

Projet G.L.

Enseignant: S. Vauttier-J. Vlasak

Projet GL

- ❑ Intervenants:
 - ❑ S. Vauttier
 - ❑ J. Vlasak
- ❑ But
 - ❑ Aborder les bonnes pratiques de développement logiciel.
 - ❑ Mener un projet en équipe de 3 personnes
- ❑ Présentation du projet
 - ❑ Carnet de note électronique

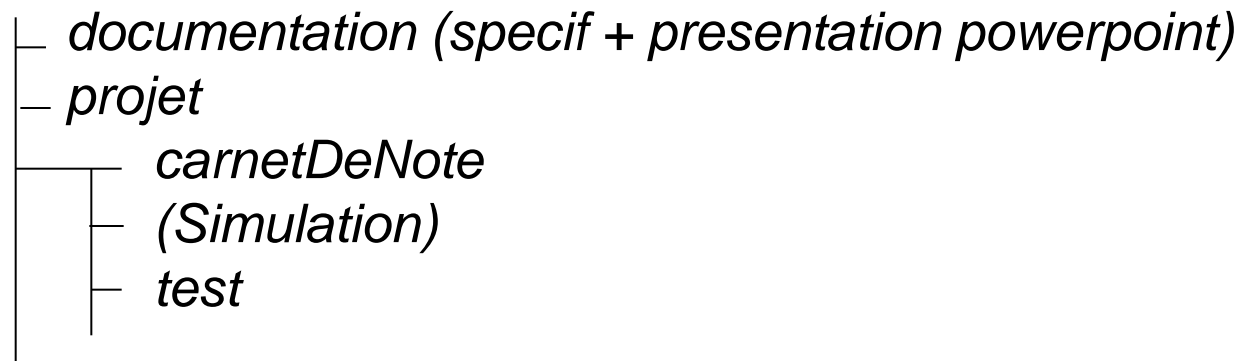
Rendu Projet GL

- ❑ Rendu:
 - ❑ Cahier des charges
 - ❑ analyse des exigences.
 - ❑ Dossier de spécification
 - ❑ Modélisation UML du modèle complet du domaine.
 - ❑ Modélisation UML du logiciel implémenté.
 - ❑ Sources commentés du projet.
 - ❑ Présentation oral et démonstration fin de la première semaine de juillet.
- ❑ Spécification du projet
 - ❑ Pas de cycle en V, méthode incrémental (agile).

Rendu Projet GL

- Source du projet:
 - Respecter la structure suivante

groupe-Nom1-Nom2



- externaliser dans un fichier properties les paramètres de la BD.

Rendu Projet GL

- La notation sera une moyenne entre:
 - Documentation
 - Source.
 - Qualité de code (via Sonar).
 - Oral.

Outils Projet GL

- ❑ En fonction de votre niveau, vous utilisez les outils de votre choix. Pas d'obligation.
- ❑ Développement de code
 - ❑ Netbeans
 - ❑ Eclipse avec le module WindowBuilder
 - ❑ <https://www.eclipse.org/windowbuilder/>
 - ❑ ...
- ❑ Gestion des sources
 - ❑ Clef usb.
 - ❑ Gestionnaire de source: mercurial, git, cvs

Outils Projet GL

- ❑ Pas de printf!!! Utilisation des log
 - ❑ Soit les log présent dans le jdk
 - ❑ Soit Log4j de la fondation apache
- ❑ Test unitaire
 - ❑ Tester le programme fonction par fonction
 - ❑ L'outils le plus connu: Junit
- ❑ Qualité de code
 - ❑ Respecter les règles de code de java
 - ❑ L'outil sonar donnera une note à votre code.

Outils Projet GL

- ❑ Ne pas sous estimer le temps de mise en place des outils.
- ❑ Autres outils:
 - ❑ Mockito : création de bouchon logiciel.
 - ❑ Readmine: suivie des anomalies du programme.
 - ❑ Cobertura ou JaCoCo: couverture du code.
 - ❑ Maven: compilation, permet de résoudre les problèmes de dépendance des libraires.
 - ❑ Jendkins: intégration continu.

Gestion des sources

□ Les principaux gestionnaires

Software	Maintainer	Development status	Repository model	Concurrency model	License	Platforms supported
Perforce	Perforce Software Inc.	actively developed	Client-server	Merge or lock	Proprietary	Unix-like, Windows, OS X
Surround SCM	Seapine Software	actively developed	Client-server	Merge or lock	Proprietary	Linux, Windows, OS X
Codeville	Ross Cohen	official site offline; latest release July 13, 2007	Distributed	precise codeville merge	BSD	Unix-like, Windows, OS X
CVS	The CVS Team ^[2]	maintained but new features not added	Client-server	Merge	GNU GPL	Unix-like, Windows, OS X
darcs	The Darcs team	actively developed	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X
Fossil	D. Richard Hipp	actively developed	Distributed	Merge	BSD	POSIX, Windows, OS X, Other
Git	Junio Hamano	actively developed	Distributed	Merge	GNU GPL	POSIX, Windows, OS X
GNU arch	Andy Tai	unmaintained	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X
GNU Bazaar	Canonical Ltd.	actively developed	Distributed ^[nb 1]	Merge	GNU GPL	Unix-like, Windows, OS X
LibreSource Synchronizer	Artenum ^[4]	maintained and new features under development	Client-server extended to "tree" ^[nb 3]	Merge	GNU GPL ^[nb 4]	Unix-like, Windows, OS X
Mercurial	Matt Mackall	actively developed	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X
Monotone	Nathaniel Smith, Graydon Hoare	actively developed	Distributed	Merge	GNU GPL	Unix-like, Windows, OS X
Subversion (SVN)	Apache Software Foundation ^[10]	actively developed	Client-server ^[nb 6]	Merge or lock ^[nb 7]	Apache/BSD style	Unix-like, Windows, OS X
SVK	Best Practical	unmaintained	Client-server, decentralized	Merge	Artistic/GPL	Unix-like, Windows, OS X
Veracity	SourceGear LLC	actively developed	Distributed	Merge or lock	Apache	Unix-like, Linux, Windows
Vesta	Kenneth Schalk; Tim Mann, ^{[12][13]}	actively developed	Distributed NFS-protocol- emulation choice to optionally confederate clients and/or servers	lock on branch; merge branch-to-branch	LGPL	Tru64, Linux

Gestion des sources

- Initialisation du projet.

```
$ hg init project
```

- Ajout des fichiers

```
$ cd project
```

```
$ echo 'print("Hello")' > hello.py
```

```
$ hg add
```

```
$ hg commit
```

- Voir l'historique des révisions

```
$ hg log
```

```
changeset: 0:a5ecbf5799c8
```

```
user: Mr. Johnson
```

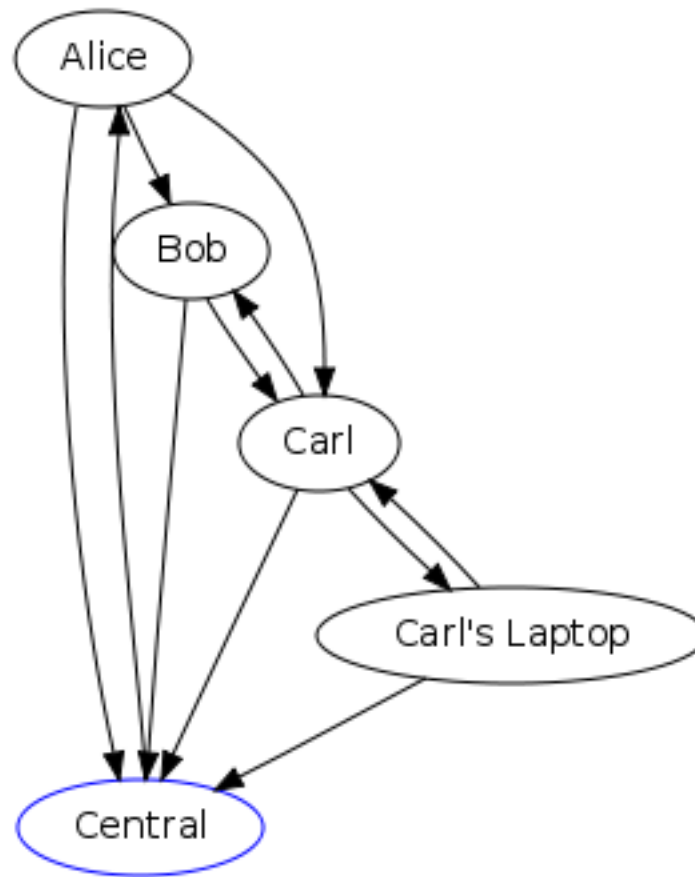
```
date: Sun Nov 20 11:00:00 2011 +0100
```

```
summary: Initial commit.
```

Gestion des sources
















- ❑ Soit en ligne de commande, ou bien intégré dans eclipse ou Netbeans.
- ❑ Développement à plusieurs:
 - ❑ Pas de site centralisé nécessaire
 - ❑ Débute par cloner le projet
 - ❑ \$ hg clone <http://selenic.com/repo/hello>
 - ❑ Récupère et soumet les modifications via push/pull
 - ❑ Possibilité d'utiliser ssh au lieu de http.

Outils Projet GL



A Mercurial Network

Outils Projet GL

Provider	Git	Mercurial	Free hosting	Premium hosting
Gitorious 	X		X	X
GitLab 	X		X	X
GitHub 	X		X	X
Codeplex 	X	X	X	
repo.or.cz 	X		X	
Codebase 	X	X	X	X
Google Code 	X	X	X	?
BitBucket 	X	X	X	X
SourceForge 	X	X	X	?
Kenai 		X	X	X
java.net 	X	X	X	X
ProjectLocker 	X		X	X
Kiln 	X	X		X
Codeplane 	X			X
Assembla 	X	X	X	X

Utilisation des log

- Log4j possède 3 composants:

- loggers
- appenders
- les layouts

- Les loggers

```
// get a logger instance named "com.foo"
Logger logger = Logger.getLogger("com.foo");

// Now set its level. Normally you do not need to set
the level of a logger programmatically. This is usually
done in configuration files.
logger.setLevel(Level.INFO);

Logger barlogger = Logger.getLogger("com.foo.Bar");

// This request is enabled, because WARN >= INFO.
logger.warn("Low fuel level.");
logger.debug("Starting search for nearest
    gas station.");
barlogger.info("Located nearest gas station.");
```

Utilisation des log

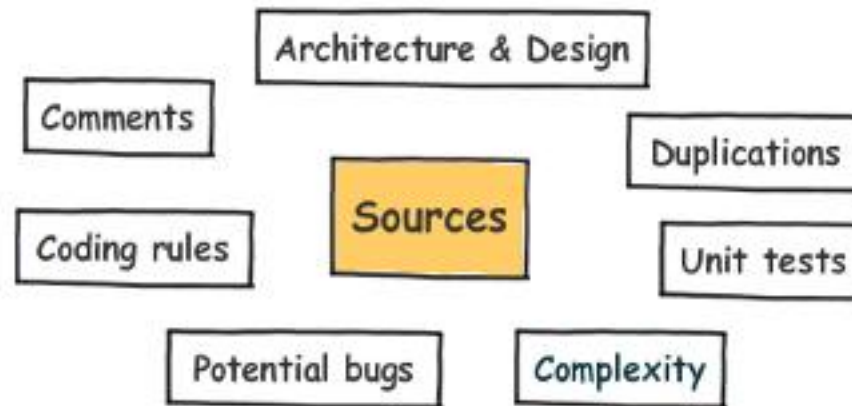
- Les appenders :
 - Leur rôle est de contrôler la sortie des messages en fonction du niveau de log
 - Il y a 6 niveaux de log
 - TRACE, DEBUG, INFO, WARN, ERROR, FATAL
 - Il est possible de les configurer par programmation ou par fichier

```
// Set up a simple configuration that logs on the console.  
BasicConfigurator.configure();
```

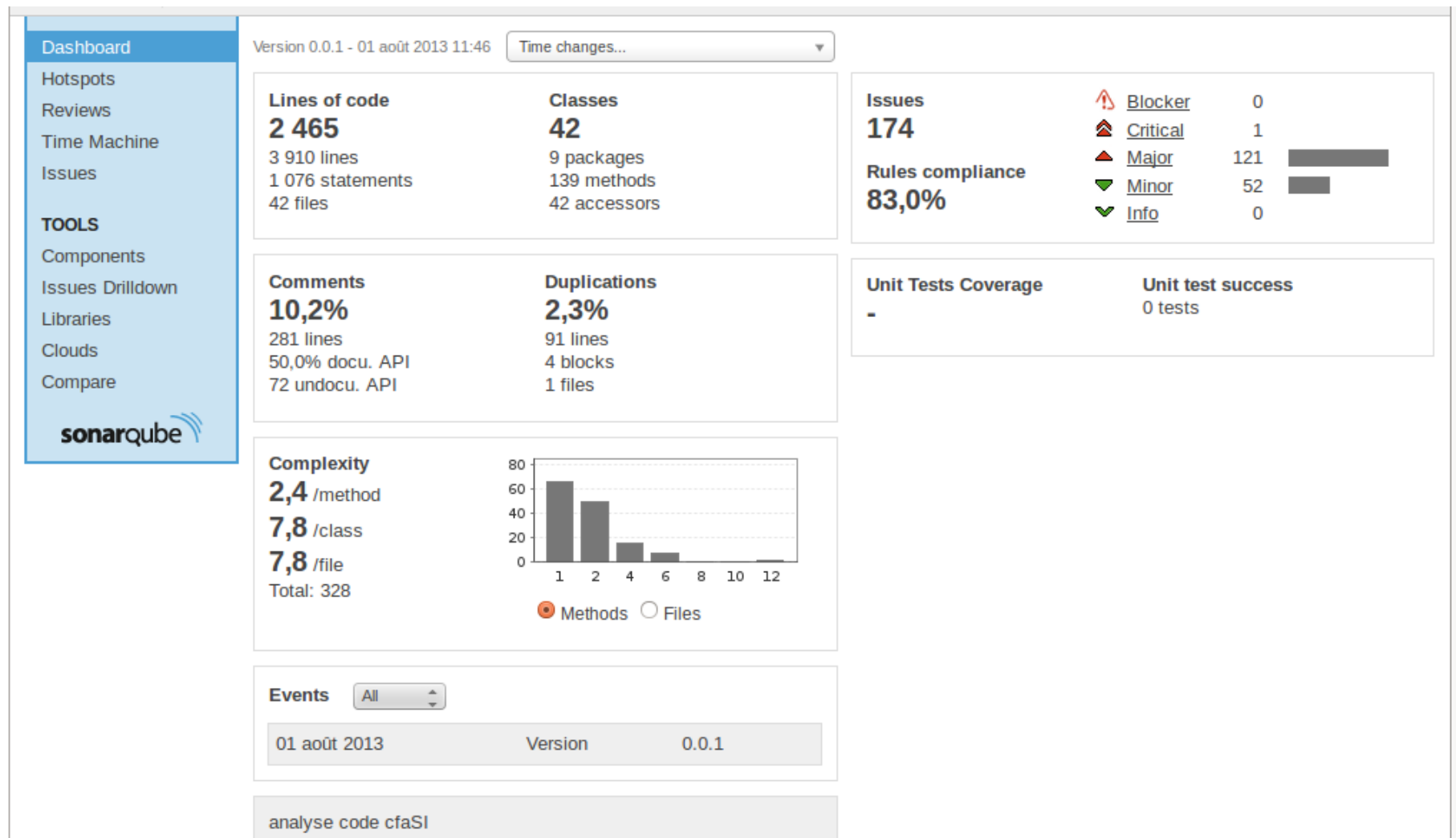
```
# Set root logger level to DEBUG and its only appender to A1.  
log4j.rootLogger=DEBUG, A1  
# A1 is set to be a ConsoleAppender.  
log4j.appender.A1=org.apache.log4j.ConsoleAppender  
# A1 uses PatternLayout.  
log4j.appender.A1.layout=org.apache.log4j.PatternLayout  
log4j.appender.A1.layout.ConversionPattern=%-4r [%t] %-5p %c %x - %m%n
```

Qualité de code: Sonar

- Permet de tester 7 caractéristiques sur un code

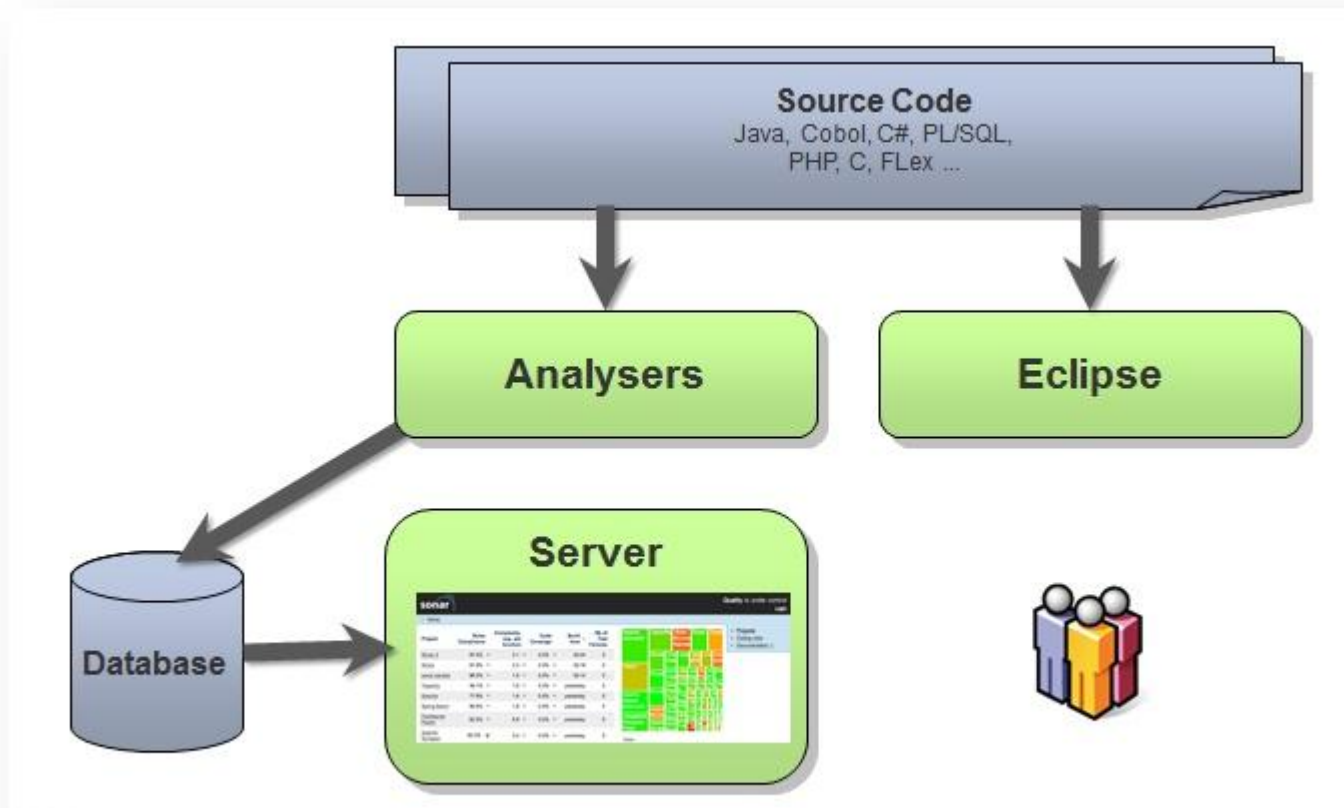


Qualité de code: Sonar



Qualité de code: Sonar

- Structure de Sonar:



Qualité de code: Sonar

- ❑ Configuration du serveur
 - ❑ Téléchargez la version à <http://www.sonarqube.org/downloads/>
 - ❑ Décompressez dans un répertoire.
 - ❑ Modifiez `<install_directory>/conf/sonar.properties`
 - ❑ `sonar.jdbc.username=sonarqube`
 - ❑ `sonar.jdbc.password=mypassword`
 - ❑ `sonar.jdbc.url=jdbc:mysql://localhost/sonarqube?useUnicode=true&characterEncoding=utf8`

Qualité de code: Sonar

- ❑ Configuration de l'analyseur (sonnar runner)
 - ❑ Téléchargez la version à <http://docs.codehaus.org/display/SONAR/Installing+and+Configuring+SonarQube+Runner>
 - ❑ Modifiez <install_directory>/conf/sonar-runner.properties
 - ❑ sonar.jdbc.username=sonarqube
 - ❑ sonar.jdbc.password=mypassword
 - ❑ sonar.jdbc.url=jdbc:mysql://localhost/sonarqube?useUnicode=true&characterEncoding=utf8
 - ❑ Configurez sonar-project.properties
 - ❑ Exécutez le script

Structure du code

- ▣ Application des patterns
 - ▣ MVC
 - ▣ DAO
 - ▣ Singleton