Tools to support development in JAVA

Highlights

- Documentation
- Organization => Compile Chain
- Debugging
- Monitoring
- Pre-compiler
- Retro-Compilation
- Obfuscator
- Performance analyzer
- Conformance test
- Continuous integration
- Versioning

Documentation

- Javadoc (JDK)
  - Automatic generation of HTML documentation
  - Using comments in java files
- Syntax

```
/**
 * This is a <b>doc</b> comment.
 * @see java.lang.Object
 * @todo fix {@underline this !}
 */
```

- Includes
  - class hierarchy, interfaces, packages
  - detailed summary of class, interface, methods, attributes
- Note
  - Add doc generation to your favorite compile chain

Coding Conventions

- To simplify code review
- Rules on the coding style :
  - Apache, Oracle and others template
- Verification tools
  - CheckStyle, PMD, JackPot, Spoon VSuite…

Tools to Improve your Source code

- Formatting tools
  - Indenteurs (Jindent), beautifiers, stylers (JavaStyle), …
- « bug fixing » tools
  - Spoon VSuite, Findbugs (sourceforge) …
- Quality report tools : code metrics
  - Number of Non Comment Code Source), Number of packages, Cyclomatic numbers, …
    - JavaNCCS, Eclipse Metrics …

Compile chain

- Tools
  - make, gmake, nmake (Win),
  - Apache ANT, Apache MAVEN, Freshmeat 7Bee ...
- To automate:
  - pre-compilation, obfuscation, verification
  - generation of .class and .jar
    - normal, tracing, debug, …
  - documentation generation
  - « stubs » generation (rmic, idl2java, javacard …)
  - test
  - …

Maven

- Goal
  - Separation of concerns applied to project build
    - Compilation, code generation, unit testing, documentation, …
  - Handle project dependencies with versions (artifacts)
- Project object model (POM)
  - abstract description of the project
  - Property inheritance from POM parents
- Tools (called plugin)
  - To compile, generate documentation, automate test …

- Note: more and more useful !

Maven Motivation

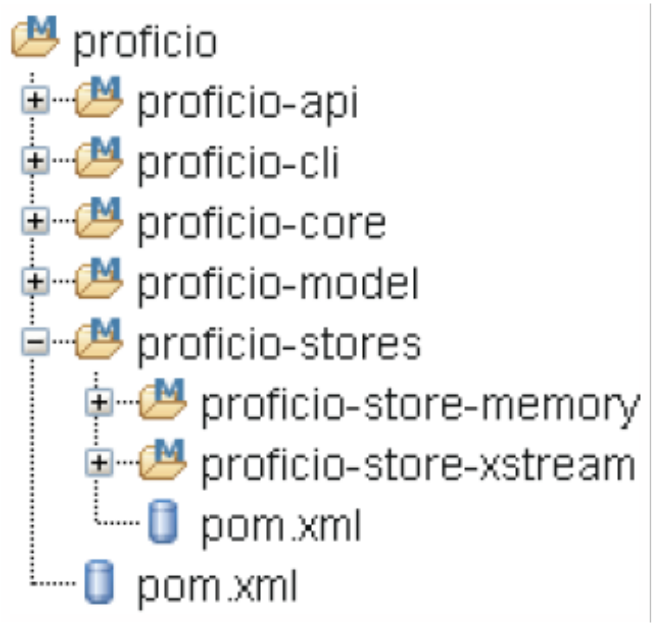
- Abstract project model (POM)
  - Object oriented, inheritance
  - Separation of concerns
- Default lifecycle
  - Default state (goals) sequence
    - plugins depend on states
- Give a project « standard » structure
  - Standard naming conventions
  - Standard lifecycle
- Automatic handling of dependencies between projects
  - Chargement des MAJ
- Project repositories
  - public or private, local or remotes
  - caching and proxy
- Extensible via external plugins

Maven plugins

- Core
  - clean, compiler, deploy, install, resources, site, surefire, verifier
- Packaging
  - ear, ejb, jar, rar, war, bundle (OSGi)
- Reporting
  - changelog, changes, checkstyle, clover, doap, docck, javadoc, jxr, pmd, project-info-reports, surefire-report
- Tools
  - ant, antrun, archetype, assembly, dependency, enforcer, gpg, help, invoker, one (interop Maven 1), patch, plugin, release, remote-resource, repository, scm
- IDEs
  - eclipse, netbeans, idea
- Others
  - exec, jdepend, castor, cargo, jetty, native, sql, taglist, javacc, obr …

Project hierarchies

- Motivations
  - Organize development in sub-projects
    - With N levels (N>=1)
- Technique
  - Create a super POM (type *pom*) for each nesting level
    - Place shared plugins/goals at the same level
  - Subprojects (called modules) inherit from this super pom
- Example



```
proficio
├── proficio-api
├── proficio-cli
├── proficio-core
├── proficio-model
├── proficio-stores
│   ├── proficio-store-memory
│   ├── proficio-store-xstream
│   └── pom.xml
└── pom.xml
```

- Command
  - mvn clean install
    - Global construction

Maven plugin for JAVA IDE

- Maven plugins exists for
  - Eclipse (does not work particularly well ☹)
  - Intellij
  - NetBeans
  - …

Debugging

- Symbolic debugging
  - javac options: -g, -g:source,vars,lines
  - command-line debugger : jdb (JDK)
    - commands look like those of dbx
  - graphical « front-ends » for jdb (AGL)
  - Misc
    - Multi-threads, Cross-Debugging (-Xdebug) on remote VM , ...

Monitoring

- Tracer
  - TRACE options of the program
  - can slow-down .class with TRACE/🗂TRACE tests
    - solution : use a pre-compiler (excluding trace calls)
  - Kernel tools, like OpenSolaris DTrace (coupled with the JVM)

- Logger
  - Record events on a registry, to be used at execution time or later on (via some event handlers)
  - Tools
    - Apache Log4J, ObjectWeb MonoLog
    - Package java.util.logging since J2SE1.4
      - Logger, LogRecord, Handler

Validation (i)

- Assertion
  - Pre-Condition, Post-Condition, Invariant
    - EIFFEL, CLU … built-in
    - Java since SE 1.4
- Tools for J2SE 1.3 et less (J2ME, JavaCard, …)
  - AssertMate (Reliable Software Technologies)
    - http://www.ddj.com/articles/1998/9801d/9801d.htm#rel
  - JML (Java Modeling Language)
    - http://www.eecs.ucf.edu/~leavens/JML/
  - iContract (Reliable Systems)
  - … Design by Contract with JML (by Gary T. Leavens and Yoonsik Cheon)

Refactoring

- Motivation
  - Reorganizing a set of code sources
    - without modifing the external behaviour

  - Package tree
  - New methods
  - Changing parameters

- Built-in support in all modern IDEs
- Batchs jarjar

Reverse-compilation

- Bytecode decompilation
  from Java .class to Java source
  - Partial/Total
- « attacker » goals
  - name of a bean, class, methods, comments, traces, debugging informations, have a meaning
  - to discover the algorithms behind a business component, to modify its source (ie, to delete license verification code), to hack it...
- Developer risks:
  - Copyright losses, loss of incomes ...

First pass: The Obfuscator

- Goal : Avoid bytecode interpretation done by retro-compilers
  - Solution : make the source code unreadable before distribution
- Techniques applied by these tools:
  - transformation of classes names, methods, attributes (a0001, …)
  - mix attribute visibility (public, private, …)
  - delete debugging informations
- Note : delete your traces and debug informations
- Attention : makes hard to mantain your code « Error a238 : send this message to our helpdesk debug@mycomp.com »
- Other usecase: reduce code footprint for very restrained platforms (J2ME/CDLC)

Second pass: Watermarking

- Goal : test retro-compilation and its usage
- How
  - http://www.cs.arizona.edu/sandmark/
  - Add a datastructure W inside a program P
  - W must be recognizable/trackable ! (for testing) and robust to translation, optimization, obfuscation;

Decompilers and Obfuscators

- Decompilers
  - JDK javap, WingDis, NMI 's Java Code Viewer, JAD, DeCafe, …
- Obfuscators
  - Mocha Source Obfuscator, WingDis, NMI 's Java Code Viewer, JAD, Hashjava, Jmangle, Zelix KlassMaste, RetroGuard, Dash-O, …
- Digg more:
  - Dave Dyer, Java decompilers compared, JavaWorld (July 1997), http://www.javaworld.com/javaworld/jw-07-1997/jw-07-decompilers.html
  - Qusay H. Mahmoud, Java Tip 22: Protect your bytecodes from reverse engineering/decompilation, http://www.javaworld.com/javatips/jw-javatip22i.html

Licensing

- Some numbers
  - Illegal use of software
    - 50% in Europe
    - 95% in Asia, South America, Eastern Europe
  - Loss of income for developers
    - $12 billions/year world-wide
    - $3 billions/year in USA only

Licensing

- Software
  - Class verify the signature of (Nom+Société+@MAC+Key)
    - Key is sent after Web registration
  - Stop lauching and execution in several point if the signature is erroneous
  - ☹ Replicated N times the same key !
    - ☹ hastalavista.sk
  - ☹ Cracking (license checking is bypassed)
    - ☹ Patch supress licence checking
- Hardware
  - Customer VM / Custom ClassLoader
    - CL or VM uses a decryption Key (present in a Dongle or SmartCard) to decrypt encrypted bytecode and verify the signature
    - Presence of the dongle is checked regulary

Performances

- Measure/Analyze
  - Benchmark
  - java.awt.Robot (to build clients for testing)
  - Accounting : http://abone.unige.ch/jraf/index.htm
  - JProfiler, OptimiszeIt …
- Optimization
- See books:
  - Steve Wilson, Jeff Kesselman, « Java Platform Performance: Strategies and Tactics (The Java Series) », 1 edition (May 25, 2000), Addison-Wesley Pub Co; ISBN: 0-201-70969-4
  - Jack Shirazi, "Java Performance Tuning", Ed Oreilly, 2000, ISBN 0-596-00015-4

Choosing the JVM and JRE

- Some criteria
  - License, redistribution, supports, performances, contraintes (embedded, servers, réal-time, …), runtimes, …
- Examples
  - Sun HotSpot JVM + JRE
  - MicroSoft JVM (deprecated)
  - IBM J9
  - BEA JRockit
  - HP Chai
  - Macromedia JRun
  - Blackdown JVM
  - Kaffe + GNU Classpath
  - CReME
  - Cacao http://www.cacaojvm.org/
  - Apache Harmony
  - JamVM
  - JRate
  - …

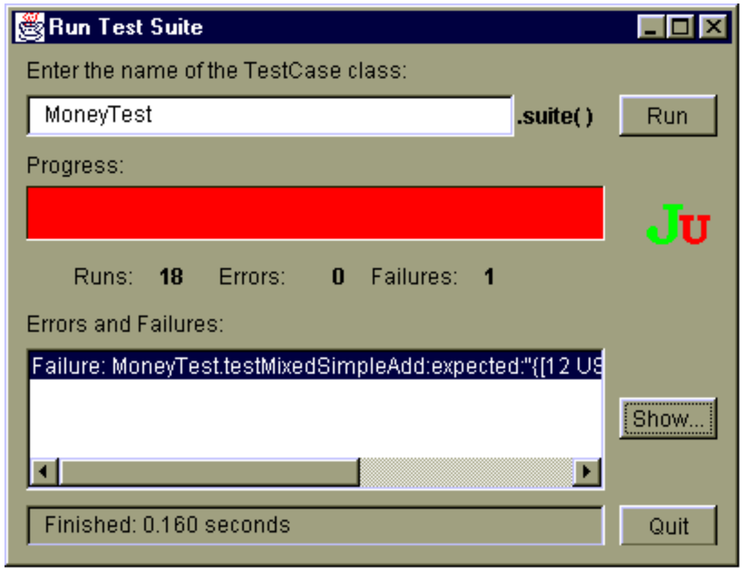
Test

- Tools to manage tests
  - « Test » every possible case
- Unit Test
  - JUNIT (http://www.junit.org) the most famous
  - Cactus (for Servlets)
  - Jcover (CodeWork)
- Coverage test
  - See methods or code branches not covered by tests.
  - Quilt (http://quilt.sourceforge.net)
- Continuous integration

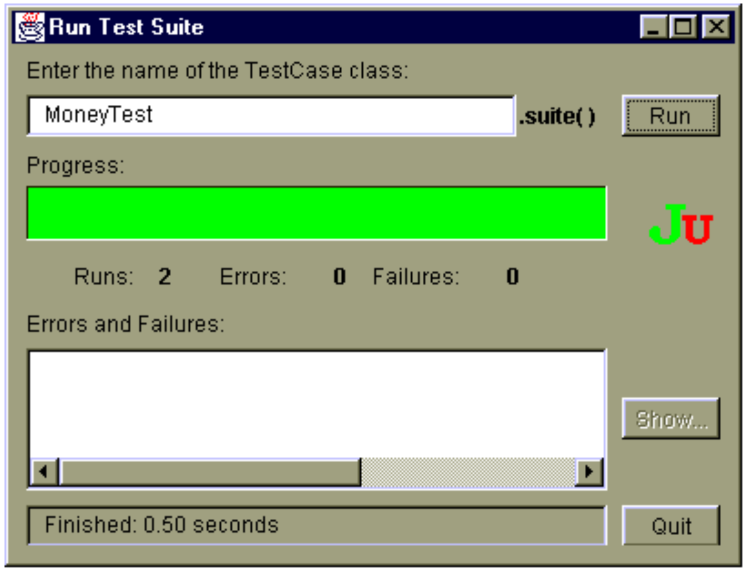
Unit Test

JUnit 3 and 4 and 5 http://www.junit.org

- Test pattern
  - Test, TestSuite, TestCase
  - Assertions (assertXX) that must be verified
- TestRunner
  - Chain tests and output a report.



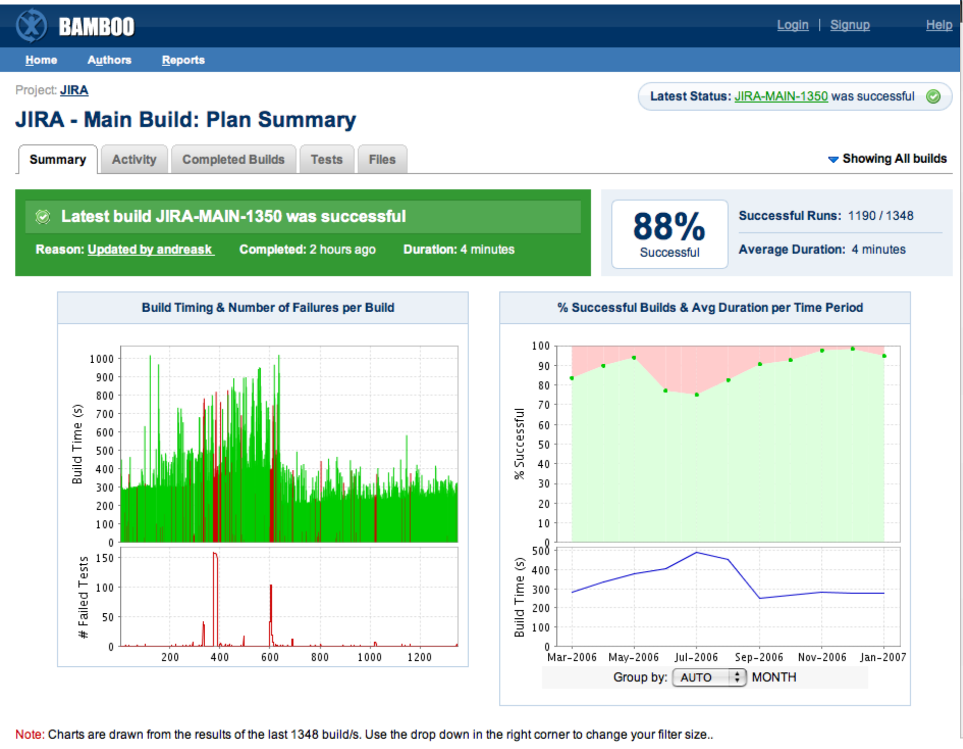
```
Run Test Suite
Enter the name of the TestCase class:
MoneyTest .suite( )  Run
Progress:
Runs: 18   Errors: 0   Failures: 1
Errors and Failures:
Failure: MoneyTest.testMixedSimpleAdd:expected:"{[12 US
Show...
Finished: 0.160 seconds  Quit
```



```
Run Test Suite
Enter the name of the TestCase class:
MoneyTest .suite( )  Run
Progress:
Runs: 2   Errors: 0   Failures: 0
Errors and Failures:
Show...
Finished: 0.50 seconds  Quit
```

Continous integration

- Principe
  - Schedule periodically test execution, nightly builds, …
- Example : Continuum, Bamboo, Hudson, JENKINS …



Project: JIRA

Latest Status: JIRA-MAIN-1350 was successful

JIRA - Main Build: Plan Summary

Latest build JIRA-MAIN-1350 was successful

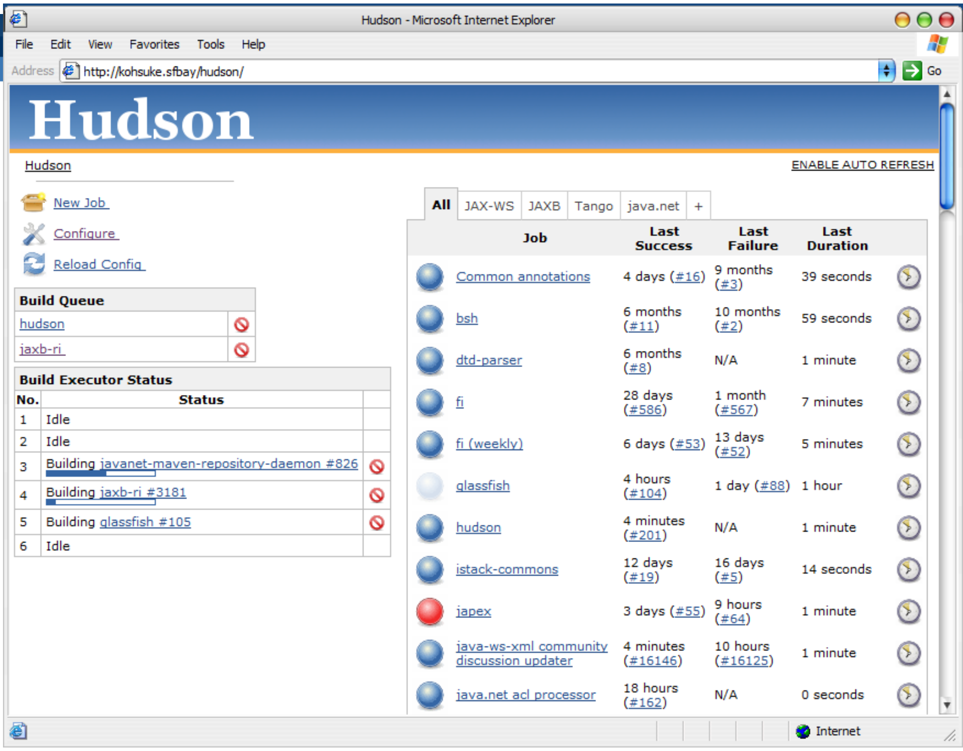
Reason: Updated by andreask Completed: 2 hours ago Duration: 4 minutes

88% Successful

Successful Runs: 1190 / 1348

Average Duration: 4 minutes

Note: Charts are drawn from the results of the last 1348 build/s. Use the drop down in the right corner to change your filter size..



Build Queue

hudson
jaxb-ri

Build Executor Status

No.	Status
1	Idle
2	Idle
3	Building javanet-maven-repository-daemon #826
4	Building jaxb-ri #3181
5	Building glassfish #105
6	Idle

Job	Last Success	Last Failure	Last Duration
Common annotations	4 days (#16)	9 months (#3)	39 seconds
bsh	6 months (#11)	10 months (#2)	59 seconds
dtd-parser	6 months (#8)	N/A	1 minute
fi	28 days (#586)	1 month (#567)	7 minutes
fi (weekly)	6 days (#53)	13 days (#52)	5 minutes
glassfish	4 hours (#104)	1 day (#88)	1 hour
hudson	4 minutes (#201)	N/A	1 minute
istack-commons	12 days (#19)	16 days (#5)	14 seconds
japex	3 days (#55)	9 hours (#64)	1 minute
java-ws-xml community discussion updater	4 minutes (#16146)	10 hours (#16125)	1 minute
java.net acl processor	18 hours (#162)	N/A	0 seconds

Improving Your Productivity

- Continuous integration can help you go faster
  - Detect build breaks sooner
  - Report failing tests more clearly
  - Make progress more visible

Jenkins for Continuous Integration

- Jenkins – open source continuous integration server
- Jenkins (http://jenkins-ci.org/) is
  - Easy to install
  - Easy to use
  - Multi-technology
  - Multi-platform
  - Widely used
  - Extensible
  - Free

Jenkins for a Developer

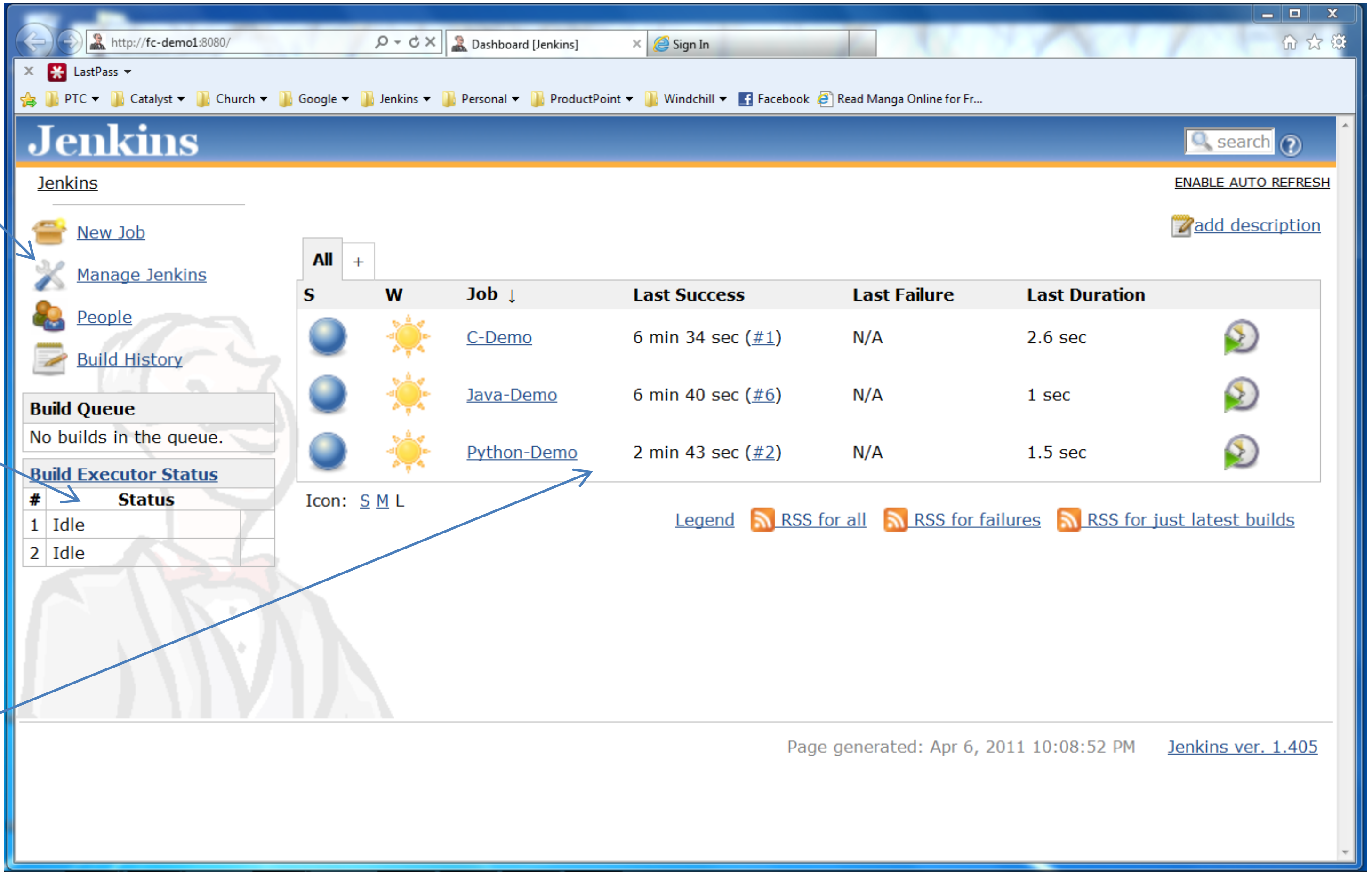
- Easy to install
  - Download one file – jenkins.war
  - Run one command – java –jar jenkins.war
- Easy to use
  - Create a new job – checkout and build a small project
  - Checkin a change – watch it build
  - Create a test – watch it build and run
  - Fix a test – checkin and watch it pass
- Multi-technology
  - Build C, Java, C#, Python, Perl, SQL, etc.
  - Test with Junit, Nunit, MSTest, etc.

Jenkins User Interface

Actions

Nodes

Jobs



Jenkins

ENABLE AUTO REFRESH

New Job

Manage Jenkins

People

Build History

add description

Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

All +

S	W	Job ↓	Last Success	Last Failure	Last Duration
●	☀	C-Demo	6 min 34 sec (#1)	N/A	2.6 sec
●	☀	Java-Demo	6 min 40 sec (#6)	N/A	1 sec
●	☀	Python-Demo	2 min 43 sec (#2)	N/A	1.5 sec

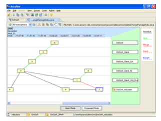
Icon: S M L

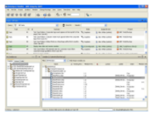
Legend RSS for all RSS for failures RSS for just latest builds

Page generated: Apr 6, 2011 10:08:52 PM Jenkins ver. 1.405

Jenkins Plugins - SCM

- Version Control Systems
  - Accurev
  - Bazaar
  - BitKeeper
  - ClearCase
  - Darcs
  - Dimensions
  - Git
  - Harvest
  - MKS Integrity
  - PVCS
  - StarTeam
  - Subversion
  - Team Foundation Server
  - Visual SourceSafe





Jenkins Plugins – Build & Test

- Build Tools
  - Ant
  - Maven
  - MSBuild
  - Cmake
  - Gradle
  - Grails
  - Scons
  - Groovy
- Test Frameworks
  - Junit
  - Nunit
  - MSTest
  - Selenium
  - Fitnesse

Jenkins Plugins – Analyzers

- Static Analysis
  - Checkstyle
  - CodeScanner
  - DRY
  - Crap4j
  - Findbugs
  - PMD
  - Fortify
  - Sonar
  - FXCop
- Code Coverage
  - Emma
  - Cobertura
  - Clover
  - GCC/GCOV

Jenkins Plugins – Other Tools

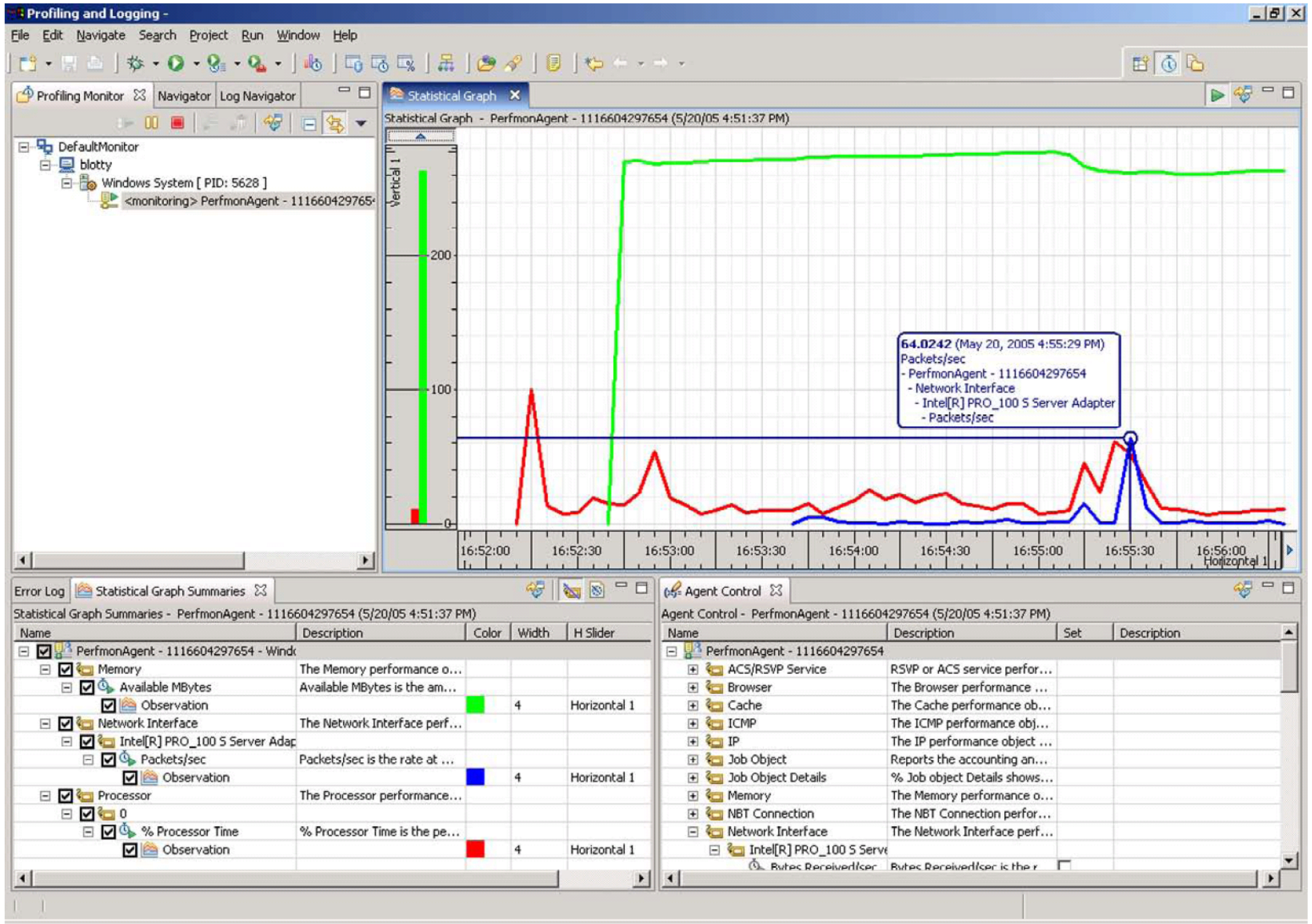
- Notification
  - Twitter
  - Campfire
  - Google Calendar
  - IM
  - IRC
  - Lava Lamp
  - Sounds
  - Speak
- Authorization
  - Active Directory
  - LDAP
- Virtual Machines
  - Amazon EC2
  - VMWare
  - VirtualBox
  - Xen
  - Libvirt

Measurements and Analysis of Performances

- Java Profiler
  - Use to profile an application
    - CPU usage, Memory, Network, time spent, and Garbage collection
    - In Total or by threads, or called methods

Exemple

- Eclipse TPTP http://www.eclipse.org/tptp/



Statistical Graph - PerfmonAgent - 1116604297654 (5/20/05 4:51:37 PM)

64.0242 (May 20, 2005 4:55:29 PM)
Packets/sec
- PerfmonAgent - 1116604297654
 - Network Interface
  - Intel[R] PRO_100 S Server Adapter
   - Packets/sec

Statistical Graph Summaries - PerfmonAgent - 1116604297654 (5/20/05 4:51:37 PM)

Name	Description	Color	Width	H Slider
PerfmonAgent - 1116604297654 - Windo				
Memory	The Memory performance o…			
Available MBytes	Available MBytes is the am…			
Observation		🟩	4	Horizontal 1
Network Interface	The Network Interface perf…			
Intel[R] PRO_100 S Server Adap				
Packets/sec	Packets/sec is the rate at …			
Observation		🟦	4	Horizontal 1
Processor	The Processor performance…			
0				
% Processor Time	% Processor Time is the pe…			
Observation		🟥	4	Horizontal 1

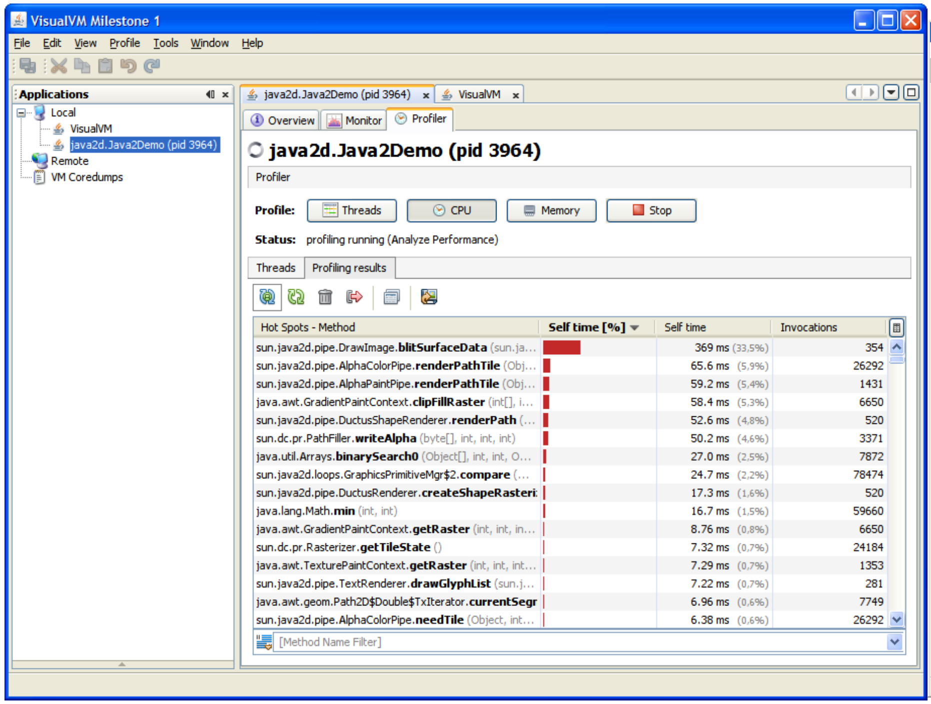
Agent Control - PerfmonAgent - 1116604297654 (5/20/05 4:51:37 PM)

Name	Description	Set	Description
PerfmonAgent - 1116604297654			
ACS/RSVP Service	RSVP or ACS service perfor…		
Browser	The Browser performance …		
Cache	The Cache performance ob…		
ICMP	The ICMP performance obj…		
IP	The IP performance object …		
Job Object	Reports the accounting an…		
Job Object Details	% Job object Details shows…		
Memory	The Memory performance o…		
NBT Connection	The NBT Connection perfor…		
Network Interface	The Network Interface perf…		
Intel[R] PRO_100 S Serve			
Bytes Received/sec	Bytes Received/sec is the r	☐	

VisualVM

https://visualvm.dev.java.net/

- « VisualVM is a visual tool that integrates several existing JDK software tools and lightweight memory and CPU profiling capabilities. This tool is designed for both production and development time use and further enhances the capability of monitoring and performance analysis for the Java SE platform.»
- **VisualVM includes the JConsole.**



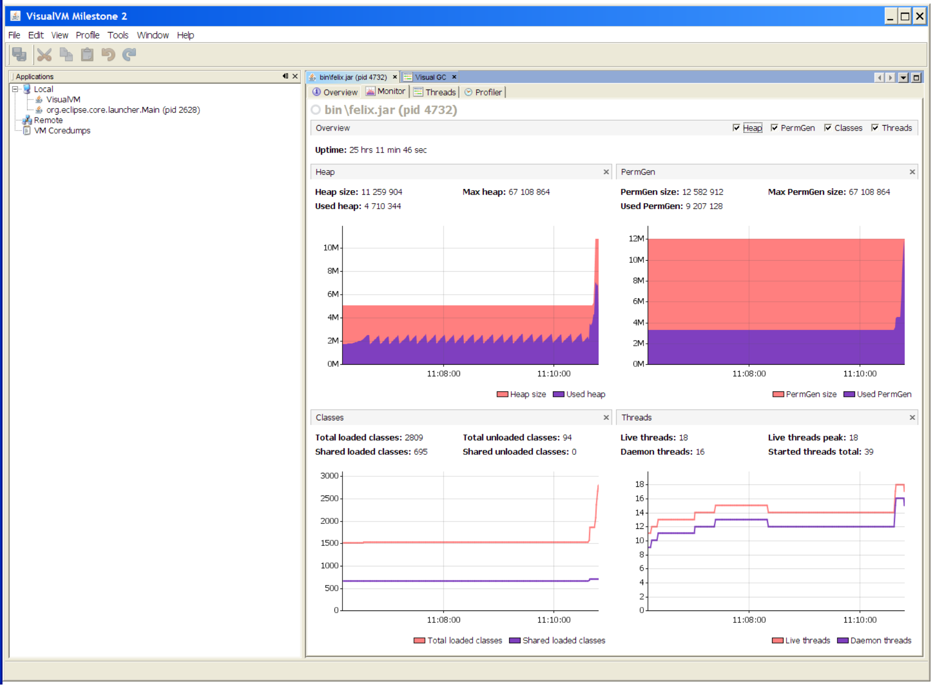
java2d.Java2Demo (pid 3964)

Profiler

Profile: Threads CPU Memory Stop

Status: profiling running (Analyze Performance)

Hot Spots - Method	Self time [%]	Self time	Invocations
sun.java2d.pipe.DrawImage.**blitSurfaceData** (sun.ja…		369 ms (33,5%)	354
sun.java2d.pipe.AlphaColorPipe.**renderPathTile** (Obj…		65.6 ms (5,9%)	26292
sun.java2d.pipe.AlphaPaintPipe.**renderPathTile** (Obj…		59.2 ms (5,4%)	1431
java.awt.GradientPaintContext.**clipFillRaster** (int[], i…		58.4 ms (5,3%)	6650
sun.java2d.pipe.DuctusShapeRenderer.**renderPath** (…		52.6 ms (4,8%)	520
sun.dc.pr.PathFiller.**writeAlpha** (byte[], int, int, int)		50.2 ms (4,6%)	3371
java.util.Arrays.**binarySearch0** (Object[], int, int, O…		27.0 ms (2,5%)	7872
sun.java2d.loops.GraphicsPrimitiveMgr$2.**compare** (…		24.7 ms (2,2%)	78474
sun.java2d.pipe.DuctusRenderer.**createShapeRasteri**		17.3 ms (1,6%)	520
java.lang.Math.**min** (int, int)		16.7 ms (1,5%)	59660
java.awt.GradientPaintContext.**getRaster** (int, int, in…		8.76 ms (0,8%)	6650
sun.dc.pr.Rasterizer.**getTileState** ()		7.32 ms (0,7%)	24184
java.awt.TexturePaintContext.**getRaster** (int, int, int…		7.29 ms (0,7%)	1353
sun.java2d.pipe.TextRenderer.**drawGlyphList** (sun.j…		7.22 ms (0,7%)	281
java.awt.geom.Path2D$Double$TxIterator.**currentSegr**		6.96 ms (0,6%)	7749
sun.java2d.pipe.AlphaColorPipe.**needTile** (Object, int…		6.38 ms (0,6%)	26292



bin \felix.jar (pid 4732)

Overview ☑ Heap ☑ PermGen ☑ Classes ☑ Threads

Uptime: 25 hrs 11 min 46 sec

Heap

Heap size: 11 259 904 **Max heap:** 67 108 864
Used heap: 4 710 344

PermGen

PermGen size: 12 582 912 **Max PermGen size:** 67 108 864
Used PermGen: 9 207 128

Classes

Total loaded classes: 2809 **Total unloaded classes:** 94
Shared loaded classes: 695 **Shared unloaded classes:** 0

Threads

Live threads: 18 **Live threads peak:** 18
Daemon threads: 16 **Started threads total:** 39

http://weblogs.java.net/blog/mandychung/archive/VisualVM-BOF-2007.pdf

Performance Optimizations

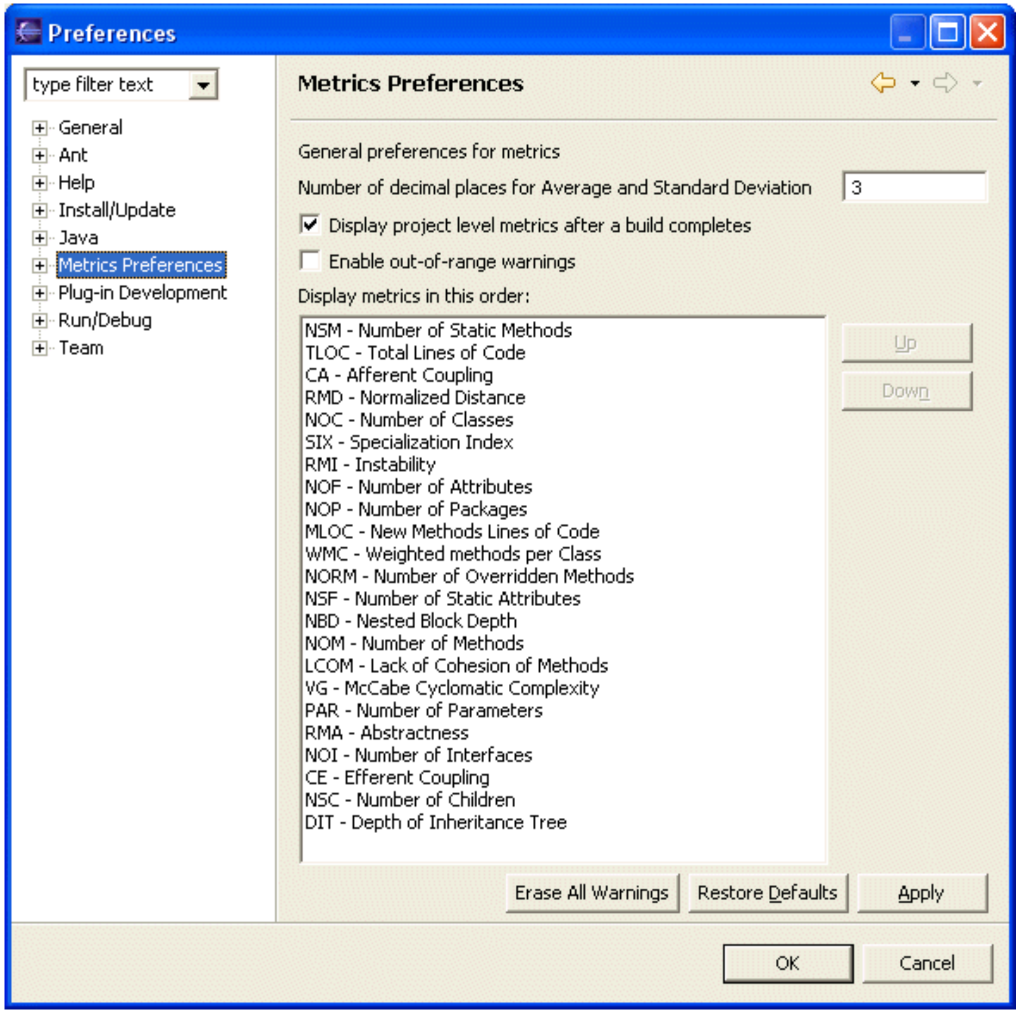
- Java's Script Engine
  - Jython (jython.sourceforge.net), …
- Bytecode interpreter
- Native compiler (static)
  - .class to .c to .s to .exe
- On-the-fly compiler (dynamic)
  - Compilation JIT (Just-In-Time) de Symantec
- HotSpot™ Optimizer
  - garbage collector
  - « method inlining »
    - with load-time verification (dynamic) of class bytecode
- Benchmark de JVM

Code Quality Metrics

- Metrics on project source code to evaluate its quality (maintenance,reverse-engineering,evolution …)
  - and how good the development team is :-)
- Metrics
  - LOC, LOCC, McCabe Cyclomatic Complexity, …
- Lectures
  - Brian Henderson-Sellers , "Object-Oriented Metrics, measures of Complexity", Ed Prentice Hall, 1996
  - Robert Martin, "OO Design Quality Metrics, An Analysis of Dependencies", 1994, http://www.objectmentor.com/resources/articles/oodmetrc.pdf
  - Chidamber and Kemerer, A Metrics Suite for Object Oriented Design, http://www.pitt.edu/~ckemerer/CK%20research%20papers/MetricForOOD_ChidamberKemerer94.pdf
  - Mariano Ceccato and Paolo Tonella, Measuring the Effects of Software Aspectization, http://homepages.cwi.nl/~tourwe/ware/ceccato.pdf
  - Robert Martin, "Agile Software Development, Principles, Patterns and Practices", Prentice Hall, 1st edition, 2002, ISBN: 978-0135974445

Code Quality Metrics

- Example: Metrics (Tache ANT + Eclipse plugin)



Preferences

type filter text

- General
- Ant
- Help
- Install/Update
- Java
- Metrics Preferences
- Plug-in Development
- Run/Debug
- Team

Metrics Preferences

General preferences for metrics

Number of decimal places for Average and Standard Deviation: 3

[✓] Display project level metrics after a build completes

[ ] Enable out-of-range warnings

Display metrics in this order:

NSM - Number of Static Methods
TLOC - Total Lines of Code
CA - Afferent Coupling
RMD - Normalized Distance
NOC - Number of Classes
SIX - Specialization Index
RMI - Instability
NOF - Number of Attributes
NOP - Number of Packages
MLOC - New Methods Lines of Code
WMC - Weighted methods per Class
NORM - Number of Overridden Methods
NSF - Number of Static Attributes
NBD - Nested Block Depth
NOM - Number of Methods
LCOM - Lack of Cohesion of Methods
VG - McCabe Cyclomatic Complexity
PAR - Number of Parameters
RMA - Abstractness
NOI - Number of Interfaces
CE - Efferent Coupling
NSC - Number of Children
DIT - Depth of Inheritance Tree

Code Quality Metrics

- Others (standalone or as IDE plugins)
  - http://metrics.sourceforge.net/
  - http://qjpro.sourceforge.net/
  - http://www.geocities.com/sivaram_subr/index.htm
  - …

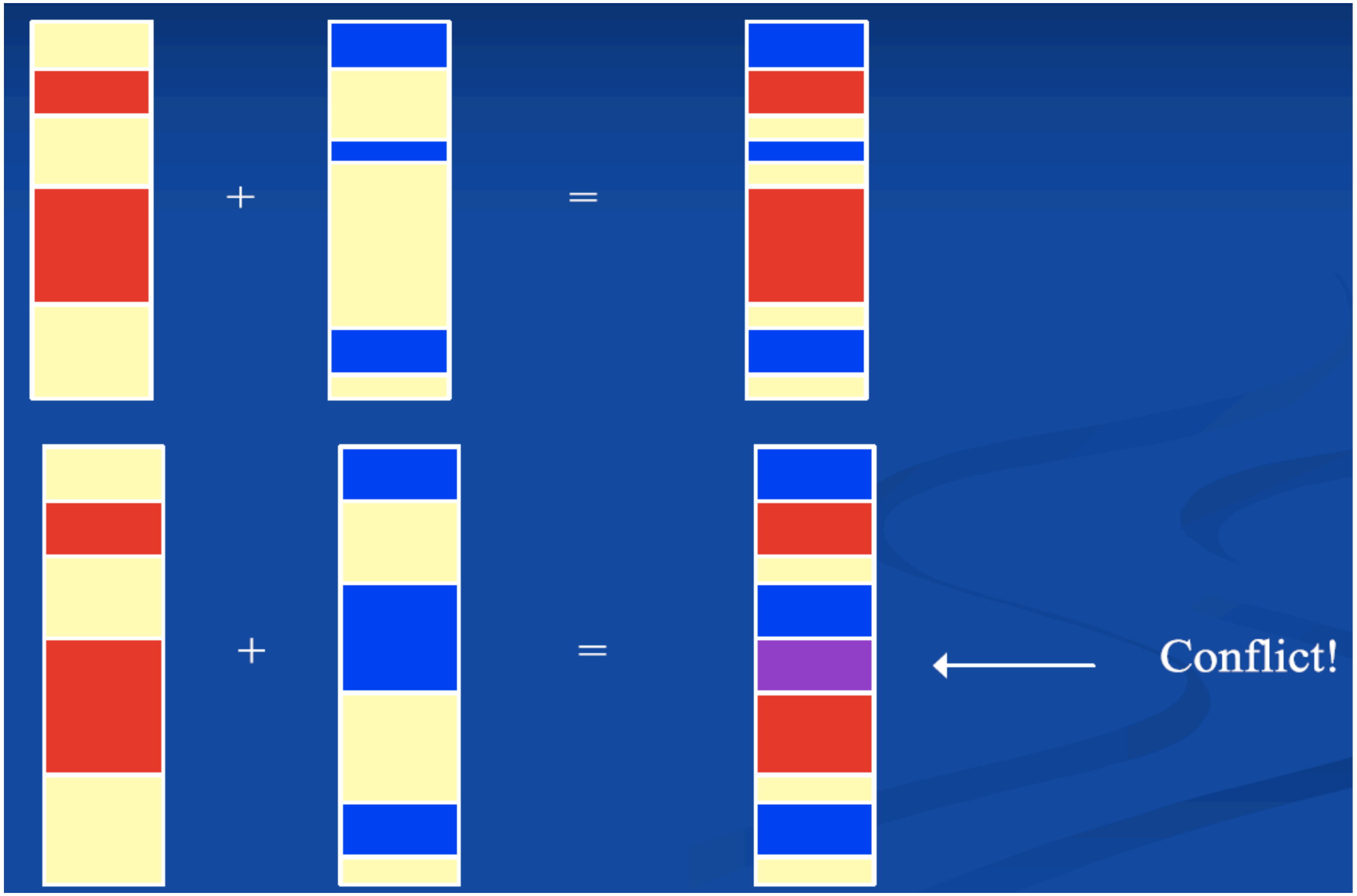
Versioning of source code

- Collaborative software engineering
- To master
  - software development by very large developer teams
  - parallel implementations (experiments, vendors)
- Goals
  - Increase productivity of developers and software robustness
  - Low-down development costs
- Manage software system configuration
  - to control software system’s evolution
  - evolution tracking (time-machine)
  - issue and bug tracking

Versioning : What for?

- History of versions
  - back to an older version in case of errors
- Alternative versions (branching)
  - different design/implementations (maybe experimentals) for the same module
- Collaborative access by many developers
  - audit modification history
    - how many commits by X ?
    - when most of the commits are done?
    - ...

Concurrency management



Concurrency control

- Doing nothing!
- Lock-Modify-Unlock (Pessimistic)
  - SCCS, RCS
  - Decrease productivity
- Copy-Modify-Merge (Optimistic)
  - Conflicts resolution when concurrent modifications (which are actually rare)
    - Merge, Selection, …
  - CVS, SVN : Client level resolution
- Policy-based
  - Merging and validation process for each code contribution

Concept of Version

- Trunk
  - main development
- Branches
  - Alternatives to trunk
    - Different design/implementation (experimental), vendor-specific
- Revisions
  - Sequence of versions
- Tags
  - Symbolic references to revisions (Tiger, LongHorn, …)
    - Represent a public release (R), a milestone (M)
- Branch merging

Tools

- Pioneers
  - SCCS, RCS, PVCS
- Current alternatives
  - CVS
  - SubVersion
  - Git
  - MS Visual SourceSafe
  - ChangeMan (Serena)
  - AllFusion Harvest (CA)
  - ClearCase (IBM Rational)
  - Perforce
  - CM Synergy (Telelogic)
  - Source Integrity (MKS)
  - PVCS (Merant)
  - TeamCode (Interwoven)
  - Surround CM (Seapine)
- Web-Oriented protocols
  - WebDAV/DeltaV

Git

- version control system
  - designed to handle very large projects with speed and efficiency
  - mainly for various open source projects, most notably the Linux kernel.
- http://git.or.cz/

Tools to use Git

- SmartSVN
- GitHub => provide the whole forge with a GIT installed
  - Free for open-source project

Centralized Version Control

- Traditional version control system
  - Server with database
  - Clients have a working version
- Examples
  - CVS
  - Subversion
  - Visual Source Safe
- Challenges
  - Multi-developer conflicts
  - Client/server communication

Distributed Version Control

- Authoritative server by convention only
- Every working checkout is a repository
- Get version control even when detached
- Backups are trivial

- Other distributed systems include
  - Mercurial
  - BitKeeper
  - Darcs
  - Bazaar

Git Advantages

- Resilience
  - No one repository has more data than any other
- Speed
  - Very fast operations compared to other VCS (I’m looking at you CVS and Subversion)
- Space
  - Compression can be done across repository not just per file
  - Minimizes local size as well as push/pull data transfers
- Simplicity
  - Object model is very simple
- Large userbase with robust tools

Git Architecture

- Index
  - Stores information about current working directory and changes made to it
- Object Database
  - Blobs (files)
    - Stored in .git/objects
    - Indexed by unique hash
    - All files are stored as blobs
  - Trees (directories)
  - Commits
    - One object for every commit
    - Contains hash of parent, name of author, time of commit, and hash of the current tree
  - Tags

Some Commands

- Getting a Repository
  - git init
  - git clone
- Commits
  - git add
  - git commit
- Get changes with
  - git fetch (fetches and merges)
  - git pull
- Propagate changes with
  - git push