

# INTÉGRATION CONTINUE

OUTILS , BONNES PRATIQUES ET QUALITÉ LOGICIEL

# What is Continuous Integration ?

2

« L'intégration continue est un ensemble de pratiques utilisées en génie logiciel. Elles consistent à **vérifier** à chaque modification de code source que le résultat des modifications ne produit pas de **régression** de l'application en cours de développement. »

Wikipedia

# 1<sup>ère</sup> Etape : Tester son application

3

- Mettre en place des tests répétables et indépendants
- Comment tester son application ?
  - Tests fonctionnels
  - Tests d'intégration
  - Test unitaires

# Importance des tests

4

- Quelques exemples :
  - Russie – Le système de contrôle aérien confond des satellites avec des missiles balistiques lancés contre eux
  - Le USS Yorktown - une division par 0 coupe les moteurs
  - Mars Orbiter – Explosion de l'engin d'étude sur Mars car le calculateur d'altitude était basé sur une mauvaise unité de mesure
  - Ariane 5 – Détruite 36 secondes après son lancement car le système de bord a été incapable de convertir une donnée

# Intérêts des tests unitaires

5

- Simplicité de débogage
- Développement accéléré
- Amélioration du code
- Contrôle des régressions
- Réduit les coûts futurs

# Test unitaire idéal

6

- Le code testé doit être isolable du reste de l'application
- Le test doit être :
  - Répétable
  - Automatisable
  - Facile d'écriture

# JUnit

7

- Framework de test pour Java
- Simple d'utilisation
- Contrôle de l'exécution d'un code par un mécanisme d'assertions

```
Assert.assertTrue(isDateValide («12/12/2012»)) ;
```

```
Assert.assertFalse(isDateValide («12/13/2012»)) ;
```

# Limites des tests unitaires

8

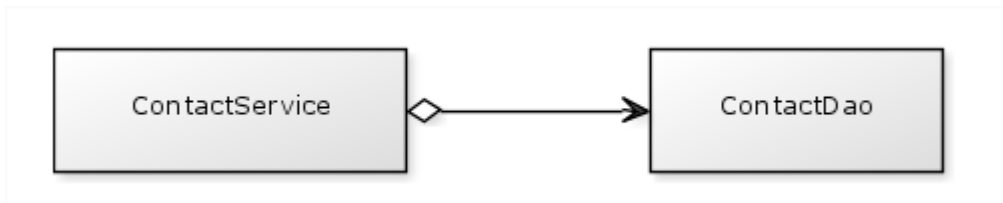
- Le test porte sur un code pré-existant
- Ne teste que des portions de code indépendantes
- Ne garantie pas toujours un fonctionnement correcte de l'application
- Certaines parties de codes ne peuvent pas être rendues indépendantes (Service lié à une base de données)
- Couverture des tests dans le code



# Cas d'usage

9

- Service de gestion de contacts
  - Création d'un contact
  - Récupération d'un contact
  - Suppression d'un contact



# Test Driven Development : Cas d'usage

10

- Je souhaite créer un contact en entrant son nom
- Le nom est obligatoire, unique et doit-être valide (min 3 caractères, max 40 caractères)
- Un message doit m'avertir lors d'une erreur de saisie, ou si le nom est déjà utilisé.

# Test Driven Development : Cas d'usage

11

```
public interface IContactService {  
  
    /**  
     * Méthode qui permet de créer un contact  
     *  
     * @param nom nom du contact a créer  
     * @return Contact créé  
     * @throws ContactException exception lorsque le contact ne peut-être créé (Nom invalide ou Non unique)  
     */  
    Contact creerContact(String nom) throws ContactException;  
}
```

```
@Test(expected = ContactException.class)
public void testCreerContactNomNull() throws ContactException {
    contactService.creerContact(null);
}

@Test(expected = ContactException.class)
public void testCreerContactNomVide() throws ContactException {
    contactService.creerContact("");
}

@Test(expected = ContactException.class)
public void testCreerContactNomTropCourt() throws ContactException {
    contactService.creerContact("AB");
}

@Test(expected = ContactException.class)
public void testCreerContactNomTropLong() throws ContactException {
    contactService
        .creerContact("NomDeContactTropLongQuiFaitPlusDe100CaracteresEtQuiNePeutPasEtreInsererEnBaseDeDonnees");
}

@Test
public void testCreerContactOk() throws ContactException {
    Contact contactCree = contactService.creerContact("Abitbol");
    Assert.assertNotNull(contactCree);
    Assert.assertEquals("Abitbol", contactCree.getNom());
}
```

# Test Driven Development : Cas d'usage

13

```
public class ContactServiceImpl implements IContactService {  
  
    @Override  
    public Contact creerContact(String nom) throws ContactException {  
        // TODO Auto-generated method stub  
        return null;  
    }  
}
```

Runs: 5/5

Errors: 0

Failures: 5

- fr.annuaire.service.ContactServiceTest [Runner: JUnit 4] (0,025 s)
  - testCreerContactNomNull (0,015 s)
  - testCreerContactNomVide (0,001 s)
  - testCreerContactNomTropCourt (0,001 s)
  - testCreerContactNomTropLong (0,001 s)
  - testCreerContactOk (0,001 s)

# Test Driven Development : Cas d'usage

14

```
@Override
public Contact creerContact(String nom) throws ContactException {
    if (nom == null || nom.length() == 0) {
        throw new ContactException();
    }

    return null;
}
```

Runs: 5/5

Errors: 0

Failures: 3

fr.annuaire.service.ContactServiceTest [Runner: JUnit 4] (0,016 s)

- testCreerContactNomNull (0,000 s)
- testCreerContactNomVide (0,000 s)
- testCreerContactNomTropCourt (0,010 s)
- testCreerContactNomTropLong (0,001 s)
- testCreerContactOk (0,000 s)

# Test Driven Development : Cas d'usage

15


```
private List<Contact> contacts = new ArrayList<Contact>();

@Override
public Contact creerContact(String nom) throws ContactException {
    if (nom == null || nom.length() < 3 || nom.length() > 40) {
        throw new ContactException();
    }
    // Enregistrement du contact ...
    Contact contact = new Contact();
    contact.setNom(nom);
    contacts.add(contact);
    return contact;
}
```

Runs: 5/5

✖ Errors: 0

✖ Failures: 0

▶  fr.annuaire.service.ContactServiceTest [Runner: JUnit 4] (0,000 s)

# Test Driven Development : Cas d'usage

16

```
@Test(expected = ContactException.class)
public void testCreerContactDoublon() throws ContactException {
    Contact contactCree = contactService.creerContact("Abitbol");
    Assert.assertNotNull(contactCree);
    Assert.assertEquals("Abitbol", contactCree.getNom());
    contactService.creerContact("Abitbol");
}
```

```
@Override
public Contact creerContact(String nom) throws ContactException {
    if (nom == null || nom.length() < 3 || nom.length() > 40) {
        throw new ContactException();
    }
    // Enregistrement du contact ...
    Contact contact = new Contact();
    contact.setNom(nom);
    if (contacts.contains(contact)) {
        // Le contact existe déjà
        throw new ContactException();
    }
    contacts.add(contact);
    return contact;
}
```



# Réponses à ces limites

17

- Test d'intégration : Garantir fonctionnement de l'application
- Mocks : Isoler une portion du code à tester en fournissant un objet simulé
- Code Coverage : Identifie les portions de codes non couvertes par un test

# Test intégration

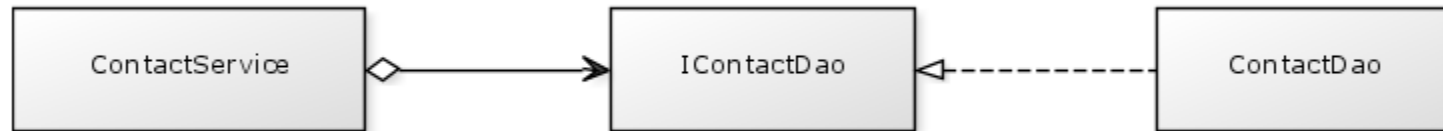
18

- Test consistant à contrôler une fonctionnalité
  - Utilisation de base de données
  - Utilisation de webservices
  - Imports/Exports de fichiers
  - Utilisation de librairie
  - Regroupement de briques unitaires

# Mocks

19

- Remplacer un comportement non déterministe
- Permettre de tester un code en spécifiant le comportement des classes utilisées
- Nécessité de créer des interfaces



# Mocks : Types de mocks

20

- Dummy : objets "vides" qui n'ont pas de fonctionnalités implémentées.
- Stub : classes qui renvoient en dur une valeur pour une méthode invoquée
- Fake : classes qui sont une implémentation partielle et qui, par exemple, renvoient toujours les mêmes réponses selon les paramètres fournis
- Spy : classe qui vérifie l'utilisation qui en est faite après l'exécution
- Mock : classes qui agissent comme un stub et un spy

# Mocks : Exemples

21

```
@Override
public Contact creerContact(String nom, String prenom, String email, String telephone) throws ContactException {
    // Contrôle de nullité
    if (StringUtils.isBlank(nom) || StringUtils.isBlank(prenom)) {
        throw new IllegalArgumentException(
            "Les champs nécessaires ne sont pas rempli (nom et prénom sont obligatoires)");
    }
    Contact contact = contactDao.rechercherContact(nom, prenom);
    if (contact != null) {
        throw new ContactException("Le contact existe déjà en base de donnée");
    } else {
        contact = new Contact();
        contact.setNom(nom);
        contact.setPrenom(prenom);
        contact.setEmail(email);
        contact.setTelephone(telephone);
        Long id = contactDao.create(contact);
        contact.setId(id);
    }
    return contact;
}
```

# Mocks : Exemples

22

```
@Test(expected = ContactException.class)
public void testCreateContactDoublon() throws ContactException {

    1 EasyMock.expect(contactDaoMock.rechercherContact("Valjean", "Jean"))
        .andReturn(new Contact()); 2
    mockControl.replay();
    contactService.creerContact("Valjean", "Jean",
        "jean.valjean@gmail.com", "0233554678");
    mockControl.verify();
}
```

- On attend l'appel de la méthode `rechercherContact`
- On retourne un contact pour simuler le cas où la donnée existe déjà en base

# Maven

23

- Maven est un outil java de build
  - Gestion de dépendances
  - Construction multi-module
  - Orienté plugin et évolutif
- Plus besoin de créer un script pour construire votre application

# Maven – Project Object Model

24

- Représente le modèle objet du projet
- Décrit le projet
  - Nom et version
  - Type d'artefact
  - Structure du projet
  - Dépendances
  - Plugins



# Maven – Description du projet

25

- Identification d'un projet pour Maven
  - GroupId : Nom du groupe de projet
  - ArtifactId : Nom du projet
  - Version : Version du projet
    - Format GOROCO
    - Ajout de -SNAPSHOT pour version en cours de développement

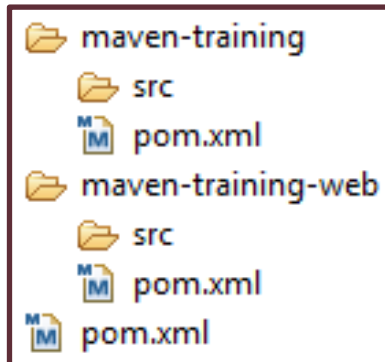
```
<?xml version="1.0" encoding="UTF-8"?>
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.lds.training</groupId>
  <artifactId>maven-training</artifactId>
  <version>1.0</version>
</project>
```

# Maven - Packaging

26

- Maven peut spécifier le type de packaging d'un projet (jar, war, ear, ...)
- Possibilité de créer un projet multi module
  - Le Pom parent est utilisé pour regrouper les modules

```
<project>
  ...
  <packaging>pom</packaging>
  <modules>
    <module>maven-training</module>
    <module>maven-training-web</module>
  </modules>
</project>
```



# Maven – Structure projet

27

- Spécifier les différents dossiers de développement
  - src/main/java: All java source files
  - src/main/webapp: All web source files
  - src/main/resources: All non compiled source files
  - src/test/java: All java test source files
  - src/test/resources: All non compiled test source files

# Maven – Cycle de vie

28

- Le build maven suit l'ordre suivant
  - generate-sources/generate-resources
  - compile
  - test
  - package
  - integration-test (pre and post)
  - install
  - deploy

# Maven – Exemple d'appels

29

- Pour effectuer une action il suffit juste d'appeler l'action voulue
  - mvn install
    - generate\*, compile, test, package, integration-test, install
  - mvn compile
    - generate\*, compile
  - mvn install -DskipTests
    - generate\*, compile, package, install

# Maven – Gestion des dépendances

30

- La description d'une dépendance :
  - Identifiant du projet (GAV)
  - Scope : compile, test, provided
  - Type : jar, pom, war, ear, zip

```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
</project>
```

# Maven – Gestion des dépendances

31

- Les dépendances sont téléchargées sur le repository central
- Les dépendances téléchargées sont mises en cache
  - repository local {Home}/.m2/repository
- Possibilité d'avoir un gestionnaire de dépendance dans le pom afin d'éviter les conflits de versions de librairies
- Maven a révolutionné la gestion des dépendances en Java

# Intégration continue

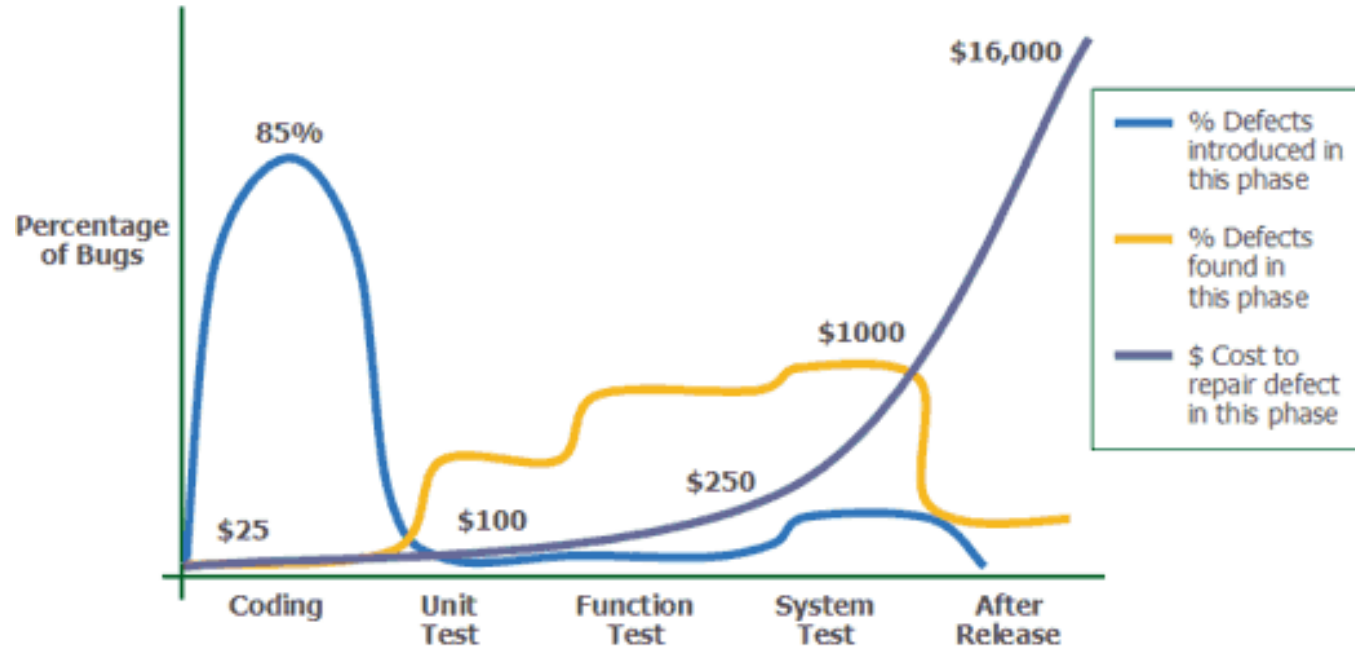
32

- Réduction des risques
  - Découplage de l'environnement de développement
  - Découverte des défauts au plus tôt
  - Qualité accrue grâce aux métriques périodiques
  - Informations sur la santé du projet



# Intégration continue

33



# Intégration continue

34

- L'intégration continue crée le concept de « non-événement »
  - Intégration des différents modules
  - Tests de l'application
  - Inspection du code
  - Déploiement de l'application sur les environnements
  - Livraison de l'application

# Intégration continue – Bonnes pratiques

35

- Automatiser les builds et les tests
- Commit fréquents
- Garder le build de l'application rapide
- Coresponsabilité de l'intégration
- Déploiement automatisé

# Jenkins

36

- Un serveur d'intégration continue Open Source
- Une communauté
- Plus de 500 plugins
- Le CIS le plus utilisé au monde
- Intégration complète avec Maven

# Jenkins – Fonctionnalités

37

- Déclencheur de build
- Récupérer les sources d'un gestionnaire de source
- Automatiser les builds et les tests
- Générer des rapports et notifier
- Déployer le build
- Installer le build sur un serveur de qualification

# Jenkins – Déclencheur de build


38

- Plusieurs déclencheurs sont possibles:
  - Manuel
  - Périodiquement
  - Après le build d'un autre projet
  - Lors d'un changement sur le gestionnaire de source

# Point de parcours

39

- Code testé
  - Unitairement
  - Par fonctionnalité
  - Simulation de cas fonctionnels difficilement reproductible en production
- Code sur Jenkins
- What else ?

A man with dark hair, wearing a grey suit, blue shirt, and yellow tie, is shown from the chest up. He has a thoughtful or slightly skeptical expression, looking off-camera to the left. The background is dark and out of focus.

On dit d'un code testé et installé sur Jenkins qu'il est

♦♦A: De qualité

♦B: Certainement buggé

♦C: Testé

♦D: La réponse D



```

package fr.annuaire.service;

import java.io.BufferedReader;







public class Texte_file_reader {

    private static String data;

    public static List<String> extract_lines(String nomFichier) {
        if (nomFichier == null || !new File(nomFichier).exists()) {
            throw new IllegalArgumentException("Le fichier n'est pas valide");
        }
        ArrayList<String> results = new ArrayList<String>();
        File file = new File(nomFichier);
        try {
            FileReader r = new FileReader(file);
            BufferedReader b = new BufferedReader(r);
            while ((data = b.readLine()) != null) {
                results.add(data);
            }
        } catch (IOException e) {
            System.out.println("bad !");
        }

        return results;
    }
}

```

-  testFileNull (0,000 s)
-  testFileEmpty (0,000 s)
-  testFileIncorrect (0,000 s)
-  testReadEmptyFile (0,000 s)
-  testReadOneLineFile (0,000 s)
-  testReadFourLinesFile (0,002 s)

```

package fr.annuaire.service;

import java.io.BufferedReader;

public class Texte_file_reader {

    private static String data;

    public static List<String> extract_lines(String nomFichier) {
        if (nomFichier == null || !new File(nomFichier).exists()) {
            throw new IllegalArgumentException("Le fichier n'est pas valide");
        }
        ArrayList<String> results = new ArrayList<String>();
        File file = new File(nomFichier);
        try {
            FileReader r = new FileReader(file);
            BufferedReader b = new BufferedReader(r);
            while ((data = b.readLine()) != null) {
                results.add(data);
            }
        } catch (IOException e) {
            System.out.println("bad !");
        }

        return results;
    }
}

```

Commentaires

Règles de nommage

Classe utilitaire non protégée

Utilisation des Sysout

Fermeture des ressources

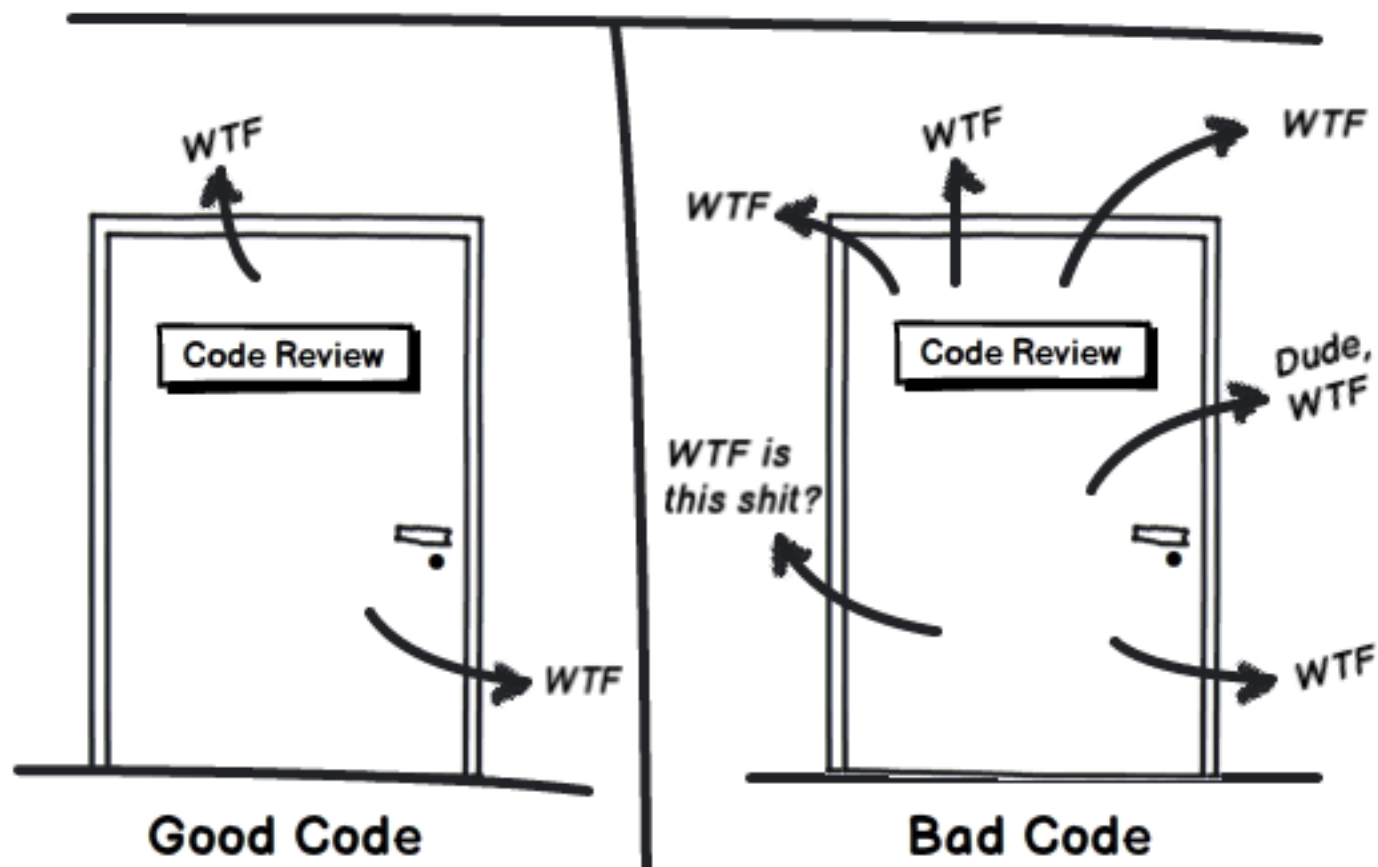
Réentrance

# S'assurer de la qualité d'un code

43

- Les tests de l'application ne sont pas suffisants
  - Faux positif, 100% test passed  $\neq$  quality code
  - Prise en compte de tous les paramètres
- Comment mesurer la qualité d'un code ?
- Quels sont les axes de qualité de code ?

# Code Quality Measurement: WTFs/Minute



# Les axes de la qualité de code

45

- Règles de codage
- Utilisation de pattern (GOF)
- Couverture des tests
- Duplication de code
- Bugs potentiels
- Complexité
- Commentaires

# Inspection du code – CheckStyle





46

- CheckStyle est un outil d'analyse statique du code
  - Cohérence de la syntaxe
  - Meilleures pratiques de développement
  - Basé sur des conventions de codage
    - Standard ou customisée

# Inspection du code – CheckStyle

47

- Exemples d'anomalie remontée par checkstyle
  - Méthodes non commentées
  - Nommage des classes, variables, constantes, méthodes
  - Imports non utilisés
  - Règles de développement

	Missing a Javadoc comment.	45
	Utility classes should not have a public or default constructor.	45
	Missing a Javadoc comment.	47
	Parameter args should be final.	47

# Inspection du code – PMD

48

- PMD est un outils d'analyse statique du code
  - Taille du code
  - Complexité du code
  - Optimisation
  - Code inutilisé

Violation	Line
Avoid empty 'if' statements	572
Avoid empty catch blocks	647
Avoid unnecessary return statements	654



# Inspection du code – Findbugs

49

- FindBugs est un outils de recherche de bug
  - Code qui ne sera jamais exécuté
  - Boucles infinies
  - Détection des potentiels NullPointerException
  - Méthodes sur des types immutables (String, Long, ...)
  - Vulnérabilité du code
  - Performances / Usage de la mémoire

# FindBugs – Quelques exemples

50

## - Eclipse 3.0.0

```
if (in == null){  
    try{  
        in.close();  
        ...  
    }  
}
```

# FindBugs – Quelques exemples

51

- JDK1.6.0, b13
- Class AnnotationTypeMismatchException

```
public String foundType() {  
    return this.foundType()  
}
```

# FindBugs – Quelques exemples

52

- JDK1.6.0, b105 – Class sun.awt.x11.XMLSelection

```
if (listeners == null){  
    listeners.remove(listener)  
}
```

# FindBugs – Quelques exemples

53

```
if (TypeEnumeration.A.equals(type.getCode())) {  
    doSomething();  
}
```

```
If( myDto.getId() != otherObject.getId() ) {  
    doSomething();  
}
```

# Inspection du code – Code Coverage

54

## - Code couvert par les tests

```
public abstract class AbstractDao<T, PK> implements IDao<T, PK> {  
  
    private Map<PK, T> values = new HashMap<PK, T>();  
  
    private PK sequence;  
  
    @Override  
    public T read(PK identifiant) {  
        return BddUtils.deproxy(values.get(identifiant));  
    }  
  
    @Override  
    public PK create(T entite) {  
        if (entite == null || getId(entite) != null) {  
            throw new IllegalStateException("Impossible de créer une entite qui possède un  
        }  
        sequence = getSeqNextVal(sequence);  
        BddUtils.setIdentifier(entite, getIdFieldName(), sequence);  
        values.put(sequence, entite);  
  
        return sequence;  
    }  
}
```

# Code Coverage

55


- Ne contrôle pas la pertinence des tests
- Donne un indicateur sur les branches non testés

Name	Lines	Total	%	Branches	Total	%
▲ All Packages (2014-01-15 00:48:32)	136	288	47,22 %	29	66	43,94 %
▶ fr.ic.contact.dao	0	0	-	0	0	-
▲ fr.ic.contact.dao.exceptions	0	4	0,00 %	0	0	-
G DataBaseException	0	4	0,00 %	0	0	-
▲ fr.ic.contact.dao.impl	6	35	17,14 %	2	26	7,69 %
G AbstractDao	2	20	10,00 %	0	16	0,00 %
G ContactDaoImpl	4	15	26,67 %	2	10	20,00 %
▶ fr.ic.contact.dao.util	0	31	0,00 %	0	8	0,00 %
▶ fr.ic.contact.exception	2	2	100,00 %	0	0	-
▶ fr.ic.contact.model	16	16	100,00 %	0	0	-
▶ fr.ic.contact.service	73	112	65,18 %	0	0	-
▶ fr.ic.contact.service.impl	31	31	100,00 %	20	20	100,00 %
▶ fr.ic.contact.util	0	19	0,00 %	0	2	0,00 %
▶ fr.ic.contact.utils	2	19	10,53 %	6	8	75,00 %
▶ fr.ic.util	6	19	31,58 %	1	2	50,00 %

# Plugins Jenkins

56

- Jenkins permet l'installation de plugins
  - Traitements après builds
  - Consultation de rapports
  - Système de packaging
  - Système de déploiement automatisés

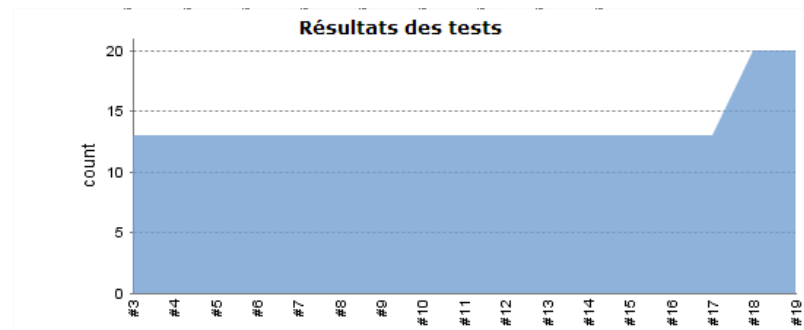
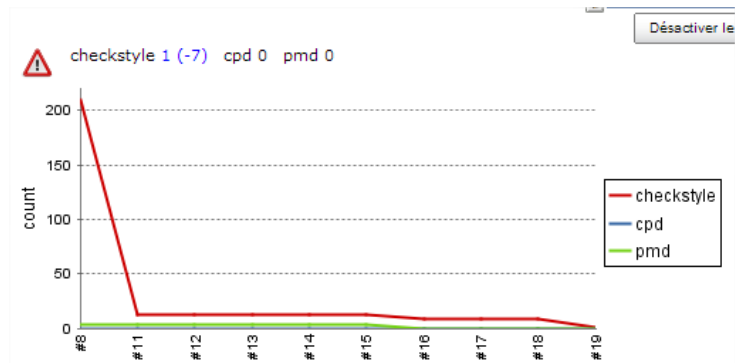
 All arrays Chuck Norris declares are of infinite size, because Chuck Norris knows no bounds.





# Plugins Jenkins

57

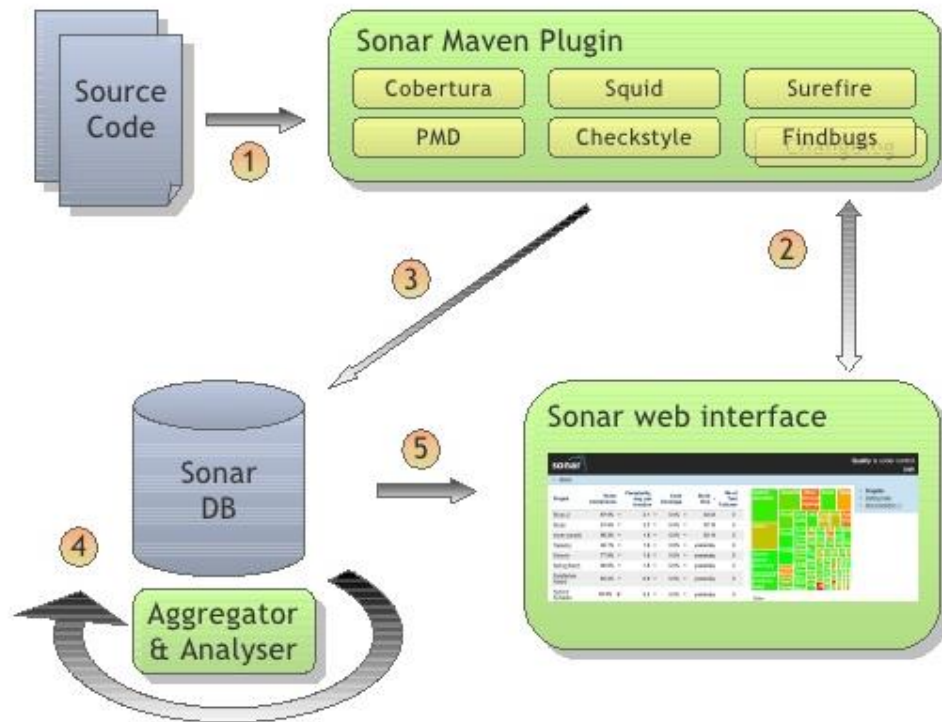


# Sonar

58

- Une plateforme qui gère la qualité du code
- Gratuite et Open Source
- Agrège des outils déjà utilisés sur les projets
  - Analyse du code : CheckStyle, PMD, Findbugs...
  - Redondances de code : CPD, Squid...
  - Couverture de code : Cobertura, JaCoCo...

# Sonar à cœur ouvert



# Sonar : Portail

60

## - Résumé du projet

Lignes de code

**2 672** 📉

4 785 lignes 📉

1 352 instructions 📉

29 fichiers 📉

Classes

**29** 📉

11 packages

200 méthodes 📉

17 accesseurs 📉

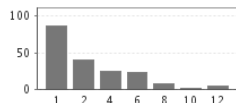
Complexité

**3,0** /méthode

**20,7** /classe 📈

**20,7** /fichier 📈

Total: 600 📉



● Méthodes ● Fichiers

Evénements

Tous ▼

14 fév. 2013

Version

0.0.1-SNAPSHOT

Défauts

**31**

Taux de conformité

**97,0%** 📈



Bloquant

0



Critique

0



Majeur

25



Mineur

6



Info

0

Indice d'interdépendance entre packages

**3,4%**

> 2 cycles

Dépendances à couper

1 entre packages

1 entre fichiers

Couverture de code

**56,8%** 📈

55,0% de couverture de ligne 📈

61,1% de couverture de branche 📈

Succès d'exécution des tests

**100,0%**

0 en échec

0 en erreur












127 tests 📉

6.1 sec 📉

# Sonar : Portail



61

## - Défauts de l'application (PMD, Findbugs)

Sévérité			Règle		
	<a href="#">Bloquant</a>	0		<a href="#">Visibility Modifier</a>	8 <div><div></div></div>
	<a href="#">Critique</a>	0		<a href="#">Cyclomatic Complexity</a>	3 <div><div></div></div>
	<a href="#">Majeur</a>	25 <div><div></div></div>		<a href="#">Loose coupling</a>	2 <div><div></div></div>
	<a href="#">Mineur</a>	6 <div><div></div></div>		<a href="#">Avoid cycle between java packages</a>	1 <div><div></div></div>
	<a href="#">Info</a>	0		<a href="#">Hide Utility Class Constructor</a>	1 <div><div></div></div>
				<a href="#">Boolean Expression Complexity</a>	1 <div><div></div></div>

261

```
private void loadClass(String tmp, Class<?> unC) {
```

 [Cyclomatic Complexity](#) | Ouvert | 12 mois 

La complexité cyclomatique de la classe est de 13 alors que le maximum autorisé est de 10.

[Commenter](#) | [Affecter \[à moi\]](#) | [Planifier](#) | [Confirmer](#) [Plus d'actions](#) ▼

## Règles les plus enfreintes

Toute sévérité ▼

[En savoir plus](#)

▲ <a href="#">Constructor Calls Overridable Method</a>	9	<div></div>
▲ <a href="#">Visibility Modifier</a>	8	<div></div>
▲ <a href="#">Cyclomatic Complexity</a>	3	<div></div>
▼ <a href="#">Collapsible If Statements</a>	3	<div></div>
▲ <a href="#">Loose coupling</a>	2	<div></div>

## Les moins respectueux des règles

[En savoir plus](#)

<a href="#">ControleAccessiWeb</a>	▲0	▲0	▲6	▼0	▼0
<a href="#">LangageChooser</a>	▲0	▲0	▲4	▼0	▼0
<a href="#">Engine</a>	▲0	▲0	▲3	▼2	▼0
<a href="#">View</a>	▲0	▲0	▲3	▼1	▼0
<a href="#">QuestionMessageBox</a>	▲0	▲0	▲3	▼0	▼0

## Points chauds par Durée des tests unitaires

[En savoir plus](#)

<a href="#">Images_1_2Test</a>	2 sec	<div></div>
<a href="#">Images_1_1Test</a>	639 ms	<div></div>
<a href="#">Images_1_3Test</a>	628 ms	<div></div>
<a href="#">structure_9_2Test</a>	540 ms	<div></div>
<a href="#">Tableaux_5_6Test</a>	344 ms	<div></div>

## Points chauds par Lignes non couvertes (TU)

[En savoir plus](#)

<a href="#">Engine</a>	322	<div></div>
<a href="#">View</a>	128	<div></div>
<a href="#">Config</a>	34	<div></div>
<a href="#">Domxpath</a>	33	<div></div>
<a href="#">Controler</a>	27	<div></div>

## Points chauds par Complexité

[En savoir plus](#)

<a href="#">Engine</a>	109	<div></div>
<a href="#">Structure_9_2</a>	55	<div></div>
<a href="#">AskImagesType</a>	54	<div></div>
<a href="#">Images_1_1</a>	43	<div></div>
<a href="#">Lien_6_1</a>	33	<div></div>

## Points chauds par Complexité /méthode

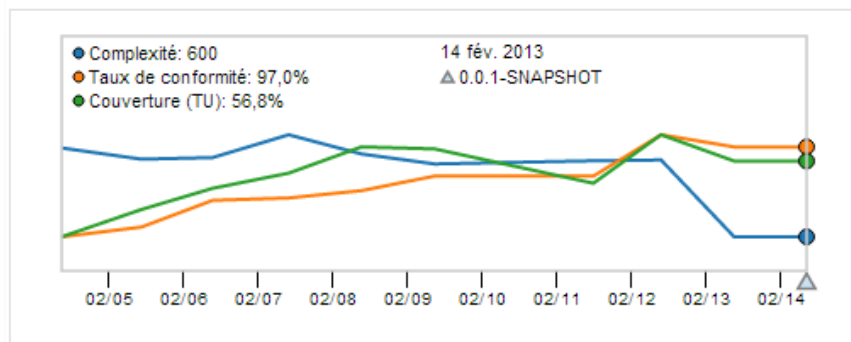
[En savoir plus](#)

<a href="#">Structure_9_2</a>	6,9	<div></div>
<a href="#">Elements_Obligatoires_8_5</a>	6,0	<div></div>
<a href="#">Lien_6_2</a>	5,0	<div></div>
<a href="#">Tableaux_5_4</a>	4,7	<div></div>
<a href="#">Tableaux_5_1</a>	4,7	<div></div>

# Sonar : Portail

63

## - Un suivi des tendances



	03 fév. 2011	11 juil. 2013 1.4.0-SNAPSHOT	14 jan. 2014 2.0.0-SNAPSHOT	
Lines of code	33 433	47 705	46 371	
Lines	52 373	72 788	71 106	
Statements	12 256	15 844	15 377	
Files	344	448	450	
Classes	359	467	469	
Methods	1 832	2 609	2 564	
Accessors	811	1 196	1 138	

	03 fév. 2011	11 juil. 2013 1.4.0-SNAPSHOT	14 jan. 2014 2.0.0-SNAPSHOT	
Comments (%)	19,3%	17,5%	17,9%	
Comment lines	8 004	10 100	10 085	
Public documented API (%)	28,5%	27,9%	28,5%	
Public undocumented API	1 218	1 735	1 712	

# Vers le déploiement continu

64

- Pouvoir définir une release candidate
- Installation automatisée en production
- Possibilité de faire marche arrière
- Green server / Blue server