Analyse de la qualité du code Java avec JDepend 2.7

Hugo Etiévant

Dernière mise à jour : 20 juin 2004

Qu’est-ce que la qualité ?

La réussite d’un projet de développement nécessite une organisation rigoureuse tant dans la gestion des ressources et des tâches que dans l’organisation technique. Cette dernière implique la définition et le respect de **normes de développement** qui assurent une uniformité du code source en vue d’une maintenance et d’une documentation aisée.

Un code de qualité assure la **pérennité**, la **diffusion** et la **maintenabilité** du projet.

La réussite d’un projet dépend de la qualité du code.

Comment se mesure la qualité ?

Sur quels critères peut-on se baser pour analyser la qualité du code ?

- lisibilité du code
- uniformité des conventions
  - de nommage (variables, méthodes, classes, packages)
  - d'indentation
  - d'organisation (répertoires du projet)
  - de documentation (commentaires, JavaDoc)
- couplage minimum (modules indépendants)
- complexité cyclomatique miminum

Présentation de JDepend

JDepend est un outil développé en Java qui analyse le code source d'un projet Java et calcule des métriques de qualité pour chaque package.

JDepend est sensible à l'extensibilité, la réutilisabilité et la maintenabilité des sources.

Il est développé par Mike Clark de la « Clarkware Consulting, Inc ».

Il est soumis à la licence « BSD License » : Les sources peuvent être modifiées et redistribuées à la condition que le programme résultant suive lui aussi la licence BSD.

Installation sous Windows

Téléchargement à l'adresse suivante :

http://www.clarkware.com/software/JDepend.html

Décompression dans un répertoire :

jdepend-2.7/

Rajout en variable d'environnement de ce répertoire et de son archive JAR contenant les classes :

```
set JDEPEND_HOME=D:\jdepend-2.7
set CLASSPATH=%JDEPEND_HOME%\lib\jdepend.jar
```

Recompilation :

```
cd $JDEPEND_HOME
ant jar
```

Installation sous Unix

Téléchargement à l'adresse suivante :

http://www.clarkware.com/software/JDepend.html

Décompression dans un répertoire :

jdepend-2.7/

Rajout en variable d'environnement de ce répertoire et de son archive JAR contenant les classes :

```
JDEPEND_HOME=D:\jdepend-2.7
chmod -R a+x $JDEPEND_HOME
export CLASSPATH=$CLASSPATH:$JDEPEND_HOME/lib/jdepend.jar
```

Recompilation :

```
cd $JDEPEND_HOME
ant jar
```

Mise en oeuvre

JDepend possède trois interfaces :

- une interface graphique
- deux interfaces en ligne de commande
  - dont l'une pour un format texte
  - et l'autre un format XML

Interface graphique (1)

Lancement :

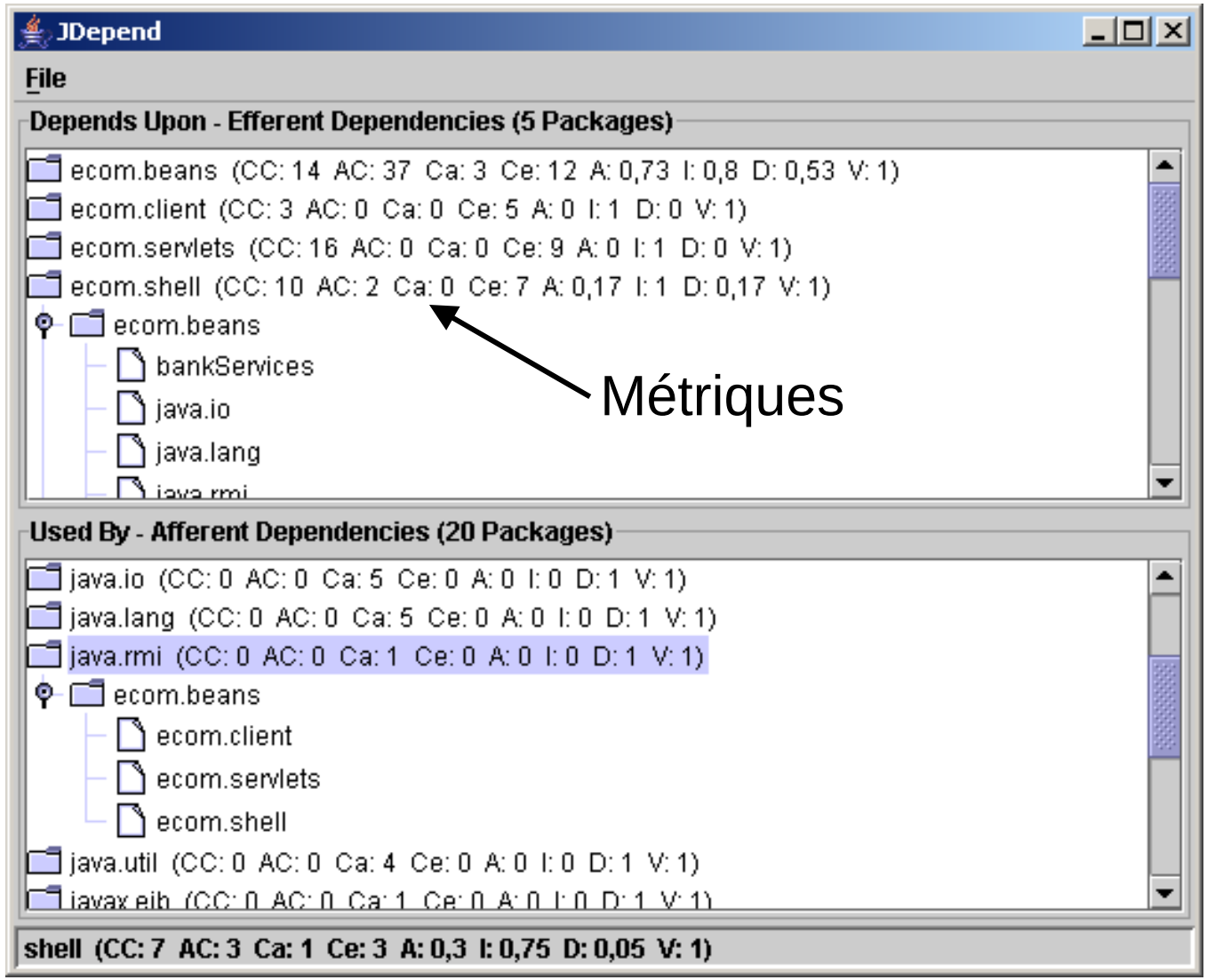
java jdepend.swingui.JDepend <répertoire>

Le « répertoire » est le chemin des classes à analyser, il peut lui-même contenir plusieurs packages. Une liste de répertoires peut également être passée en paramètre, il suffit de les séparer d'un espace.

Exemple :

java jdepend.swingui.JDepend d:\projet\classes

Interface graphique (2)



Arbre des dépendances descendantes

utilise

Arbre des dépendances ascendantes

est utilisé par

Interface graphique (3)

L'interface graphique montre les dépendances des packages et classes ainsi que les métriques de qualité associées à chacun d'eux.

L'exemple précédent montre dans sa vue descendante que le package ecom.shell utilise le package ecom.beans qui lui-même utilise les classes java.io, etc.

La vue ascendante montre que le package java.rmi est utilisé par le package ecom.beans, en particulier par les classes ecom.client, ecom.servlets et ecom.shell.

Les métriques du package ecom.shell sont les suivantes : CC:10 AC:2 Ca:0 Ce:7 A:0.17 I:1 D:0.17 V:1.

Interface en ligne de commande

Format texte :

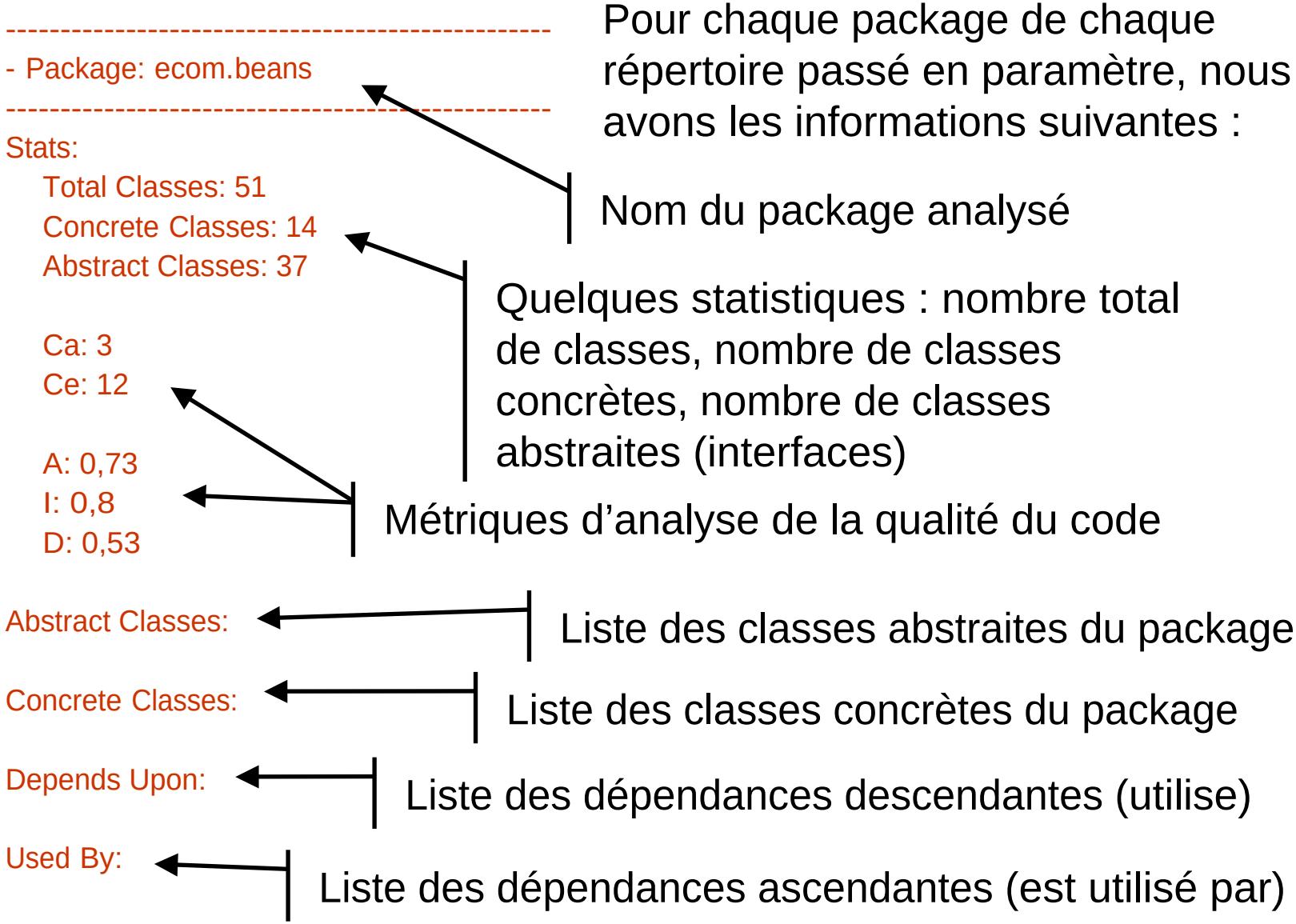
java jdepend.textui.Jdepend [–file <fichier>] <répertoire>

Format XML :

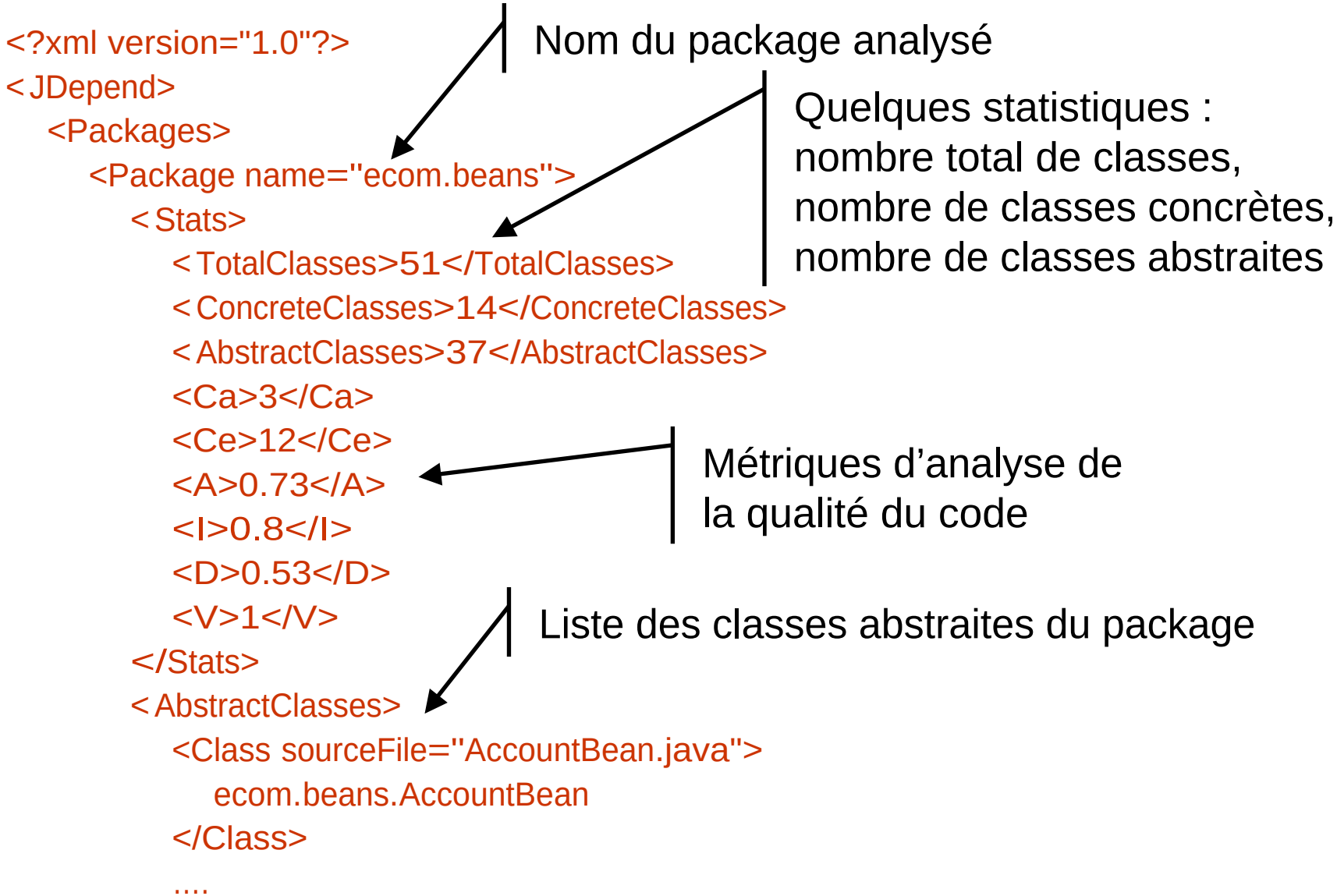
java jdepend.xmlui.JDepend [–file <fichier>] <répertoire>

L'option –file permet de spécifier le « fichier » dans lequel seront écrites les métriques. Si aucun fichier n'est spécifié, c'est la sortie standard qui est utilisée.

Format texte



Format XML


<?xml version="1.0"?>
<JDepend>
<Packages>
<Package name="ecom.beans">
<Stats>
<TotalClasses>51</TotalClasses>
<ConcreteClasses>14</ConcreteClasses>
<AbstractClasses>37</AbstractClasses>
<Ca>3</Ca>
<Ce>12</Ce>
<A>0.73</A>
<I>0.8</I>
<D>0.53</D>
<V>1</V>
</Stats>
<AbstractClasses>
<Class sourceFile="AccountBean.java">
ecom.beans.AccountBean
</Class>
....
Nom du package analysé
Quelques statistiques :
nombre total de classes,
nombre de classes concrètes,
nombre de classes abstraites
Métriques d'analyse de
la qualité du code
Liste des classes abstraites du package

Intégration à Ant (1)

Pour ceux qui déploient leurs projets avec Ant, une tâche spécifique peut être intégrée au fichier build.xml afin de générer automatiquement l'analyse de la qualité de votre code.

Dans les exemples qui vont suivre, les variables suivantes sont utilisées : docs.dir est le répertoire où seront générés les fichiers, classes.dir est le répertoire où se trouvent vos sources et ant.home est le répertoire d'installation de Ant.

Exemple de déclaration de variables :

```
<property environment="myenv"/>
<property name="ant.home" value="${myenv.ANT_HOME}" />
<property name="classes.dir" value="classes" />
<property name="docs.dir" value="docs" />
```

Intégration à Ant (2) – format texte

La tâche utilisateur suivante « jdepend » permet de générer un rapport de qualité au format texte. Cette tâche dépend de la tâche de compilation « compile » car Jdepend travaille sur les fichiers de classes et non pas sur les fichiers sources. La tâche Ant à exécuter est « jdepend », elle prend pour paramètre (« outputfile ») le nom complet du fichier à générer. Elle possède deux éléments obligatoires : « sourcespath » et « classpath » afin se spécifier les chemins des des classes à analyser.

```
<target name="jdepend" depends="compile">
    <jdepend outputfile="${docs.dir}/jdepend-report.txt">
        <sourcespath>
            <pathelement location="${classes.dir}" />
        </sourcespath>
        <classpath>
            <pathelement location="${classes.dir}" />
        </classpath>
    </jdepend>
</target>
```

Intégration à Ant (3) – format XML

Pour la génération au format XML, il faut spécifier le format « xml » via le paramètre « format ».

```
<target name="jdepend" depends="compile">
    <jdepend format="xml" outputfile="${docs.dir}/jdepend-report.xml">
        <sourcespath>
            <pathelement location="${classes.dir}" />
        </sourcespath>
        <classpath>
            <pathelement location="${classes.dir}" />
        </classpath>
    </jdepend>
</target>
```

Intégration à Ant (4) – format HTML

Pour la génération au format HTML, on rajoute à la tâche permettant de générer du XML une tâche de transposition de style via un fichier xsl et à partir du fichier XML généré précédemment.

```
<target name="jdepend" depends="compile">
    <jdepend format="xml" outputfile="${docs.dir}/jdepend-report.xml">
        <sourcespath>
            <pathelement location="${classes.dir}" />
        </sourcespath>
        <classpath>
            <pathelement location="${classes.dir}" />
        </classpath>
    </jdepend>
    <style basedir="${docs.dir}" destdir="${docs.dir}"
        includes="jdepend-report.xml"
        style="${ant.home}/etc/ jdepend.xsl" />
</target>
```

Intégration à Ant (5) – format HTML

Le fichier HTML contient les mêmes informations que les formats texte et XML mais organisées avec des liens hypertextes permettant de naviguer facilement parmi les métriques.

Package	Total Classes	Abstract Classes	Concrete Classes	Afferent Couplings	Efferent Couplings	Abstractness	Instability	Distance
ecom.beans	51	37	14	3	12	0.73	0.8	0.53
ecom.client	3	0	3	0	5	0	1	0
ecom.servlets	16	0	16	0	9	0	1	0
ecom.shell	12	2	10	0	7	0.17	1	0.17
shell	10	3	7	1	3	0.3	0.75	0.05

shell

Afferent Couplings: 1 Efferent Couplings: 3 Abstractness: 0.3 Instability: 0.75 Distance: 0.05

Abstract Classes	Concrete Classes	Used by Packages	Uses Packages
shell.ShellCommand shell.ShellConstantes shell.ShellContext	shell.CartCommandImpl shell.EchoCommandImpl shell.HelpCommandImpl shell.Shell shell.ShellContextImpl shell.SimpleQuitCommandImpl shell.VarCommandImpl	ecom.shell	java.io java.lang java.util

Critères de qualité – CC, AC

Nombre de classes (CC & AC).

Le nombre de classes abstraites (AC) et concrètes (CC) est un indicateur d'extensibilité d'un package. Plus ce nombre est important, plus les entités qu'elles implémentent peuvent être étendues indépendamment les unes des autres.

Critères de qualité – Ca

Afferent coupling (Ca) où couplage par dépendance descendante.

Le nombre de packages qui utilisent les classes du package analysé peut être un indicateur de mauvaise gestion des dépendances ou être le signe que le package est le centre de l'application. Plus ce nombre est grand, plus il est nécessaire de fragmenter le package.

Critères de qualité – Ce

Efferent coupling (Ce) où couplage par dépendance ascendante.

Le nombre de packages utilisés par les classes du package analysé. C'est un indicateur d'indépendance du code. Plus ce nombre est faible, mieux c'est.

Critères de qualité – A

Abstractness (A) où degré d'abstraction.

Pourcentage de classes concrètes par rapport aux classes abstraites. Proche de 0 : package concret, proche de 1 : package abstrait.

Le degré d'abstraction d'un package doit tendre vers l'une ou l'autre des deux borne : 0 ou 1. Une valeur proche de 0.5 montrerait une mauvaise écriture du code.

Critères de qualité – I

Instability (I) où degré de stabilité.

Indicateur de résilience du package : propriété de stabilité par rapport à la mise à jour d'autres packages. Proche de 0 : package stable, proche de 1 : package instable.

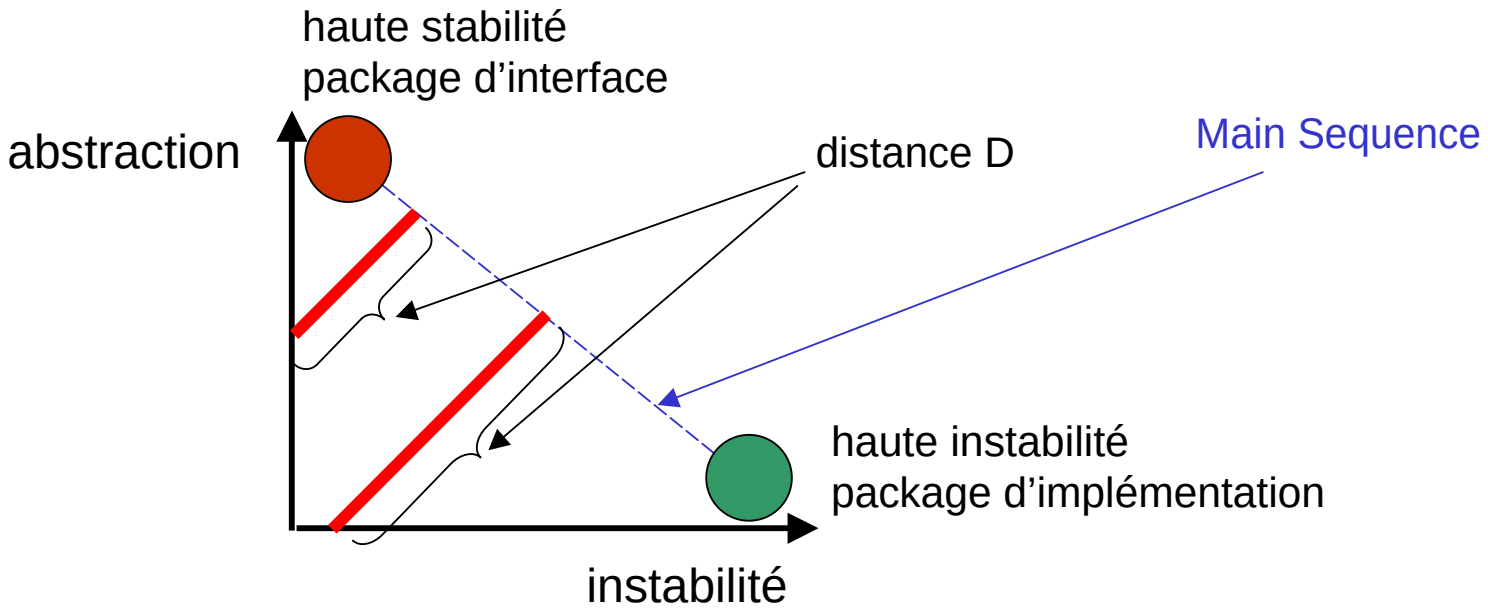
Lié aux critères de dépendances ascendante et descendante.

Critères de qualité – D

Distance from the Main Sequence (D)

Distance normale (perpendiculaire) à la droite A + I = 1.

Proche de 0 : le package coïncide avec la « Main sequence », proche de 1 : très éloigné de la « Main séquence ». Dans le cas idéal (D = 0), un package est soit complètement abtrait et stable (A=1, I = 0) ou complètement concret et instable (A=0, I=0). Si D est proche de 1, c'est mauvais et est le signe d'un package mal géré.



Critères de qualité – V

Package dependency cycle : Volatility (V)

La volatilité d'un package est sa propention à évoluer au fil du temps. Un V est proche de 0 dénote un package qui est le noyau dur d'une application du fait du couplage avec un grand nombre d'autres packages qui font référence à ce premier.

Interprétation des résultats

JDepend ne livre aucune interprétation, seulement des métriques. En revanche, ces métriques fournissent beaucoup d'informations de haut niveau d'agrégation sur vos classes.

Un package faiblement couplé, abstrait et stable est d'excellente qualité.

Historique

- **20 juin 2004** : mise à jour critères Ca, A, D, V
- **17 avril 2004** : création du document (27 diapos)

Agissez sur la qualité de ce document en envoyant vos critiques et suggestions à l'auteur.

Remerciements à Vincent Brabant pour ses remarques.

Pour toute question technique, se reporter au forum Java de Developpez.com

Hugo Etiévant
cyberzoide@yahoo.fr
http://cyberzoide.developpez.com/