-duplicate-files is set");
        return;
      }
      current.add(signature);
    }

    if ( !file.getCanonicalPath().equals(new File(file.getAbsolutePath()).getCanonicalPath())) {
      System.err.println("Skipping " + file + " since it appears to be a symlink");
System Println : System.out.print is used
      return;
    }

    listener.addFile(fileCount, file);
    SourceCode sourceCode = new SourceCode(new FileCodeLoaderWithoutCache(file, encoding));
```

http://nemo.sonarsource.org/

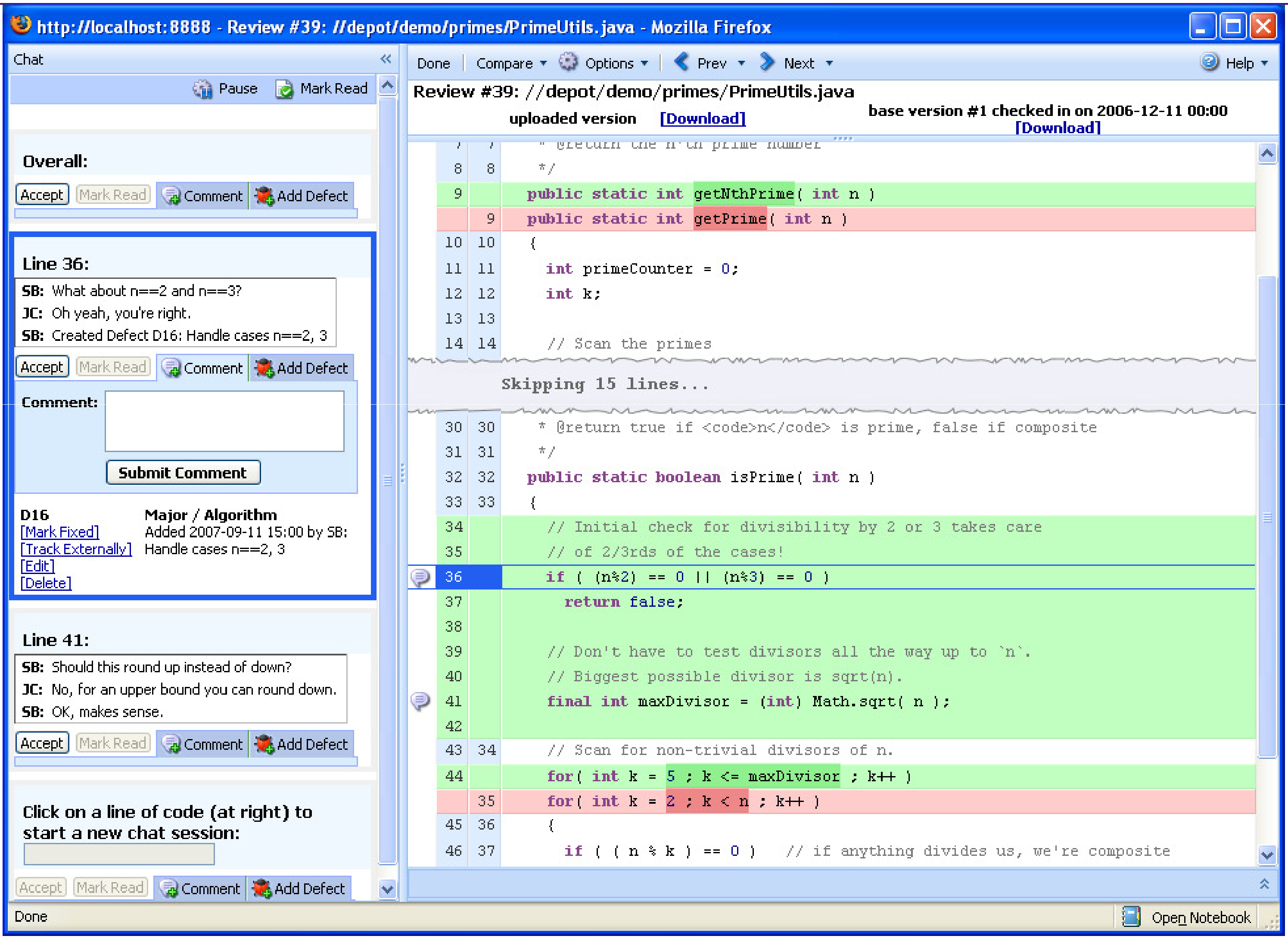
Sample Rules

Unused Private Field	UnusedPrivateField	Maintainability	pmd	BLOCKER	ACTIVE	
Unused formal parameter	UnusedFormalParameter	Maintainability	pmd	MAJOR	ACTIVE	
Unused local variable	UnusedLocalVariable	Maintainability	pmd	BLOCKER	ACTIVE	
Unused private method	UnusedPrivateMethod	Maintainability	pmd	BLOCKER	ACTIVE	
Use Array List Instead Of Vector	UseArrayListInsteadOfVector	Efficiency	pmd	MINOR	ACTIVE	
Use Arrays As List	UseArraysAsList	Efficiency	pmd	MAJOR	ACTIVE	
Use Correct Exception Logging	UseCorrectExceptionLogging	Maintainability	pmd	CRITICAL	ACTIVE	
Use Index Of Char	UseIndexOfChar	Efficiency	pmd	MAJOR	ACTIVE	
Use String Buffer Length	UseStringBufferLength	Efficiency	pmd	MAJOR	ACTIVE	
Useless Operation On Immutable	UselessOperationOnImmutable	Reliability	pmd	BLOCKER	ACTIVE	
Useless Overriding Method	UselessOverridingMethod	Maintainability	pmd	BLOCKER	ACTIVE	
Useless String Value Of	UselessStringValueOf	Efficiency	pmd	MAJOR	ACTIVE	
Visibility Modifier	com.puppycrawl.tools.checkstyle.check	Maintainability	checkstyle	MAJOR	ACTIVE	
While Loops Must Use Braces	WhileLoopsMustUseBraces	Usability	pmd	BLOCKER	ACTIVE	

Code Review/Inspection

- Possibly the most effective method of finding bugs, design issues in code
- Pair Programming (an aspect of Extreme programming) encourages this
- Important to note that code review should be about the code not the person
- Tools help to do this in a distributed or time shifted groups
  - CodeCollaborator
  - Crucible/Fisheye
  - Or just print it out and read through the code

Code Collaborator



http://localhost:8888 - Review #39: //depot/demo/primes/PrimeUtils.java - Mozilla Firefox

Overall:

Line 36:

SB: What about n==2 and n==3?
JC: Oh yeah, you're right.
SB: Created Defect D16: Handle cases n==2, 3

Comment:

D16	**Major / Algorithm**
[Mark Fixed]	Added 2007-09-11 15:00 by SB:
[Track Externally]	Handle cases n==2, 3
[Edit]	
[Delete]	

Line 41:

SB: Should this round up instead of down?
JC: No, for an upper bound you can round down.
SB: OK, makes sense.

Click on a line of code (at right) to start a new chat session:

Review #39: //depot/demo/primes/PrimeUtils.java

uploaded version [Download]

base version #1 checked in on 2006-12-11 00:00 [Download]

```
 7  7     * @return the n'th prime number
 8  8     */
 9       public static int getNthPrime( int n )
    9    public static int getPrime( int n )
10 10    {
11 11      int primeCounter = 0;
12 12      int k;
13 13
14 14      // Scan the primes

         Skipping 15 lines...

30 30     * @return true if <code>n</code> is prime, false if composite
31 31     */
32 32    public static boolean isPrime( int n )
33 33    {
34         // Initial check for divisibility by 2 or 3 takes care
35         // of 2/3rds of the cases!
36         if ( (n%2) == 0 || (n%3) == 0 )
37           return false;
38
39         // Don't have to test divisors all the way up to `n`.
40         // Biggest possible divisor is sqrt(n).
41         final int maxDivisor = (int) Math.sqrt( n );
42
43 34      // Scan for non-trivial divisors of n.
44         for( int k = 5 ; k <= maxDivisor ; k++ )
   35      for( int k = 2 ; k < n ; k++ )
45 36      {
46 37        if ( ( n % k ) == 0 )   // if anything divides us, we're composite
```

Done

http://smartbear.com/codecollab.php

PERFORMANCE & MISCELLANOUS

Performance

- Begins with the architecture
  - Think about how your system would scale to the number of users
  - How responsive does the UI need to be users won't use your site if it appears slow
- Needs to be thought about when coding
  - Database usage, sql indexes for example
  - Web Service calls are expensive
  - Use of caches
  - Check the code another use of single stepping in the debugger
- Superficially cheap activities soon add up when called millions of times
- Measure performance first then optimize where needed
  - You can spend a lot of time optimizing something that doesn't need to be

Measuring Performance

- Response times
  - Under load
  - How many concurrent users do you have
- Soak testing
  - Long term testing looking for memory leaks
  - Would like to see the classic Java sawtooth pattern
  - Degradation in performance over time
  - Usually takes several weeks to run
- Tools
  - JMeter
  - Grinder
- Performance profiling tools
  - Tell you how often a method was called how long it took
    - Built into JDK 1.5 and later
    - Rational Tools
  - May have to use logging on servers with timers
    - Spring AOP can be used to measure calls without affecting the code

Logging

- Log4j
  - Imitated in other languages
    - Python
    - C++
- Needed for server products to trace/track issues
- Log4j has a set of log levels (Info, Debug, Warning, Error)
  - Log level determines what to print
  - It is faster to check the log level in your code then call the logger rather than letting strings be constructed that are discarded

Database Usage

- Don’t just use it as a place to store object data
- Use the power of the database
  - i.e. don’t try to do the databases job in code
  - Sort in the database for example
- Use persistence frameworks such as Hibernate are good to a certain level
  - When it comes to making a system perform you almost always end up wanting to be in control of the SQL

UI Development

- Good easy to use UI development is hard
- User driven
  - Not just tables on databases
- Requires multi-disciplinary team
  - User interaction
  - Visual design
  - Web Developer, HTML/CSS
  - Server developer to provide apis
    - APIs should be driven from user usage

Web Containers

- Tomcat
  - Mainly used in development
  - Simple to deploy
  - Integrated with Eclipse
- JBoss
  - Used in deployment
  - Can be used in development
  - Eclipse Integration
- WebSphere (IBM)
  - Used in deployment
  - Installs can be scripted

Monitoring

- Essential for long running server products
- Simple Network Management Protocol
  - http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
- Java Management Extensions
  - Standard part of JDK 1.5
  - Allow you to change properties of the system
  - http://en.wikipedia.org/wiki/JMX

OPEN SOURCE

Open Source

- Used with care provides a huge amount of time saving for projects
  - Headcount is usually the biggest expense on projects

- Lots of contributors developing code usually means it’s well tested
  - It doesn’t guarantee it’s well documented, you do have access to the source though

- Understand the licences (http://www.opensource.org/licenses/alphabetical)

- Some licences are more commercial friendly for example
  - Apache 2.0
  - MIT
  - BSD

- Less commercial friendly include
  - GPL
  - LGPL
  - Mozilla
  - Eclipse

Open Source

- Licences determine the conditions of usage
  - Respect them
  - Know what your implications are before using them
  - Does the code contain encryption (see export compliance)
  - What happens if you change the code
- You can’t just lift code from other sites
- Companies now make tools to check companies use of Open Source
  - BlackDuck (http://www.blackducksoftware.com/)
  - Home grown scanning tools

Open Source Usage

- Spring Framework
- UI
  - Spring Web Flow
  - Dojo (Javascript Library)
- Apache
  - JUnit
  - Commons
  - Maven
  - Tomcat

Export Compliance

- USA Based companies must comply when any software is shipped outside the USA
  - Companies must apply to the US Government for an export compliance status
- UK and other countries have export compliance rules
- Mainly concerned with encryption

Conclusion

- Taster of the sorts of things we need to think about when developing code
- Projects/Products last years
  - You must be able to maintain it as the team of developers change
  - You must be able to change it with confidence
    - A regression suite is invaluable in allowing you to do that
  - Performance counts
    - Testable
    - Scalable
- Open Source usage matters
- The tools are there to help you use them
  - In the Java/Python/Ruby world a great deal of them are free

THANK YOU FOR YOUR TIME

ANY QUESTIONS