Les Outils de Développement pour Java

Didier Donsez

Université Joseph Fourier - Grenoble 1
PolyTech' Grenoble - LIG / ADELE
Didier.Donsez@imag.fr
Didier.Donsez@ieee.org

Licence

- Cette présentation est couverte
  par le contrat Creative Commons By NC ND
  - http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

Objectifs

- Documentation
- Organiser
- Deboggage
- Observer
- Précompilateur
- Retro-Compilation
- Ofuscateur
- Analyseur de Performance
- Test de Compatibilité

Documentation

- Javadoc (JDK)
  - Génération automatique de la documentation HTML
  - à partir des commentaires présents dans les .java
- Commentaires et Tags

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

Tags non standards

- Documentation Standard (HTML avec/sans frame)
  - hiérarchie des classes et des interfaces, liste des packages
  - résumé et détail d 'une classe, interface, méthode, propriété,…
- Documentation Customisée (RTF, XML, MIF, HTML, …)
  - Doclet : classe Java chargée par Javadoc
    pour personnaliser le résultat de la génération de la documentation
  - Taglet : classe Java personnalisant la sortie HTML lié à un tag (bloc ou inline)

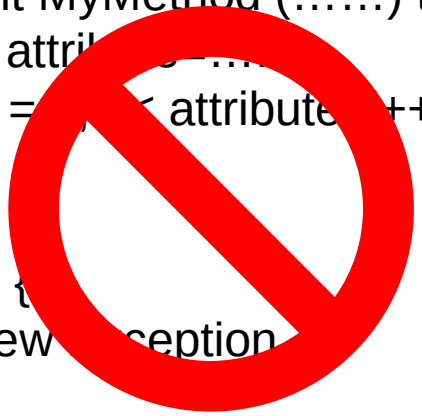
Remarque

- N 'oubliez pas d'ajouter la génération au Makefile/Build/POM file

Normes de programmation (i)

- Facilite la lecture des sources (lecture croisée)
- Normes
  - SUN, autres extensions (template)

```
public int MyMethod (……) throws … {
    int     attri
for (int i =     < attribute
{
…}
if (……) {
throw new     ception
}
}
```



```
/**
* Description of the method
* @param What is it ?
* @return What the function return
*/
public int myMethod (……) throws … {
    int     attribute=…. ;// description
    for (int i = 0; i < attribute; i++) {
            …….
      }
      if (……) {
          throw new Exception
      }
```

Normes de programmation (ii)

- Outils de vérification
  - CheckStyle, PMD, JackPot, Spoon VSuite…
- Outils de reformatage
  - Indenteurs (Jindent), *beautiful*eurs, styleur (JavaStyle), …
- Outils de « bug fixing »
  - Spoon VSuite, Findbugs (sourceforge) …
- Outils de rapport de qualité
  - métriques sur le code pour la qualité
    - NCCS (number of Non Comment Code Source), Nombre de paquetage, …
    - JavaNCCS, Eclispe Metrics, Bamboo …

- Exemple de rapport CheckStyle

```
File.java:338: Unused @throws tag for 'Exception'.
File.java:420: 'if' construct must use '{}'s.
File.java:427:37: variable ‘MyVar' must
  match pattern '^[a-z][a-zA-Z0-9]*$'.
```

Organiser (i)

- N ’oubliez pas Make & Co
  - make, gmake, nmake (Win),
  - Apache ANT, Apache MAVEN, Freshmeat 7Bee ...
- Pour
  - précompilation, obfuscation, prévérification (J2ME)
  - génération des .class et des .jar
    - en mode normal, trace, debug, …
  - génération de la documentation
  - génération des « stubs » (rmic, idl2java, javacard …)
  - contrôle de la compatibilité
  - …
- en fonction des dépendances temporelles
  - des fichiers (make & co) ou des taches (ANT)

Organiser (ii) : ANT

- Outil de construction (build tool)
  - Indépendant de la plateforme et open-source (Apache Jakarta)
  - Extensible à d'autres taches ou des commandes de la plateforme
- Exemple : build.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project basedir="." default="compile" name="test">
   <property name="build" value="build"/>
   <property name="src" value="src"/>
   <!-- [ path and patternset ] -->
   <target name="init">
         <mkdir dir="${build}"/>
   </target>
   <target name="compile" depends="init">
         <javac srcdir="${src}" destdir="${build}"/>
   </target>
</project>
```

Voir cours ANT

- http://www-adele.imag.fr/users/Didier.Donsez/cours/ant.pdf

Organiser (iii)

- Maven
  - Application du principe de la séparation des préoccupations à la construction d'un projet
    - Compilation, code generation, unit testing, documentation, …
  - Gestion des dépendances versionnées vers d'autres projets (artifacts)
- Project object model (POM)
  - description abstraite/informationnel du projet
    - Pas de listes de tâche à enchainer (sauf personnalisation du projet type)
  - Hérite des propriétés de POM parents
- Les outils (appelés plugin) supposent
  - une structure de projet type
  - un cycle de vie de projet type

Remarque: de plus en plus utilisé !!!

Voir cours Maven

- `http://www-adele.imag.fr/users/Didier.Donsez/cours/maven.pdf`

Organiser (iv)

- Versions de code
  - Branches de code
- Développement collaboratif
  - Import, Update, Commit, Merge
- Produits
  - SourceSafe (MS), Rational ClearCase, Perforce, VSS, …
  - RCS, CVS, SubVersion
  - JIRA
  - *Celine (ADELE)*

Voir cours

- **http://www-adele.imag.fr/users/Didier.Donsez/cours/version.pdf**
- **http://www-adele.imag.fr/users/Didier.Donsez/cours/subversion.pdf**

Deboggeur – Dévermineur - *Debugging*

- Motivation
  - Comportement différent de celui qui est spécifié
  - Plantage (*core dump*)
  - Faille
  - → Bogue, Vermine, Cafard, *Bug*
- Déboggage symbolique
  - option de javac : -g, -g:source,vars,lines
  - déboggage en ligne de commande : jdb (JDK)
    - commandes semblables à celles de dbx
  - « front-ends » graphiques à jdb (liés aux AGL)
  - Misc
    - Multi-threads, Cross-Deboggage (-Xdebug) sur VM distante, ...
  - Interface JPDA et maintenant JVMTI

Surveiller/Observer

- Surveiller/Observer l'activité de votre programme
  - Serveur, parc de serveurs, …
- Tracer
- Journaliser

Tracer

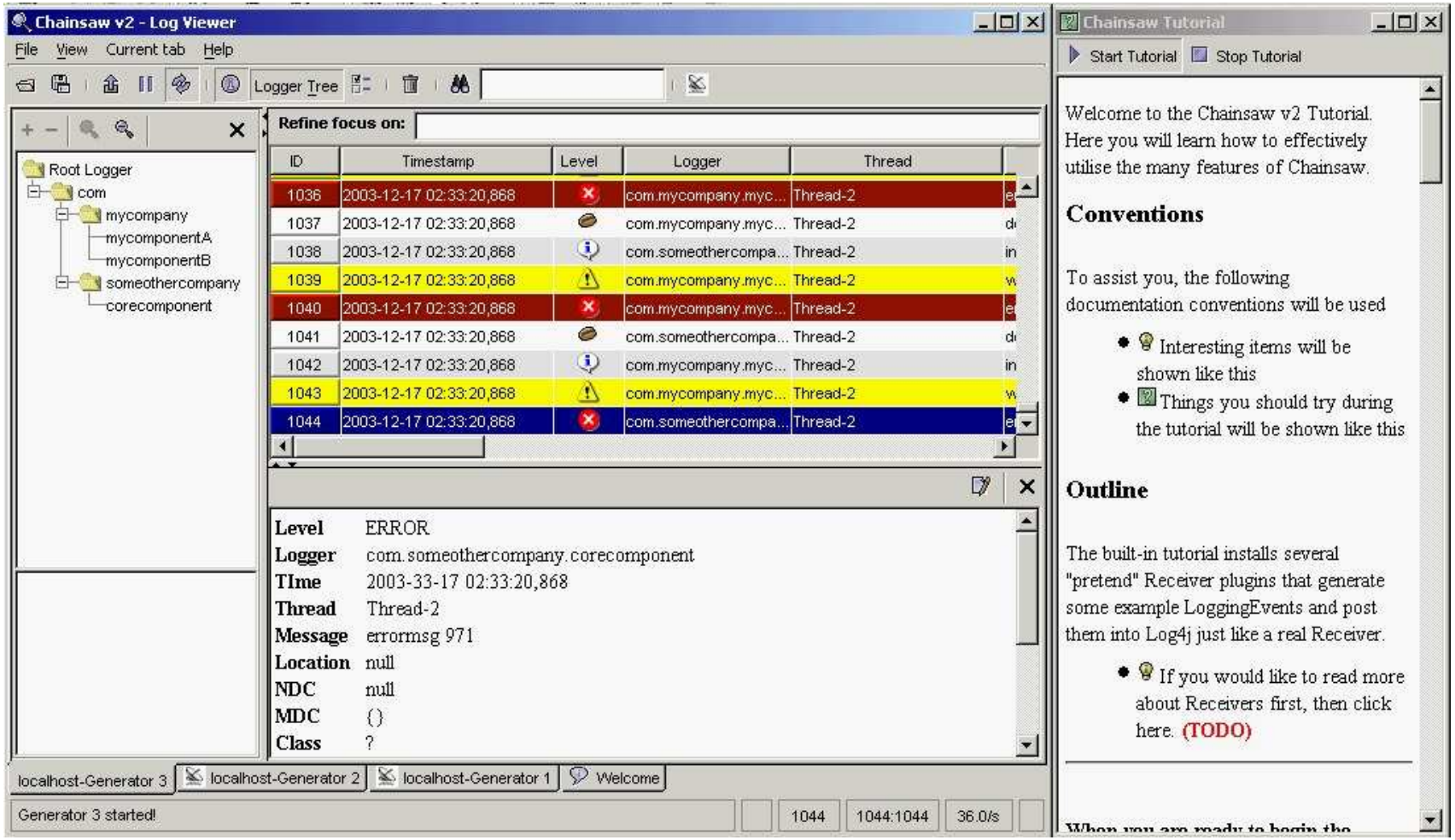
- option de TRACE du programme
- peut ralentir le .class avec les tests TRACE/¬TRACE
  - solution : utiliser un précompilateur (excluant les appels aux traces)
- Outils noyau
  - OpenSolaris/BSD DTrace (couplage avec la JVM)
  - Linux SystemTap (pas de couplage)

Journaliser (i)

- Enregistrer les événements dans un journal exploitable en cours d'exécution ou à posteriori (par un handler)
- Outils
  - Log Backend
    - JUL package java.util.logging du J2SE1.4
      - Logger, LogRecord, Handler
    - Apache Log4J http://logging.apache.org/
    - LogBack http://logback.qos.ch/
    - OW2 MonoLog
    - Simple Logging Facade for Java (SLF4J) http://www.slf4j.org/
    - OSGi LogService
  - Visualisateurs
    - Chainsaw, Lilith
  - Analyseurs
    - Logback-audit
  - Serveurs (bases de données)
    - TBD

Journaliser (ii)

- Apache Chainsaw
  - Try http://logging.apache.org/log4j/docs/webstart/chainsaw/chainsawWebStart.jnlp



Valider (i)

- Assertion
  - Pré-Condition, Post-Condition, Invariant
    - EIFFEL, CLU ... les supportent
    - Et Java depuis la version SE 1.4
- Outils pour J2SE 1.3 et moins (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)

Valider (ii)
Les assertions 1.4

- Un nouveau mot-clé :assert
  - assert expression1; expression2; …
  - Activation (-ea) et Désactivation (-da) à l'exécution
    - java -ea:com.wombat.fruitbat... -da:com.wombat.fruitbat.Brickbat Main.class
- Exemple : Pré-Condition

```
public void setRefreshRate(int rate) {
// Enforce specified precondition in public method
  if (rate <= 0 || rate > MAX_REFRESH_RATE)
     throw new IllegalArgumentException("Illegal rate: " + rate);
  setRefreshInterval(1000/rate);
}
private void setRefreshInterval(int interval) {
     // Confirm adherence to precondition in nonpublic method
     assert interval > 0 && interval <= 1000/MAX_REFRESH_RATE;
     ... // Set the refresh interval
}
```

Valider (iii)
Les assertions 1.4

- Exemple : Post-Condition

```
public BigInteger modInverse(BigInteger m) {
        if (m.signum <= 0)
            throw new ArithmeticException("Modulus not positive: " + m);
        if (!this.gcd(m).equals(ONE))
            throw new ArithmeticException(this + " not invertible mod " + m);

        ... // Do the computation

        assert this.multiply(result).mod(m).equals(ONE);
        return result;
    }
```

- Exemple : Invariant de Classe

```
// Returns true if this tree is properly balanced
    private boolean balanced() {
        ...
    }
// This method is a class invariant.
// It should always be true before and after any method completes.
// To check that this is indeed the case, each public method and
// constructor should contain the line:
    assert balanced();
```

Précompilateur

- Insertion de directive de précompilation
  - comme dans cpp (phase 1 de CC) : #include, #define, #ifdef, ...
- Intérêt
  - 1 source .java -> plusieurs .java alternatifs
    - exemple d 'application: .java avec trace et .java sans trace
      - évite d 'avoir un .class ralenti par les tests de trace
    - Ligne de produits, Logicile d'évaluation
- Outils : Mocha Source Obfuscator, ...
- Exemple

```
private void myfunction(int x, int y){
//#ifdef TRACE
Globallog.log(DEBUG,"hello world, I am in myfunction " + x + " " +y);
//#endif
```

  - donne (si TRACE n'est pas défini)

```
private void myfunction(int x, int y){
//#ifdef TRACE
//Globallog.log(DEBUG,"hello world, I am in myfunction " + x + " " +y);
//#endif
```

Remarque sur les traces

- L'optimiseur de javac supprime les branches jamais atteintes

```
static final boolean TRACE=false;
…

private void myfunction(int x, int y){
  if(TRACE) { // n'est pas inclus dans le bytecode
    Globallog.log(DEBUG,"hello world, I am in myfunction " + x + " " +y);
  }
  …
}
```

Restructuration (*Refactoring*)

- Motivation
  - Réorganisation d'un ensemble de sources
  - Arborescence de paquetage
  - Nouvelles méthodes
  - Chargement de paramêtres
- Dans la plupart des IDE
- Batchs jarjar

Rétro-Compilation

- Décompilation du bytecode
  d 'un .class Java en un source .java
  - Partielle/Totale
- Objectif de l ' « attaquant »
  - le nom d 'un bean, d 'une classe, des méthodes, les commentaires des traces, les informations de deboggage… ont une signification
  - retrouver les algorithmes d 'un composant métier, modifier le source (par exemple pour supprimer le code de vérification de la licence) , le détourner, le pirater ...
- Risque pour le développeur
  - Perte des droits d 'auteur, Manque à gagner, ...

La première parade: l ’ Ofuscateur

- But : Eviter l’interprétation d ’un bytecode par la rétro-compilation
  - Solution : rendre inintelligible le source avant distribution
- Méthodes appliquées par ces outils
  - brouillage du nom des classes, des méthodes, des propriétés, et les variables par renommage (a0001, …)
  - mélanger les propriétés d ’accès (public, private, …)
  - supprimer l ’information de deboggage
- Remarque : supprimer vos traces et l ’option de deboggage
- Attention : ne facilite pas la maintenance si vous manglez les exceptions « Erreur a238 : envoyez ce message à notre service de maintenance debug@mycomp.com »
- Une autre utilisation : rendre le code plus compact pour des cibles comme J2ME/CDLC

La deuxième parade: le Watermarking

- But : Prouver la retro-compilation d'un code et sa réutilisation
- Comment
  - http://www.cs.arizona.edu/sandmark/
  - Ajouter une structure W dans un programme P
  - W doit être décelable ! (pour la preuve) et robuste à la translation, optimization, obfuscation;

Décompilateurs et Ofuscateurs

- Decompilateurs
  - JDK javap, WingDis, NMI 's Java Code Viewer, JAD, DeCafe, …
- Ofuscateurs
  - Mocha Source Obfuscator, WingDis, NMI 's Java Code Viewer, JAD, Hashjava, Jmangle, Zelix KlassMaste, RetroGuard, Dash-O, …

Pour en savoir plus

- 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

Inspecter

- Inspecter, transformer, générer du bytecode

- pour
  - Ofuscateur
  - Vérifieur
  - Convertisseur
  - Intercepter
  - Monitoring
  - …

Inspecter Aspect Oriented Programming (AOP)

- Principe
  - Poser des ‘cutpoints’ dans le source/bytecode
  - pour insérer des traitements avant (before), au lieu de (arround) et/ou après (after) l’invocation de la méthode
- Outils
  - AspectJ (Statique)
  - JAC (A la volée)
  - AOP Alliance: http://sourceforge.net/projects/aopalliance

Modifier et générer Byte Code Engineering Tools

- Modification et génération de bytecode
  - A la contruction
  - A l'exécution
    - Modification au chargement des classes
- Outils
  - BCEL http://jakarta.apache.org/bcel
  - SERP http://serp.sourceforge.net
  - JOIE
  - ASM http://www.objectweb.org/asm (visitor pattern)
  - gnu.bytecode http://www.cygnus.com/~bothner/gnu.bytecode/
  - **Package java.lang.instrument du J2SE1.5**
- Attention ! Bytecode Modification Problem
  - Lot of serialization/deserialization detail
  - Remove/Add in constant pool
  - Jump offset
  - Stack Size

Modifier et générer
Mixin

- Principe
  - Similaire à l'héritage multiple (perdu du C++)
- Outils
  - MixJuice
  - Fractal/Julia possède un mixer
  - Scala

Modifier et générer
Exemple de Mixin

```
class Printer {
  void printPage() {
      <fontionnalité de base>
  }
}
```

```
abstract class Printer_Facturation {
  int count = 0;
  abstract __super__printPage();

  void printPage() {
      facturation();
      __super__printPage();
  }
  void facturation() { count++; }
}
```

```
abstract class Printer_Log {
  abstract __super__printPage();
  void printPage() {
      log();
      __super__printPage();
  }
  void log() {
      System.out.println("printPage"+System.currentTimeMillis();
  } }
```

Mixin transformation

```
class Printer {
    int count = 0;
    void printPage() {
        facturation();
        log();
        <fontionnalité de base>
    }
    void facturation() {
        count++;
    }
    void log() {
        System.out.println("printPage"+
        System.currentTimeMillis();
    }
}
```

Emballage

- Motivations:
  - 1 ou plusieurs unités de déploiement groupant class files et ressources (*.properties, *.html, *.png, …)
- Formats d'emballage
  - Jar File
    - Zip file + META-INF/MANIFEST.MF + …
  - OSGi bundle
    - Jar file + dependances de packages explicitées dans le MANIFEST.MF
  - Pack200 (JSR-200): A Packed Class Deployment Format For Java Applications
    - Orienté transmission sur les réseaux
    - Techniques: Représentation compacte des constantes primitives, constant pool commun, …
  - JPackage : basé sur les RPMs (Linux)
  - EAR,WAR, RAR pour les applications JavaEE
  - JEFF : orienté vers la ROMification (voir cours J2ME)
  - CAP : orienté JavaCard
  - JSR 277 : le futur format universel (prévu pour Java Platform 7) ?
  - …

Installateur (1/4)

- Installation d 'une application
  - copie des classes ou d 'archives JAR
  - vérifie la présence d 'une JVM (JRE)
    - de sa version
    - de ses extensions (Swing, SWT, JCE, XML, JMF, …)
  - élabore la procédure de désinstallation en fonction de la personalisation
- Outils
  - Win32
    - Installer for Java, InstallShield, NSIS (NullSoft), Apache Ant, install4J, BitRock installBuilder,
    - IZPack http://izpack.org …
    - JIFI http://www.openinstallation.org/
      - *JSmooth, launch4j for JRE checking*
  - Unix
    - Apache Ant, Make, RPM, JPackage, IZPack …
  - Java Plugin / Java Web Start : pour les applets et applications standalone

Installateur 2/4

- Java Plugin / Java Web Start
  - pour les applets et applications standalone
  - Fonctionnalités
    - JNLP (Java Network Launching Protocol)
      - Descripteur XML
      - API javax.jnlp
    - Plusieurs JRE
    - Cache d'applets/appli,
    - Versions
    - mises à jour incrementales
    - Préchargement / Chargement à la demande des ressources
    - WAR et JnlpDownloadServlet
    - Voir site SUN et http://www.vamphq.com/

Installateur 3/4

- OSGi (http://www.osgi.org)
  - Évite l'enfer du CLASSPATH
    - *et du %JAVA_HOME%\jre\lib\ext*
  - Décrire l'application comme un ensemble de bundles
    - Graphe de dépendance de Packages
    - Graphe de dépendance de Services
      - Les services sont qualifiés (courtage sur propriétés)
      - Les dépendances peuvent ëtre dynamiques !
  - OSGi résout les dépendances de packages
    - avec version de spécification
  - ServiceBinder résout les dépendances de services

Installateur 4/4

- JPackage (http://www.jpackage.org)
  - Conditionnement d'applications Java (Jar File) en RPM Linux
    - Version JRE, JVM, JNI, working dir (/var/cache/tomcat4), export policies (jce), …
- http://www.jpackage.org et mirroirs
  - Index des JPackage disponibles par téléchargement gratuitement ou non

Deamon Unix et Service Win32

- Motivation
  - Développer un service Win32 ou un démon Unix en Java
    - TomCat, James, JOnAS …
- Exemple
  - JavaService
    - http://javaservice.objectweb.org

Licensing

- Quelques chiffres
  - Utilisation illégale de logiciel
    - 50% en Europe
    - 95% en Asie, Amerique latine, Europe de l'Est
  - Perte de revenu pour les développeurs
    - $12 milliards par an sur le monde entier
    - $3 milliards par an aux USA seul.

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 (licence 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

- Mesure/Analyse
  - Benchmark
  - java.awt.Robot (pour construire des clients de test)
  - Accounting : http://abone.unige.ch/jraf/index.htm
  - JProfiler, OptimiszeIt, Stackprobe (http://www.stackprobe.com) …
- Optimisation
- Voir le livre
  - 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

Choix de la JVM et du JRE

- Quelques aspects
  - Licence, redistribution, supports, performances, contraintes (embarqué, serveurs, temps réel, …), runtimes, …
- Exemples
  - 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
  - …

Jikes RVM

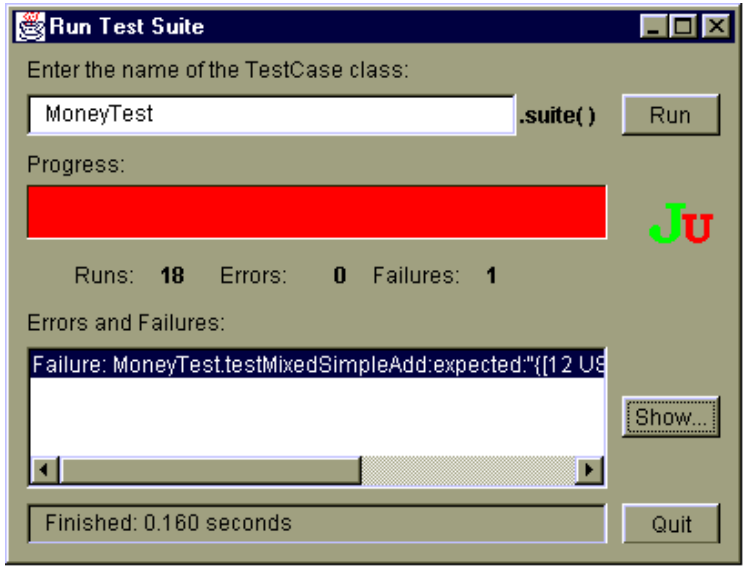
- JVM écrite en Java !
  - http://www-124.ibm.com/developerworks/oss/jikesrvm/
- Orienté recherche
- Permet d'expérimenter des techniques nouvelles
  - Garbage collection, Scheduling TR, Tissage d'aspect, échappement d'exécution, …

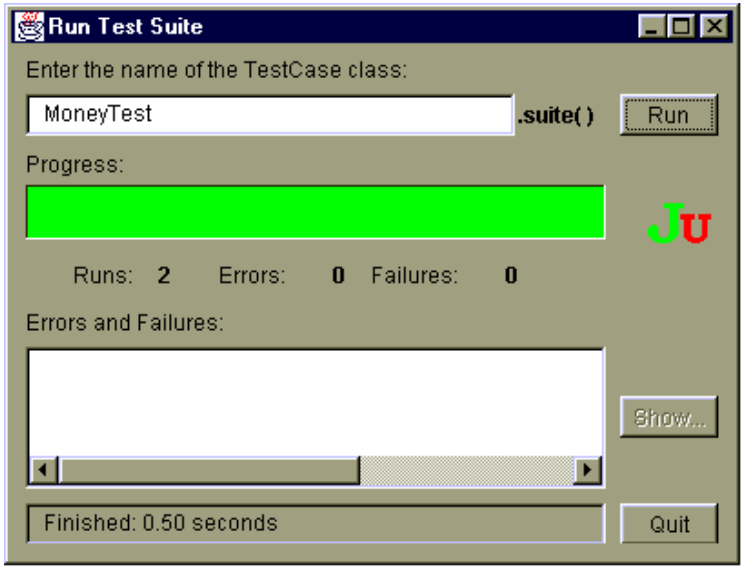
Test

- Outils de gestion de tests
  - « Tester » tous les cas possibles
- Test unitaire
  - JUNIT (http://www.junit.org) *le plus fameux*
  - Cactus (pour Servlets)
  - Jcover (CodeWork), Clover
  - SWTBot (pour applications SWT et RCP), XTest (NetBeans) :
- Test de couverture résiduelle
  - Détermine les fonctions ou les branches non testées.
  - Quilt (http://quilt.sourceforge.net)
- Intégration en continue (*Continuous integration*)

Test Unitaire
JUnit 3 et 4 *http://www.junit.org*

- Patron de conception de test
  - Test, TestSuite, TestCase
  - Assertions (assertXX) dévrant être vérifiées
- TestRunner
  - Enchaîne les tests et produit un rapport.

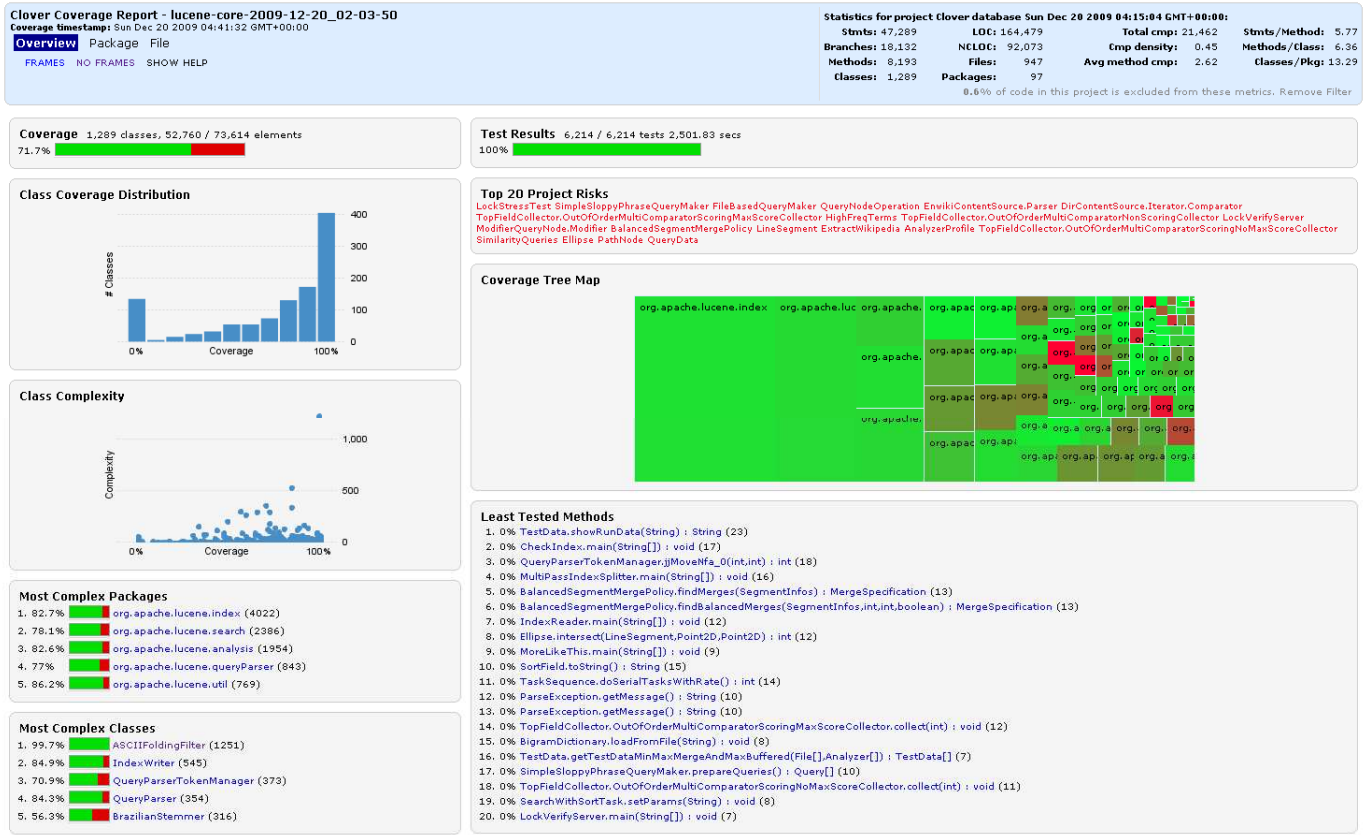




- Voir cours JUnit
  - `http://www-adele.imag.fr/users/Didier.Donsez/cours/junit.pdf`

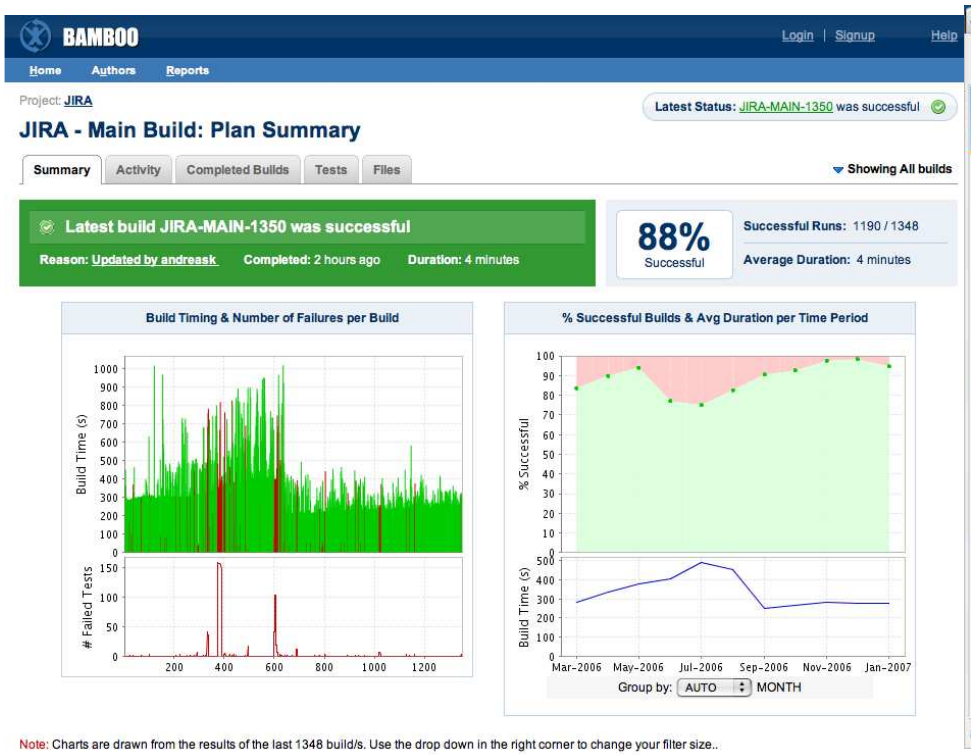
Couverture de code (Code coverage)

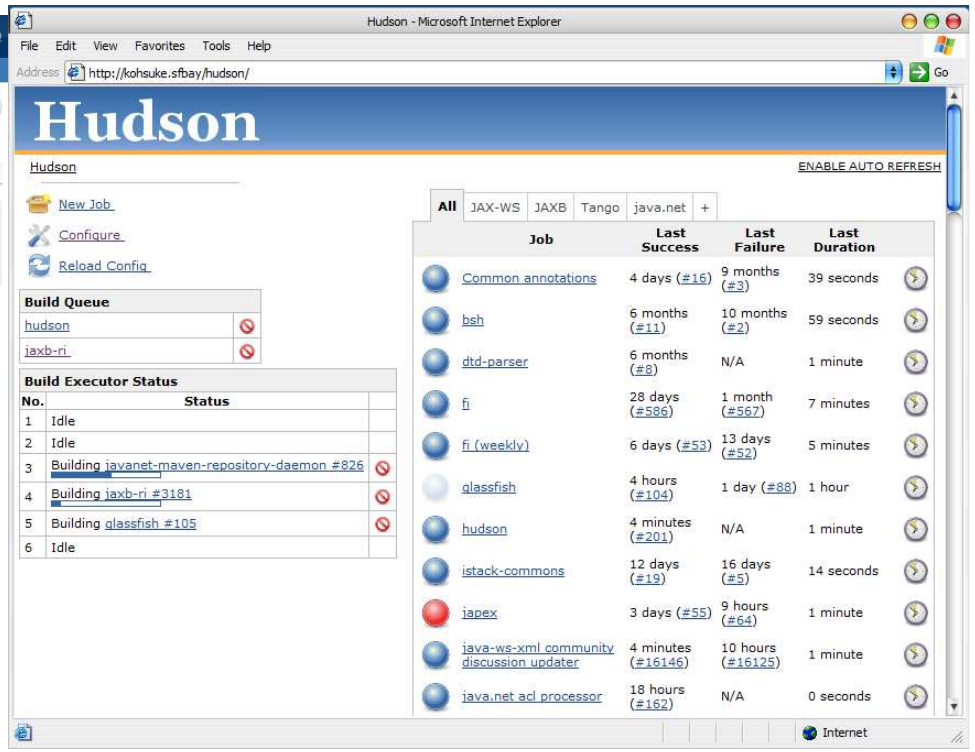
- Mesure décrivant le taux de code source testé d'un programme.
  - permet de mesurer la qualité des tests effectués.
  - Cobertura, EMMA, Clover, …



Intégration en continue

- Principe
  - Lancement à intervalle régulier des tests, des nightly builds, …
- Exemple : Continuum, Bamboo, Hudson …





Mesure et Analyse des Performances

- Option -Xrunhprof

```
C:\users>java -Xrunhprof:help
Hprof usage: -Xrunhprof[:help]|[<option>=<value>, ...]
Option Name and Value   Description             Default
---------------------   -----------             -------
heap=dump|sites|all     heap profiling          all
cpu=samples|times|old   CPU usage               off
monitor=y|n             monitor contention      n
format=a|b              ascii or binary output  a
file=<file>             write data to file      java.hprof(.txt fo
net=<host>:<port>       send data over a socket write to file
depth=<size>            stack trace depth       4
cutoff=<value>          output cutoff point     0.0001
lineno=y|n              line number in traces?  y
thread=y|n              thread in traces?       n
doe=y|n                 dump on exit?           y
Example: java -Xrunhprof:cpu=samples,file=log.txt,depth=3 FooClass
```

- Outils commerciaux
  - Consommation mémoire, détection des bottlenecks, …

Mesure des Performances avec -Xrunhprof

java -Xrunhprof:cpu=samples,depth=6 com.develop.demos.TestHprof

CPU SAMPLES BEGIN (total = 7131) Wed Jan 12 13:12:40 2000

rank	self	accum	count	trace	method
1	20.57%	20.57%	1467	47	demos/TestHprof.makeStringInline
2	20.40%	40.98%	1455	39	demos/TestHprof.addToCat
3	20.28%	61.25%	1446	53	demos/TestHprof.makeStringWithLocal
4	11.85%	73.10%	845	55	java/lang/String.getChars
5	11.75%	84.85%	838	42	java/lang/String.getChars
6	11.72%	96.58%	836	50	java/lang/String.getChars

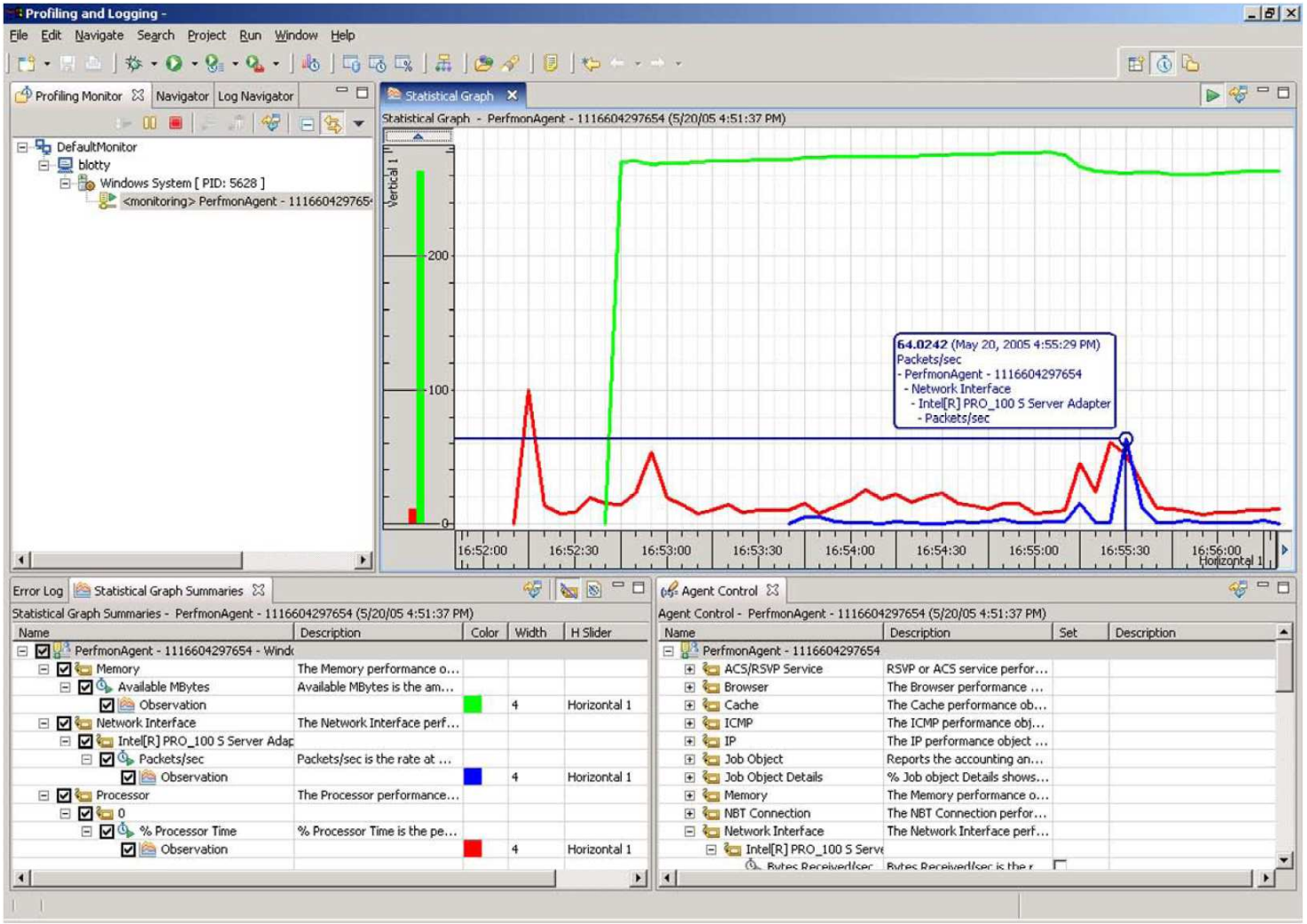
(remaining entries less than 1% each, omitted for brevity)

Mesure et Analyse des Performances (ii)

- JVMStat
  - http://java.sun.com/performance/jvmstat/
  - adds light weight performance and configuration instrumentation to the HotSpot JVM
  - provides a set of monitoring APIs and tools for monitoring the performance of the HotSpot JVM in production environments.
    - The monitoring interfaces added to the HotSpot JVM are proprietary and may or may not be supported in future versions of the HotSpot JVM.
  - negligible performance impact.

Mesure et Analyse des Performances (iii) Exemple

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



Optimisation des Performances

- Moteur de Script Java
  - Jython (jython.sourceforge.net), …
- Interpréteur de Bytecode
- Compilateur Natif (statique)
  - .class en .c en .s en .exe
- Compilateur à la volée (dynamique)
  - Compilation JIT (Just-In-Time) de Symantec
- Optimiseur HotSpot™
  - analyse de la taille des tableaux et vecteurs
  - garbage collector
  - « method inlining »
    - avec vérification au chargement (dynamique) d ’une classe
- Benchmark de JVM

Compilateur Natif (statique)

- Transformation .java/.class en .c en .s en .exe
  - Remarque : Les archives (rt.jar, …) doivent être aussi compilées et linkées
- Avantages
  - Performance
    - et encore …
  - Empreinte mémoire réduite (informatique embarquée)
    - et encore
- Inconvénients
  - Dépendance face au processeur cible
  - Pas de chargement sécurisé (car non vérifiable)
    - Nuit au concept d 'applets téléchargeables
  - Le code natif occupe 2,5 fois la place du bytecode
- Voir
  - GCJ http://gcc.gnu.org/java/ fait partie de la distribution GCC
    - Moins performant qu'une JVM JIT
  - http://www.towerj.com/
  - http://sourceware.cygnus.com/java/

Compilateur à la volée (dynamique)

- Compilation JIT (Just-In-Time)
- Principe
  - Téléchargement du bytecode
    puis compilation du bytecode vers le langage machine
    puis exécution
- Inconvénients
  - consomme de la mémoire
  - durée de la compilation non déterministe
  - peu rentable si les sections « compilées » sont peu utilisées

Optimiseur HotSpot™

- Compilation à la volée des sections critiques
  - 5% du bytecode occupant 95% la CPU
  - Interprétation du reste
- Techniques
  - analyse de la taille des tableaux et vecteurs
  - garbage collector
  - « method inlining »
    - avec vérification au chargement (dynamique) d 'une classe

Voir HotSpot FAQ

- http://java.sun.com/docs/hotspot/PerformanceFAQ.html

Test de Compatibilité JavaCheck

- Outil de test de compatibilité
  d ’une application ou d ’une applet
- avec une plateforme Java
  - PersonalJava version x.y, ... J2ME, …
  - La description de la plateforme et de ses périphériques est dans un fichier .spc
  - JavaCheck vérifie les versions et les types des VM et si l ’application n ’utilise pas des classes absentes des packages du .spc

More tools (JavaSE 6)

http://java.sun.com/javase/6/docs/technotes/tools/

- Monitoring Tools
  - **jps**: JVM Process Status Tool
    - Lists instrumented HotSpot Java virtual machines on a target system
  - **jstat**: JVM Statistics Monitoring Tool
    - Attaches to an instrumented HotSpot Java virtual machine and collects and logs performance statistics as specified by the command line options.
  - **jstatd**: JVM jstat Daemon
    - Launches an RMI server application that monitors for the creation and termination of instrumented HotSpot Java virtual machines and provides a interface to allow remote monitoring tools to attach to Java virtual machines running on the local system.
- Troubleshooting Tools
  - **jinfo**: Configuration Info for Java
    - Prints configuration information for for a given process or core file or a remote debug server.
  - **jhat**: Heap Dump Browser
    - Starts a web server on a heap dump file (eg, produced by jmap -dump), allowing the heap to be browsed.
  - **jmap**: Memory Map for Java
    - Prints shared object memory maps or heap memory details of a given process or core file or a remote debug server.
    - See also Solaris' pmap (Print the address space map of each process).
  - **jsadebugd**: Serviceability Agent Debug Daemon for Java
    - Attaches to a process or core file and acts as a debug server.
  - **jstack**: Stack Trace for Java
    - Prints a stack trace of threads for a given process or core file or remote debug server.

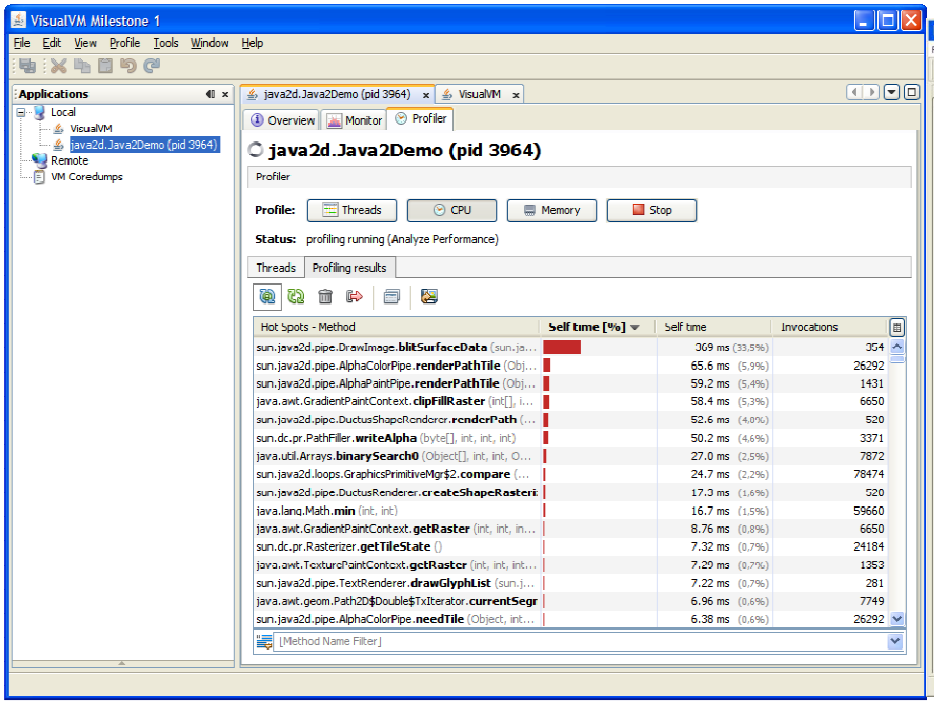
More tools

- Hot redeployment
  - Motivation : limit the unavailabilty time of applications
    - Web frameworks, Application servers, …
  - Solution : class reloading
    - Java 1.4 HotSwap
    - JavaRebel http://www.zeroturnaround.com/javarebel/

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.**



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

Benchmark des machines virtuelles Java

- Plusieurs machines virtuelles
  - Sun, Symantec, IBM J9 & JikesRvm, *MicroSoft*, Kaffe, …
  - Différence de performances
- Benchmarks
  - http://www.volano.com/report.html
  - Java Grande Forum Benchmark Suite

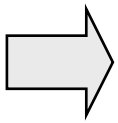
Interpréteurs Java

- Boucle interactive
  ou script sans compilation vers du bytecode
- BeanShell
- Remarque :
  - Langages de syntaxe non Java (plus de 200) supportant l'appel à/par des objets Java
    - Jython, Rhino/JavaScript, Jacl, NetRexx, JRuby, JudoScript, Groovy, ObjectScript , …
    - Voir http://robert-tolksdorf.de/vmlanguages.html
    - Voir http://www.ociweb.com/jnb/archive/jnbMar2001.html
  - JSR 223 Java Scripting
  - Implémentation optimisée avec le futur JavaSE 7

Retro-compilateur

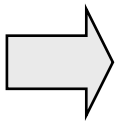
- Retroweaver http://retroweaver.sourceforge.net/
  - *tool that enables you to take advantage of the new Java 1.5 language features (***generics, extended for loops, static imports, autoboxing/unboxing, varargs, enumerations, annotations***) in your source code, while still retaining compatability with 1.4 (and older) virtual machines. Retroweaver operates by transforming Java class files compiled by a 1.5 compiler into class files which can be run on an older virtual machine or J2ME KVM.*
- Exemples

```
public void foo( String... ) {
}
```



```
public void foo( String[] ) {
}
```

```
public class Foo<T extends Comparable> {
public void foo( T t ) { ... }
}
```



```
public class Foo {
public void foo( Comparable t ) { ... }
}
```

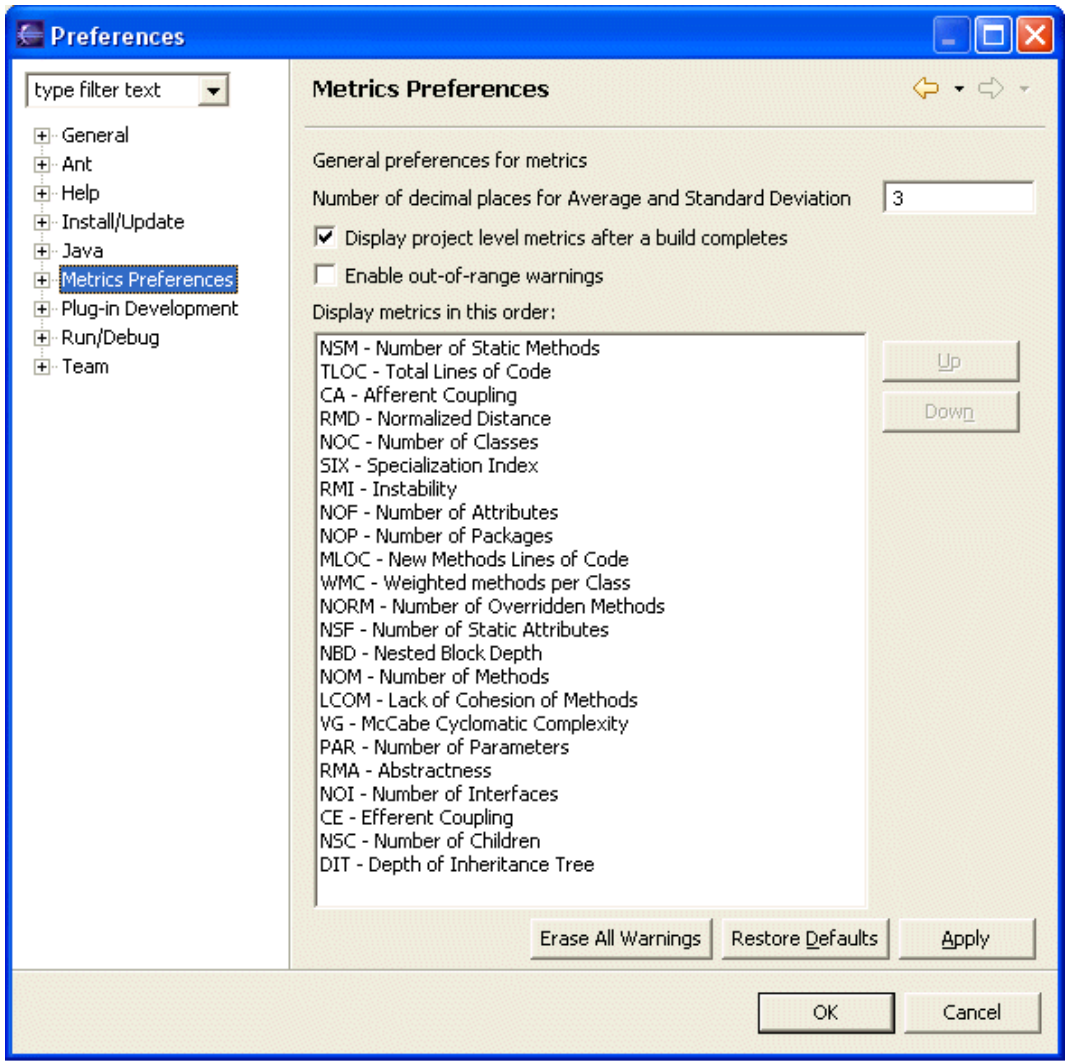
Mesures de qualité du code

- Métriques sur les sources d'un projet permettant d'évaluer sa qualité (maintenance, retro-ingénierie, évolution …)
  - et l'habilité de l'équipe de développement ;-)
- Exemple de métriques
  - 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

Mesures de qualité du code

- Exemple: Metrics (Tache ANT + Eclipse plugin)



Mesures de qualité du code

- Sonar

Mesures de qualité du code

- Autres (standalone ou sous forme de plugins d'IDE)
  - http://metrics.sourceforge.net/
  - http://qjpro.sourceforge.net/
  - http://www.geocities.com/sivaram_subr/index.htm
  - http://www.kclee.de/clemens/java/javancss/
  - https://loc-counter.dev.java.net/
  - …

Analyseurs Lexical et Grammatical

- Permet de produire un « parser » en java à partir d ’une grammaire et d ’actions en java
- Outils
  - Analyseurs Grammaticaux (LALR)
    - JavaCC (le plus utilisé)
    - BYACC/Java
      - yacc de Berkeley avec des actions C/C++ ou Java
    - Jacc, JavaCup, ANTLR, QJJ, JDT parser …
  - Analyseurs Lexicaux
    - JavaLex

JCP Java Community Process http://www.jcp.org

- Proposition of JSR (Java Specification Request)
- Experts Group then Ballots

- Open source reference implementation

Java en Open Source

- Compilateur
  - kaffec, jikes, javac (depuis 6.0) …
- VMs
  - Kaffe, Apache Harmony, JikesRVM et maintenant Java (depuis 6.0), Dalvik, …
  - Autres
    - VVM (extensible), MVM (Isolates), Maxime (écrite en Java)
- Environnement
  - GNU Classpath
    - http://www.gnu.org/software/classpath/
  - Apache Harmony
  - Android
  - …

IDE

- Editeur
  - JEdit
  - Notepad 2
  - …
- Pro
  - Eclipse
  - Sun NetBeans
  - Borland JBuilder
  - IntelliJ IDEA
  - Oracle JDeveloper
  - …
- Pédagogique
  - BlueJ
    - http://www.bluej.org/

Forges

- Motivations
  - Plateformes collaborative de développement logiciel
- Features
  - Source repository
  - Release/Artifact repository
  - Code quality metrics
  - Wiki
  - Developer scoring
    - Competences
  - Activity reporting
  - Bug tracking
  - Feature tracking
  - Continuous integration (nightly builds, …)
  - Surveys
  - IM (Jabber, …)
  - License checking
  - …
- Platforms
  - CodeX
  - GForge
  - LibreSource
  - …

Bibliographie

- Jack Shirazi, “Java Performance Tuning”, Ed Oreilly, 2000, ISBN 0-596-00015-4
- Richard Hightower, Nicholas Lesiecki, «Java Tools for Extreme Programming: Mastering Open Source Tools, including Ant, JUnit, and Cactus», Ed Wiley, ISBN: 0-471-20708-X, November 2001
- Eric M. Burke, Brian M. Coyner « Java Extreme Programming Cookbook », Ed O’ReillyMarch 2003, ISBN: 0-596-00387-0, 288 pages

- http://www.tigris.org/servlets/ProjectList