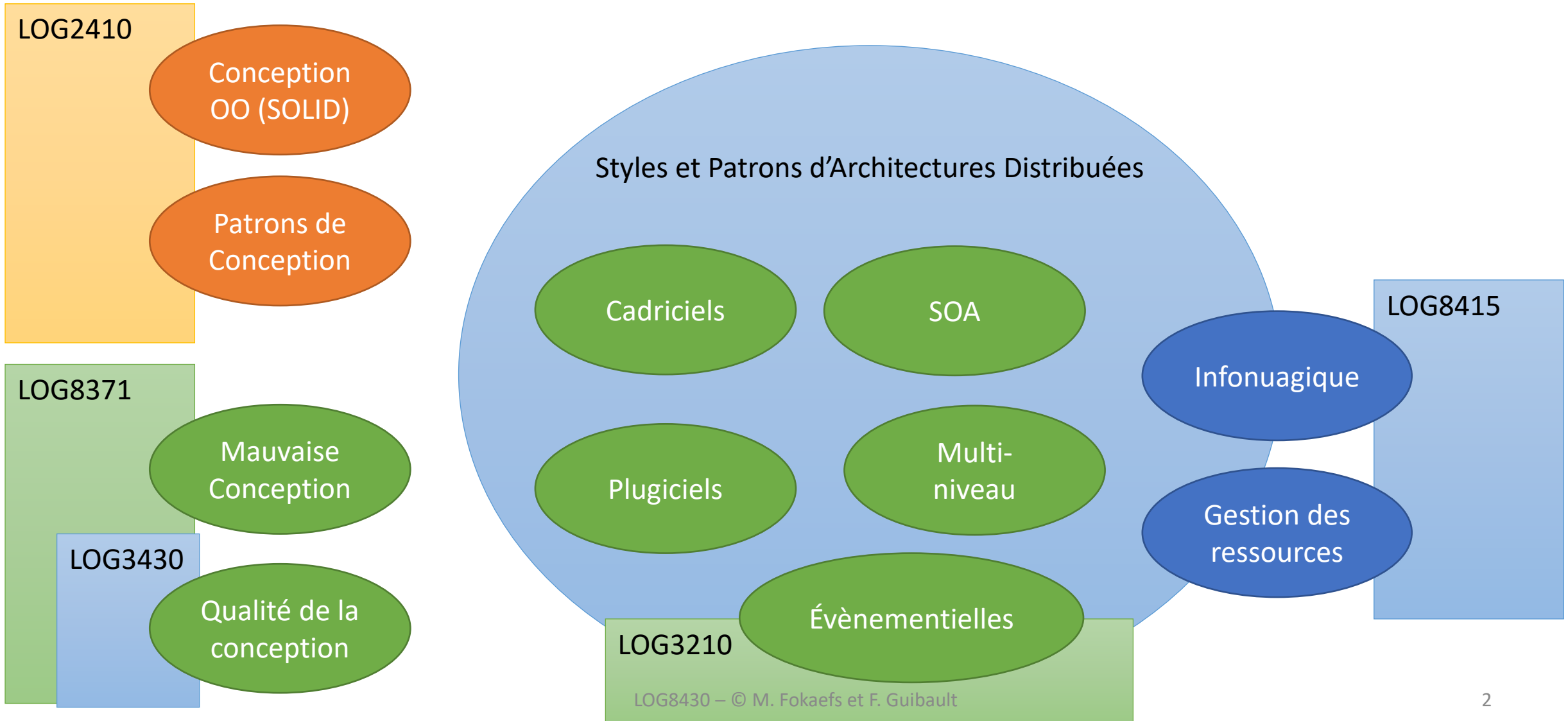


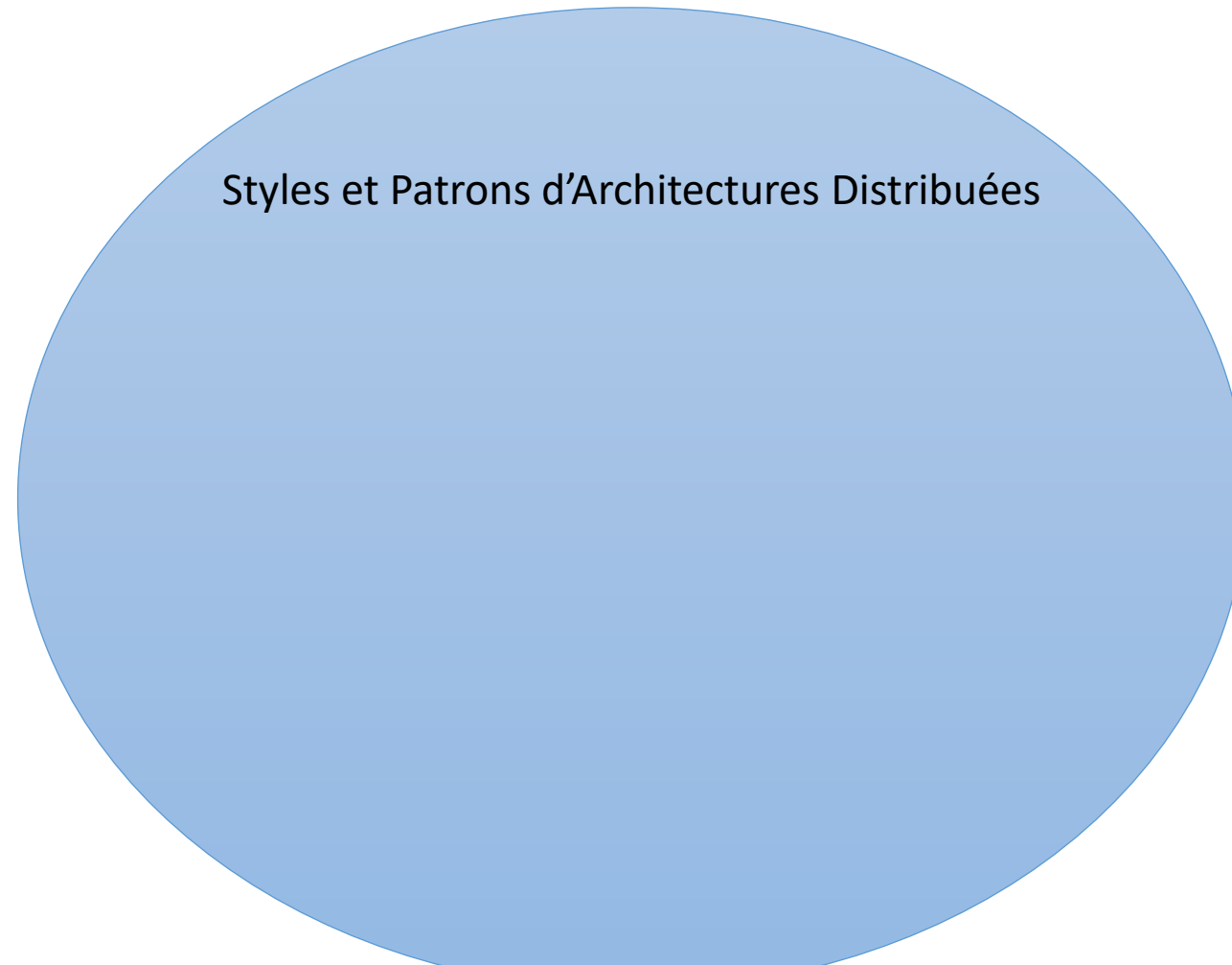
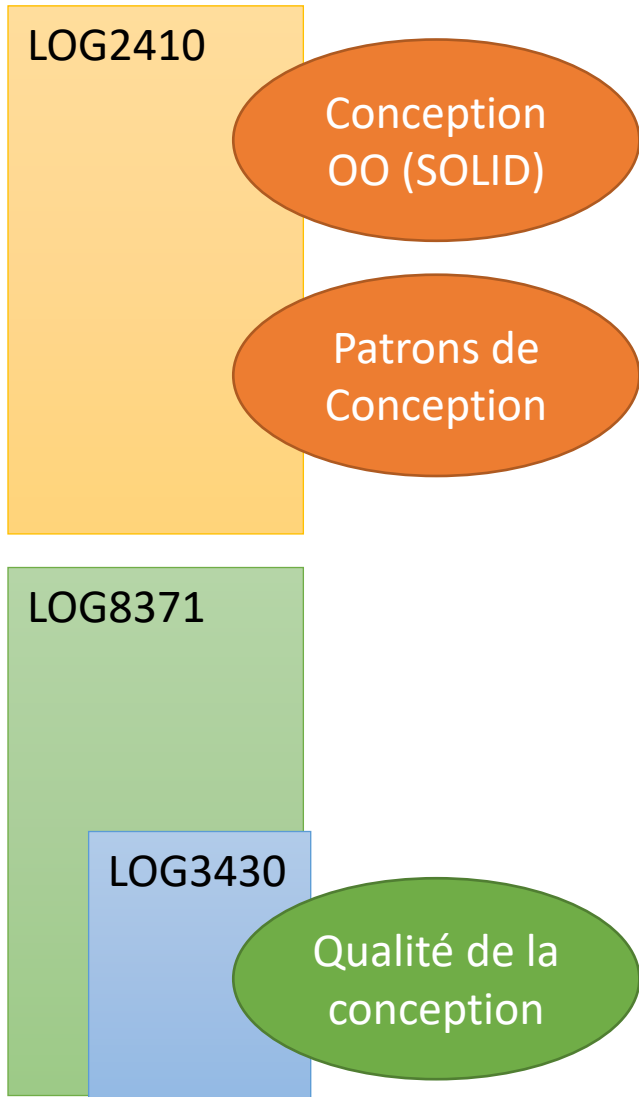


LOG8430 : Mauvaise Conception

Carte du cours



Précédemment



Aujourd'hui

LOG2410

Conception
OO (SOLID)

Patrons de
Conception

LOG8371

Mauvaise
Conception

LOG3430

Qualité de la
conception

Styles et Patrons d'Architectures Distribuées

Agenda

Antipatrons

- Antipatrons architecturaux
- Antipatrons de conception
- Antipatrons de code

Détection

- Seuils des métriques
- Stratégies de détection

Correction

- Refactoring
- Outils (JDeodorant, Eclipse)

Qu'est-ce qu'un antipatron?

- Un patron est une solution à un problème récurrent et qui facilite les activités périphériques au développement comme la maintenance, l'assurance de la qualité et d'autres.
 - C'est une bonne pratique!
- Un antipatron est une approche de conception ou une construction récurrente, qui est inefficace et contreproductive.
 - C'est une mauvaise pratique!
- Mais pourquoi les antipatrons existent-ils?
 - Incapacité ou connaissance insuffisante.
 - Bonnes solutions à court terme avec des conséquences néfastes à long terme.
 - Utilisation de patrons de conception de façon ou dans un contexte inapproprié.

Antipatrons d'architecture

- Architecture by implication
- Autogenerated Stovepipe
- Cover your Assets
- Design by Committee
- Jumble
- Reinvent the Wheel
- Spaghetti Code
- Stovepipe System
- Swiss Army Knife
- Vendor Lock-in

Architecture by Implication

- Définition
 - Les architectes ont beaucoup d'expérience dans le développement de systèmes semblables et ils ne préparent pas de spécifications architecturales suffisantes:
 - Réutilisation inappropriée d'une architecture existante/connue.
- Symptômes et Conséquences
 - Manque de planification et de spécification de l'architecture.
 - Ignorances de nouvelles technologies.
 - Risques cachés qui émergent en cours de projet: manque de connaissance du domaine, des technologies et complexité inattendue.
 - Échec imminent.
- Causes
 - Absence de gestion de risque.
 - Excès de confiance des gestionnaires, architectes et développeurs.
 - Dépendance envers des expériences antérieures qui diffèrent sous certains aspects critiques.
 - Enjeux architecturaux non résolus ou implicites causés par des déficiences dans l'ingénierie du système.

Architecture by Implication

- Refactoring
 - Modulariser la spécification de l'architecture en vues et points de vue.
 - Chaque vue représente une partie prenante pertinente (architecte, développeur, client etc.).
 - Exprimer l'architecture à l'aide d'objectifs et de questions (GQA): objectifs, questions, vues, analyse des vues, intégration des vues, correspondance entre les vues et les exigences, raffinement des vues, communication de l'architecture, validation de l'implémentation.
- Exceptions
 - Si les différences entre 2 projets sont petites ou localisées.
 - Pour un projet exploratoire.

Autogenerated Stovepipe

- Définition

- Le problème se produit lorsque nous migrons un système vers une version distribuée.
- Les interfaces du système original peuvent être inappropriées pour le système distribué.
- Les opérations locales font souvent diverses hypothèses sur l'emplacement, p.ex. l'accès au système de fichiers local.

- Refactoring

- Définir les interfaces distribuées à nouveau (en utilisant le patron façade?).
- L'interopérabilité et la stabilité doivent être une haute priorité.

Cover your assets

- Définition
 - La documentation est trop détaillée et tente de couvrir toutes les options et les exceptions. Elle devient très compliquée.
- Refactoring
 - Utiliser des abstractions, comme des diagrammes et des tableaux, pour communiquer l'architecture de façon plus efficace.

Design by committee

- Définition
 - Tous les membres de l'équipe peuvent influencer les décisions d'architecture et de conception. Le résultat est de la documentation très complexe avec tous les caractéristiques possibles.
- Symptômes et conséquences
 - Des réunions fréquentes, mais pas organisées et sans objectif.
 - Beaucoup de conflits parmi les architectes et les développeurs.
 - Beaucoup d'efforts requis pour interpréter et développer les spécifications.
- Causes
 - Les rôles ne sont pas bien définis.
 - Un effort pour satisfaire tout le monde.
 - Réunions longues et mal organisées.

Design by committee

- Refactoring
 - Définir des rôles clairs pour les membres de l'équipe.
 - Désigner un architecte principal pour le projet qui va prendre les décisions finales.
 - Prioriser les exigences.
 - Organiser les réunions et diminuer leur durée.
 - Possiblement diviser les équipes pour mieux organiser les réunions (DevOps).
- Exceptions
 - Lorsque les équipes sont petites.

Jumble

- Définition
 - Lorsque les éléments horizontaux se mélangent avec les éléments verticaux. Les éléments horizontaux sont les couches d'une architecture générale. Les éléments verticaux sont les tiers de l'architecture spécifiques à une application. (comme une violation de DIP).
- Symptômes et conséquences
 - Réutilisabilité et stabilité réduites.
- Causes
 - Manque de communication entre les développeurs et les architectes.
- Refactoring
 - Traiter les éléments horizontaux en tant qu'abstractions et les éléments verticaux en tant qu'implémentations.
 - Ajouter des éléments verticaux pour la fonctionnalité spécialisée ou pour la performance.

Reinvent the Wheel

- Définition
 - Lorsqu'un système est conçu à nouveau. Les architectures et les exemples existants ne sont pas pris en considération.
- Symptômes et conséquences
 - La réutilisabilité et l'interopérabilité sont diminuées. L'effort est augmenté.
 - Des architectures fermées, pertinentes uniquement pour un système spécifique (comme une violation d'OCP).
 - Réplication inutile des conceptions existantes.
- Causes
 - « Greenfield Assumptions » : le système doit être conçu à nouveau. Nous sommes les premiers qui font quelque chose comme ça.
 - Connaissance insuffisante d'autres projets.

Reinvent the Wheel

- Refactoring
 - Faire le minage des architectures pour identifier les exemples pertinents.
 - Mettre en place un processus logiciel en spirale, agile et itératif.
 - Raffiner les exigences et les décisions de conception, et identifier des prototypes d'architectures dans d'autres systèmes.
- Exceptions
 - Lorsque la communication est difficile et que l'on veut minimiser les frais de la coordination.

Spaghetti Code

- Définition
 - Le logiciel manque de structure et de cohérence.
 - Blague : « Le problème avec la programmation visuelle est que le code spaghetti ressemble vraiment à du spaghetti! »
- Symptômes et conséquences
 - La compréhensibilité et la réutilisabilité sont diminuées.
 - Le système est plus procédural qu'orienté-objet.
 - L'effort pour la maintenance est augmenté.
- Causes
 - Manque d'expérience.
- Refactoring
 - Séparation des responsabilités, encapsulation.
 - Utilisation des patrons et de bonnes pratiques de conception.
- Exceptions
 - Du code spaghetti est tolérable lorsqu'il se produit dans l'implémentation et pas dans les interfaces publiques.

Stovepipe System

- Définition
 - L'antipatron se produit lorsqu'on veut intégrer des sous-systèmes. Chaque intégration est spéciale et il manque un plan d'intégration générale pour le système entier.
- Symptômes et conséquences
 - La maintenabilité et l'interopérabilité sont diminuées.
 - L'implémentation ne correspond pas à la documentation.
 - Le système devient très complexe.
 - La portabilité est réduite.
- Causes
 - Manque d'abstraction. L'interface de chaque sous-système est unique.
 - Fort couplage.
 - Manque d'une architecture centrale.
- Refactoring
 - Définition d'une interface commune (SOA).

Swiss Army Knife

- Définition
 - Une interface très complexe qui expose beaucoup de fonctionnalités. (une violation de SRP et d'ISP)
- Refactoring
 - Extraire des méthodes de l'interface ou de la classe.
 - Utiliser les patrons façade ou *adapter*.
 - Documenter l'usage et l'implémentation de l'interface. Créer un profil.

Vendor Lock-in

- Définition
 - Lorsque le système dépend directement de logiciels ou de matériel propriétaires.
- Symptômes et conséquences
 - Les mises à niveau des produits commerciaux pilotent le cycle de maintenance des logiciels d'application.
 - Des échecs dans les applications se produisent à cause des délais induits par les produits commerciaux
 - Des violations de DIP et d'OCP.
 - Portabilité est réduite.
- Causes
 - Il n'existe pas un standard universel.
 - Le produit commercial ne se conforme pas aux standards ouverts.
 - Choix des produits pour des raisons économiques et non pas techniques.
- Refactoring
 - Architecture multi-niveaux
 - Utilisation d'abstractions.

Antipatrons de conception

- Missing abstraction
- Multifaceted abstraction
- Duplicate abstraction
- Deficient encapsulation
- Unexploited encapsulation
- Broken modularization
- Insufficient modularization
- Cyclically-dependend modularization
- Unfactored hierarchy
- Broken hierarchy
- Cyclic hierarchy

Missing abstraction

- Définition
 - Lorsque des groupes de données se produisent ensemble souvent.
 - P.ex. `telephoneNumber`, `areaCode`, `countryCode`
- Symptômes et conséquences
 - Violation de SRP.
 - Complexité augmentée.
 - Maintenabilité et compréhensibilité réduites.
- Refactoring
 - Extraire de la classe.

Multifaceted abstraction

- Définition
 - Une abstraction a plus d'une responsabilité
- Symptômes et conséquences
 - Violation de SRP.
- Refactoring
 - Extraire de la classe ou de la superclasse.

Duplicate abstraction

- Définition
 - Deux abstractions ont le même nom ou la même implémentation (cloning).
 - Remarque : Le même nom est permis entre deux classes dans des paquets différents.
- Causes
 - Copier-coller
 - Manque de communication
- Refactoring
 - Extraire de la classe ou de la superclasse.

Deficient encapsulation

- Définition
 - Lorsque une abstraction donne plus de permissions qu'il est nécessaire, p.ex. lorsque tous les attributs sont déclarés comme publiques.
- Symptômes et conséquences
 - Maintenabilité et sécurité réduites.
- Causes
 - Pensée procédurale dans le contexte orienté-objet.
 - Maintenance négligente.
 - Pour faciliter les tests.
- Refactoring
 - Encapsuler les données et fournir des méthodes d'accès.

Unexploited encapsulation

- Définition
 - L'utilisation des vérifications de types explicites lorsque le polymorphisme est disponible.
- Symptômes et conséquences
 - Complexité augmentée.
 - Maintenabilité et compréhensibilité réduites.
- Causes
 - Pensée procédurale dans le contexte orienté-objet.
 - Échec d'application des principes orientés-objet.
- Refactoring
 - Remplacer les vérifications de types par du polymorphisme.

Broken modularization

- Définition
 - Des données ou des méthodes qui doivent être ensemble, par rapport à la similarité sémantique ou à l'utilisation, font parties de plusieurs abstractions.
- Symptômes et conséquences
 - Haut couplage, faible cohésion
 - Maintenabilité est réduite.
 - Performance se détériore.
- Causes
 - Pensée procédurale dans le contexte orienté-objet.
 - Manque de connaissance de la conception existante.
- Refactoring
 - Déplacer des méthodes ou des attributs
 - Imbriquer des classes

Insufficient modularization

- Définition
 - Une abstraction a beaucoup de membres publiques ou des méthodes très complexes.
- Symptômes et conséquences
 - Violation de SRP et d'ISP.
- Causes
 - Contrôle centralisé.
- Refactoring
 - Extraire des méthodes ou attributs de la classe ou de l'interface.

Cyclically-dependend modularization

- Définition
 - Deux abstractions utilisent beaucoup de membres entre elles.
- Symptômes et conséquences
 - Haut couplage
- Causes
 - Passage du pointeur « this »
 - L'abstraction n'est pas conçu correctement.
- Refactoring
 - Déplacer des méthodes ou des attributs
 - Imbriquer des classes

Unfactored hierarchy

- Définition
 - Dans une hiérarchie, Il existe de la duplication entre les classes dérivées ou les classes dérivées et la classe de base.
- Symptômes et conséquences
 - Maintenabilité est réduite.
- Causes
 - Duplication de code (Cloning)
 - Copier-coller
- Refactoring
 - Extraire une superclasse
 - Remonter des méthodes ou des attributs

Broken hierarchy

- Définition
 - La classe de base et les classes dérivées n'ont pas une relation « est-un ».
- Symptômes et conséquences
 - Violation de LSP.
- Causes
 - L'héritage est implémentée pour des raisons d'implémentation, pas de conception.
- Refactoring
 - Remplacer l'héritage par de la délégation
 - Inverser la relation d'héritage

Cyclic hierarchy

- Définition
 - Une classe de base a une association avec une ou plusieurs de ses classes dérivées.
- Symptômes et conséquences
 - Haut couplage
 - Compréhensibilité réduite.
- Causes
 - L'héritage n'est pas conçu correctement.
- Refactoring
 - Extraire une classe.
 - Déplacer des méthodes.
 - Imbriquer des classes.
 - Implémenter un patron Stratégie ou État.

Antipatrons de code

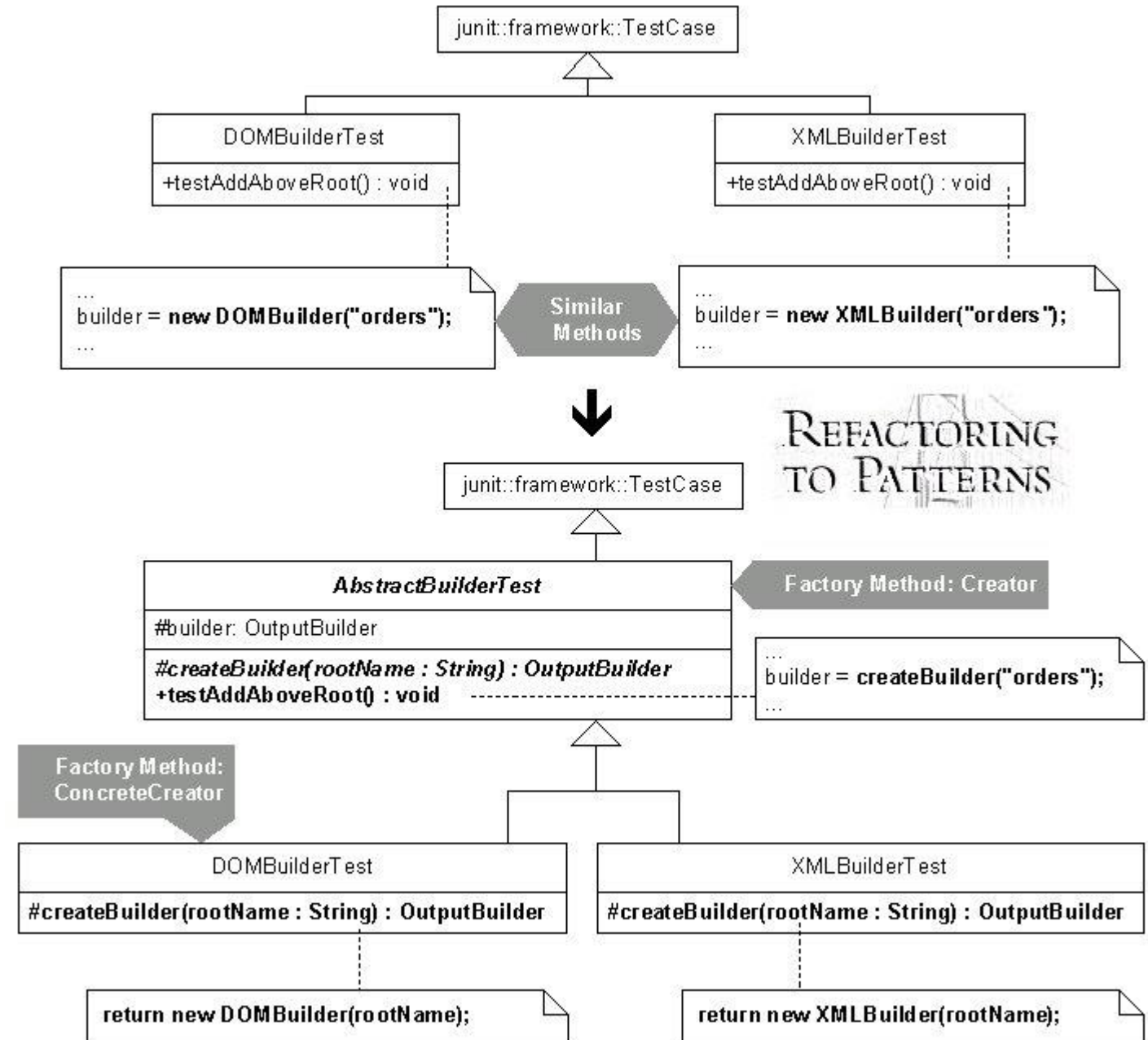
- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Switch Statements
- Message Chains
- Inappropriate Intimacy
- Refused Bequest

Duplicated Code

- Définition
 - Le même code existe à plusieurs endroits dans le système.
- Refactoring
 - Extraire des méthodes
 - Construire une classe générique
 - Extraire des classes
 - Enchaîner les constructeurs
 - Introduire un mécanisme de création d'objet polymorphique grâce au patron Factory Method



Introduce Polymorphic Creation with Factory Method



Long Method

- Définition
 - Une méthode a une longue implémentation qui fait beaucoup de choses.
- Refactoring
 - Extraire des méthodes
 - Décomposer les énoncés conditionnels
 - Assembler une méthode (Compose Method)

Compose Method

```
public void add(Object element) {
    if (!readOnly) {
        int new Size = size + 1;
        if (new Size > elements.length) {
            Object[] new Elements =
                new Object[elements.length + 10];
            for (int i = 0; i < size; i++)
                new Elements[i] = elements[i];
            elements = new Elements;
        }
        elements[size++] = element;
    }
}
```



```
public void add(Object element) {
    if (readOnly)
        return;
    if (atCapacity())
        grow ();
    addElement(element);
}
```

**REFACTORING
TO PATTERNS**

Large Class

- Définition
 - Une classe qui a beaucoup de membres ou beaucoup de responsabilités.
- Refactoring
 - Extraire une/des classe(s)
 - Extraire une/des superclasse(s)
 - Extraire une/des sous-classe(s)

Long Parameter List

- Définition
 - Une méthode a une longue liste des paramètres.
- Refactoring
 - Remplacer des paramètres par une méthode.
 - Introduire des objets en paramètres.

Divergent Change

- Définition
 - Une classe change beaucoup, plusieurs fois, ou chaque fois qu'il y a un changement aux exigences.
- Refactoring
 - Extraire une/des classes

Shotgun Surgery

- Définition
 - Lorsqu'il y a un changement, on doit changer le code à plusieurs endroits.
- Refactoring
 - Déplacer des méthodes.
 - Imbriquer des classes.

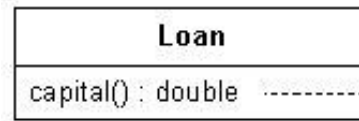
Feature Envy

- Définition
 - Une méthode utilise plus de membres d'autres classes que de la classe à laquelle elle appartient
- Refactoring
 - Déplacer des méthodes.
 - Extraire et déplacer des attributs ou des méthodes.

Switch statements

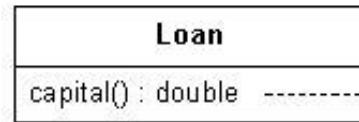
- Définition
 - Une méthode est longue et complexe au point de contenir plusieurs instructions « switch » ou conditionnelles.
- Refactoring
 - Remplacer des vérifications de types par des sous-classes.
 - Remplacer des verifications de types par un instance du patron Stratégie ou État.
 - Remplacer des énoncés conditionnels par du polymorphisme.

Replace Conditional Logic with Strategy



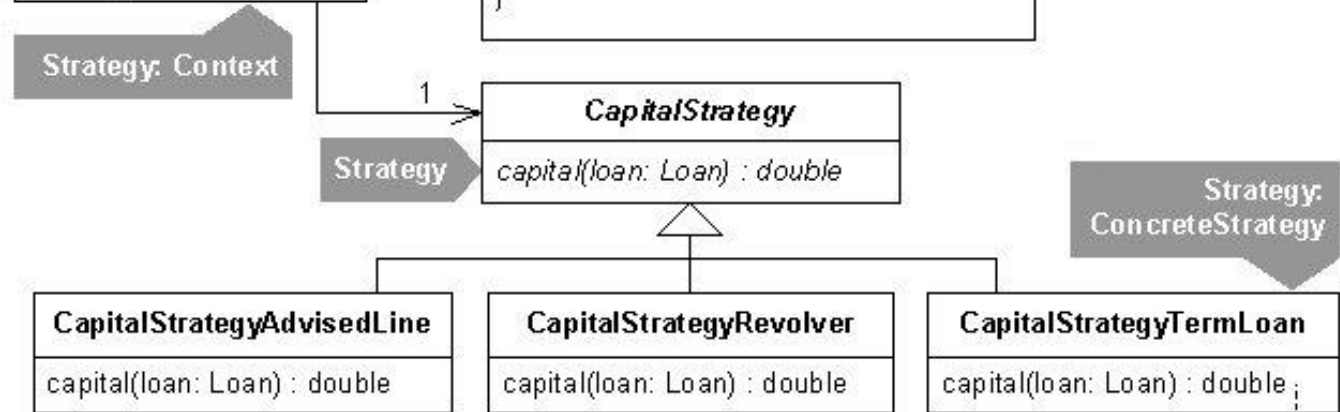
```

capital() {
    if (expiry == null && maturity != null)
        return commitment * duration() * riskFactor();
    if (expiry != null && maturity == null) {
        if (getUnusedPercentage() != 1.0)
            return commitment * getUnusedPercentage()
                * duration() * riskFactor();
        else
            return (outstandingRiskAmount() * duration() * riskFactor())
                + (unusedRiskAmount() * duration() * unusedRiskFactor());
    }
    return 0.0;
}
    
```



```

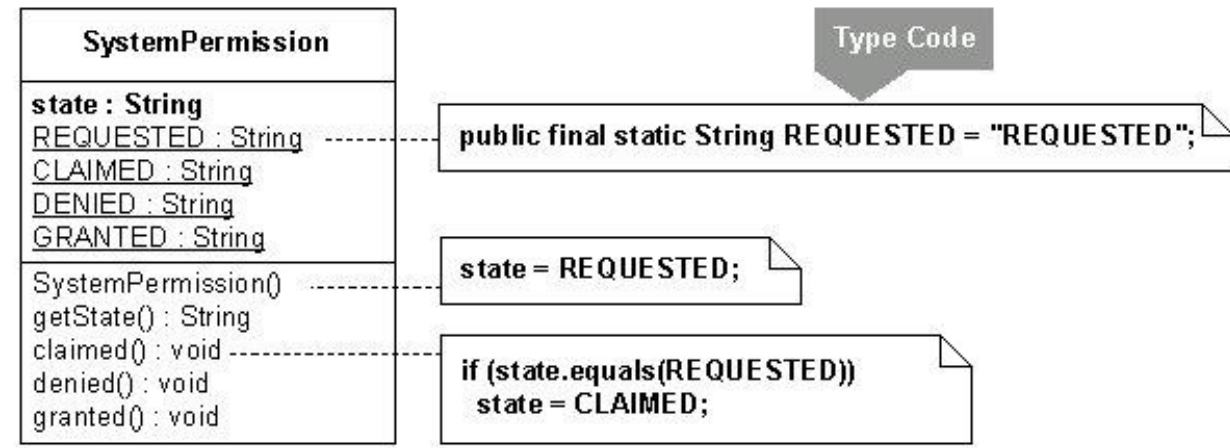
capital() {
    return capitalStrategy.capital(this);
}
    
```



```

capital(Loan loan) {
    return loan.getCommitment() * duration(loan) * riskFactorFor(loan);
}
    
```

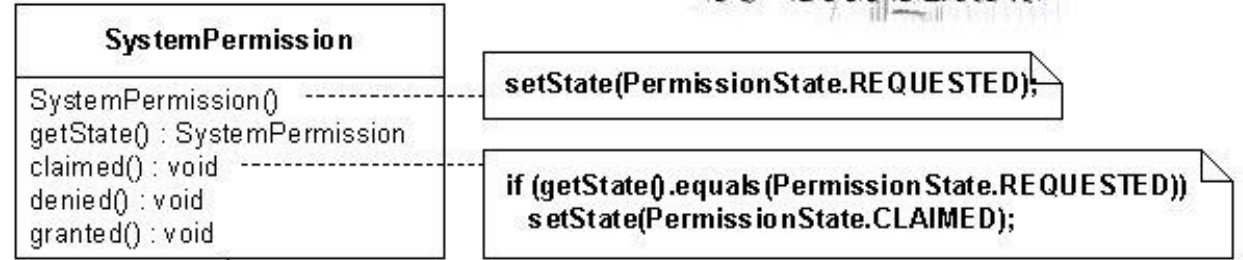
Replace Type Code with Class



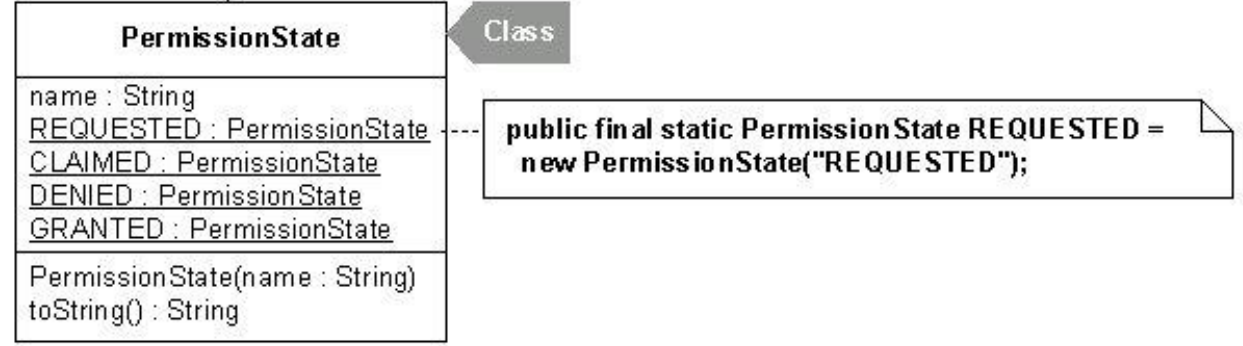
Type Code



REFACTORING TO PATTERNS



1



Class

Message Chains

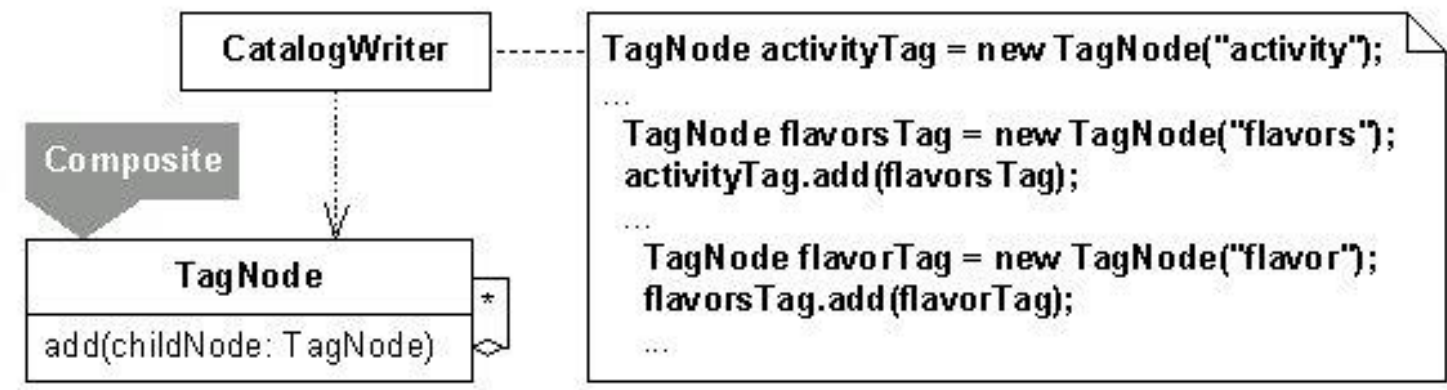
- Définition
 - Il existe de longues chaînes d'invocations de méthodes en cascade.
- Refactoring
 - Extraire et déplacer des méthodes.

Inappropriate Intimacy

- Définition
 - Une classe a beaucoup d'associations avec les membres privés d'une autre classe.
- Refactoring
 - Déplacer des méthodes ou des attributs.
 - Extraire une/des classe(s).
 - Cacher les délégations.
 - Encapsuler un Composite avec un patron Builder.



Encapsulate Composite with Builder



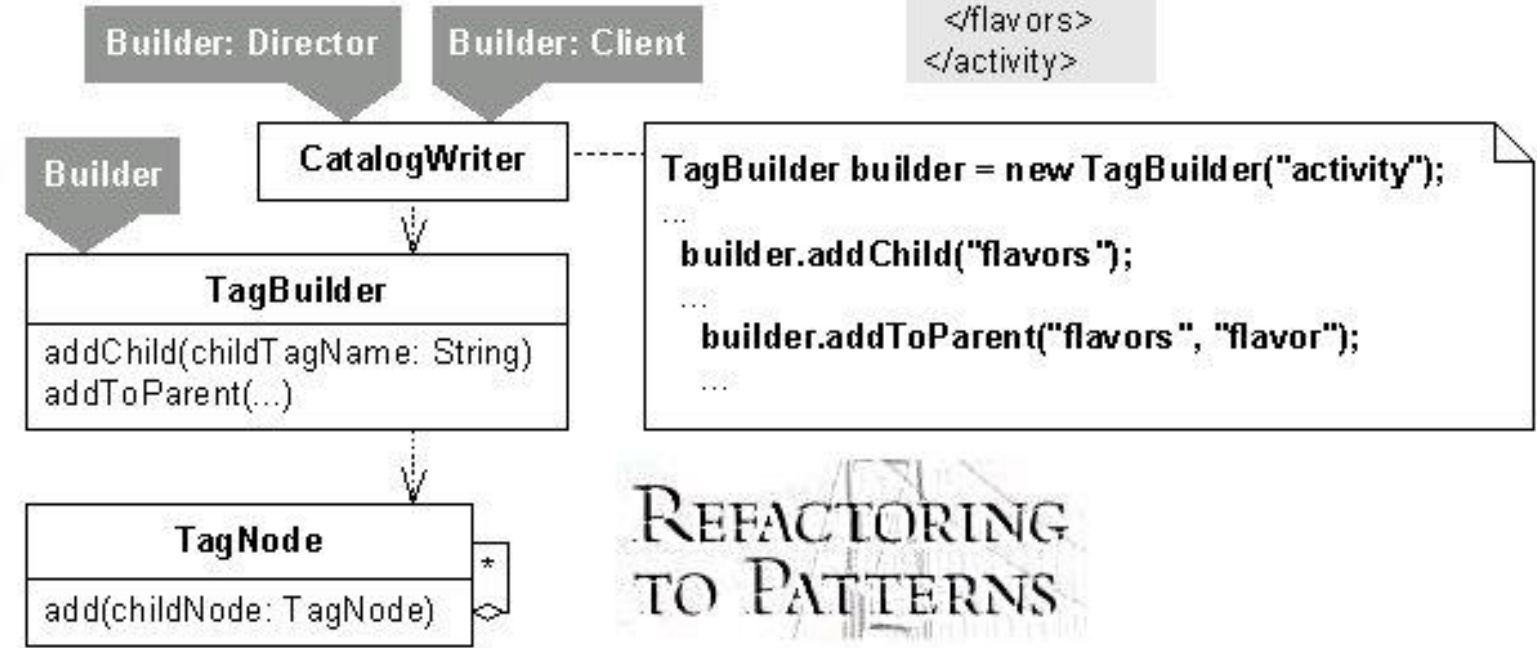
```

TagNode activityTag = new TagNode("activity");
...
TagNode flavorsTag = new TagNode("flavors");
activityTag.add(flavorsTag);
...
TagNode flavorTag = new TagNode("flavor");
flavorsTag.add(flavorTag);
...
  
```



```

<activity>
  <flavors>
    <flavor>
      ...
    </flavor>
  </flavors>
</activity>
  
```



```

TagBuilder builder = new TagBuilder("activity");
...
builder.addChild("flavors");
...
builder.addToParent("flavors", "flavor");
...
  
```



Refused Bequest

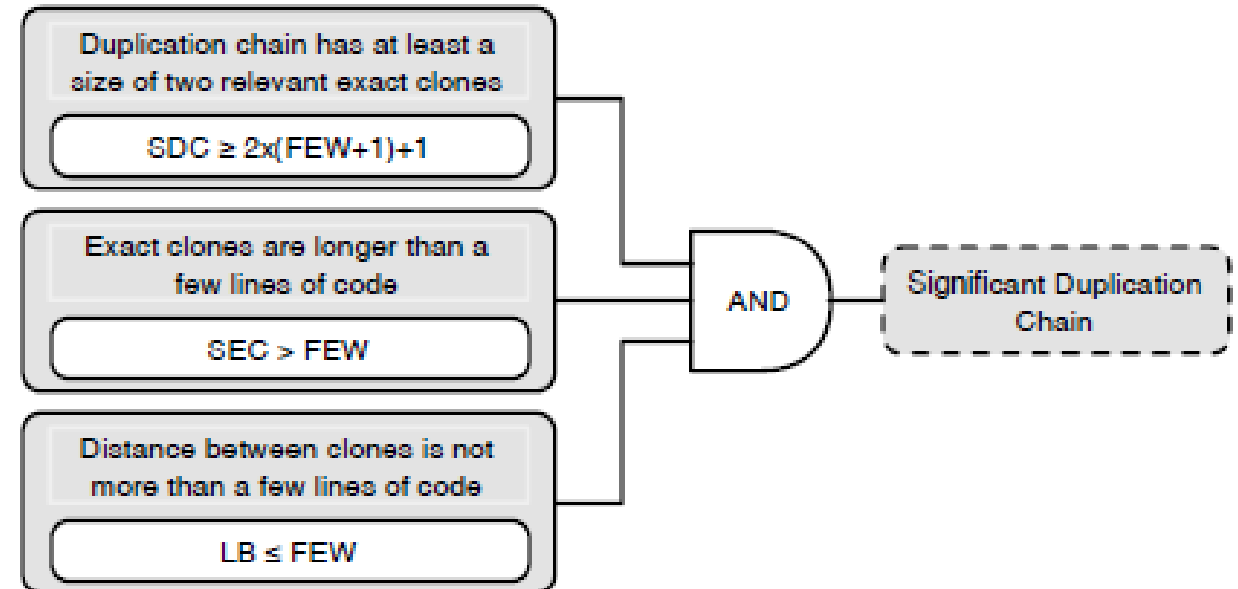
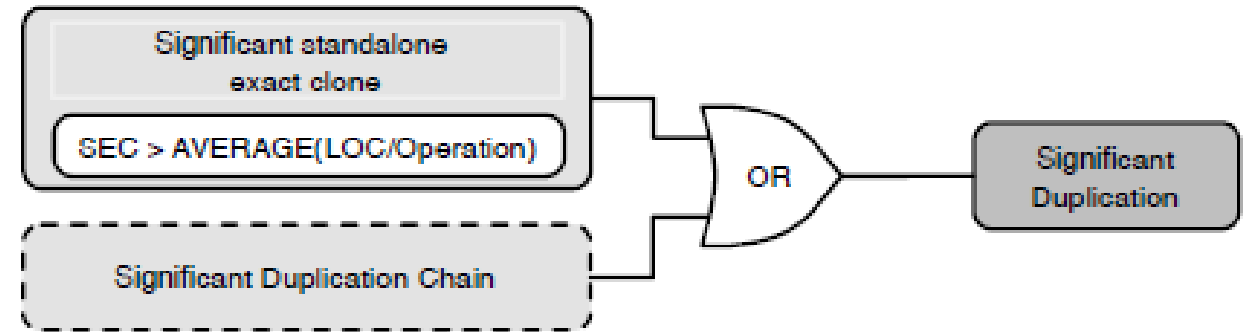
- Définition
 - Une classe dérivée n'a pas besoin d'hériter de certains membres de la classe de base.
- Refactoring
 - Descendre certaines méthodes ou attributs dans les sous-classes.

Détection des antipatrons

- Il est possible de détecter certains antipatrons de code en se basant sur les métriques.
- On peut définir des seuils sur les métriques, combiner les conditions pertinentes et détecter les antipatrons.
- La définition de seuils
 - Statistiques : $LOW = AVG - ST.DEV.$, $HIGH = AVG + ST.DEV.$,
 $VERY_HIGH = (AVG + ST.DEV.) * 1.5$
 - Sémantiques : un quart, un tier, la moitié, deux tiers, trois quarts
 - Généraux : 0 = NONE, 1 = ONE, 2-5 = FEW, 7 = Short Memory Cap, >7 = MANY

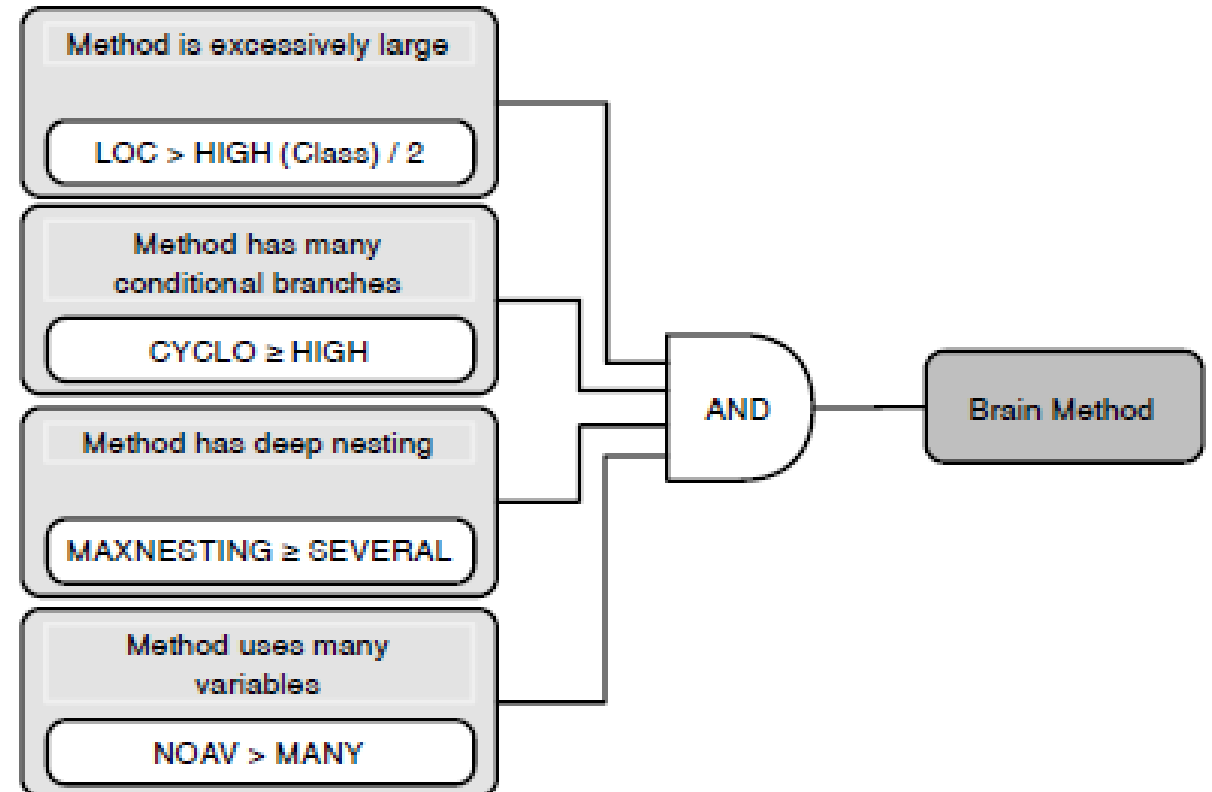
Duplicated Code

- SEC = Size of exact clone
 - Nombre des lignes d'un clone exact.
- LB = Line Bias
 - Distance de deux clones exacts.
 - Nombre des lignes pas égales entre les deux clones.
- SDC = Size of Duplication Chain
 - Nombre des clones exacts dans une chaîne de duplication.
 - Chaîne de duplication = des clones exacts qui sont très proches selon le LB.



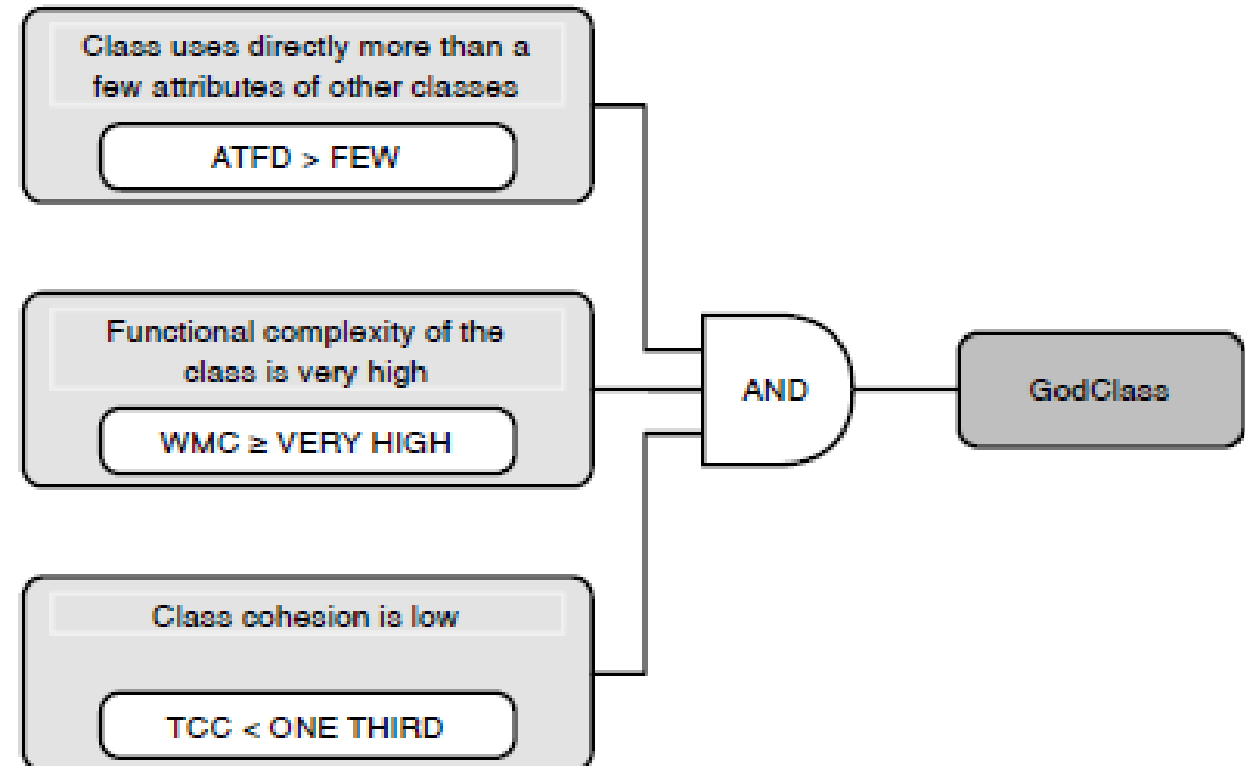
Long Method

- NOAV = Number of Accessed Variables
- MAXNESTING = Maximum Nesting Level



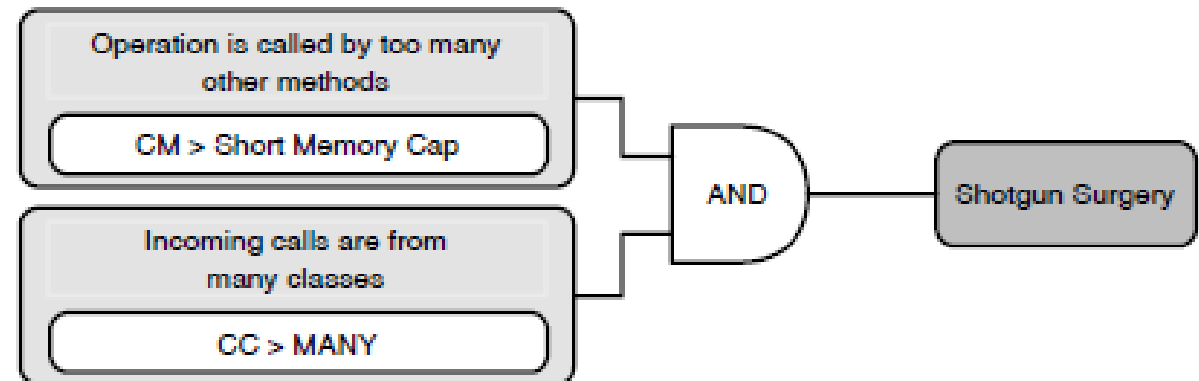
Large Class

- ATFD = Access to Foreign Data
 - Nombre de classes desquelles la classe accède à des attributs.



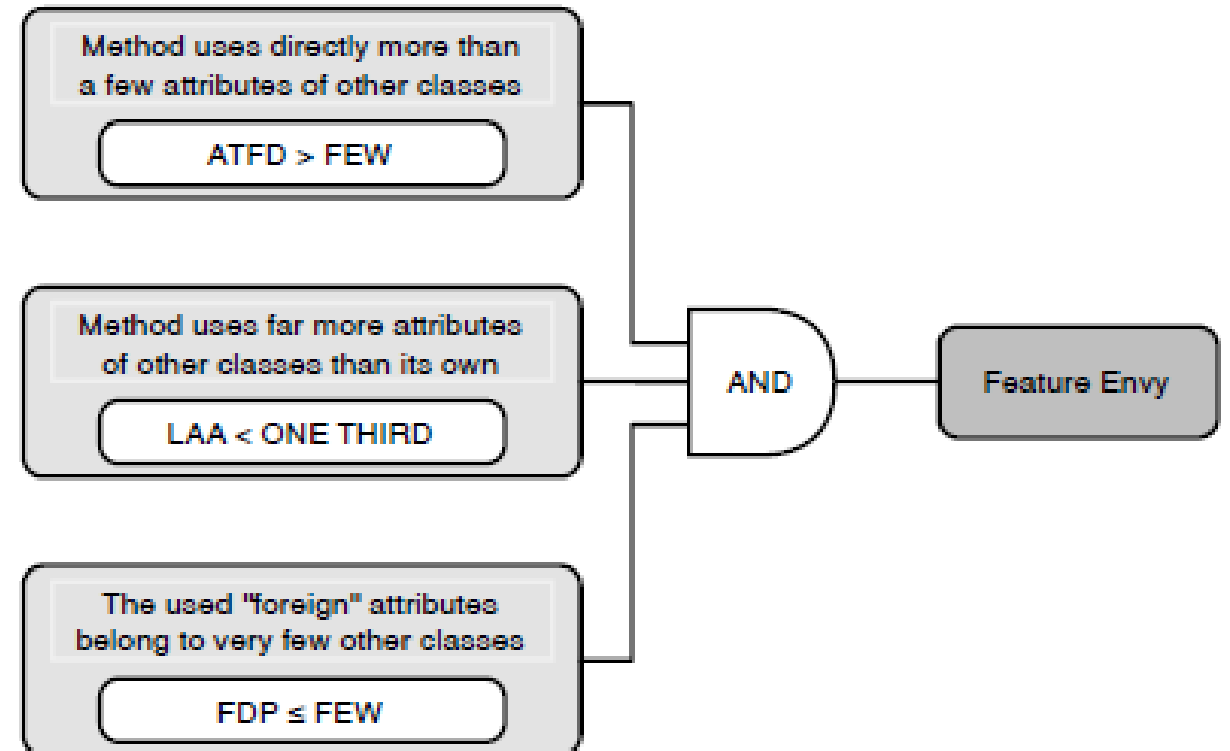
Shotgun Surgery

- CM = Changing Methods
 - Nombre des méthodes qui appellent la méthode mesurée.
- CC = Changing Classes
 - Nombre des classes où les CM sont définies.



Feature Envy

- LAA = Locality of Attribute Accesses
 - Le nombre des attributs de la classe de la méthode divisés par tous les attributs accédés par la méthode.
- FDP = Foreign Data Providers
 - Le nombre des classes où les attributs d'ATFD sont définis.



Refused Bequest

- NProtM = Number of Protected Members
- BUR = Base-class Usage Ratio
 - Ratio de l'usage des membres protégés.
- BOvR = Base-class Overriding Ratio
 - Ratio des membres surchargés.
- AMW = Average Method Weight
 - La complexité moyenne de toutes les méthodes d'une classe.

