

Projet Informatique

2019

Steve Hostettler

Software Modeling and Verification Group

University of Geneva



Compilation et gestion de dépendances



Pourquoi standardiser la compilation?

- Compilation en dehors de l'IDE
 - chez le client par exemple
 - sur le serveur d'intégration continue
 - facilement reproductible / nouveaux développeurs
- Générations répétitives
 - test unitaires, doc, analyses, déploiement



Pourquoi gérer les dépendances?

- Complexité du graphe de dépendances
 - Qui a besoin de quoi?
 - Où trouver quoi?



Maven



Maven

- Basé sur un modèle objet
 - Project Object Model
 - Fichier XML (pom.xml)
- Utilise des conventions pour simplifier la configuration
- Gestion du cycle de vie étendue



Maven vs Ant ou Make

- ANT et Make sont basés scripts qui définissent le cycle de vie.
- Ils décrivent un procédure.
- Pas de structure standard
- Pas de gestion des dépendances / M.A.J
- Maven configure un cycle de vie et décrit un projet
- Basé sur un mécanisme de template (archetype)



Project Object Model

```
<project xmlns="..." >

  <modelVersion>4.0.0</modelVersion>

  <groupId>MyProject</groupId>
  <artifactId>ServiceLayer</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <packaging>jar</packaging>

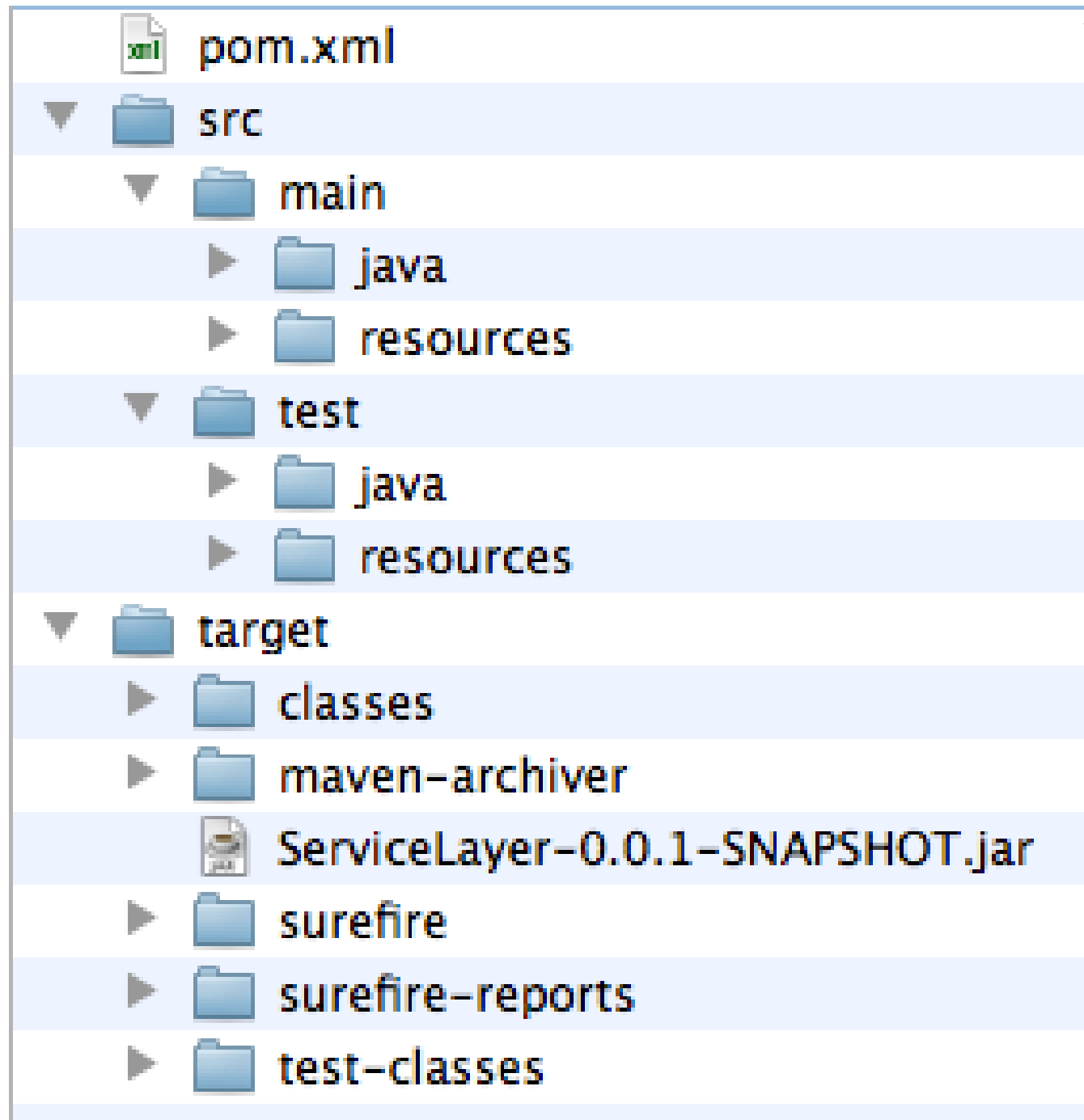
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Identification
unique
de l'artifact

Type de projet
(jar, war, ear,...)

dépendance vers
un autre projet

Structure de projet

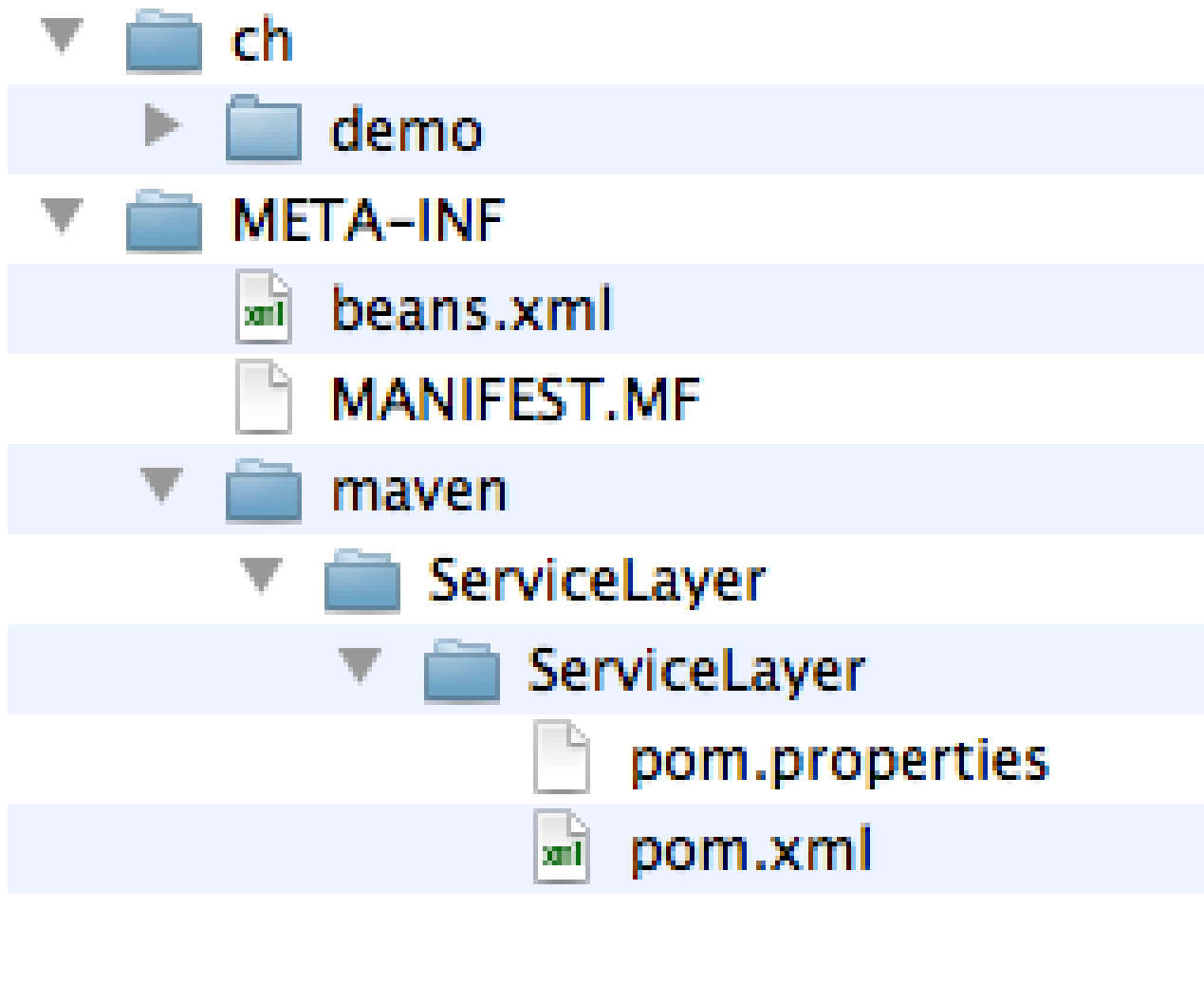


description du projet

Sources du projet

Résultat de compilation

Artifact résultat



Classe du projet

Meta-information

Information de dépendances
permettant la résolution
transitive



Installation de maven

- Téléchargez maven
- Décompressez
- Ajoutez \$MAVEN_HOME/bin au PATH
- Utilisez la commande “mvn -version”

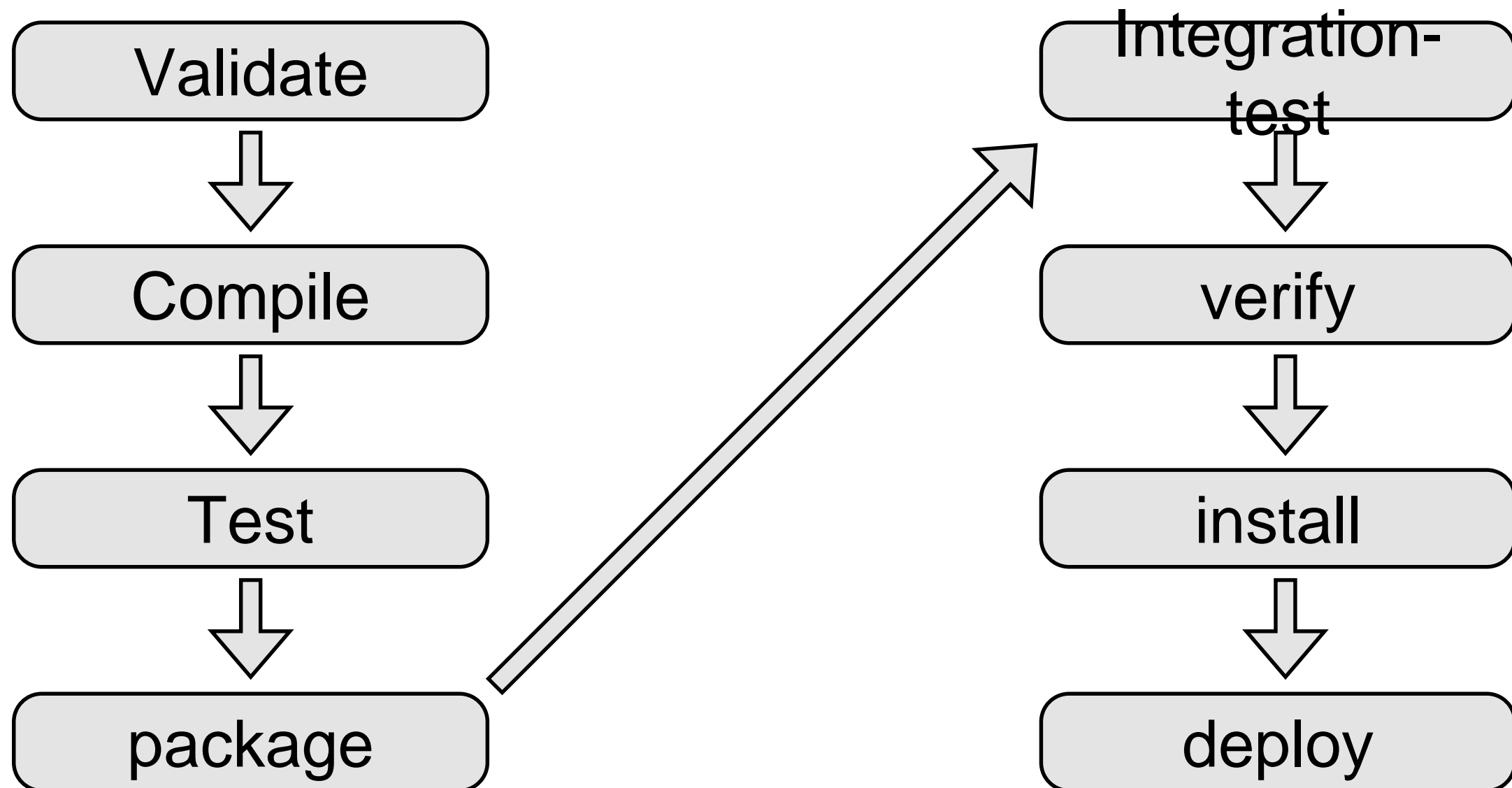




Creation et inspection d'un projet maven

```
mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-webapp  
-DarchetypeArtifactId=maven-archetype-webapp
```

Cycle de vie par défaut (7 phases principales)



“mvn deploy” execute toutes les phases jusqu’à deploy



Compiler et tester un projet

Exécutez `"mvn compile"` et constatez la compilation

Exécutez `"mvn test"` et vérifiez le résultat des tests

Exécutez `"mvn deploy"` et trouvez le package résultat

Cycle de vie par défaut

- 7 phases principales
- 21 phases en tout
- Chaque phase est associée a un ou plusieurs buts (goals) provenant de différents plugins
- Un plugin fournit un service/but (goal)
 - `compiler:compile`, `compiler:testCompile`
 - `jar:jar`
 - `surefire:test`



Numéro de versions

<major>.<minor>[.<micro>][-<qualifier>][-<buildnumber>]

Changement majeur.
Pas de rétro-compatibilité.



Numéro de versions

<major>.<minor>[.<micro>][-<qualifier>][-<buildnumber>]

Nouvelles fonctionnalités.
Garantie de rétro-
compatibilité.



Numéro de versions

<major>.<minor>[.<micro>][-<qualifier>[-<buildnumber>]]



Bug Fixes



Numéro de versions

<major>.<minor>[.<micro>][-<qualifier>[-<buildnumber>]]

SNAPSHOT: Evolution (dernière sources)

alpha: instable et incomplète

beta: instable

rc: release candidate

m: milestones

Spécification de versions

() : exclusive

[] : inclusive

, : choix

Domaine	Signification
(,1.0]	≤ 1.0
[1.2,1.3]	$1.2 \leq \dots \leq 1.3$
[1.5 , 2.0)	$1.2 \leq \dots < 2.0$
[1.5,)	≥ 1.5
(,1.1),(1.1,)	$\neq 1.1$
SNAPSHOT	Version la plus récente



Gestion des dépendances

```
<project xmlns="..." >
```

```
...
```

```
<dependencies>
```

```
<dependency>
```

```
<groupId>junit</groupId>
```

```
<artifactId>junit</artifactId>
```

```
<version>4.0</version>
```

```
<scope>test</scope>
```

```
</dependency>
```

```
</dependencies>
```

```
...
```

```
</project>
```

Groupe d'artefacts

Identifiant d'artefacts

Version

Visibilité:

Compile
test
provided
runtime

Visibilité

- compile
 - transitive
 - disponible pour tous les classpaths
- test
 - classpaths de compilation de tests et d'exécution de test
- runtime
 - classpath d'exécution standard et test
- provided
 - compilation et test mais le jar est fournit par le conteneur





Rajouter une nouvelle dépendance

Ajoutez au projet de Demo un dépendances vers $\log_4 J$

Modularité

- Maven permet de modulariser un projet et réutiliser les configurations communes
- Super POM qui contient la liste des modules
- Chaque module à une référence vers le Super POM



Modularité: projet parent

```
<project ...>
  <modelVersion>4.0.0</modelVersion>
  <groupId>JEE6-Demo</groupId>
  <artifactId>JEE6-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>JEE6-Demo</name>

  <modules>
    <module>../PresentationLayer</module>
    <module>../ServiceLayer</module>
  </modules>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        ...
      </plugin>
    </plugins>
  </build>
</project>
```

Packaging:
pom indique qu'il
s'agit d'un projet parent

Liste des modules
de ce projet

Configuration qui sera
héritée par les modules

Modularité: module

```
<project ...>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>JEE6-Demo</groupId>
    <artifactId>JEE6-Demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <relativePath>../JEE6-Demo/pom.xml</relativePath>
  </parent>

  <artifactId>ServiceLayer</artifactId>
  <packaging>jar</packaging>
  <name>Service Layer</name>
  <url>../../../../ServiceLayer/target/site</url>

  <dependencies>
    <dependency>
      <groupId>org.jboss.weld.se</groupId>
      <artifactId>weld-se</artifactId>
      <version>1.1.5.Final</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
</project>
```

Déclaration
du parent

Identification
du module

Configuration
spécifique du module



Misc

- Edition des rapports de tests et de qualité du code
- Création automatique d'un site internet sur le developement du projet
- Intégration continue facilité
- Intégration avec la gestion des défauts (bugzilla)
- Tag automatique de version





Etude de l'architecture du projet de démonstration

Téléchargez la dernière version du projet de démonstration à <http://code.google.com/p/jee6-demo/>



Bibliographie

<http://maven-guide-fr.erwan-alliaume.com/>

<http://maven.apache.org/apache-maven.pdf>



Intégration continue

- Détection des problème d'intégration très tôt
 - Plus facile
- Détection rapide des regressions
- Il y a toujours une version de disponible pour les démonstrations et les tests utilisateurs
- Détection rapide de code incompatible ou manquant
- Calcul constant de métriques sur le code



Principes

- Dépôt de sources versionnées
- Automatiser la compilation
- Automatiser les tests / auto-tests
- “Commit” fréquents
- Tous “commit” doit compiler
- Temps de compilation court (et donc incrémental)
- Accès facile aux livrables
- Tout le monde voit ce qui se passe (transparence)
- Automatisation du déploiement



Désavantages

- Setup initial couteux en temps
- La qualité de la détection des problèmes dépend fortement de la qualité des tests automatiques
- Nécessite une machine puissante (compilation, tests)

