Détection semi-automatique des patrons de mauvaise conception dans les architectures orientées-objet

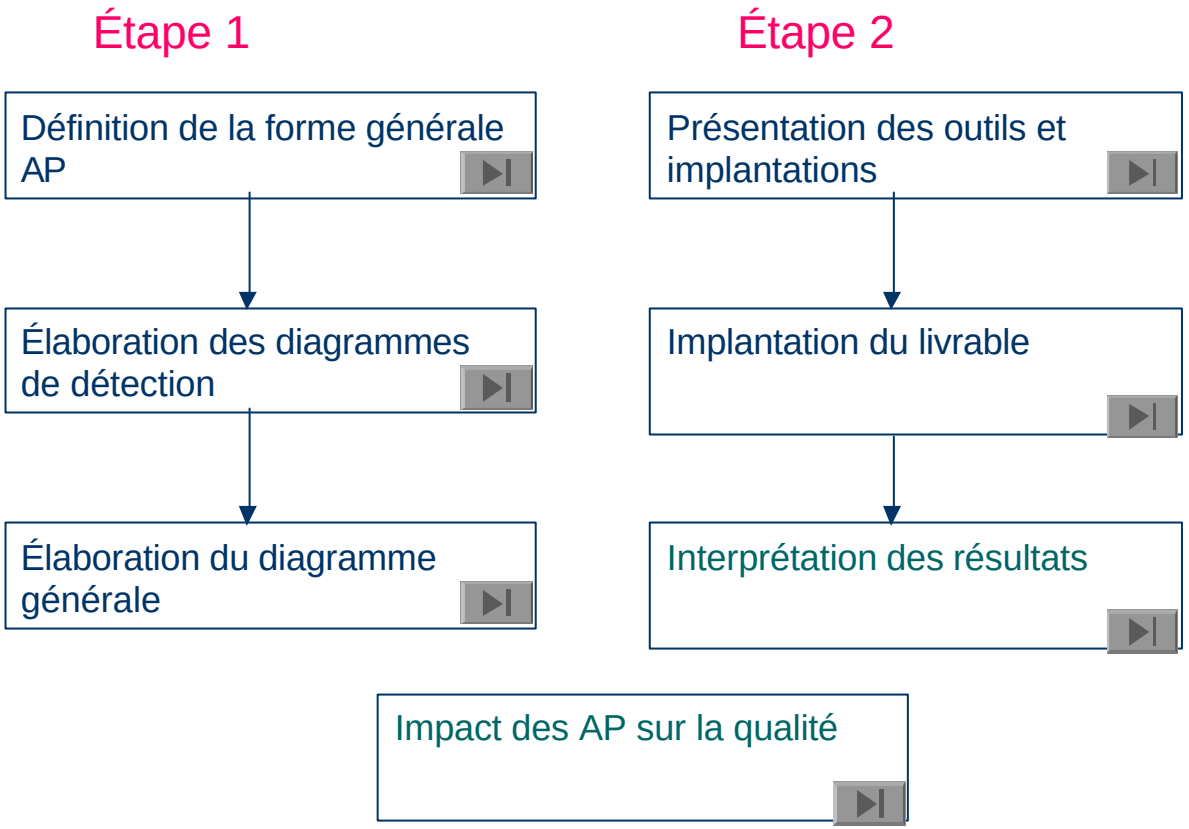
Introduction

- Contrat de la conception Orientée Objet.
- Anti-patrons # non fonctionnelle.
- Avantages des anti-patrons

Cadre du travail

- ✓ Les objectifs de l’étude.

Cheminement global du projet



BLOB

Définition :

- Identifié par une classe qui monopolise tout le traitement et qui encapsule toutes les données.
- Programmation structurée

Symptômes :

- Une classe avec un grand nombre d'attributs et d'opérations.
- Cette large classe communique avec des classes de données.
- Cohésion faible dans la large classe.
- L'absence de conception orientée objet (plutôt orientée procédurale).

Conséquences :

- Limitation de la possibilité de modification des systèmes sans affecter d'autres composants.
- Difficulté à être réutilisée et à être testée
- Utilisation énorme de ressources.

Spaghetti code

Définition :

- C'est un système où on trouve peu de structure, le système inclut un petit nombre d'objets avec des méthodes trop grandes qui sont appelées une seule fois.
- Il y a un faible degré d'interaction entre les objets.

Symptômes :

- Des objets nommés comme des processus, les méthodes sont orientées processus.
- Le flux d'exécution est dédié par l'implémentation des objets, non par les clients des objets.
- Relation minimale entre les objets
- Beaucoup de méthodes n'ont pas de paramètres, et utilisent des variables globales ou de classes.

Conséquences :

- Difficulté de réutilisation du code.
- La perte des avantages de la programmation orientée objet.

Functional Decomposition

Définition :

- Quand les programmeurs sont à l'aise avec les « main » routines qui appellent d'autres routines du système, alors ils essayent de transformer les sous routines en classes ignorant ainsi le design orienté objet.

Symptômes :

- Une classe nommée comme des routines (Afficher_table)
- Tous les attributs sont **Private** et sont utilisés seulement dans la classe.
- Une classe avec une seule action.

Conséquences :

- - Violation des principes de la programmation orientée objet
- rendant difficile la maintenance du logiciel.
- - Difficulté de clairement documenter le code.
- - Difficulté de réutiliser le code et de le tester.

Poltergeists

Définition :

- Des classes qui ont des responsabilités et des rôles limités dans le système.
- Une classe qui apparaît pendant une courte durée.

Symptômes :

- Redondance dans la navigation du flux.
- Classe sans états.
- Une classe qui apparaît pendant une courte durée.
- Une classe avec une seule action.
- Une classe nommée comme (init_processus_1).

Conséquences :

- Utilisation énorme de ressources.

LavaFlow

Définition :

- Le lavaFlow est un anti-patron qui présente des bouts de code ignorés ou des bouts de code dit critiques qu'il ne faut pas toucher.

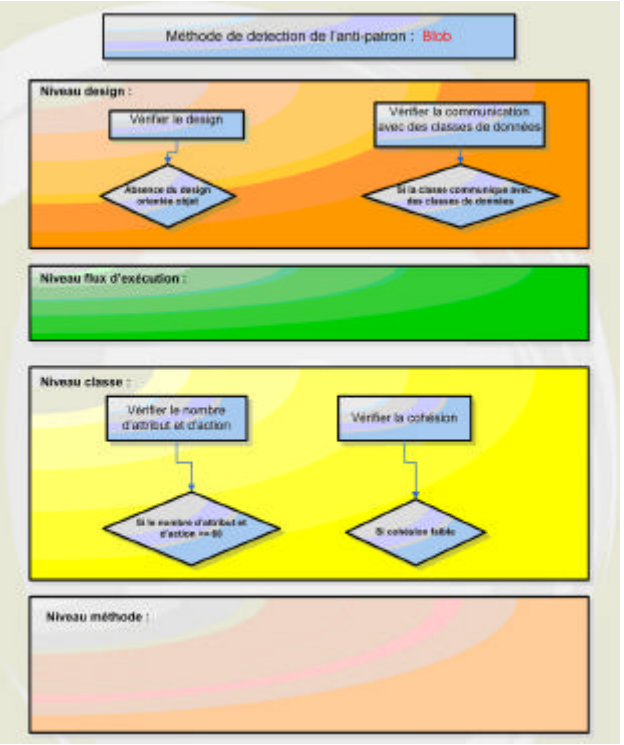
Symptômes :

- Présence fréquente de variables et de fragments de code injustifiés.
- Manque de documentation des composants critiques du système.
- Présence de code inutilisé.
- Présence des mots « in flux », « to be replaced » et « to do ».
- Des interfaces inexplicables ou obsolètes.

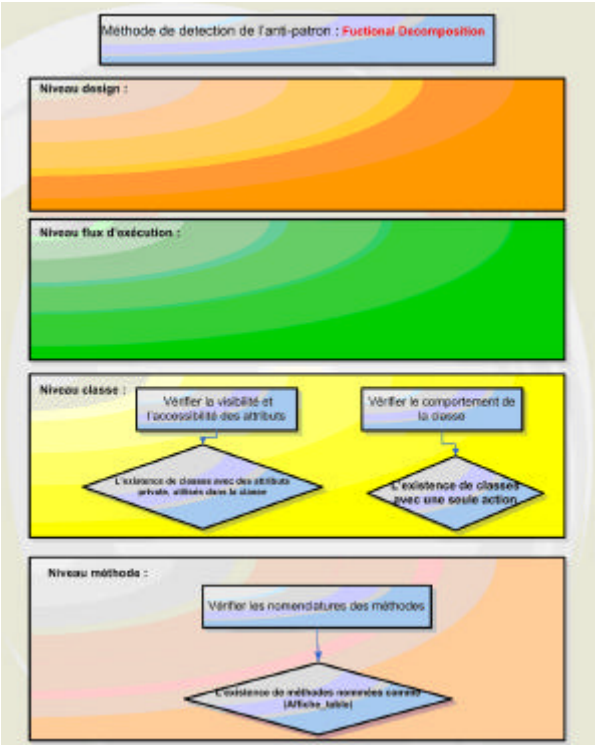
Conséquences :

- Si le lavaFlow n'est pas éliminé il peut se propager en cas de réutilisation du code.
- En cas de propagation du lavaFlow il devient très difficile de documenter le code, et de comprendre son architecture.

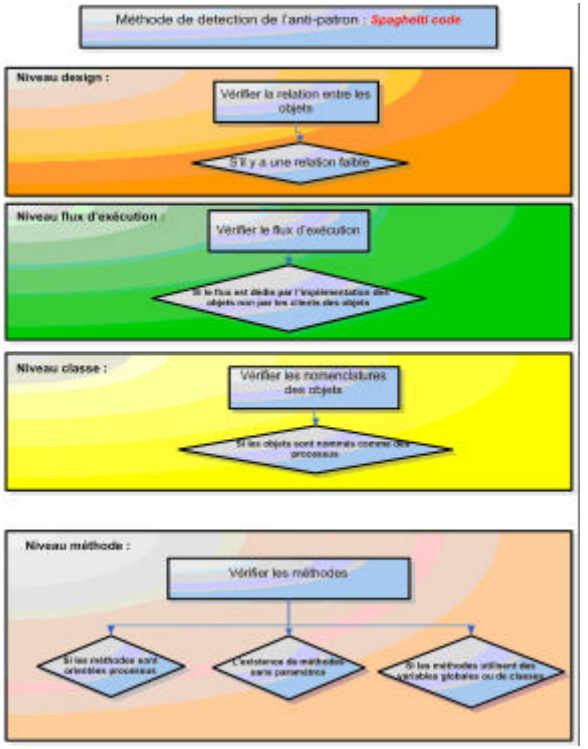
BLOB


Méthode de detection de l'anti-patron : Blob
Niveau design :
Vérifier le design
Absence de design orienté objet
Vérifier la communication avec des classes de données
Si la classe communique avec des classes de données
Niveau flux d'exécution :
Niveau classe :
Vérifier le nombre d'attribut et d'action
Si le nombre d'attribut et d'action >= 60
Vérifier la cohésion
Si cohésion faible
Niveau méthode :

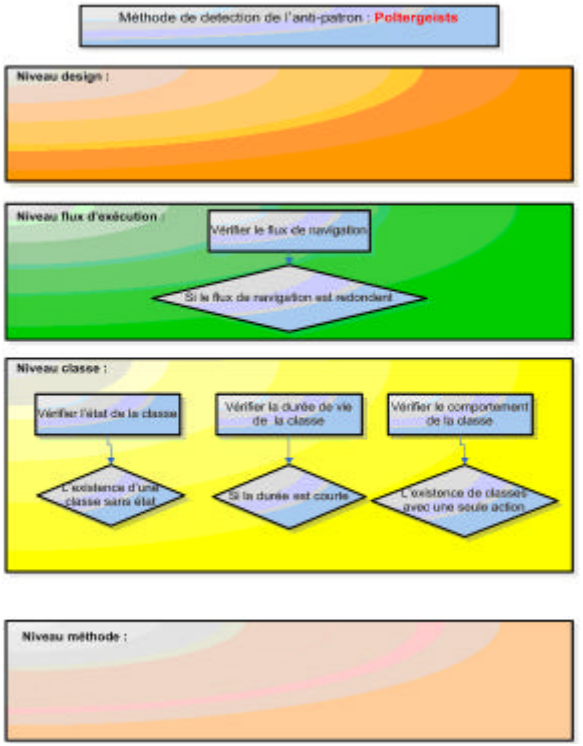
Spaghetti code



Functional Decomposition



Poltergeists



Implantations des outils de détection

- Familiarisation avec les outils de détection : POM, PADL, PADL CLASS CREATOR, CPL.
- Implantation des détecteurs des mauvais patrons de conception.
  - Utilisation des métriques.
  - Utilisation de la notation ETVX.

Détection du Blob

- **Étape 1 : Détermination de la large classe**
  - (NAD + NMD) >= 60.
- **Étape 2 : Détermination des classes de cohésion faible**
  - **(NAD + NMD) >= 60 && Faible (valeur LCOM5)**
- **Étape 3 : Identification des classes de données**
  - NbCDCLC le nombre de classes de données qui communiquent avec la large classe.

? **(NAD + NMD) >= 60 && Faible (valeur LCOM5) && (NbCDCLC >=1)**

Détection Spaghetti Code

- **Étape 1 : Détermination des classes de faible CBO**
  - **CBO** 
- **Étape 2 : Détermination des classes de cohésion forte**
  - **LCOM5** 

→Faible (valeur CBO) && Forte (valeur LCOM5)

- **Étape 3 : Vérification des méthodes**
  - vérification visuelle

Détection du Poltergeists

- **Étape 1 : Détermination des SDO (Short duration objects)**
  - DOC (Duration object calcultor)
- **Etape2 : Déterminer si la classe fait une seule action**
  - la classe a un seul point d`entrée

→ **Minimal(DOC) && classe fait une seule action && Classe sans accessibilité (non public, non private ….)**

- **Étape 3 : Vérification des noms de classes**
  - Classe nommée comme un processus.

Détection Functional Decomposition

- **Étape 1 : Déterminer si l'utilisation des attributs est locale**
- **Etape2 : Déterminer si la classe fait une seule action**
  - la classe a un seul point d`entré.
- **Étape 3 : Vérification des noms des classes**
  - la classe a un nom d'une action.

BLOB

Large classe	NAD	NMD	Les classes de données
org.argouml. model.uml.**CoreFactoryImpl** (0.9294871770991728)	3.0	106.0	org.argouml.application.configuration.ConfigurationFactory org.argouml.uml.diagram.ui.CompartmentFigText org.argouml.uml.UUIDHelper org.argouml.uml.ui.SourcePathTableModel org.argouml.cognitive.ui.WizDescription org.argouml.model.uml.ExtensionMechanismsHelperImpl org.argouml.uml.diagram.use_case.ui.UseCaseDiagramRenderer org.argouml.uml.diagram.collaboration.ui.CollabDiagramRenderer org.argouml.uml.diagram.static_structure.ui.ClassDiagramRenderer org.argouml.ui.explorer.TransferableModelElement org.argouml.uml.diagram.sequence.ui.SequenceDiagramRenderer org.argouml.application.security.ArgoJarClassLoader org.argouml.uml.diagram.static_structure.layout. ClassdiagramModelElementFactory org.argouml.uml.cognitive.critics.CrMultipleAgg org.argouml.persistence.LastLoadInfo org.argouml.uml.cognitive.critics.CrMultiComposite org.argouml.uml.diagram.ui.SPFigEdgeModelElement org.argouml.uml.diagram.state.ui.StateDiagramRenderer org.argouml.uml.diagram.ui.ActionAddAssociationRole org.argouml.model.uml.CopyHelper org.argouml.uml.diagram.deployment.ui.DeploymentDiagramRenderer org.argouml.application.security.ArgoAwtExceptionHandler org.argouml.uml.cognitive.critics.CrUnnavigableAssoc
~~**net.sourceforge.ganttproject.gui.GanttTaskPropertiesBean**~~	~~67.0~~	~~34.0~~	~~* net.sourceforge.ganttproject.shape.JPaintCombo * net.sourceforge.ganttproject.shape.ShapeConstants * net.sourceforge.ganttproject.shape.ColorConstants~~	~~0.9673507477564272~~

Spaghetti code

Nom de la Classe	Valeur du CBO	Valeur du LCOM5	Méthodes sans paramètres
~~org.argouml.uml.cognitive.ProjectMemberTodoList~~	~~4.0~~	~~1.2222222222222222~~	~~getResolvedCritiquesList~~ ~~getToDoList~~ ~~getType~~ ~~getZipFileExtension~~
org.argouml.util.ExprSeparatorWithStrings	2.0	1.3333333333333333	hasFreePart reset tokenLength

Interprétation des résultats

- BLOB
  - Évolution et distribution du Blob.
  - Problème de la nature de la classe
  - MVC
- Spaghetti Code (moins critique).
- Poltergeists
  - package JDI
  - SDO
  - Difficulté de détection
- Functional Decomposition
  - Peu d'expérience
  - Méthode de détection pas assez raffinée.

Impact des AP sur la qualité logiciel

- Terminologie
- la conformité des anti-patrons de conception
- Coûts des AP.

Conclusion