

Test et Maintenance de Logiciel

Chapitre 5

Métriques et outils de test

Ileana Ober

Maître de conférences

Université Paul Sabatier

IRIT

<http://www.irit.fr/~Ileana.Ober/>



Année Universitaire 2012-2013

©Ileana Ober

Plan du cours

- ❖ Métriques
- ❖ Outils de test
- ❖ Organisation et documentation

Difficulté majeure du test

- ❖ On ne peut pas tout tester
- ❖ Difficile de décider à quel moment arrêter le test
- ❖ Le test de certaines fonctionnalité n'est pas automatisable

Quand arrêter le test

Quand arrêter le test

- ❖ Plus le temps

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts
- ❖ On ne trouve plus de nouveau cas de test

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts
- ❖ On ne trouve plus de nouveau cas de test
- ❖ Quand l'objectif de couverture a été atteint

Quand arrêter le test

- ✿ Plus le temps
- ✿ Plus de ressource
- ✿ On ne trouve plus de nouveau défauts
- ✿ On ne trouve plus de nouveau cas de test
- ✿ Quand l'objectif de couverture a été atteint
- ✿ Quand il n'y a plus de faute

Quand arrêter le test

- ✿ Plus le temps
- ✿ Plus de ressource
- ✿ On ne trouve plus de nouveau défauts
- ✿ On ne trouve plus de nouveau cas de test
- ✿ Quand l'objectif de couverture a été atteint
- ✿ Quand il n'y a plus de faute

Paul C Jorgensen

©Ileana Ober, 2012

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts
- ❖ On ne trouve plus de nouveau cas de test
- ❖ Quand l'objectif de couverture a été atteint
- ❖ Quand il n'y a plus de faute

Paul C Jorgensen

©Ileana Ober, 2012

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts
- ❖ On ne trouve plus de nouveau cas de test
- ❖ Quand l'objectif de couverture a été atteint
- ❖ Quand il n'y a plus de faute

Paul C Jorgensen

©Ileana Ober, 2012

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts
- ❖ On ne trouve plus de nouveau cas de test
- ❖ Quand l'objectif de couverture a été atteint
- ❖ Quand il n'y a plus de faute

Paul C Jorgensen

©Ileana Ober, 2012

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts
- ❖ On ne trouve plus de nouveau cas de test
- ❖ Quand l'objectif de couverture a été atteint
- ❖ Quand il n'y a plus de faute

Paul C Jorgensen

©Ileana Ober, 2012

Quand arrêter le test

- ❖ Plus le temps
- ❖ Plus de ressource
- ❖ On ne trouve plus de nouveau défauts
- ❖ On ne trouve plus de nouveau cas de test
- ❖ Quand l'objectif de couverture a été atteint
- ❖ Quand il n'y a plus de faute

Paul C Jorgensen

©Ileana Ober, 2012

Quand arrêter le test

- Plus le temps
 - Plus de ressource
 - On ne trouve plus de nouveau défauts
 - On ne trouve plus de nouveau cas de test
 - Quand l'objectif de couverture a été atteint
 - Quand il n'y a plus de faute

Autant des choses à mesurer...

Paul C Jorgensen

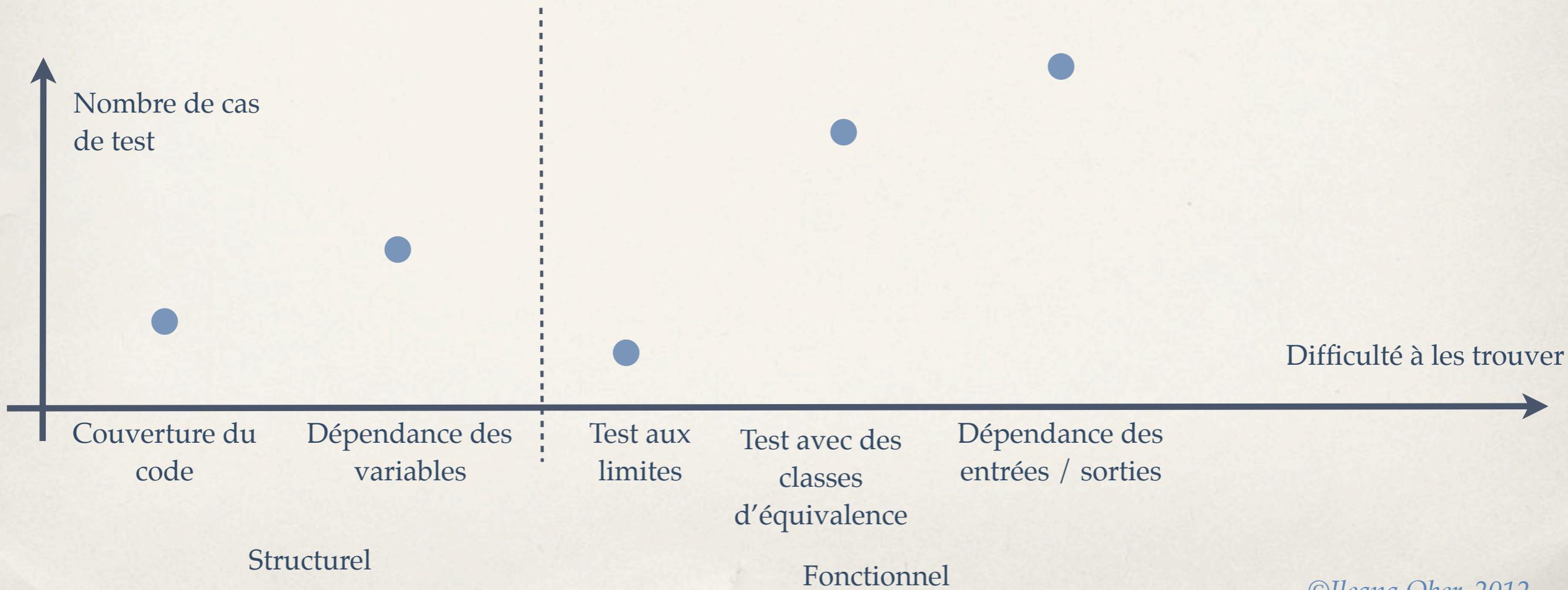
©Ileana Ober, 2012

Critère d'arrêt

- ✿ Basé sur
 - ✿ effort de test
 - ✿ efficacité du test
 - ✿ efficience du test
- ✿ Des **métriques** aident à obtenir des estimations numériques
- ✿ Problème multidimensionnel

Effort pour tester

- ✿ Les techniques de test varient par rapport aux
 - ✿ Nombre de cas de test générés
 - ✿ Difficulté à les trouver



Efficacité et efficience

- ❖ Efficacité - effectuer le test le plus vite possible et avec le minimum d'effort
 - ❖ le rapport entre le travail effectué et l'énergie dépensée par unité de temps
 - ❖ typiquement: nombre de cas de test effectués par jour
- ❖ Efficience - faire du bon travail
 - ❖ dans quelle mesure le programme testé satisfait les spécifications
 - ❖ typiquement: trouver le maximum de défauts

Efficacité du test

- ❖ Intuitivement : le nombre de tests qui permettent de trouver le maximum d'erreurs dans un minimum de temps
- ❖ Il faut trouver le bon compromis entre
 - ❖ laisser des parties non testées
 - ❖ effectuer des tests redondants
- ❖ Difficile à quantifier

Efficacité du test

- ❖ Test fonctionnel
 - ❖ Annoter chaque cas de test avec “objectif de test” (pourquoi on l’a ajouté”)
 - ❖ Si plusieurs test ont le même objectif, vérifier s’ils ne sont pas redondants
- ❖ Test structurel
 - ❖ Si le même chemin est parcouru par plusieurs chemins vérifier s’il n’y a pas de redondance (surtout au niveau des PLCS)
- ❖ Trouver des manques
 - ❖ Beaucoup plus difficile
 - ❖ Pour éviter les manques il faut accepter les redondances
 - ❖ Particulièrement important pour les logiciels critiques

Efficiency du test

- ⊕ Nous donne une mesure de l'efficiency dans l'identification des défauts:

$$\frac{N}{N+S}$$

N - Nombre de défauts identifiés

S - Nombre d'erreurs non identifiés (impossible à savoir)

- ⊕ Difficile à estimer car on ne connaît pas le nombre défauts
- ⊕ En pratique
 - ⊕ Identifier le type et la fréquence des bogues pendant le développement
 - ⊕ Faire des analyses de la décroissance du nombre d'erreurs

Efficiency du test

- ⊕ Nous donne une mesure de l'efficiency dans l'identification des défauts:

$$\frac{N}{N+S}$$

N - Nombre de défauts identifiés

S - Nombre d'erreurs non identifiés (impossible à savoir)

- ⊕ Difficile à estimer car on ne connaît pas le nombre défauts
- ⊕ En pratique
 - ⊕ Identifier le type et la fréquence des bogues pendant le développement
 - ⊕ Faire des analyses de la décroissance du nombre d'erreurs
 - ⊕ Typiquement 85% en pre-release et 15% en maintenance

Métriques

- ❖ Liées au test
- ❖ Liées au logiciel développée (en lien avec la testabilité)

Exemples de métriques liées au test

- ❖ Nombre d'exigences à tester
 - ❖ par fonctionnalité
 - ❖ par priorité
- ❖ Nombre de cas de test
 - ❖ créés
 - ❖ exécutés
 - ❖ passés
 - ❖ échoués
- ❖ Nombre de défauts
 - ❖ par sévérité (dérangeant, critique, catastrophique, ...)
 - ❖ par état (récurrent,
- ❖ En pratique plusieurs métriques utilisées ensemble

Métriques de test

- ❖ % de cas de test exécutés
 - ✿ No de cas de test **passés** / No de cas de test à exécuter
 - ✿ Calculé par composant, priorité, testeur
 - ✿ Donne un indice sur la maturité du développement
- ❖ % couverture de test
 - ✿ No de cas de test **exécutés** / No de cas de test à exécuter
 - ✿ Calculé par composant, priorité
 - ✿ Donne un indice sur l'état d'avancement du test
- ❖ % tests passés
 - ✿ No de cas de test **passés** / No de cas de test **exécutés**
 - ✿ Calculé par composant, équipe
 - ✿ Donne un indice sur l'état d'avancement du test
- ❖ % défaillances à la première exécution
 - ✿ No de cas de test **échoués au premier passage** / No de cas de test **exécutés**
 - ✿ Calculé par composant, équipe
 - ✿ Donne un indice sur l'état de "propreté" du code à l'issue du développement

Métriques de test (suite)

- ❖ **% de tests échoués**
 - ❖ No de cas de test échoués / No de cas de test exécutés
 - ❖ Calculé par composant, priorité, testeur
 - ❖ Donne un indice sur la qualité de l'application et est utilisé pour l'historique
- ❖ **% défauts corrigés**
 - ❖ No de défauts corrigés / No de défauts identifiés
 - ❖ Calculé par composant, priorité, équipe
 - ❖ Donne un indice sur la prise en compte des retours du test
- ❖ **% de mauvaises corrections**
 - ❖ (No de cas de test échoués - No de cas de test échoués au premier passage) / No de cas de test échoués au premier passage
 - ❖ Calculé par composant, priorité, équipe
 - ❖ Donne un indice sur la qualité des corrections
- ❖ **Ratio découverte de défaut**
 - ❖ No défauts trouvés / No de journées de travail
 - ❖ Calculé par composant, équipe, itération
 - ❖ Donne un indice sur la qualité du développement

Exemple

Test Metrics	
Metric	Value
% Complete	90.2%
% Test Coverage	95.0%
% TCs Passed	94.9%
% 1st Run Failures	10.3%
% Failures	11.1%
% Defects Corrected	59.2%
% Rework	10.0%
% Bad Fixes	18.0%
Defect Discovery Rate	0.52
% Test Effectiveness	98.7%

Métriques en lien avec le logiciel

- ✿ Les métriques qui caractérisent le code peuvent être corrélés avec la testabilité
- ✿ Métriques générales
- ✿ Métriques OO

Métriques OO

- ✿ nombre de méthodes / fonctions / procédures par classe
- ✿ couplage avec d'autres classes (en termes d'utilisation)
- ✿ profondeur de la hiérarchie d'héritage
- ✿ largeur de l'arbre d'héritage
- ✿ ...

NNOM, NNOF, NNOP

- ✿ **NNOM, NNOF, NNOP :**
Nominal Numbers Of Methods, Functions and Procedures per class
- ✿ **NNOM, NNOF, NNOP** = respectivement nombres de méthodes, fonctions et procédures de la classe (sauf celles qui sont héritées)
 $NNOM = NNOF + NNOP$
- ✿ En général: max. suggéré pour **NNOM** : 20

TNOM, TNOF, TNOP

- ❖ TNOM, TNOF, TNOP : Total Number Of Methods, Functions and Procedures per class
- ❖ Comme NNOM, NNOF et NNOP, mais inclut les méthodes, fonctions et procédures héritées $TNOM = TNOF + TNOP$

CBO

- ❖ CBO : Coupling Between Objects
- ❖ nombre de liens avec d'autres classes, *à l'exception* des liens d'héritage
- ❖ e.g. dans une classe A, avoir un attribut de type b:B (une autre classe)
- ❖ CBO élevé
 - ⇒ mise au point et test longs et difficiles, beaucoup d'éléments externes, chaque interface doit être testée
 - ⇒ difficile à modifier
 - ⇒ nombre de défauts potentiels élevé
 - ⇒ tester davantage
- ❖ CBO élevé - nuit à une conception modulaire, empêche la réutilisation

DIT

- ✿ DIT : Depth of Inheritance Tree - profondeur de la classe dans son arbre d'héritage, i.e. distance à la racine (profondeur maximum, dans le cas d'héritage multiple)
- ✿ mesure du nombre d'ancêtres pouvant affecter la classe
- ✿ En pratique:
 - ✿ DIT dépend de l'environnement
 - ✿ hiérarchies d'héritage relativement plates DIT max. faible (≤ 10) volontaire, pour meilleure compréhensibilité de l'architecture ?
 - ✿ beaucoup de classes à DIT faible : volontaire, pour meilleure compréhensibilité ?

Autres exemples en vrac

- ❖ NOC : Number Of Children
- ❖ NOR : Number Of Root classes
- ❖ LCOM : Lack of COhesion in Methods
- ❖ PAP : percent Public And Protected
- ❖ PAD : Public Access to Data members
- ❖ OVR : percent of non-OVeRloaded calls
- ❖ DYN : percent of DYNamic calls

Intérêt des métriques

- ⊕ Aider à découvrir des défauts ou des violations de règles de conception (DIT)
- ⊕ Aider à gérer les possibilités de réutilisation (DIT, NOC)
- ⊕ Aider à allouer les ressources pour le test (CBO, RFC)
- ⊕ Aider à cibler les tests (CBO, RFC)
- ⊕ Contrôler l'évolution d'une application OO

Plan du cours

- ❖ Métriques
- ❖ Outils de test
- ❖ Organisation et documentation

Outils d'automatisation du test

- ✿ le test est une activité consommatrice en ressources
- ✿ l'aide des outils est essentielle (environ 300 outils recensées)
- ✿ différents sites web proposent des listes d'outils de test
(e.g. <http://www.aptest.com/resources.html>)
- ✿ Catégories d'outils:
 - ✿ L'exécution des tests
 - ✿ La gestion des campagnes
 - ✿ Le test de performance
 - ✿ La génération de tests fonctionnels
 - ✿ La génération de tests structurels

Outils de test : Cantata

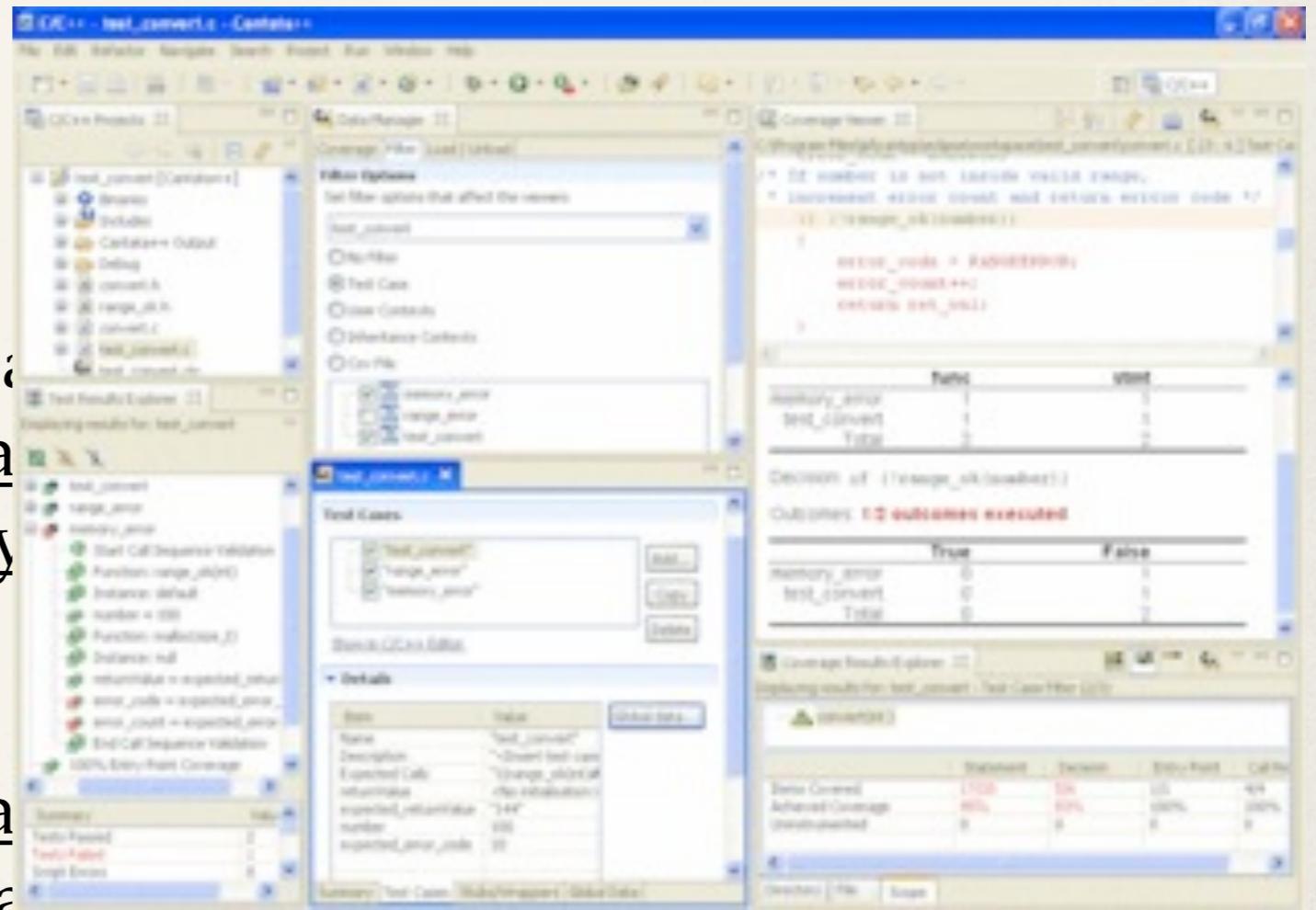
test dynamique

- ❖ Test unitaire et d'intégration Cantata ++
[http://www.youtube.com/watch?
v=bs7XzoG5YLo&feature=player_embedded](http://www.youtube.com/watch?v=bs7XzoG5YLo&feature=player_embedded)
- ❖ Test de robustesse Cantata ++
[http://www.youtube.com/watch?
v=SOYOxr1DsM4&feature=player_embedded](http://www.youtube.com/watch?v=SOYOxr1DsM4&feature=player_embedded)

Outils de test : Cantata

test dynamique

- Test unitaire et d'intégration C++
http://www.youtube.com/watch?v=bs7XzoG5YLo&feature=player_embedded
- Test de robustesse Cantata ++
http://www.youtube.com/watch?v=SOYOxr1DsM4&feature=player_embedded

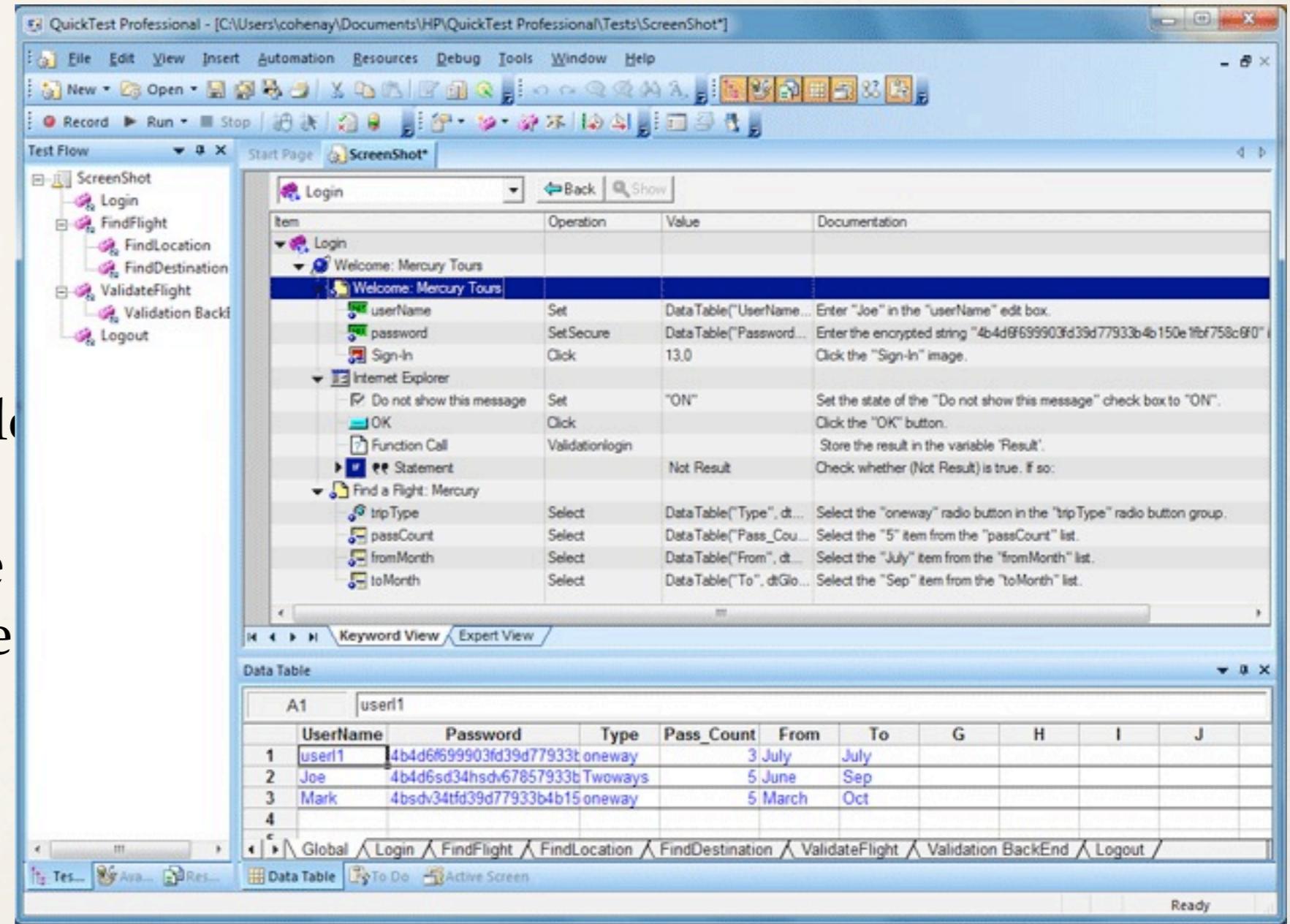


HP QuickTest Professionnel

- ✿ test fonctionnel et de non régression
- ✿ interface graphique propre, mais fonctionnement en ligne de commande possible aussi

HP QuickTest Professional

- ✿ test fonctionnel et de performance
- ✿ interface graphique et commande possible

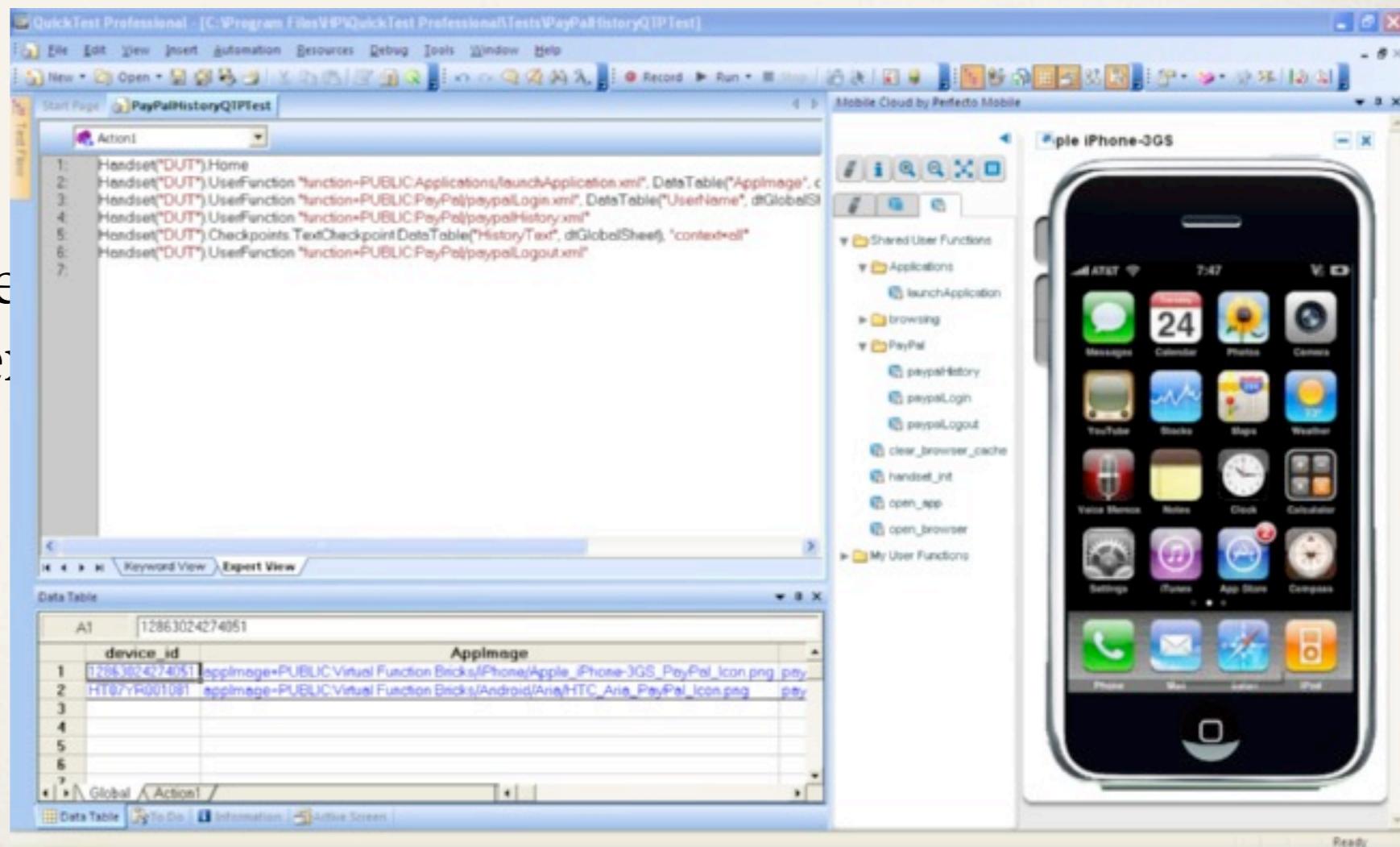


Outils spécifiques

- ✿ Perfecto Mobile intègre la possibilité de faire du test sur les plateformes mobiles (extension HP QuickTest Professionnel)

Outils spécifiques

- Perfecto Mobile intègre plusieurs plateformes mobiles (ex: iPhone)



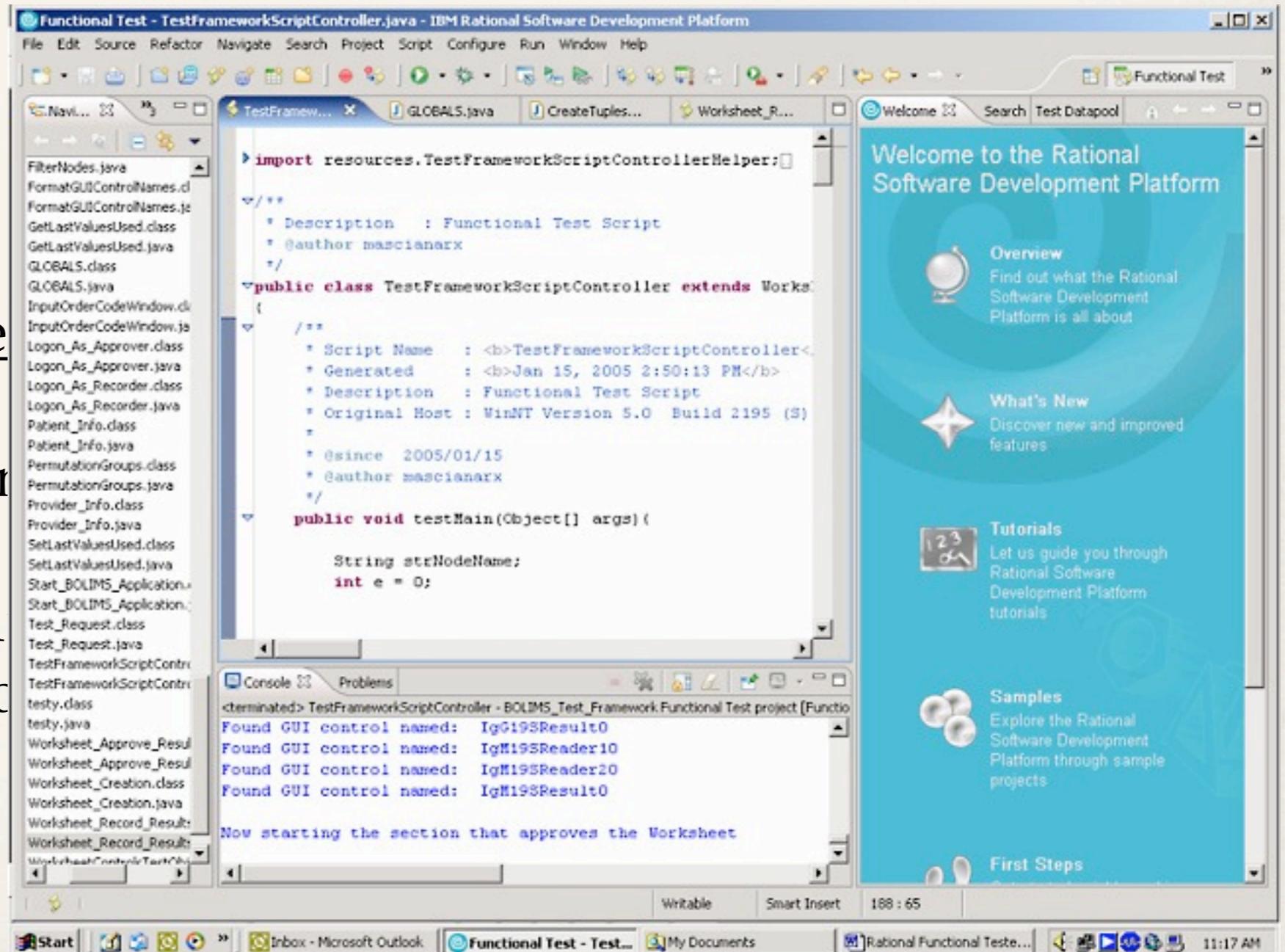
IBM Rational Functional Tester

IBM Rational Functional Tester

- ✿ <http://www.youtube.com/watch?v=j35ox9kFYrw>
- ✿ L'outil permet d'automatiser les activités liées au test
- ✿ permet aux testeurs d'éditer des scripts de test à partir des captures d'écran de l'application à tester

IBM Rational Functional Tester

- ✿ <http://www.youtube.com>
- ✿ L'outil permet d'automatiser les tests
- ✿ permet aux testeurs de tester l'application à l'écran de l'applicati



Selenium

- ✿ outil orienté vers applications web
- ✿ à la base un plug-in FireFox
- ✿ exécution de script de test
- ✿ navigation dans la base de tests
- ✿ dépannage et points d'arrêt
- ✿ <http://seleniumhq.org/>

Selenium

- * outil ori

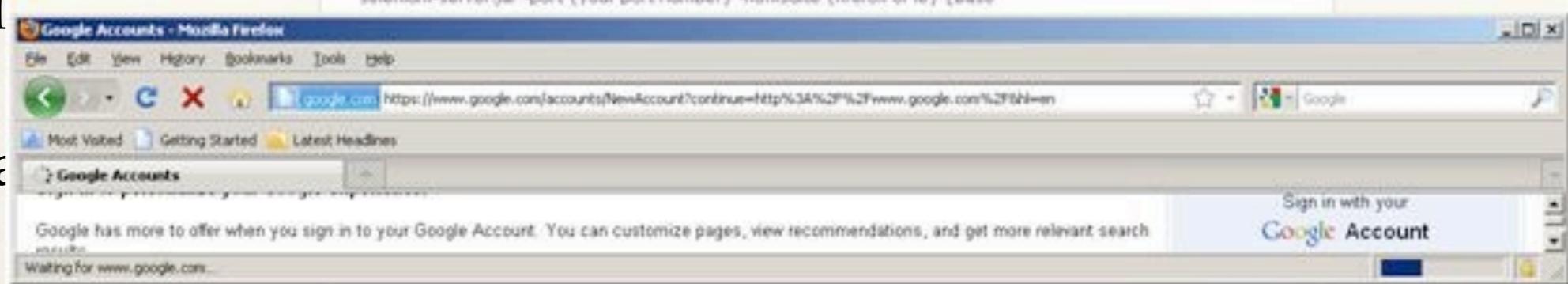
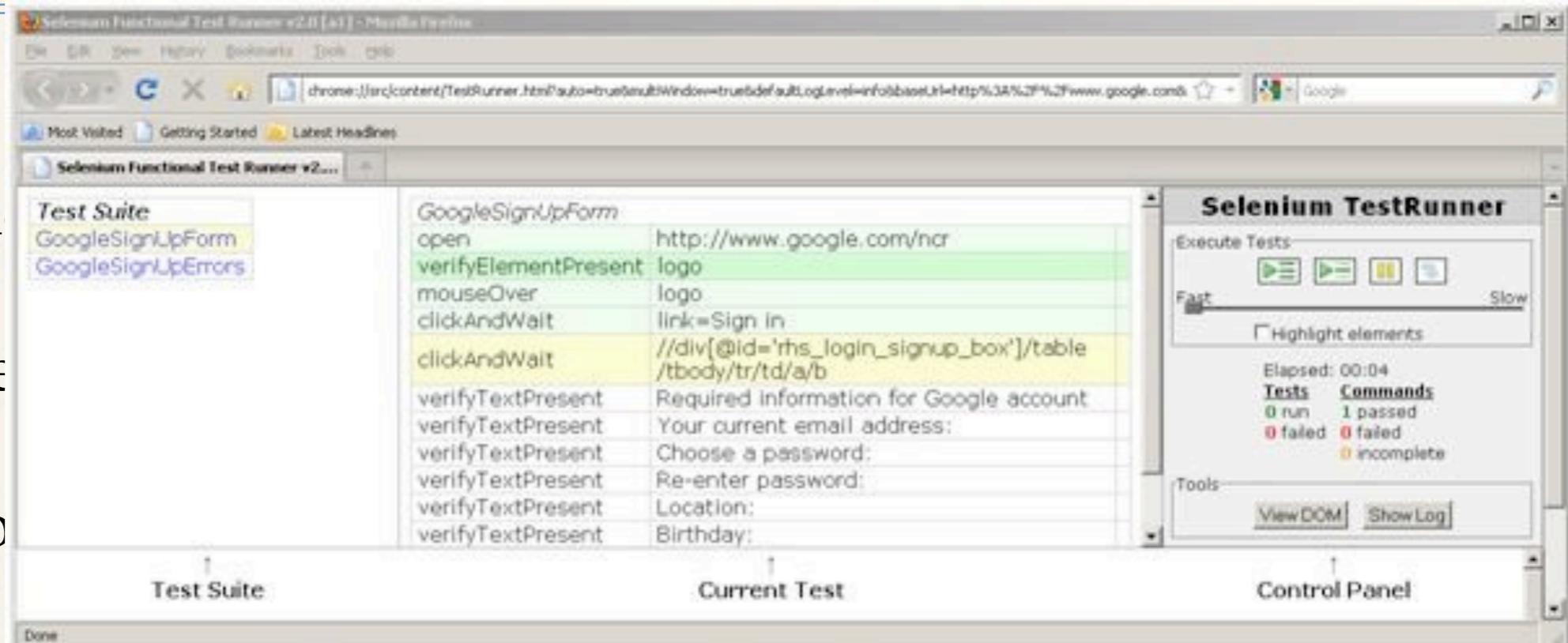
- * à la base

- * exécu

- * navigati

- * dépanna

- * <http://seleniumhq.org/>



Emma

- ❖ outil d'aide au test
- ❖ mesure les couvertures de code Java
- ❖ mesure de couvertures cumulés

Emma

COVERAGE BREAKDOWN BY CLASS AND METHOD

name	class, %	method, %	block, %	line, %
class App	100 % (1/1)	100 % (2/2)	76 % (19/25)	95 % (7,6/8)
method (): void		100 % (1/1)	73 % (16/22)	94 % (6,6/7)
App (): void		100 % (1/1)	100 % (3/3)	100 % (1/1)

❖ outil c

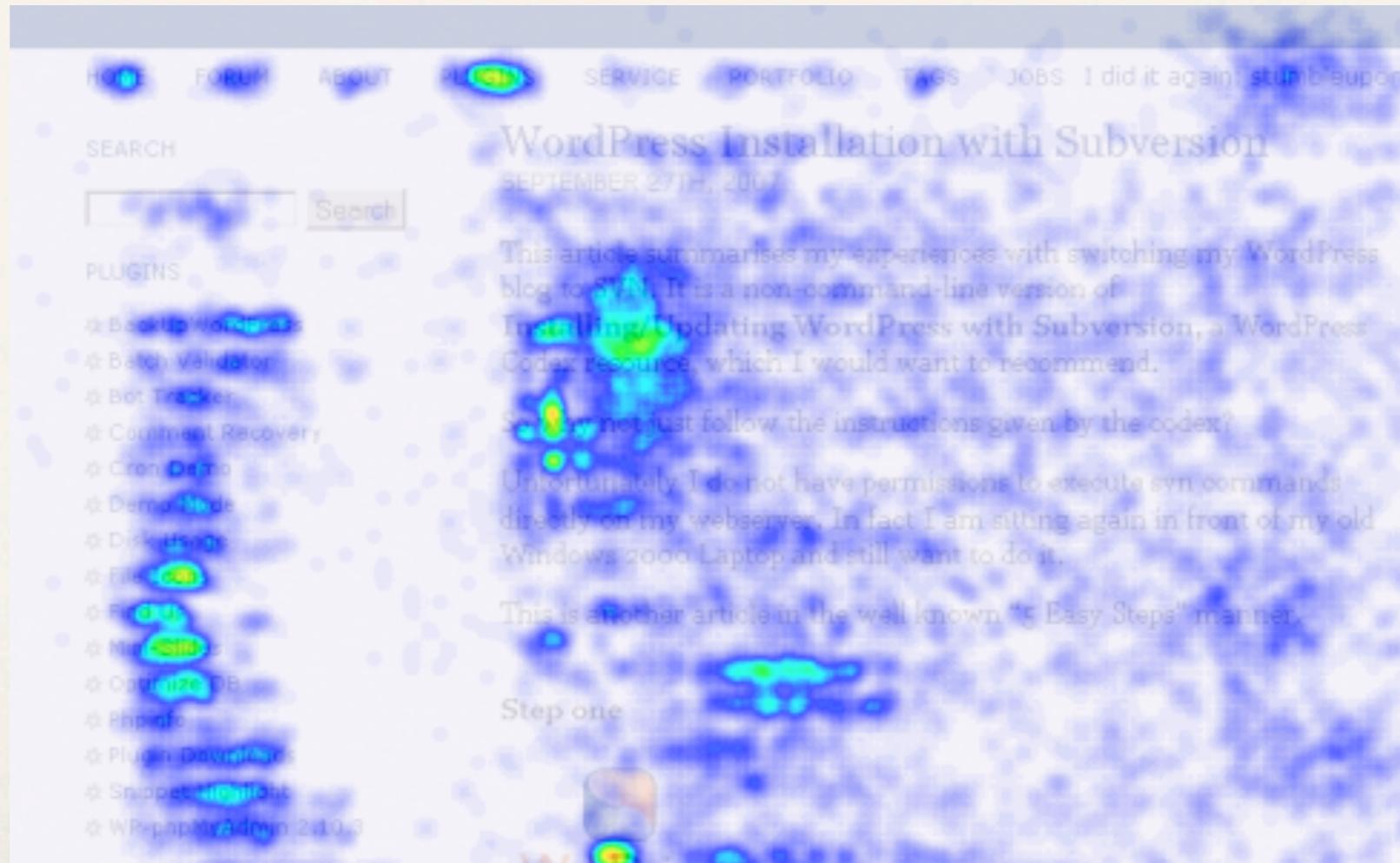
```
1 package com.jh.test_coverage_tools;
2
3 public class App {
4     public void method() {
5         try {
6             System.out.println("Throw exception");
7             throw new NullPointerException();
8         }
9         catch (NullPointerException e) {
10             System.out.println("Exception catched");
11         }
12         finally {
13             System.out.println("Finally... ");
14         }
15     }
16 }
```

❖ mesure

❖ mesure

ClickHeat

- générateur de cartes de température pour visualiser les emplacements des clics des utilisateurs (par rapport à l'application ou à un site web)

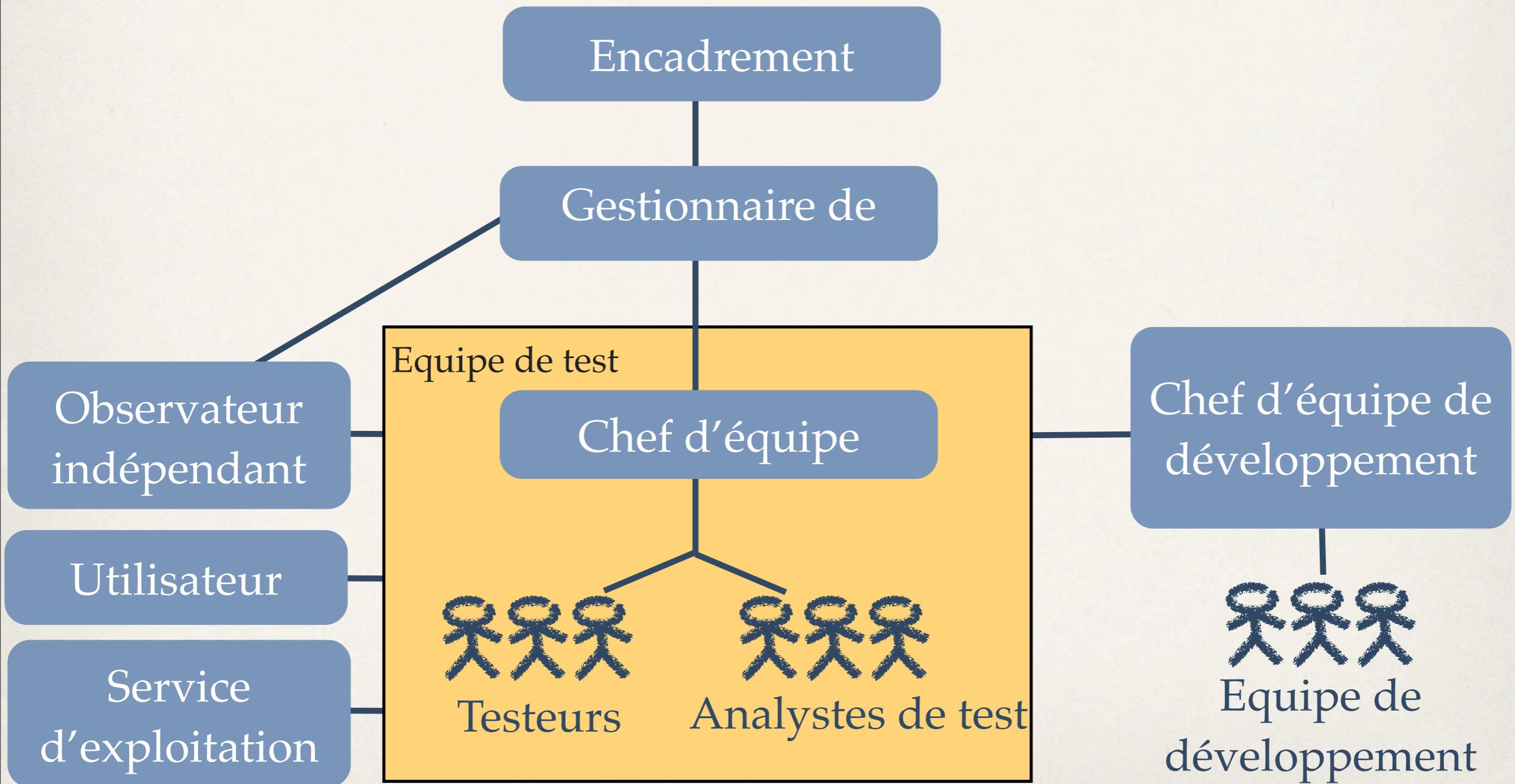


©Ileana Ober, 2012

Plan du cours

- ❖ Métriques
- ❖ Outils de test
- ❖ Organisation et documentation

Organisation des tests - exemple



Gestion du test

- ❖ Politique de test
- ❖ Stratégie de test
- ❖ Test
- ❖ Contrôle et supervision

Gestion du test

- ❖ Politique de test
 - ❖ Quels sont les standard qualité qu'on veut appliquer
 - ❖ Comment mesurer notre performance
- ❖ Stratégie de test
- ❖ Test
- ❖ Contrôle et supervision

Politique de test

- ❖ Permet de s'assurer que les produits logiciel satisfont les exigences métier fixées
- ❖ Est décrite par un document haut niveau décrivant brièvement l'organisation

Gestion du test

- * Politique de test
- * **Stratégie de test**

(Comment organiser le test? Quoi tester? Quand? Avec quelle méthode? Quand s'arrêter? Combien dépenser?)

- * Test
- * Contrôle et supervision

Stratégie de test

- ❖ IEEE Std. 829-1998
- ❖ précise les contraintes spécifiques à l'organisation
- ❖ présente les phases de test et leur description haut-niveau
 - ❖ descriptions des entrées et sorties
 - ❖ approche de test (top-down, bottom-up)
 - ❖ techniques de conception des tests
 - ❖ informations sur les standards, métriques, etc

Gestion du test

- ❖ Politique de test
- ❖ Stratégie de test
- ❖ **Test**
 - ❖ **Préparation**
 - ❖ **Exécution**
- ❖ Contrôle et supervision

Préparation des tests

- ❖ Planification de chaque phase de test
 - ❖ unitaire, d'intégration, système
- ❖ Spécification de la conception des tests
 - ❖ spécification des jeux de test
 - ❖ spécification de la procédure de test (comment ces tests s'exécutent)
 - ❖ inclut la spécification de la configuration de test (logiciels, outils, librairies, bases de données, versions etc.)

Exécution des tests

- ❖ Demande d'effectuer des mesures pour monitorer le déroulement du test
 - ❖ No de test exécutés
 - ❖ Tests passés / échoués
 - ❖ Incidents apparus
- ❖ Informations nécessaires aux chefs d'équipe de test pour décider:
 - ❖ Si on a besoin de ressources supplémentaires pour le test
 - ❖ Si on aura des difficultés à atteindre les objectifs (temps)

Gestion du test

- ❖ Politique de test
- ❖ Stratégie de test
- ❖ Test
- ❖ **Contrôle et supervision**

Contrôle et supervision

- ❖ En continu: surveiller l'évolution des tests, le bon déroulement des activités, le respect des règles et du planning
- ❖ Permet de gérer les changements et les imprévus (dans le planning, dans la constitution des équipes, dans l'environnement de test)

Documentation du test

- ❖ doit être facile à maintenir
- ❖ un document en permanente évolution
- ❖ des exemples sont référencés par la page *moodle* du cours

Normes

- ✿ IEEE 829-2008 - standard pour la documentation de test logiciel
- ✿ IEEE 1008 - standard pour le test unitaire
- ✿ IEEE 1044 - standard pour la classification des anomalies d'un système
- ✿ ...

Conclusions

- ❖ Pour avoir des caractérisations quantitatives d'un logiciel on peut utiliser des *métriques*.
- ❖ De la même façon que pour avoir des mesures justes il faut faire plusieurs mesures avec des instruments différents, pour avoir une bonne caractérisation d'un état, on doit utiliser plusieurs métriques.
- ❖ Il y a plusieurs types d'outils de test
- ❖ L'organisation et la documentation du test doivent se faire avec beaucoup de soin
- ❖ Plusieurs normes régissent ces activités