

Analyse de l'introduction d'anti-patrons de testabilité au cours de développement

Equipe VASCO

MUHAMMAD RABEE SHAHEEN

LYDIE DU BOUSQUET

22 Octobre 2009

Analyse de l'introduction d'anti-patrons de testabilité au cours de développement

Equipe VASCO

MUHAMMAD RABEE SHAHEEN

LYDIE DU BOUSQUET



22 Octobre 2009



Plan

- 1 Introduction
- 2 Test d'intégration & anti-patrons
- 3 Conclusions & Perspectives

Plan

- 1 Introduction
- 2 Test d'intégration & anti-patterns
- 3 Conclusions & Perspectives

Introduction

Tester un logiciel

processus d'exécuter un logiciel dans le but de trouver des erreurs [Myers]

Caractéristiques

- processus long
- méthode de validation la plus utilisée
- critères de couverture
- 40% de coût total selon les stratégies de test

Introduction

Tester un logiciel

processus d'exécuter un logiciel dans le but de trouver des erreurs [Myers]

Caractéristiques

- processus long
- méthode de validation la plus utilisée
- critères de couverture
- 40% de coût total selon les stratégies de test

Introduction

Tester un logiciel

processus d'exécuter un logiciel dans le but de trouver des erreurs [Myers]

Caractéristiques

- processus long
- méthode de validation la plus utilisée
- critères de couverture
- 40% de coût total selon les stratégies de test

Introduction

Tester un logiciel

processus d'exécuter un logiciel dans le but de trouver des erreurs [Myers]

Caractéristiques

- processus long
- méthode de validation la plus utilisée
- critères de couverture
- 40% de coût total selon les stratégies de test

Introduction

Tester un logiciel

processus d'exécuter un logiciel dans le but de trouver des erreurs [Myers]

Caractéristiques

- processus long
- méthode de validation la plus utilisée
- critères de couverture
- 40% de coût total selon les stratégies de test

La testabilité des logiciels

Des solutions

- Réduire le nombre de cas de test
- Réduire le temps par l'automatisation (génération, exécution, analyse)

EST Concevoir testable

Testabilité matérielle

- contrôlabilité
- observabilité

La testabilité des logiciels

Des solutions

- Réduire le nombre de cas de test
- Réduire le temps par l'automatisation (génération, exécution, analyse)

 **Concevoir testable**

Testabilité matérielle

- contrôlabilité
- observabilité

La testabilité des logiciels

Des solutions

- Réduire le nombre de cas de test
- Réduire le temps par l'automatisation (génération, exécution, analyse)

☞ **Concevoir testable**

Testabilité matérielle

- contrôlabilité
- observabilité

La testabilité des logiciels

Des solutions

- Réduire le nombre de cas de test
- Réduire le temps par l'automatisation (génération, exécution, analyse)

 Concevoir testable

Testabilité matérielle

- contrôlabilité
- observabilité

La testabilité des logiciels

Des solutions

- Réduire le nombre de cas de test
- Réduire le temps par l'automatisation (génération, exécution, analyse)

☞ **Concevoir testable**

Testabilité matérielle

- contrôlabilité
- observabilité

La testabilité des logiciels

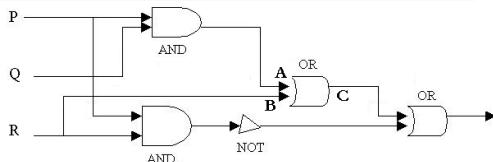
Des solutions

- Réduire le nombre de cas de test
- Réduire le temps par l'automatisation (génération, exécution, analyse)

☞ **Concevoir testable**

Testabilité matérielle

- contrôlabilité
- observabilité



Définition de la testabilité logicielle

ISO

Attributs logiciels qui portent sur l'effort nécessaire pour valider le logiciel

IEEE

Le degré auquel un système facilite l'établissement de critères de test et la performance de tests pour déterminer si ces critères ont été accomplis

Voas

La probabilité qu'une partie de logiciel va échouer dans sa prochaine exécution au cours de test s'il inclut une faute

Définition de la testabilité logicielle

ISO

Attributs logiciels qui portent sur l'effort nécessaire pour valider le logiciel

IEEE

Le degré auquel un système facilite l'établissement de critères de test et la performance de tests pour déterminer si ces critères ont été accomplis

Voas

La probabilité qu'une partie de logiciel va échouer dans sa prochaine exécution au cours de test s'il inclut une faute

Définition de la testabilité logicielle

ISO

Attributs logiciels qui portent sur l'effort nécessaire pour valider le logiciel

IEEE

Le degré auquel un système facilite l'établissement de critères de test et la performance de tests pour déterminer si ces critères ont été accomplis

Voas

La probabilité qu'une partie de logiciel va échouer dans sa prochaine exécution au cours de test s'il inclut une faute

Plan

- 1 Introduction
- 2 Test d'intégration & anti-patterns**
- 3 Conclusions & Perspectives

Test d'intégration

Définition

Le *test d'intégration* est l'étape suivante du test unitaire. A ce niveau, l'objectif est de tester l'intégration de composants (ou des unités de code)

Pourquoi ?

Des tests unitaires OK ne garantissent pas le succès des tests d'intégration

Éléments influençant le test d'intégration

- #composants à intégrer
- des composants incomplets

Test d'intégration

Définition

Le *test d'intégration* est l'étape suivante du test unitaire. A ce niveau, l'objectif est de tester l'intégration de composants (ou des unités de code)

Pourquoi ?

Des tests unitaires OK ne garantissent pas le succès des tests d'intégration

Éléments influençant le test d'intégration

- #composants à intégrer
- des composants incomplets

Test d'intégration

Définition

Le *test d'intégration* est l'étape suivante du test unitaire. A ce niveau, l'objectif est de tester l'intégration de composants (ou des unités de code)

Pourquoi ?

Des tests unitaires OK ne garantissent pas le succès des tests d'intégration

Éléments influençant le test d'intégration

- #composants à intégrer
- des composants incomplets

Patrons d'anti-testabilité

Définition

- un facteur qui influence négativement la testabilité
- une solution de conception qui rend le test difficile
- augmente le #tests

peu d'études afin d'identifier les antipatrons [*Y. Le Traon et al.*]

Patrons d'anti-testabilité

Définition

- un facteur qui influence négativement la testabilité
- une solution de conception qui rend le test difficile
- augmente le #tests

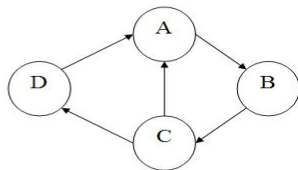
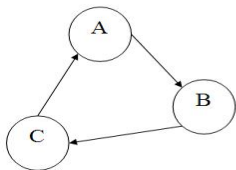
peu d'études afin d'identifier les antipatrons [*Y. Le Traon et al.*]

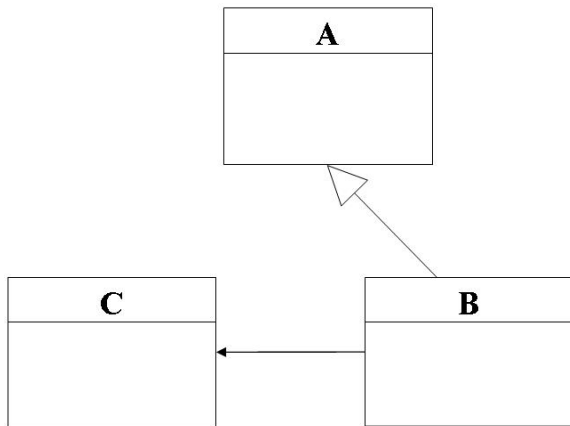
Dépendance entre les composants

- La dépendance est une relation entre deux ou plusieurs composants
- La dépendance peut être *physique* ou *logique*
- La force de dépendance augmente avec le nombre de composants

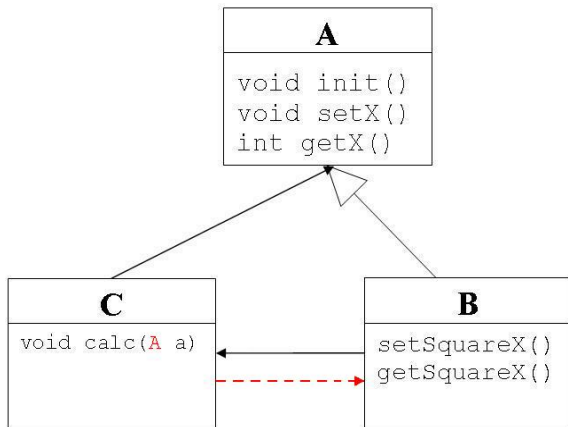
Dépendance entre les composants

- La dépendance est une relation entre deux ou plusieurs composants
- La dépendance peut être *physique* ou *logique*
- La force de dépendance augmente avec le nombre de composants





Modèle abstrait



Modèle détaillé

L'objectif

- 1 Cycles (# + taille) vs. différents stades de conception
- 2 A quel moment les cycles sont-ils introduits ?
- 3 Coût prédictif vs. coût effectif

Démarche d'analyse

- 1 Récolte des données
(sans modèles)
- 2 Simulation de modélisation
- 3 Analyse de données

Retro Engineering

Démarche d'analyse

- 1 Récolte des données
(sans modèles)
- 2 Simulation de modélisation
- 3 Analyse de données

Retro Engineering

Démarche d'analyse

- 1 Récolte des données
(sans modèles)
- 2 Simulation de modélisation
- 3 Analyse de données

Retro Engineering

Code Java

Démarche d'analyse

- 1 Récolte des données
(sans modèles)
- 2 Simulation de modélisation
- 3 Analyse de données

Retro Engineering

Modèle détaillé

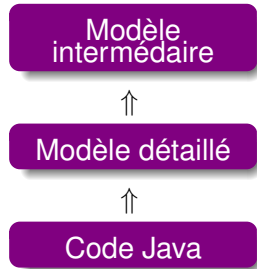


Code Java

Démarche d'analyse

- 1 Récolte des données
(sans modèles)
- 2 Simulation de modélisation
- 3 Analyse de données

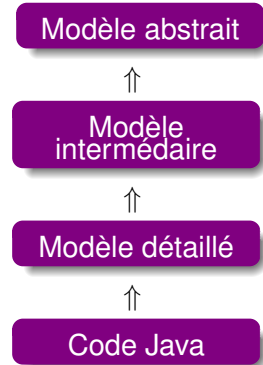
Retro Engineering



Démarche d'analyse

- 1 Récolte des données
(sans modèles)
- 2 Simulation de modélisation
- 3 Analyse de données

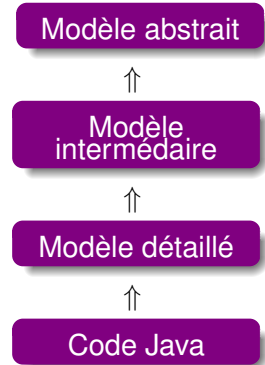
Retro Engineering



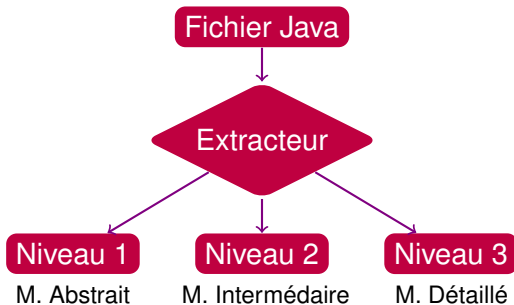
Démarche d'analyse

- 1 Récolte des données
(sans modèles)
- 2 Simulation de modélisation
- 3 Analyse de données

Retro Engineering



Extraction des niveaux



Niveau d'attributs

Attributs : (sans les attributs de type classe interne (inner class))

Niveau 1

```
public class SampleExample{  
    /* Attributes */  
    int attribute1 ;  
    String attribute2 ;  
    byte [ ] attributearray ;  
    Non_innerClassType attribute_complex ;  
}
```

Niveau 2

Attributs + Signatures des méthodes : pas de classes internes dans les attributs ou les paramètres

Niveau 2

```
public class SampleExample{  
/* Attributes */  
    int attribute1 ;  
    String attribute2 ;  
    byte [ ] attributearray ;  
    Non_innerClassType attribute_complex ;  
  
/* Methods */  
    public int getAttribute1() { return attribute1 ;}  
    public void setAttribute1(int para0){}  
    public void init(){}  
}
```

Niveau 3

```
public class SampleExample{  
  /* Attributes */  
  int attribute1 ;  
  String attribute2 ;  
  byte [ ] attributearray ;  
  Non_innerClassType attribute_complex ;  
  InnerClass Exattribute3 ;  
  /* Inner Class Type */  
  class InnerClassEx()  
    { // some attributes and methods }  
  /* Methods */  
  public int getAttribute1() { return attribute1 ;}  
  public void setAttribute1(int p0){}  
  public void init(){}  
  public void calc(int p0, InnerClassEx p1){}  
}
```

Niveau 4 : (source code) implémentation complete

```
public class SampleExample{
  /* Attributes */
  int attribute1 = 7 ;
  String attribute2 = "source code level" ;
  byte [ ] attributearray ;
  Non_innerClassType attribute_complex ;
  InnerClass Exattribute3 ;
  /* Inner Class Type */
  class InnerClassEx()
    { int x = 10 ; public void setX(int p){ x = p} }
  /* Methods */
  public int getAttribute1() { return attribute1 ;}
  public void setAttribute1(int p0){ attribute1 = p0}
  public void init(){attribute_complex = null }
  public void calc(int p0, InnerClassEx p1){System.out.print(p0*p1.x) ;}
}
```

Fréquence de cycles (niveau 4)

Application/Cycle size	2	3	4	5-10	11-20	21-30	>30
CEF	1	4	1	3	0	1	1
Bluepad	0	0	0	1	0	0	0
HtmlCleaner	0	0	0	0	0	1	0
JavaGUIBuilder	6	4	1	1	0	0	0
Jaxe	6	2	0	0	0	0	1
Jdom	2	4	1	0	0	1	0
JfreeChart	20	5	1	2	1	0	1
JSXE	12	2	2	5	1	1	4
KoLmafia	13	4	4	3	0	1	1
MegaMek	38	18	3	11	3	0	4
Weirdx	2	1	0	0	0	0	1
NanoXML	2	0	0	0	0	0	0
XmlMath	1	0	1	0	0	0	0
Total	103	44	14	26	5	5	13
Percent	49,05	20,95	6,67	12,38	2,38	2,38	6,19

Cycles aux niveaux 1,2 et 3

Application/Cycle size	Level 1			Level 2			Level 3		
	#Max	#Min	#Cycles	#Max	#Min	#Cycles	#Max	#Min	#Cycles
1-NanoXML	0	0	0	0	0	0	2	2	1
2-CEF	0	0	0	3	3	1	3	3	3
3-Jsxe	2	2	2	8	2	6	46	2	43
4-XMLMath	0	0	0	0	0	0	0	0	0
5-HTMLCleaner	0	0	0	8	8	1	10	10	1
6-MegaMek	13	2	5	39	2	15	45	2	47
7-weirdx	13	2	4	27	2	2	27	2	9
8-Java Gui Builder	0	0	0	0	0	0	3	2	7
9-Bluepad	0	0	0	0	0	0	0	0	0
10-Jaxe	4	3	3	9	3	3	24	2	19
11-JDom	3	3	3	8	8	1	10	2	7
12-JFreeChart	2	2	3	31	2	9	31	2	24
13-Kolmafia	2	2	1	29	2	4	29	2	99
Total			21			42			260

- Niveau 1 : Min [2,3] Max [2,13] #Cycles 21
- Niveau 2 : Min [2,8] Max [3,39] #Cycles 42
- Niveau 3 : Min [2,10] Max [2,46] #Cycles 260

Tailles de cycles & classes internes

Application/Cycle size	#Max	#Min	#Cycles	#Classes in all cycles	#Inner Classes	%Inner classes
NanoXML	2	2	2	2	1	50
CEF	38	2	11	105	91	86.67
JSXE	129	2	27	361	209	57.89
XmlMath	4	2	2	6	1	16.67
HtmlCleaner	21	21	1	21	0	0
MegaMek	101	2	77	571	321	56.22
Weirdx	58	2	4	65	10	15.38
JavaGUIBuilder	6	2	12	34	22	64.71
Bluepad	10	10	1	10	0	0
Jaxe	132	2	9	150	84	56
JDom	24	2	8	44	13	29.55
JFreeChart	36	2	30	121	22	18.18
KoLmafia	598	2	26	703	396	56.33
Total	-	-	210	2193	1170	53.35

Plan

- 1 Introduction
- 2 Test d'intégration & anti-patterns
- 3 Conclusions & Perspectives**

Résultats

- 1 #Cycles augmente considérablement au niveau 3 et 4
- 2 #Cycles de taille 2 et 3 représentent environ 70%
- 3 Les classes internes participent fortement à la formation de cycles
- 4 Détecter les cycles au niveau modèle est important, mais pas suffisant
- 5 Besoin d'un outil pour détecter les cycles aux différents niveaux

Perspectives

- Outil aide à détecter les cycles aux différentes phases
- même étude avec des modèles réel

Merci de votre attention !

Questions ?