DECOR : DEtection et CORrection des défauts dans les systèmes orientés objet

Soutenance de thèse

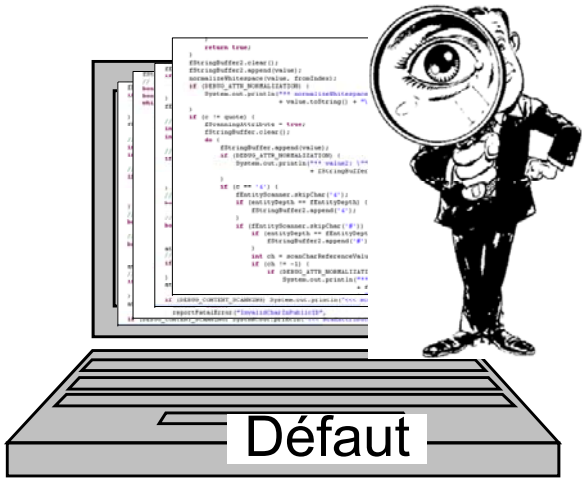
Naouel MOHA

Ptidej Team, GEODES, Université de Montréal, Canada
LIFL, INRIA Lille - Nord Europe / ADAM Team, Université de Lille, France

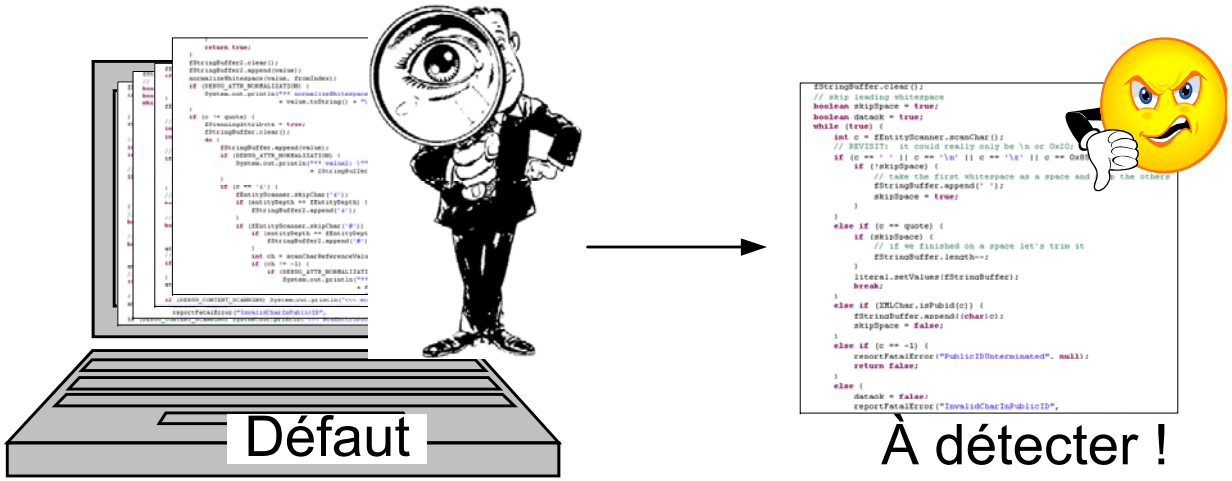
Directeur	:	Yann-Gaël Guéhéneuc
Co-directrice	:	Laurence Duchien
Co-encadrante	:	Anne-Françoise Le Meur

26 août 2008

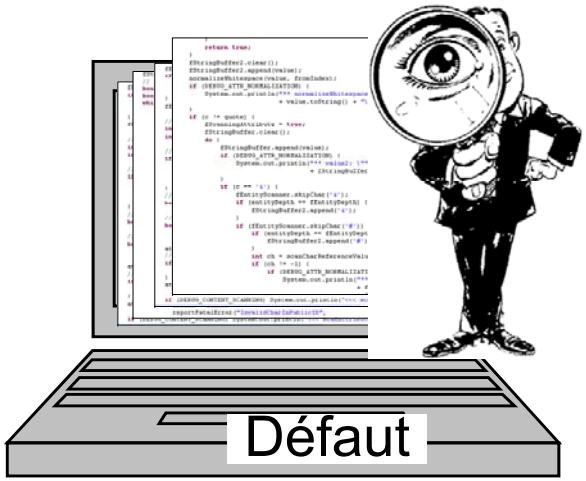
Contexte

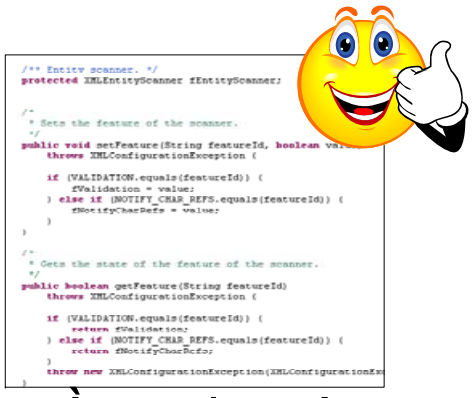


Contexte

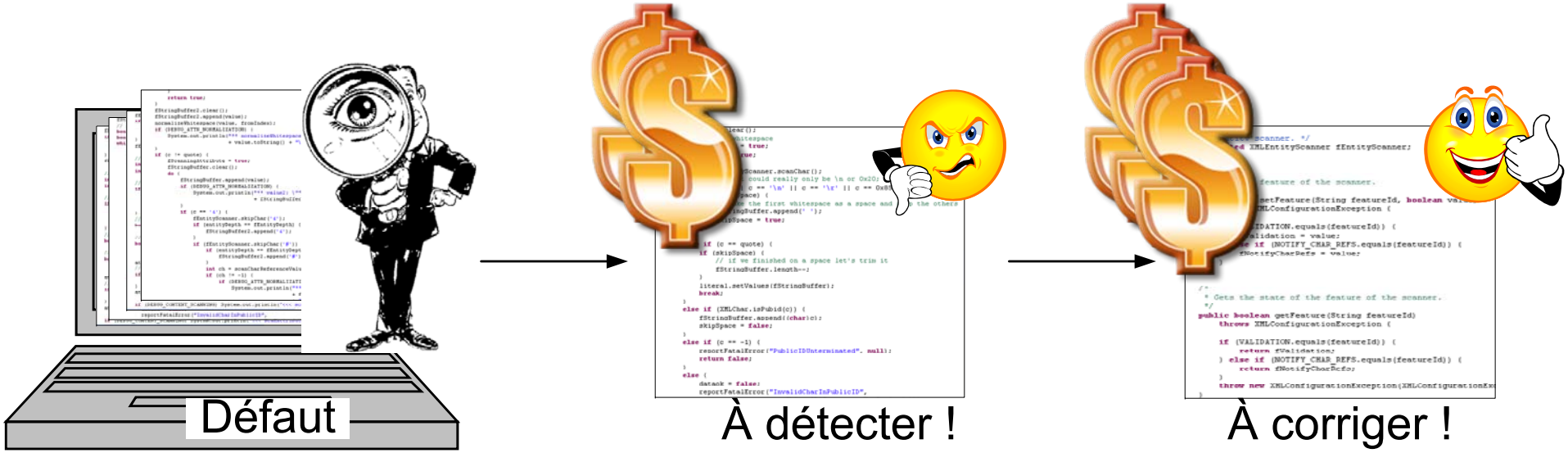


Contexte

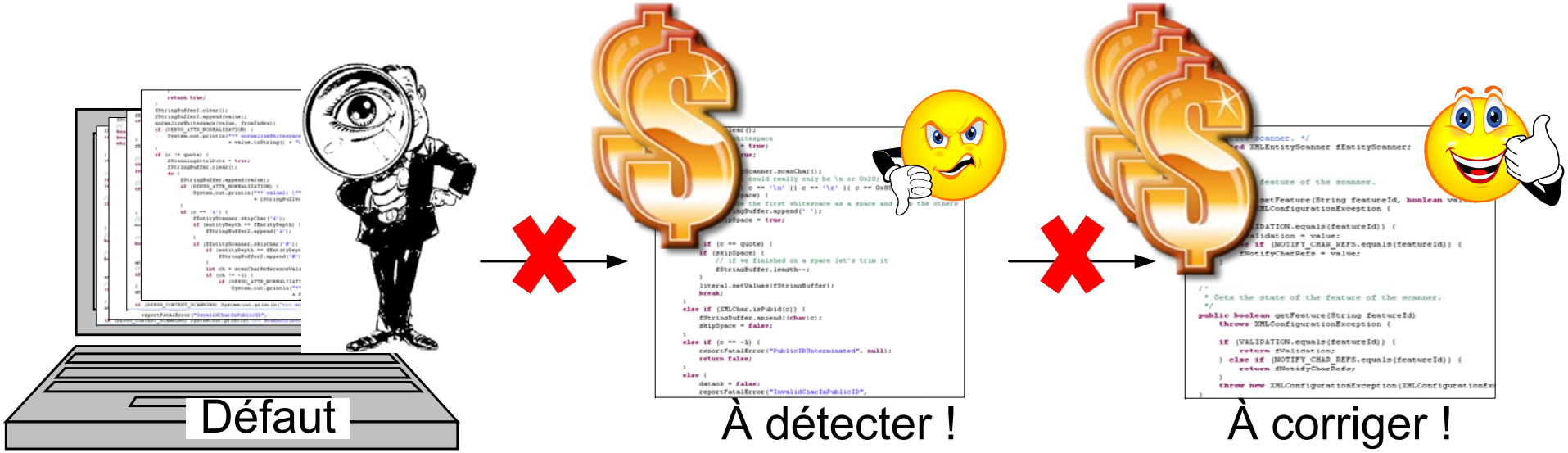




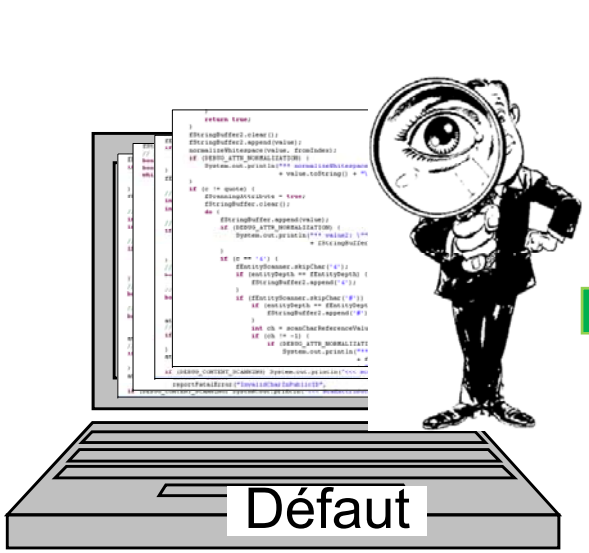
Contexte



Contexte



Contexte



Plan

- Contexte
- Thèse
- État de l’art et contributions
- Détection
- Correction
- Conclusion et perspectives

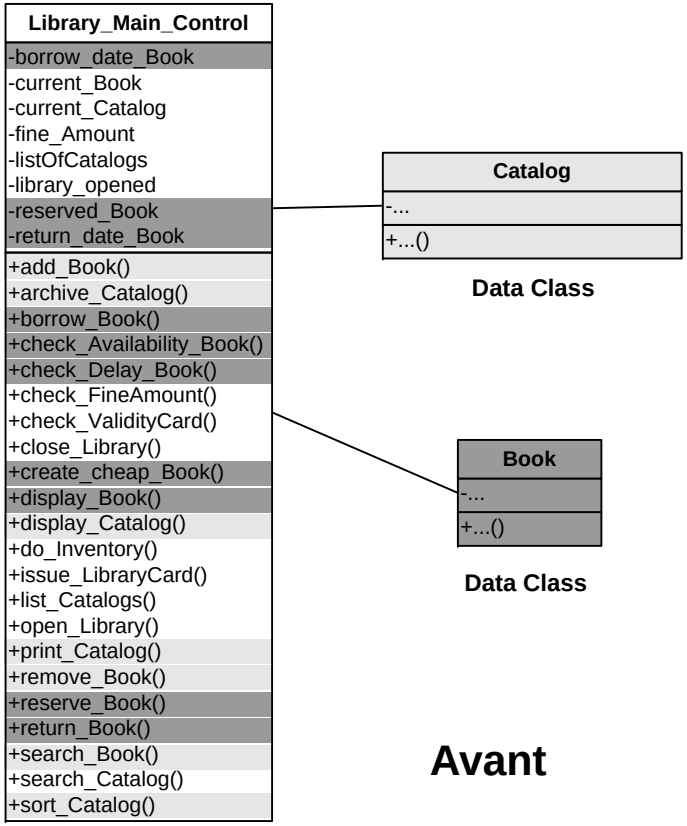
Thèse

« Définir une approche globale pour la détection et la correction des défauts »

- Nos contributions
  - Méthode DECOR
  - Technique de détection DETEX (DETection EXpert)
  - Technique de correction COREX (CORection EXpert)

L'exemple du Blob

Exemple extrait du livre "*AntiPatterns* in Project Management" de W. Brown *et al.* 1998

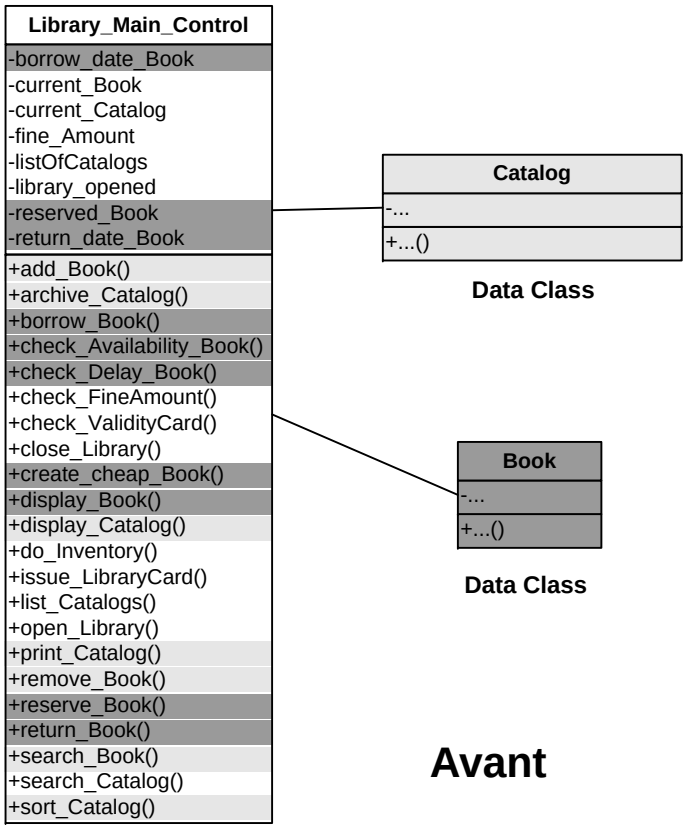


Avant

- Large classe complexe
- Petites classes de données
- Non orienté objet

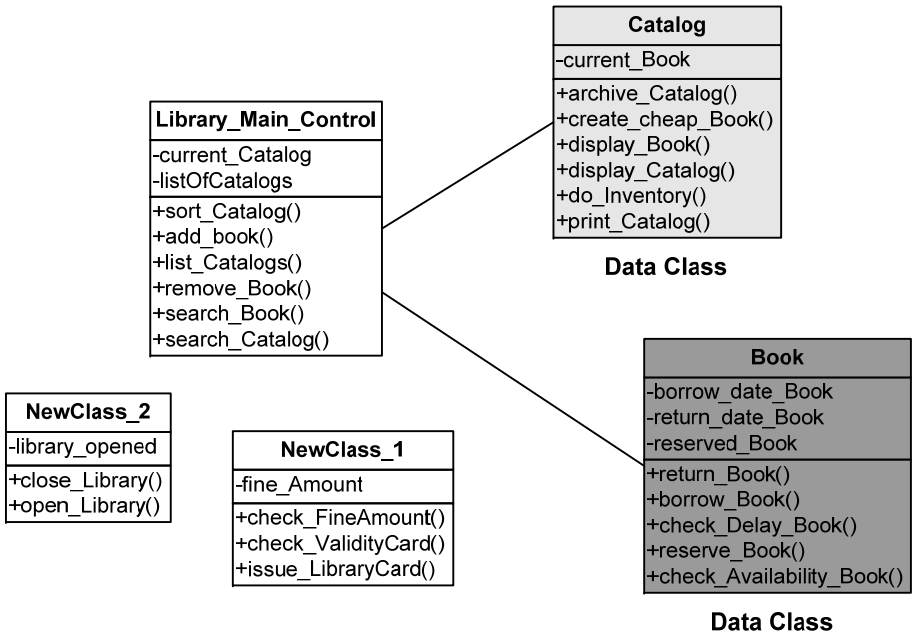
L'exemple du Blob

Exemple extrait du livre "*AntiPatterns* in Project Management" de W. Brown *et al.* 1998



Avant

- Large classe complexe
- Petites classes de données
- Non orienté objet

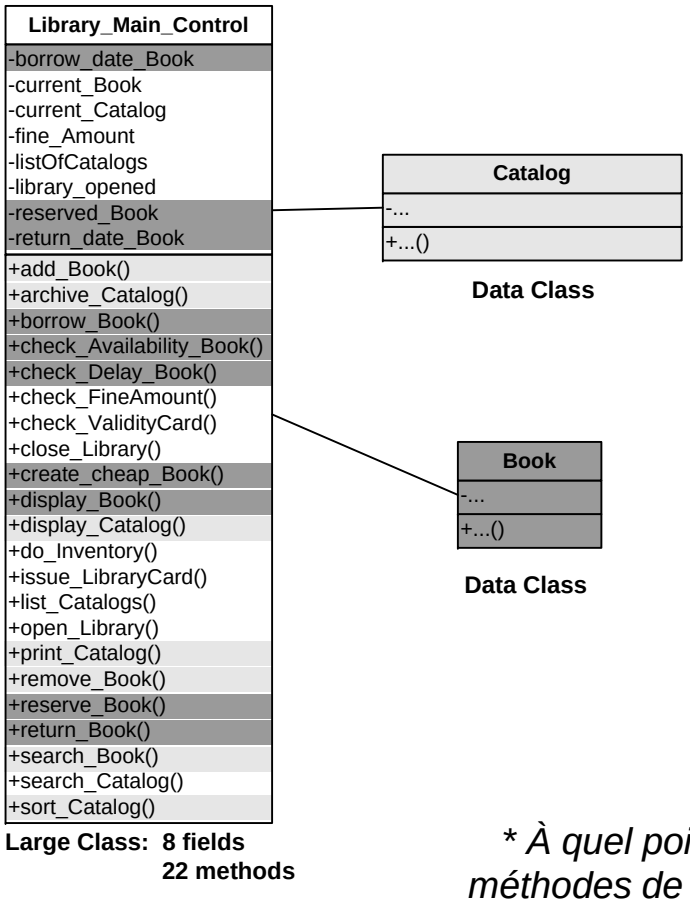


Après

- Large classe devient moins complexe
- Classes de données avec plus de comportement
- Plus orienté objet

L'exemple du Blob

*Comment **détecter** ce défaut ?*



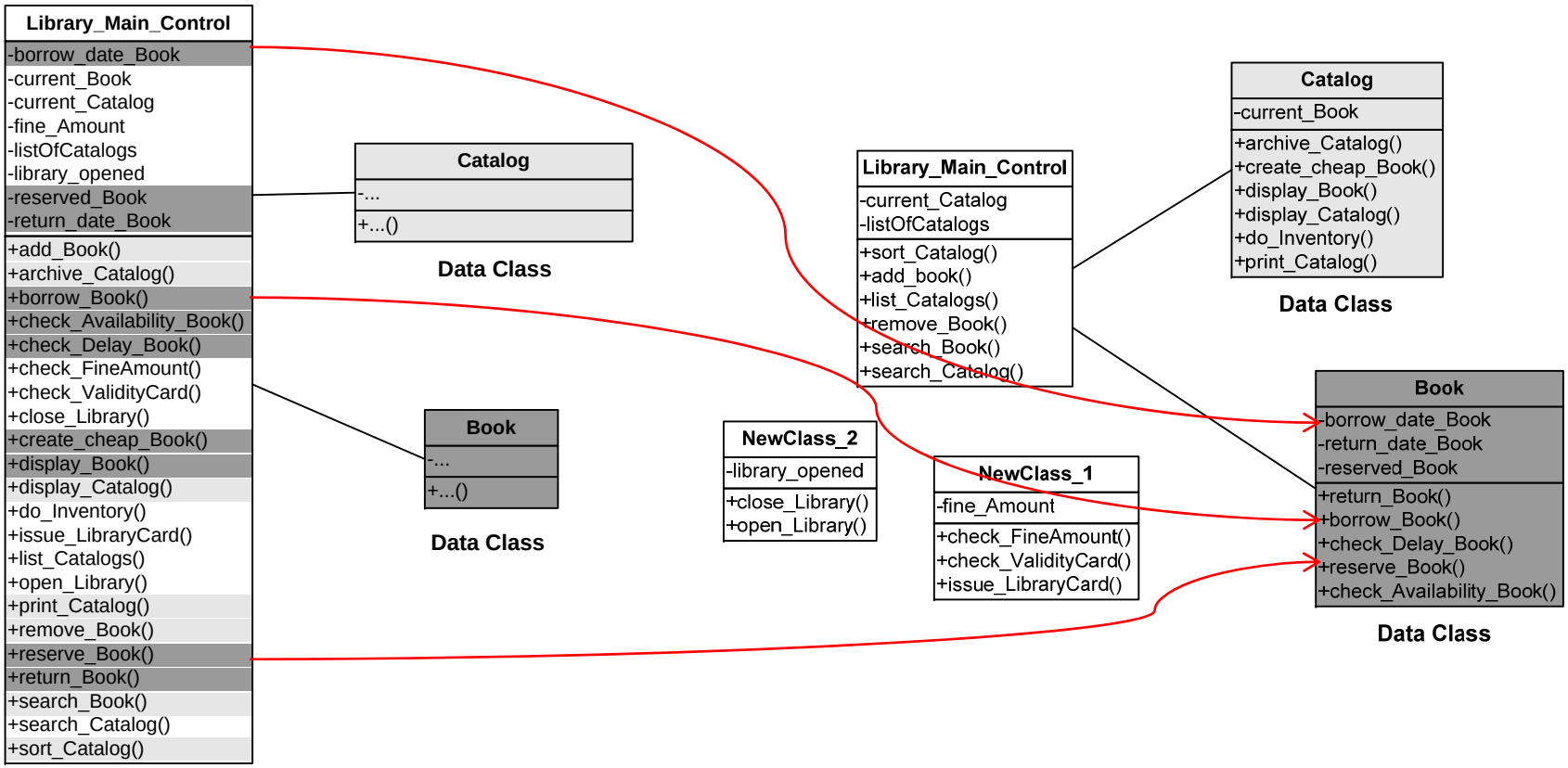
- Règles de détection
  - Nombre important de méthodes et d'attributs
  - Faible cohésion*
  - Fort couplage**
  - Classes de données

** À quel point les méthodes sont étroitement liées aux attributs et aux méthodes de la classe.*

*** Le degré de dépendance des services fournis aux autres classes.*

L'exemple du Blob

Comment corriger ce défaut ?



- Redistribuer les éléments parmi les classes existantes (avec possiblement de nouvelles classes) pour augmenter la cohésion et/ou diminuer le couplage

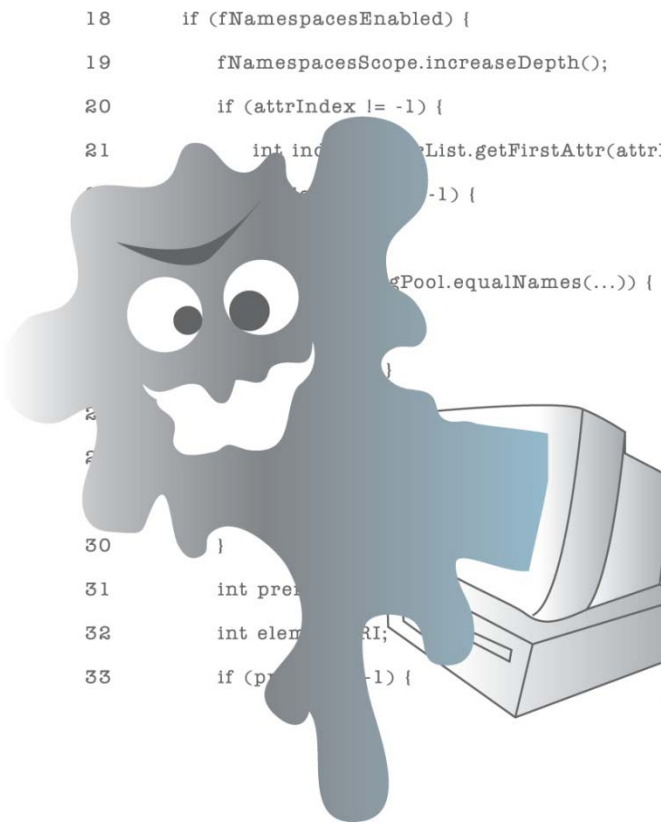
Quels types de défauts ?

- Patrons de conception sont de “bonnes” solutions à des problèmes de conception récurrents

- Défauts de conception
  - sont de “mauvaises” solutions à des problèmes récurrents
  - 2 catégories:
    - Défauts de haut niveau : anti-patrons [Brown 98]
    - Défauts de bas niveau : mauvaises odeurs [Fowler 99]

Quels types de défauts ?

- 2 exemples d'anti-patrons [Brown 98]
  - Blob *(God Class)*



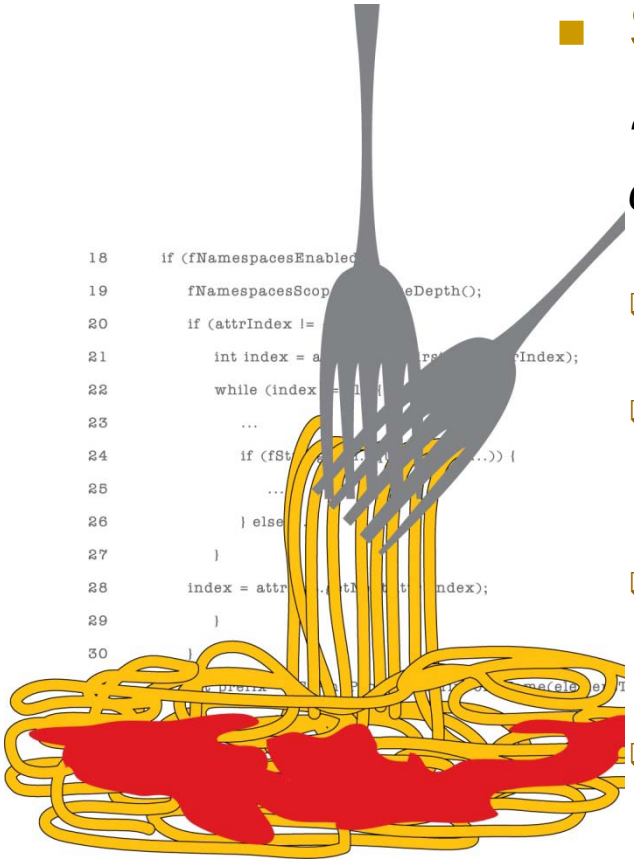
" Procedural-style design leads to one object with a lion's share of the responsibilities while most other objects only hold data or execute simple processes "

- Conception procédurale en programmation OO
- Large classe contrôleur
- Beaucoup d'attributs et méthodes avec une faible cohésion*
- Dépend de classes de données

** À quel point les méthodes sont étroitement liées aux attributs et aux méthodes de la classe.*

Quels types de défauts ?

- 2 exemples d'anti-patrons [Brown 98]



- Spaghetti Code

" Ad hoc software structure makes it difficult to extend and optimize code. "

  - Conception procédurale en programmation OO
  - Manque de structure : pas d'héritage, pas de réutilisation, pas de polymorphisme
  - Noms des classes suggèrent une programmation procédurale
  - Longues méthodes sans paramètres avec une faible cohésion
  - Utilisation excessive de variables globales

Plan

- Contexte
- Thèse
- État de l’art et contributions
- Détection
- Correction
- Conclusion et perspectives

État de l'art

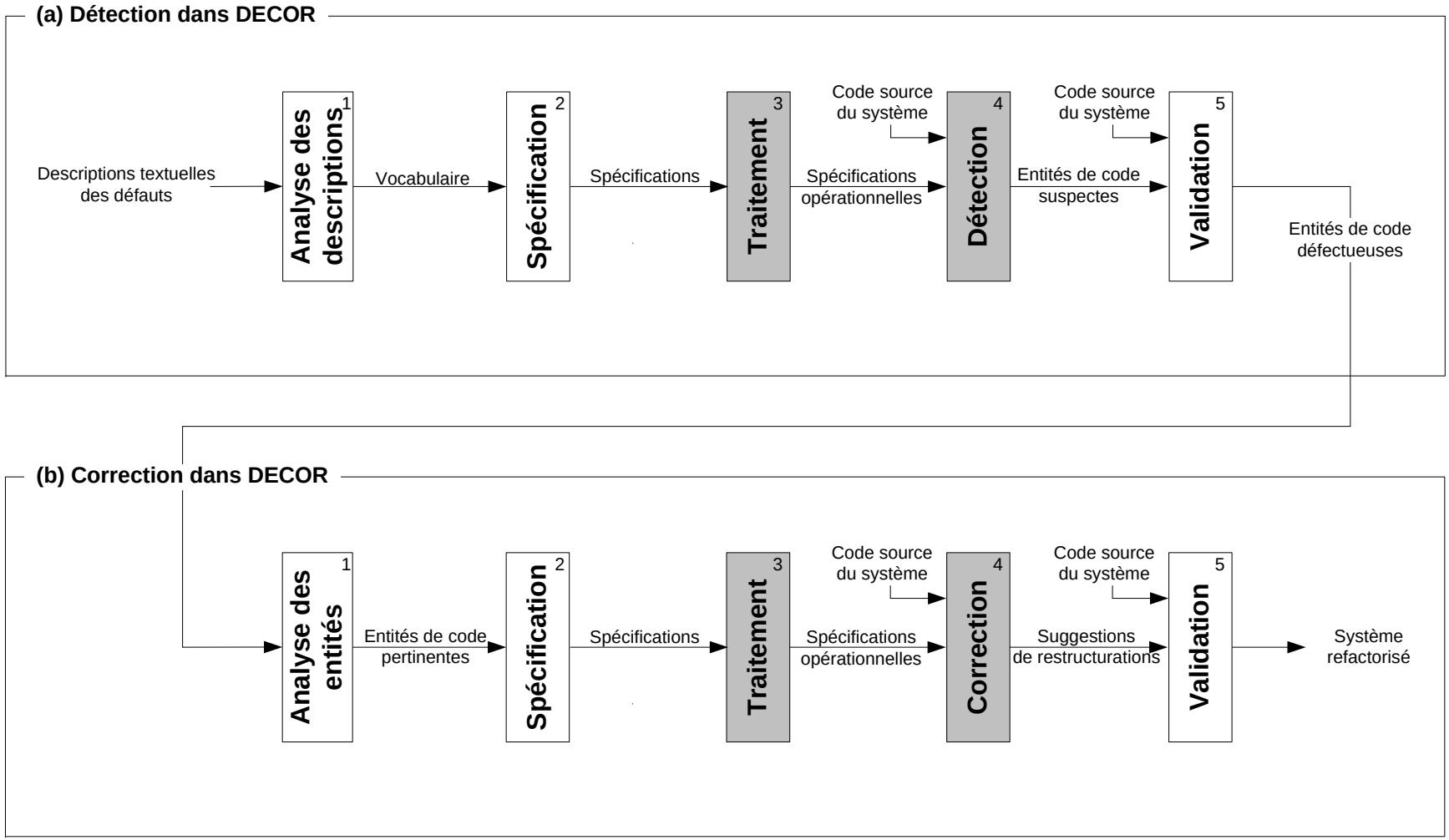
- Constat

  Absence d'une solution complète pour la détection et la correction

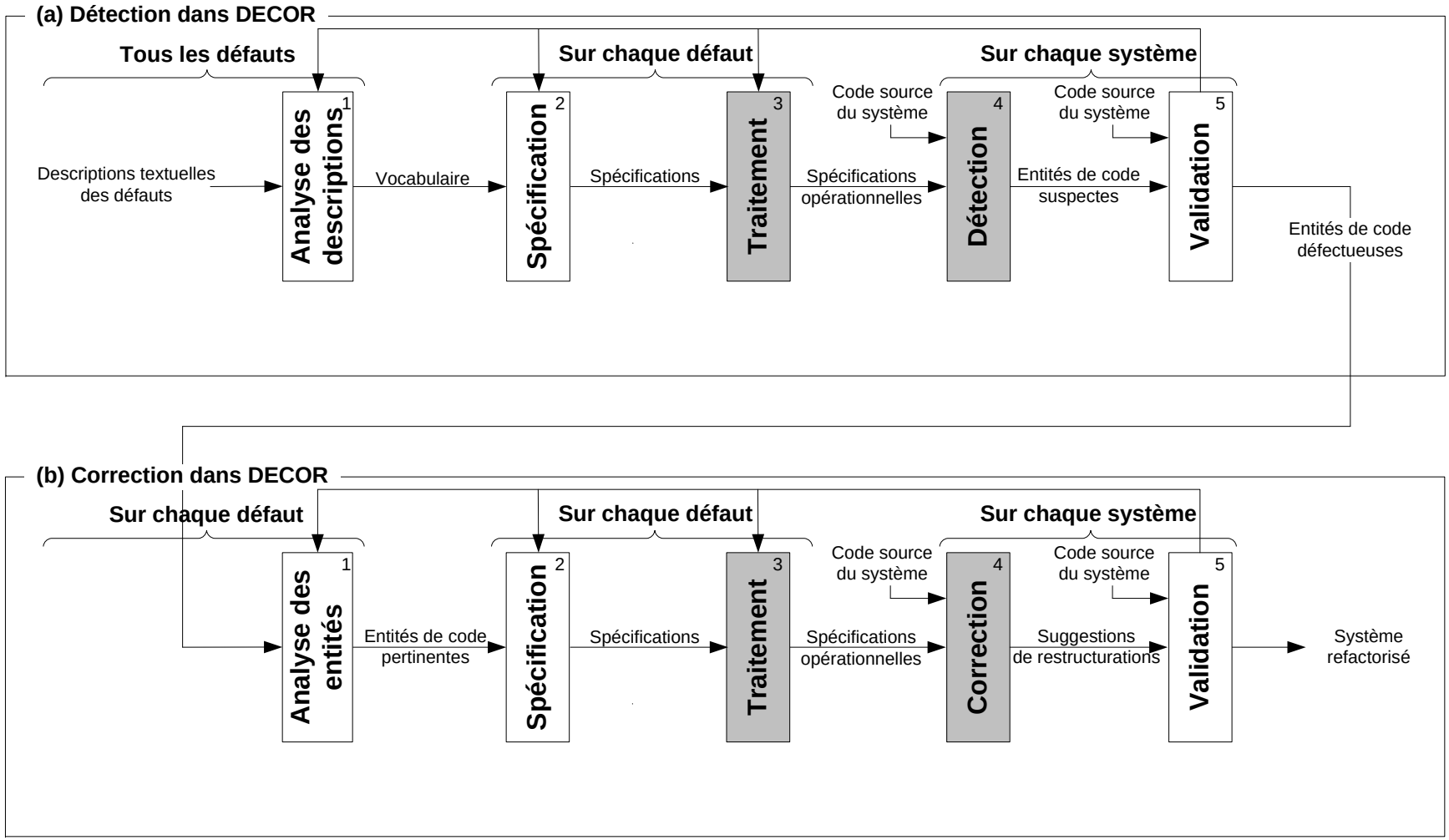
- Solution

  Méthode DECOR

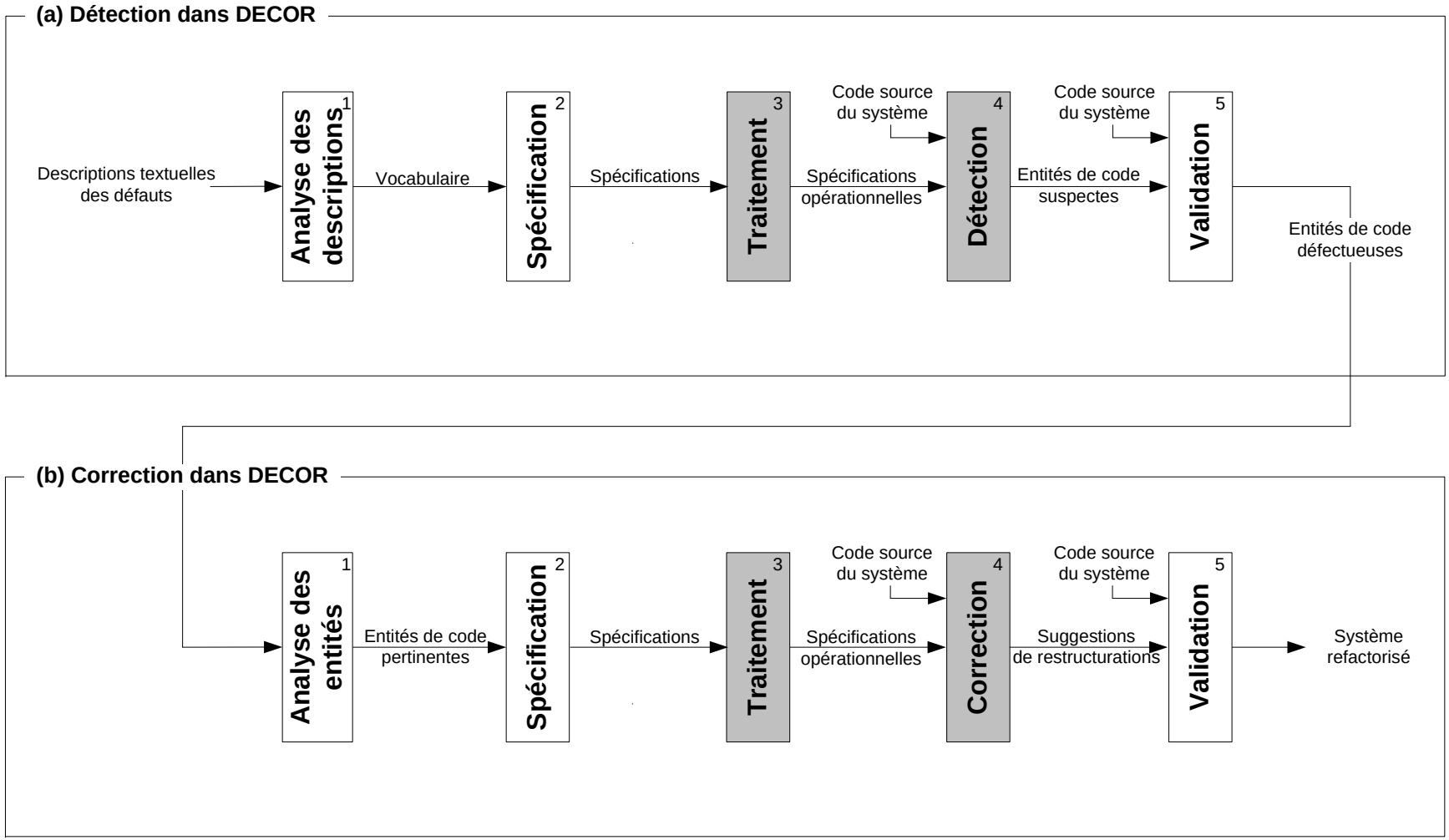
État de l'art


(a) Détection dans DECOR
Descriptions textuelles des défauts
1
Analyse des descriptions
Vocabulaire
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Détection
Code source du système
Entités de code suspectes
5
Validation
Entités de code défectueuses
(b) Correction dans DECOR
1
Analyse des entités
Entités de code pertinentes
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Correction
Code source du système
Suggestions de restructurations
5
Validation
Système refactorisé

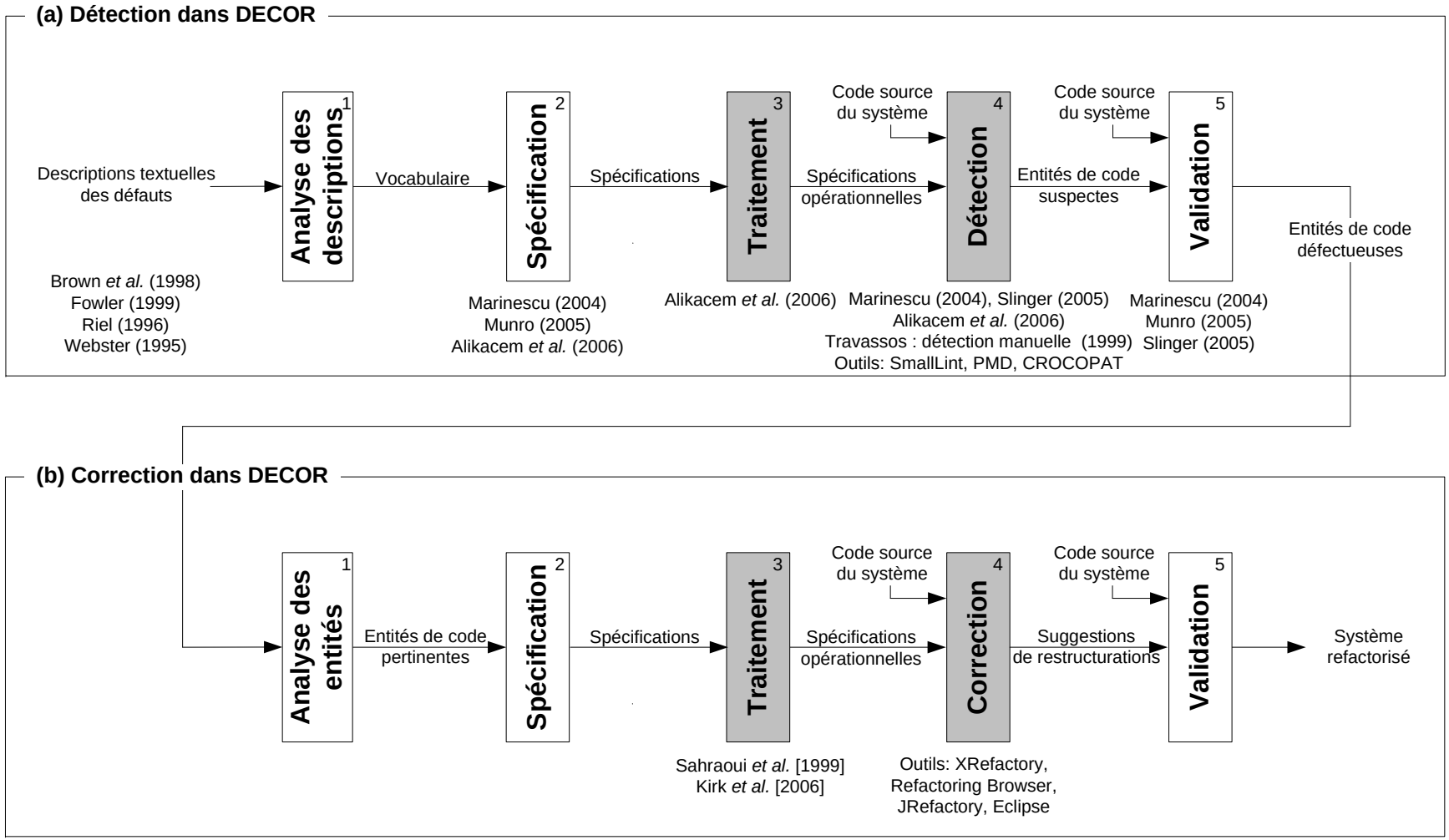
État de l'art


(a) Détection dans DECOR
Tous les défauts
Sur chaque défaut
Sur chaque système
Descriptions textuelles des défauts
1
Analyse des descriptions
Vocabulaire
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Détection
Code source du système
Entités de code suspectes
5
Validation
Entités de code défectueuses
(b) Correction dans DECOR
Sur chaque défaut
Sur chaque défaut
Sur chaque système
1
Analyse des entités
Entités de code pertinentes
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Correction
Code source du système
Suggestions de restructurations
5
Validation
Système refactorisé

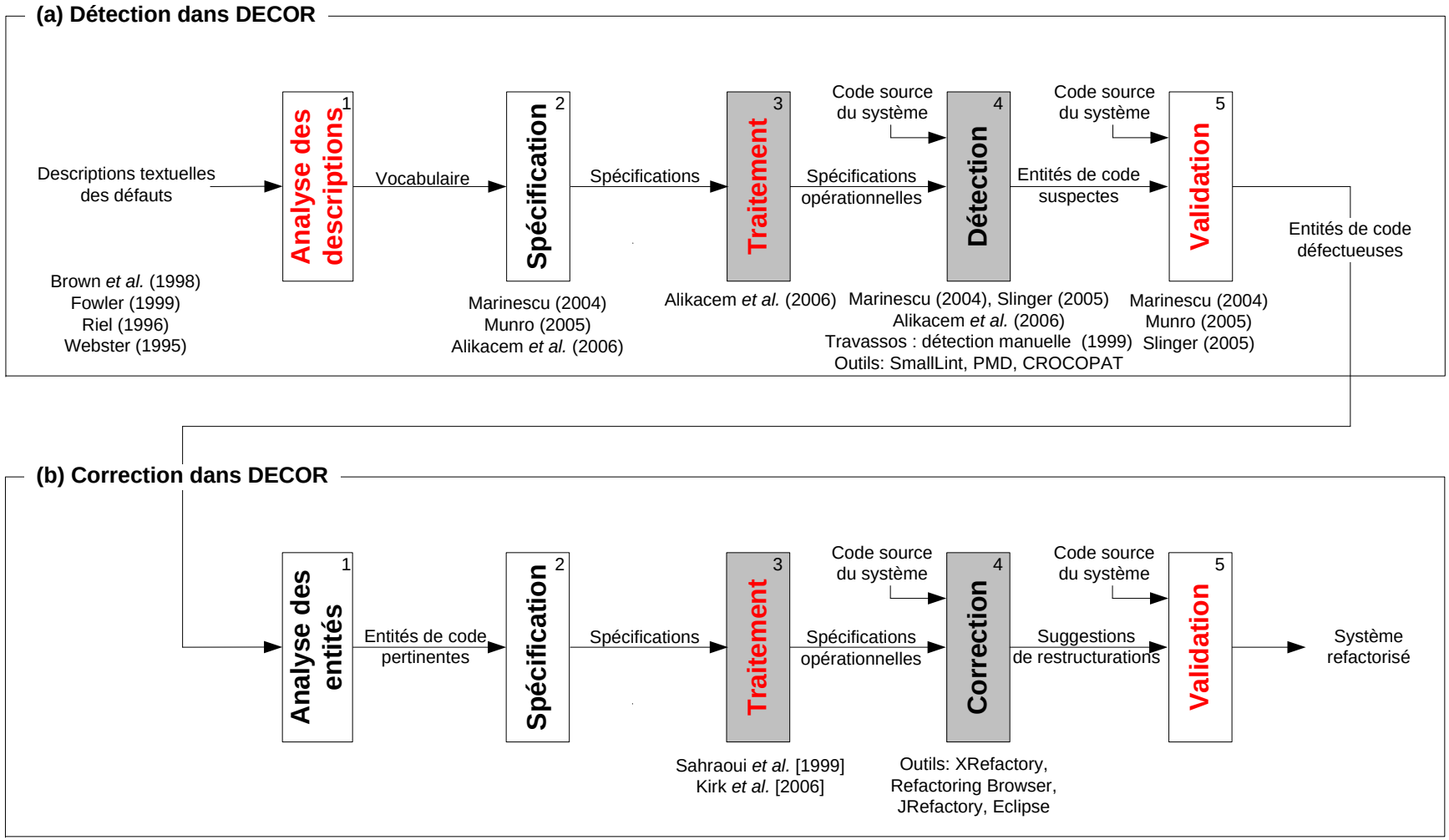
État de l'art


(a) Détection dans DECOR
Descriptions textuelles des défauts
1
Analyse des descriptions
Vocabulaire
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Détection
Code source du système
Entités de code suspectes
5
Validation
Entités de code défectueuses
(b) Correction dans DECOR
1
Analyse des entités
Entités de code pertinentes
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Correction
Code source du système
Suggestions de restructurations
5
Validation
Système refactorisé

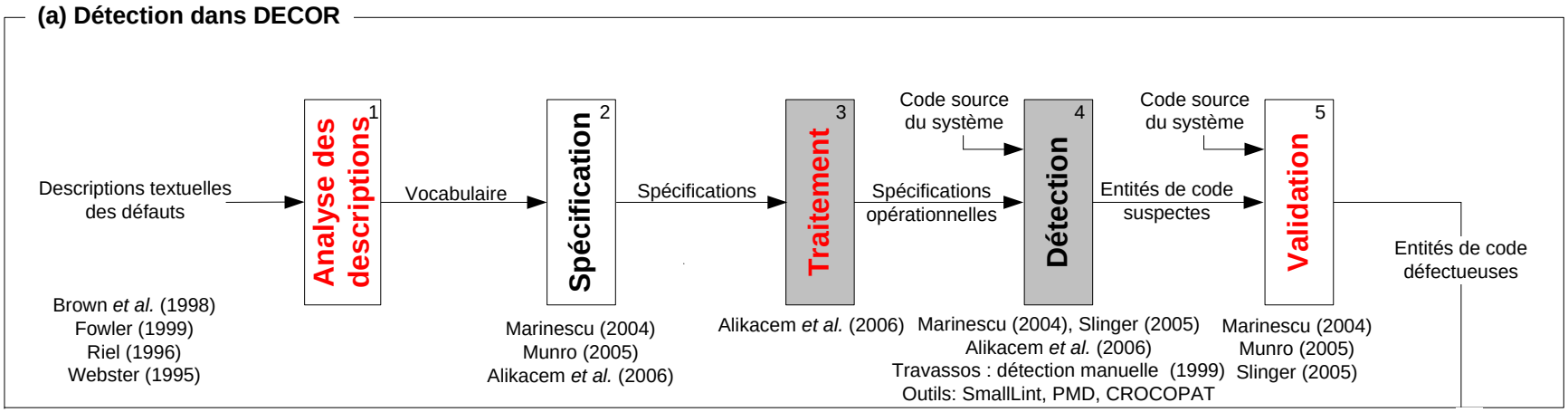
État de l'art


(a) Détection dans DECOR
Descriptions textuelles des défauts
1
Analyse des descriptions
Vocabulaire
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Détection
Code source du système
Entités de code suspectes
5
Validation
Entités de code défectueuses
Brown et al. (1998)
Fowler (1999)
Riel (1996)
Webster (1995)
Marinescu (2004)
Munro (2005)
Alikacem et al. (2006)
Alikacem et al. (2006)
Marinescu (2004), Slinger (2005)
Alikacem et al. (2006)
Travassos : détection manuelle (1999)
Outils: SmallLint, PMD, CROCOPAT
Marinescu (2004)
Munro (2005)
Slinger (2005)
(b) Correction dans DECOR
1
Analyse des entités
Entités de code pertinentes
2
Spécification
Spécifications
3
Traitement
Code source du système
Spécifications opérationnelles
4
Correction
Code source du système
Suggestions de restructurations
5
Validation
Système refactorisé
Sahraoui et al. [1999]
Kirk et al. [2006]
Outils: XRefactory, Refactoring Browser, JRefactory, Eclipse

État de l'art


(a) Détection dans DECOR
Descriptions textuelles des défauts
Brown et al. (1998)
Fowler (1999)
Riel (1996)
Webster (1995)
1
Analyse des descriptions
Vocabulaire
2
Spécification
Spécifications
Marinescu (2004)
Munro (2005)
Alikacem et al. (2006)
3
Traitement
Alikacem et al. (2006)
Code source du système
Spécifications opérationnelles
4
Détection
Marinescu (2004), Slinger (2005)
Alikacem et al. (2006)
Travassos : détection manuelle (1999)
Outils: SmallLint, PMD, CROCOPAT
Code source du système
Entités de code suspectes
5
Validation
Marinescu (2004)
Munro (2005)
Slinger (2005)
Entités de code défectueuses
(b) Correction dans DECOR
1
Analyse des entités
Entités de code pertinentes
2
Spécification
Spécifications
3
Traitement
Sahraoui et al. [1999]
Kirk et al. [2006]
Code source du système
Spécifications opérationnelles
4
Correction
Outils: XRefactory, Refactoring Browser, JRefactory, Eclipse
Code source du système
Suggestions de restructurations
5
Validation
Système refactorisé

État de l'art



Problèmes

Analyse des descriptions[1]

Difficile de spécifier de nouvelles règles selon le contexte

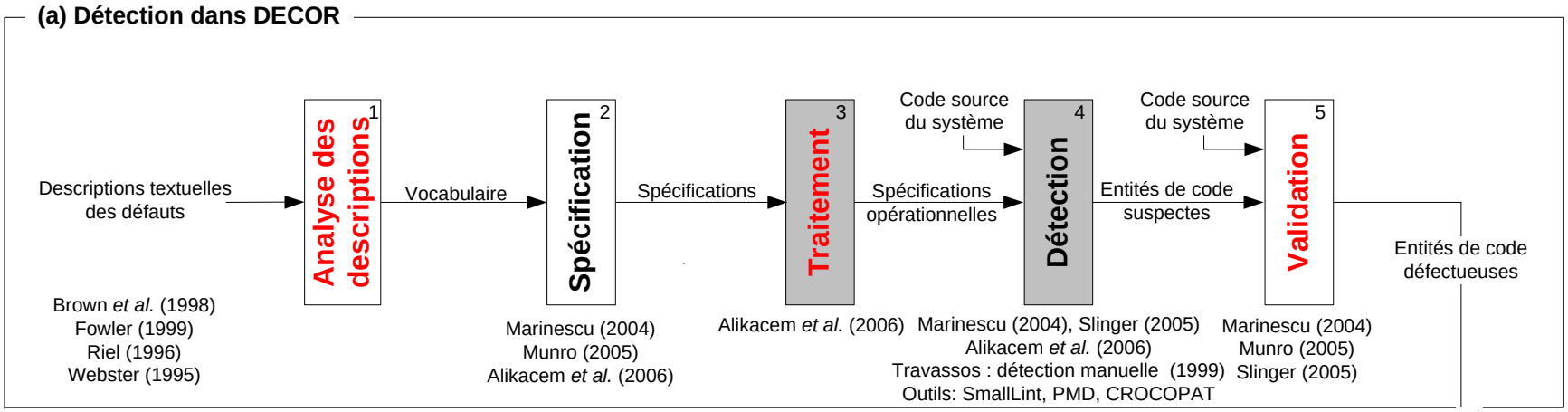
Traitement[3]

Passage des spécifications des défauts à la détection = boîte noire

Validation[5]

Validation partielle des techniques de détection

État de l'art et contributions



Contributions

- Analyse des descriptions — La technique de détection DETEX *(DETection EXpert)* avec son langage spécifique au domaine
- Traitement — Génération automatique des algorithmes de détection à partir des spécifications
- Validation — Validation de DETEX

État de l'art

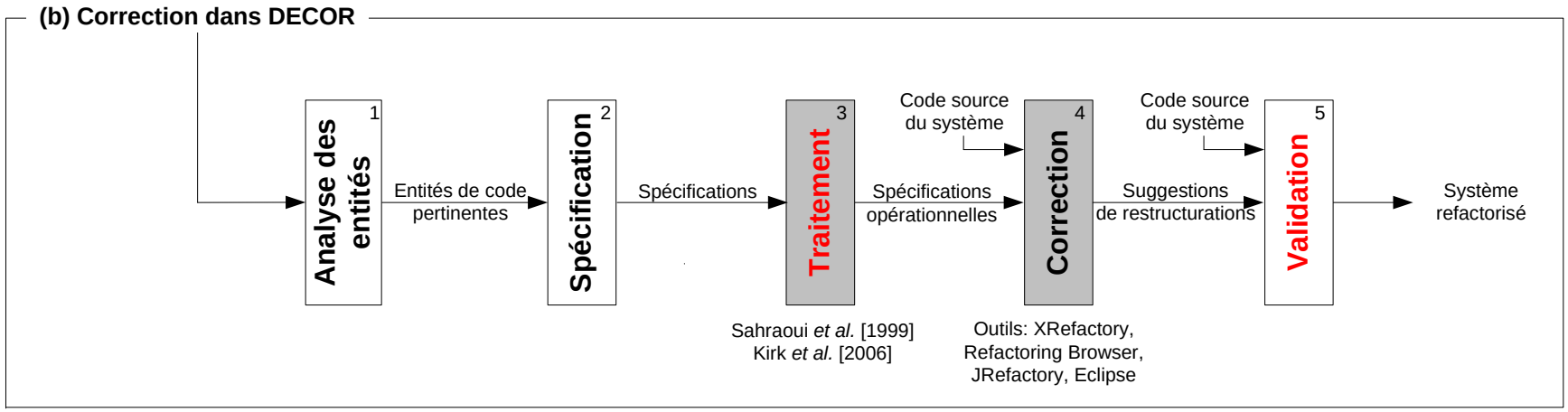
Problèmes

Traitement 3

Identification manuelle des restructurations

Validation 5

Pas de validation pour la correction



État de l'art et contributions

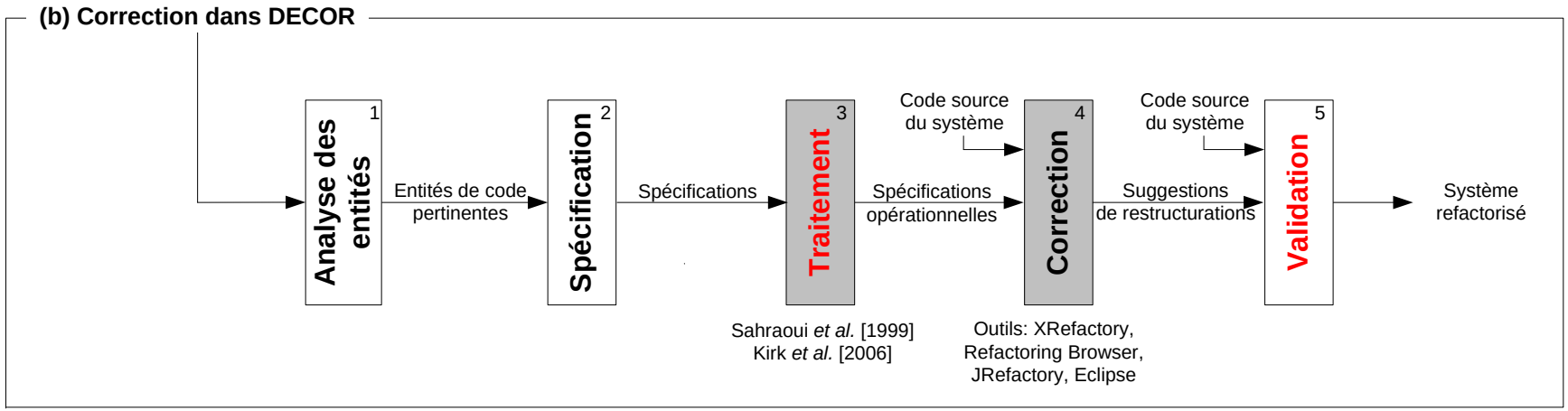
Contributions

Traitement 3

Technique de correction COREX *(CORrection EXpert)* : suggestion automatique de restructurations

Validation 5

Validation de COREX

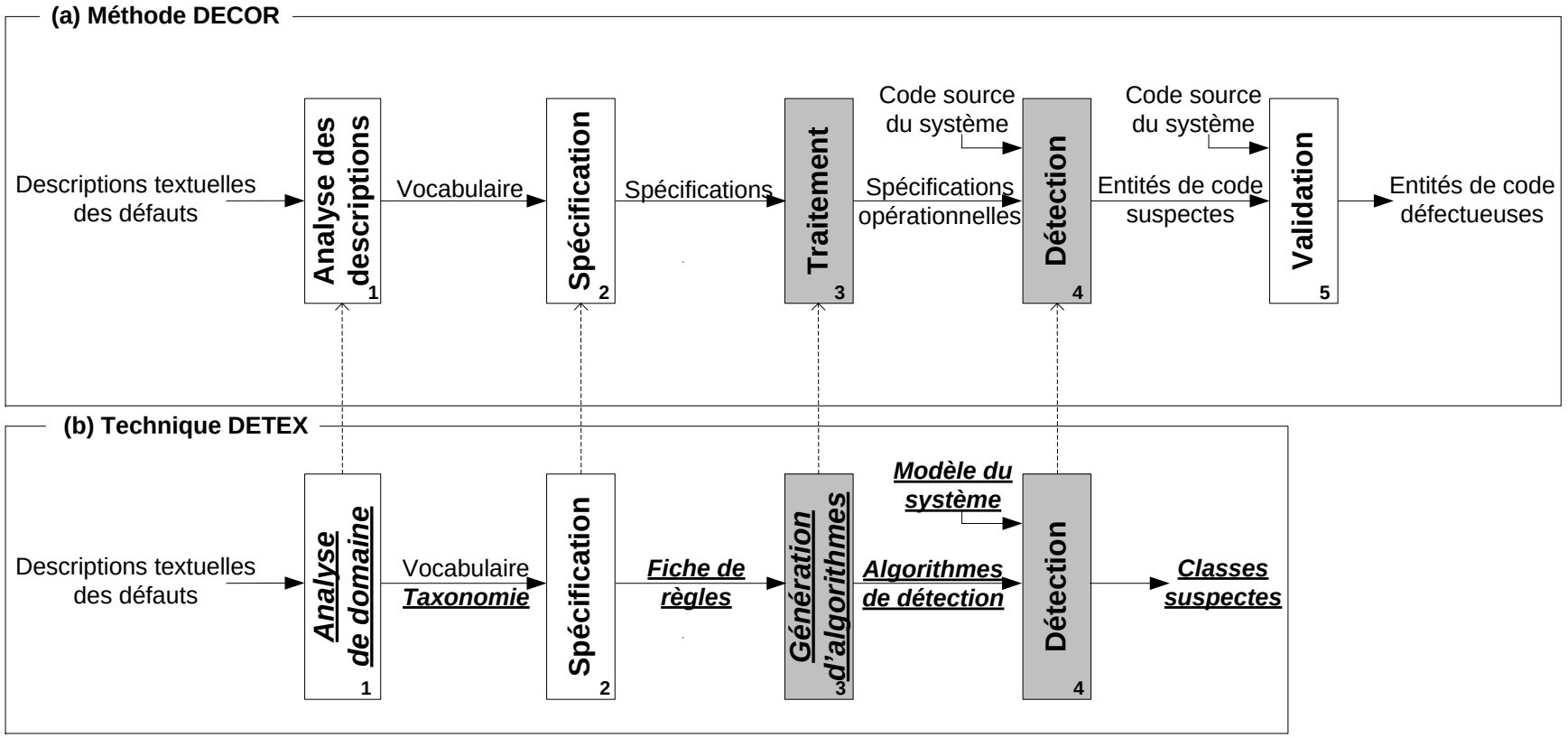


Plan

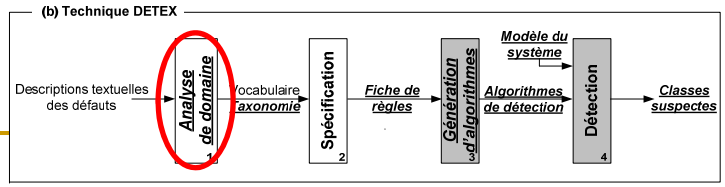
- Contexte
- Thèse
- État de l’art et contributions
- Détection
- Correction
- Conclusion et perspectives

Technique de détection DETEX

DETEX (DETection EXpert)



Analyse de domaine

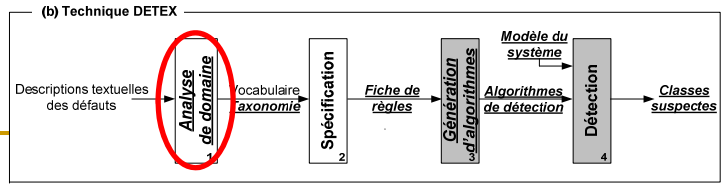


Chapter 1

Descriptions of antipatterns

The **Spaghetti Code** is an antipattern that is characteristic of procedural thinking in object-oriented programming. Spaghetti Code is revealed by classes with no structure, declaring long methods with no parameters, and utilising global variables for processing. Names of classes and methods may suggest procedural programming. Spaghetti Code does not exploit and prevents the use of object-orientation mechanisms, polymorphism and inheritance.

Analyse de domaine



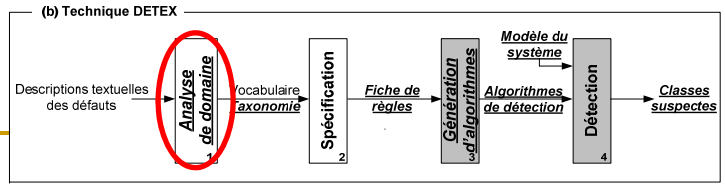
Chapter 1

Descriptions of antipatterns

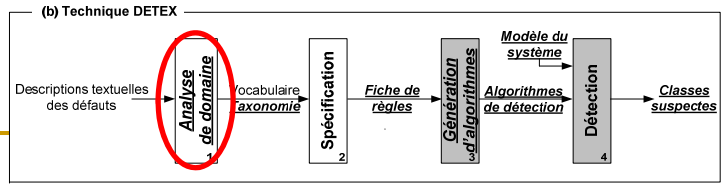
- Identification des concepts clefs

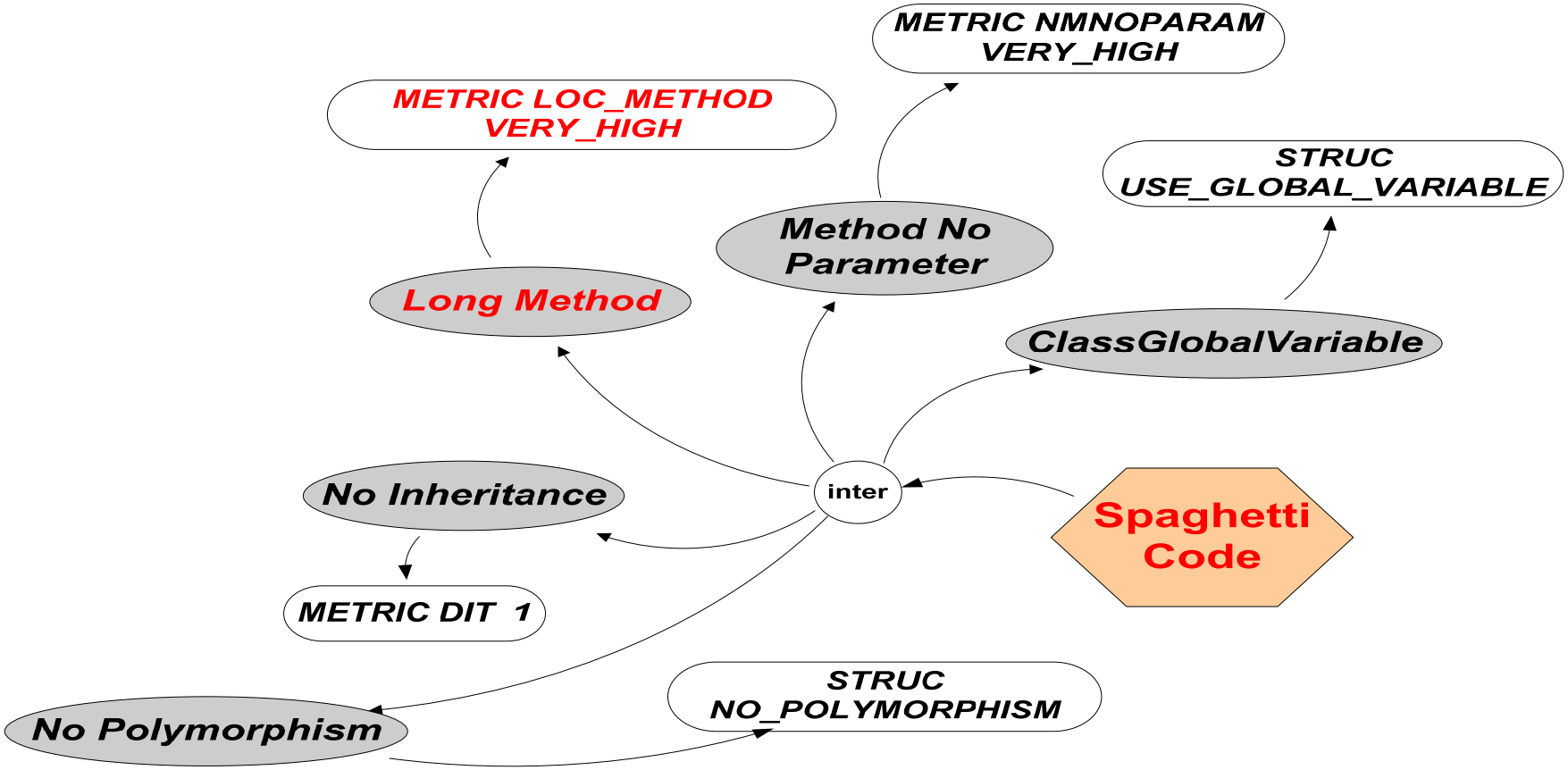
The **Spaghetti Code** is an antipattern that is characteristic of procedural thinking in object-oriented programming. Spaghetti Code is revealed by classes with no structure, declaring long methods with no parameters, and utilising global variables for processing. Names of classes and methods may suggest procedural programming. Spaghetti Code does not exploit and prevents the use of object-orientation mechanisms, polymorphism and inheritance.

Analyse de domaine

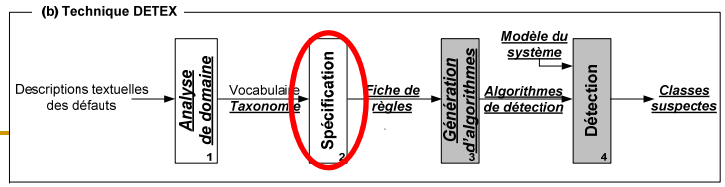


Analyse de domaine





Spécifications

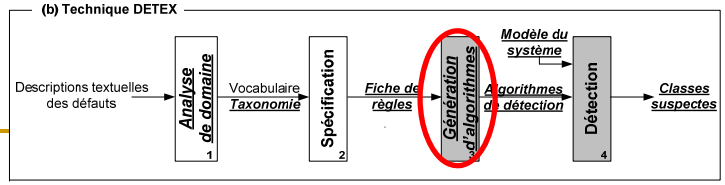


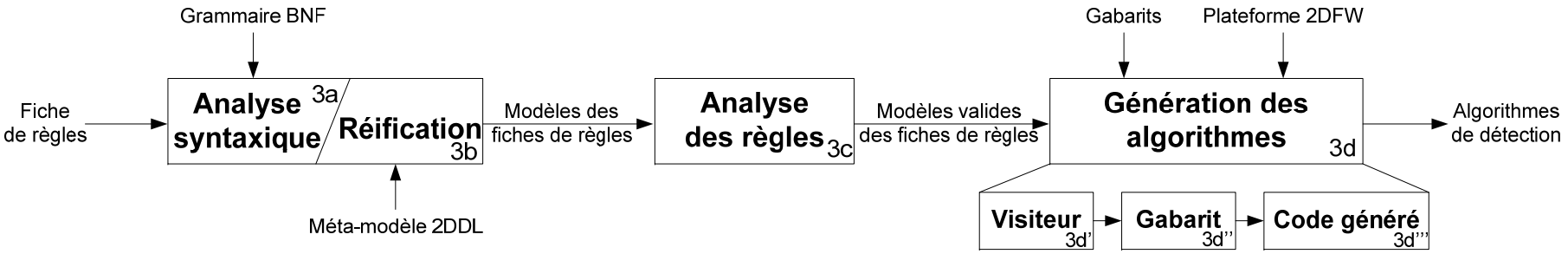
DSL *(Domain Specific Language)*

- Langage pour spécifier les défauts

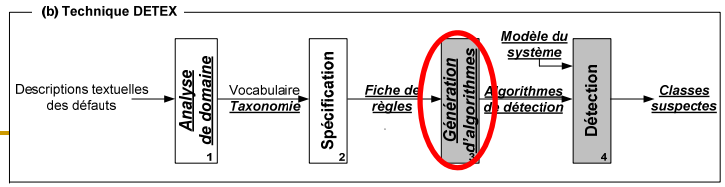
```
RULE_CARD : SpaghettiCode {
 ...
 RULE: LongMethodMethodNoParameter { INTER LongMethod MethodNoParameter };
 RULE: LongMethod          { METRIC  LOC_METHOD   VERY_HIGH  10.0 };
 RULE: MethodNoParameter { METRIC  NMNOPARAM  VERY_HIGH  10.0 };
 ...
 RULE: NoInheritance       { METRIC  DIT  1  0.0 } ;
 RULE: NoPolymorphism      { STRUCT  NO_POLYMORPHISM } ;

 RULE: ProceduralNameGlobalVariable { UNION ProceduralName GlobalVariable };
 RULE: ProceduralName      { LEXIC  CLASS_NAME  (Make, Create, System, Exec) };
 RULE: UseGlobalVariable { STRUCT  USE_GLOBAL_VARIABLE };
} ;
```

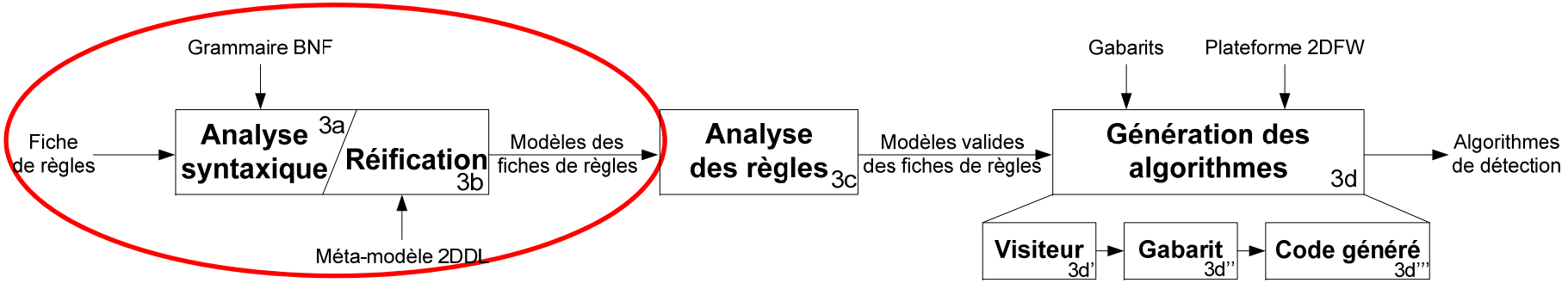
Génération des algos





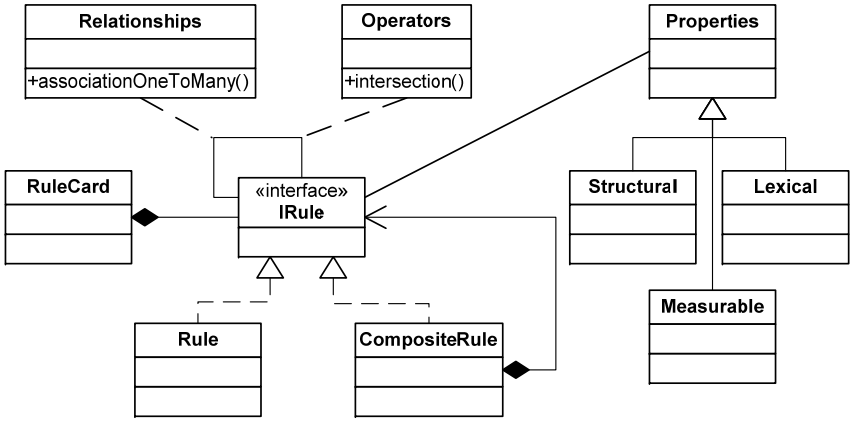
Génération des algos



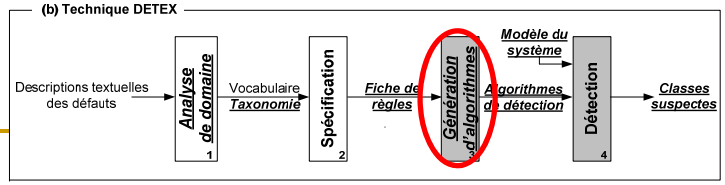


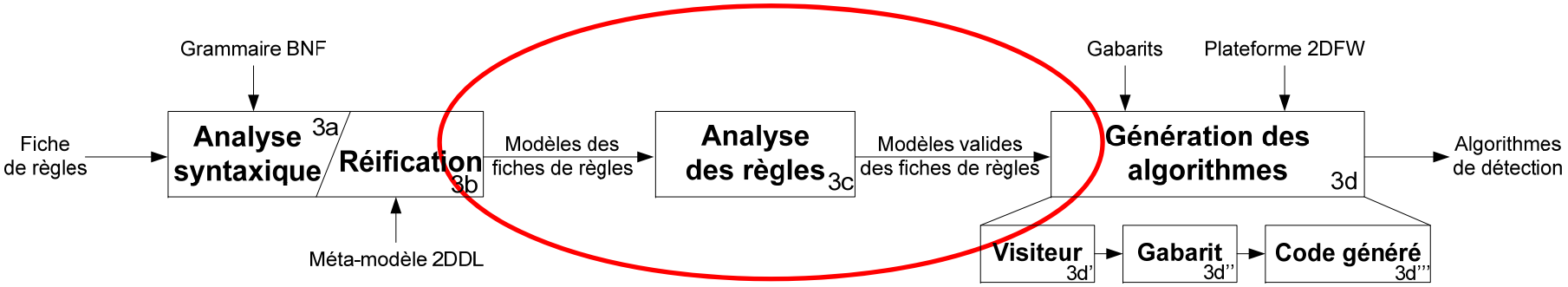

```
RULE_CARD : SpaghettiCode {
 ...
 RULE: LongMethodMethodNoParameter { INTER LongMethod MethodNoParameter };
 RULE: LongMethod         { METRIC  LOC_METHOD  VERY_HIGH  10.0 };
 RULE: MethodNoParameter { METRIC  NMNOPARAM  VERY_HIGH  10.0 };
 ...
 RULE: NoInheritance      { METRIC  DIT  1  0.0 } ;
 RULE: NoPolymorphism     { STRUCT  NO_POLYMORPHISM } ;

 RULE: ProceduralNameGlobalVariable { UNION ProceduralName GlobalVariable };
 RULE: ProceduralName     { LEXIC  CLASS_NAME  (Make, Create, System, Exec) };
 RULE: UseGlobalVariable { STRUCT  USE_GLOBAL_VARIABLE };
} ;
```

Méta-modèle 2DDL



Génération des algos





Analyses de cohérence et spécifiques au domaine

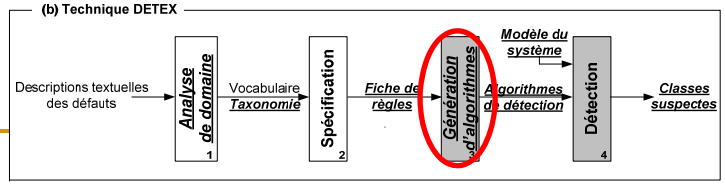
- Vérifier que les spécifications ne sont pas inconsistantes, redondantes ou incomplètes

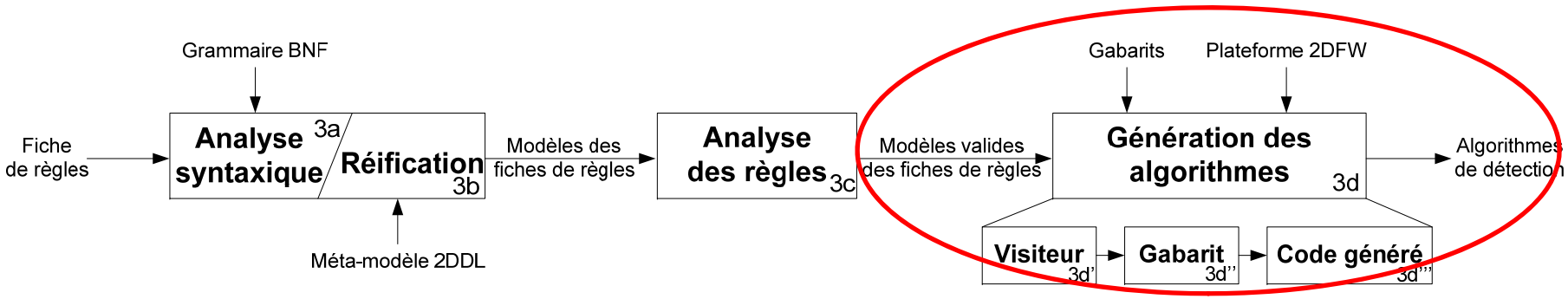
  Ex: 2 règles avec les mêmes noms mais différentes propriétés

- Vérifier que les règles sont conformes au domaine

  Ex: valeur associée à une métrique

Génération des algos





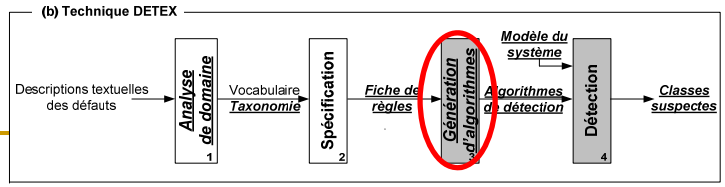
Complexité : $O((c+op) \times n)$

- c : le nombre de propriétés
- op : le nombre d'opérateurs
- n : le nombre de classes

Plateforme 2DFW *(Design Defects FrameWork)*

- Méta-modèle pour représenter les programmes OO
- Un répertoire de métriques
- Des services pour analyser des relations structurelles
- Des services pour appliquer des analyses structurelles et lexicales

Génération des algos




```
RULE: LongMethod  { METRIC  LOC_METHOD  VERY_HIGH  10.0 };
```

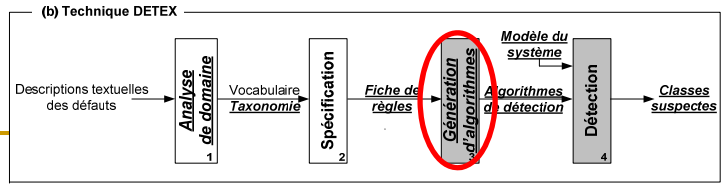
```
1  public void visit(IMetric aMetric) {
2    replaceTAG("<CODESMELL>", aRule.getName());
3    replaceTAG("<METRIC>", aMetric.getName());
4    replaceTAG(<FUZZINESS>,
5      aMetric.getFuzziness());
6    replaceTAG(<ORDINAL_VALUE>,
7      aMetric.getOrdinalValue());
8  }
9  private String getOrdinalValue(int value) {
10   switch (value) {
11     case VERY_HIGH :
12       "getHighOutliers";
13     case HIGH :
14       "getHighValues";
15     case MEDIUM :
16       "getNormalValues";
17     ...
18 }
```

Visiteur

```
1 public class <CODESMELL>Detection
2   extends CodeSmellDetection
3   implements ICodeSmellDetection {
4   public Set performDetection() {
5     IClass c = iteratorOnClasses.next();
6     LOCofSetOfClasses.add(
7     Metrics.compute(<METRIC>, c));
8     ...
9     BoxPlot boxPlot = new BoxPlot(
10      <METRIC>ofSetOfClasses, <FUZZINESS>);
11    Map setOfOutliers =
12      boxPlot.<ORDINAL_VALUE>();
13    ...
14    suspiciousCodeSmells.add( new CodeSmell(
15      <CODESMELL>, setOfOutliers));
16    ...
17    return suspiciousCodeSmells;
18 }
```

Gabarit

Génération des algos




```
RULE: LongMethod  { METRIC  LOC_METHOD  VERY_HIGH  10.0 };
```

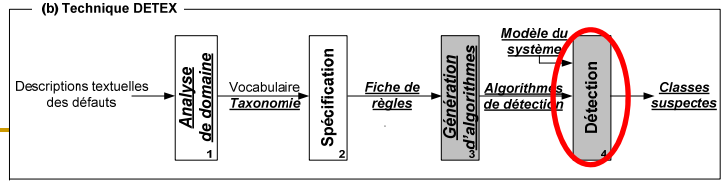
```
1  public void visit(IMetric aMetric) {
2    replaceTAG("<CODESMELL>", aRule.getName());
3    replaceTAG("<METRIC>", aMetric.getName());
4    replaceTAG(<FUZZINESS>,
5      aMetric.getFuzziness());
6    replaceTAG(<ORDINAL_VALUE>,
7      aMetric.getOrdinalValue());
8  }
9  private String getOrdinalValue(int value) {
10   switch (value) {
11     case VERY_HIGH :
12       "getHighOutliers";
13     case HIGH :
14       "getHighValues";
15     case MEDIUM :
16       "getNormalValues";
17     ...
18 }
```

Visiteur

```
1 public class LongMethodDetection
2   extends CodeSmellDetection
3   implements ICodeSmellDetection {
4   public Set performDetection() {
5     IClass c = iteratorOnClasses.next();
6     LOCofSetOfClasses.add(
7     Metrics.compute("LOC_METHOD", c));
8     ...
9     BoxPlot boxPlot = new BoxPlot(
10      LOC_METHODofSetOfClasses, 10.0);
11    Map setOfOutliers =
12      boxPlot.getHighOutliers();
13    ...
14    suspiciousCodeSmells.add( new CodeSmell(
15      LongMethod, setOfOutliers));
16    ...
17    return suspiciousCodeSmells;
18 }
```

Code généré

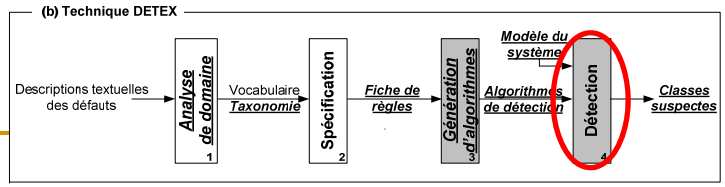
Détection

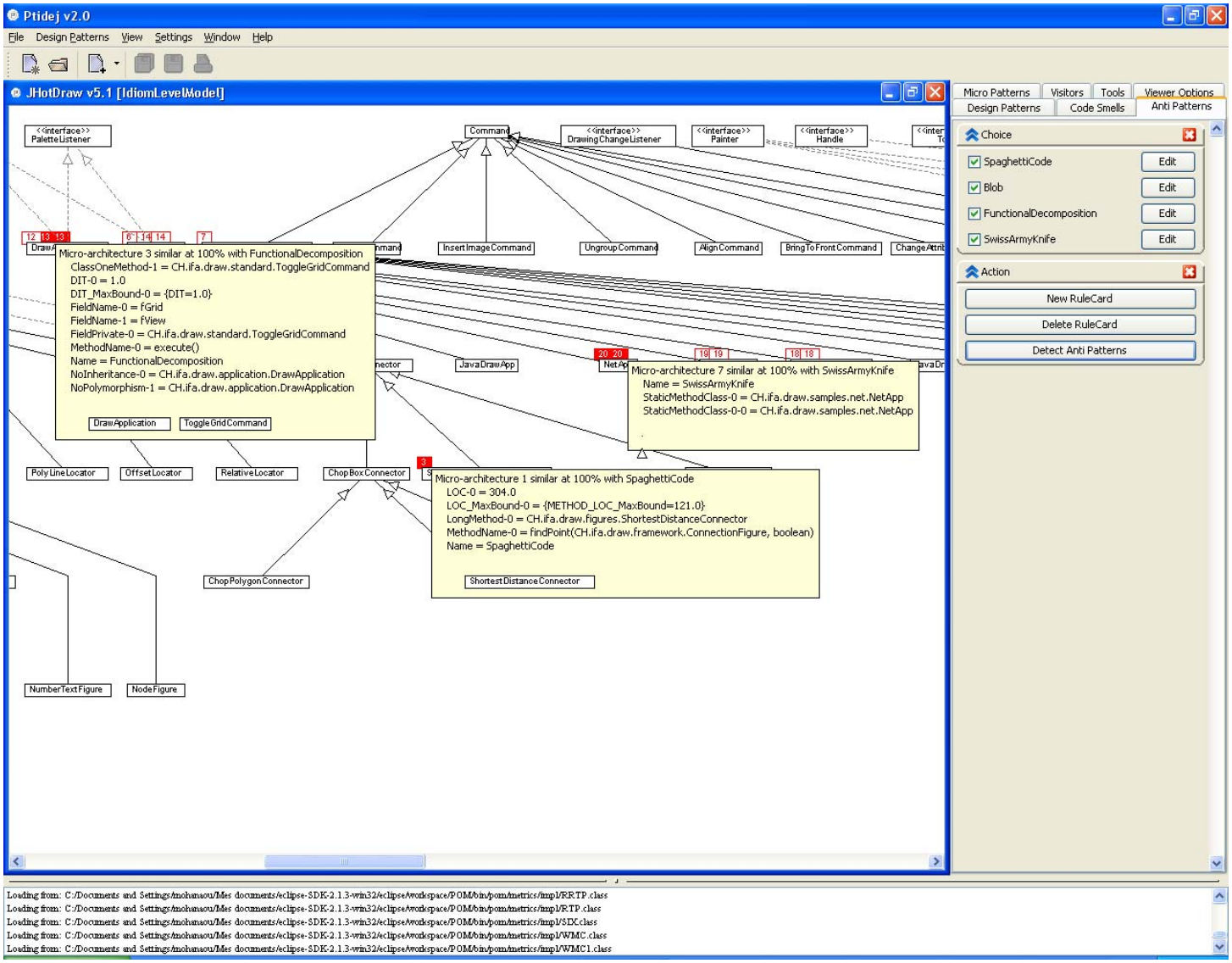


Fichier résultat de la détection

```
1.Name = SpaghettiCode
1.Class = org.apache.xerces.xinclude.XIncludeHandler
1.NoInheritance.DIT-0 = 1.0
1.LongMethod.MethodName = handleIncludeElement(XMLAttributes)
1.LongMethod.LOC_METHOD = 759.0
1.LongMethod.LOC_METHOD_Max = 254.5
1.GlobalVariable-0 = SYMBOL_TABLE
1.GlobalVariable-1 = ERROR_REPORTER
1.GlobalVariable-2 = ENTITY_RESOLVER
1.GlobalVariable-3 = BUFFER_SIZE
1.GlobalVariable-4 = PARSER_SETTINGS
...
2.Name = SpaghettiCode
2.Class = org.apache.xerces.impl.xpath.regex.RegularExpression
2.NoInheritance.DIT-0 = 1.0
2.LongMethod.MethodName = matchCharArray(Context,Op,int,int,int)
2.LongMethod.LOC_METHOD = 1246.0
2.LongMethod.LOC_METHOD_Max = 254.5
2.GlobalVariable-0 = WT_OTHER
2.GlobalVariable-1 = WT_IGNORE
2.GlobalVariable-2 = EXTENDED_COMMENT
2.GlobalVariable-3 = CARRIAGE_RETURN
2.GlobalVariable-4 = IGNORE_CASE
...
```

Détection


(b) Technique DETEX
Descriptions textuelles des défauts
Analyse de domaine
1
Vocabulaire
Taxonomie
Spécification
2
Fiche de règles
Génération d'algorithmes
Modèle du système
Algorithmes de détection
Détection
4
Classes suspectes


Ptidej v2.0
File Design Patterns View Settings Window Help
JHotDraw v5.1 [IdiomLevelModel]
Micro-architecture 3 similar at 100% with FunctionalDecomposition
ClassOneMethod-1 = CH.ifa.draw.standard.ToggleGridCommand
DIT-0 = 1.0
DIT_MaxBound-0 = {DIT=1.0}
FieldName-0 = fGrid
FieldName-1 = fView
FieldPrivate-0 = CH.ifa.draw.standard.ToggleGridCommand
MethodName-0 = execute()
Name = FunctionalDecomposition
NoInheritance-0 = CH.ifa.draw.application.DrawApplication
NoPolymorphism-1 = CH.ifa.draw.application.DrawApplication
Micro-architecture 7 similar at 100% with SwissArmyKnife
Name = SwissArmyKnife
StaticMethodClass-0 = CH.ifa.draw.samples.net.NetApp
StaticMethodClass-0-0 = CH.ifa.draw.samples.net.NetApp
Micro-architecture 1 similar at 100% with SpaghettiCode
LOC-0 = 304.0
LOC_MaxBound-0 = {METHOD_LOC_MaxBound=121.0}
LongMethod-0 = CH.ifa.draw.figures.ShortestDistanceConnector
MethodName-0 = findPoint(CH.ifa.draw.framework.ConnectionFigure, boolean)
Name = SpaghettiCode
Micro Patterns
Visitors
Tools
Viewer Options
Design Patterns
Code Smells
Anti Patterns
Choice
SpaghettiCode
Blob
FunctionalDecomposition
SwissArmyKnife
Edit
Action
New RuleCard
Delete RuleCard
Detect Anti Patterns

Expérimentations

- But: Valider l'efficacité de DETEX et indirectement l'utilité de DECOR
- Résultats attendus
  - Le DSL permet de spécifier plusieurs défauts
  - Les algorithmes de détection générés ont un rappel de 100% et une précision supérieure à 50%
  - Les temps de détection sont de l'ordre de la minute
- Sujets
  - 4 anti-patrons et 15 défauts de bas niveau associées
- Objets
  - 9 systèmes JAVA libres + XERCES + ECLIPSE
- Processus
  - 9 systèmes : précision, XERCES : précision et rappel, ECLIPSE : détection

Expérimentations

- 4 anti-patrons
  - Blob
  - Spaghetti Code
  - Functional Decomposition
    - Conception procédurale en OO
    - Routine principale qui appelle plusieurs sous-routines
    - Nom procédural
    - Pas d'héritage, pas de polymorphisme
    - Association avec plusieurs petites classes
  - Swiss Army Knife
    - Classe complexe qui offre un grand nombre de services
    - Classe utilitaire

Expérimentations

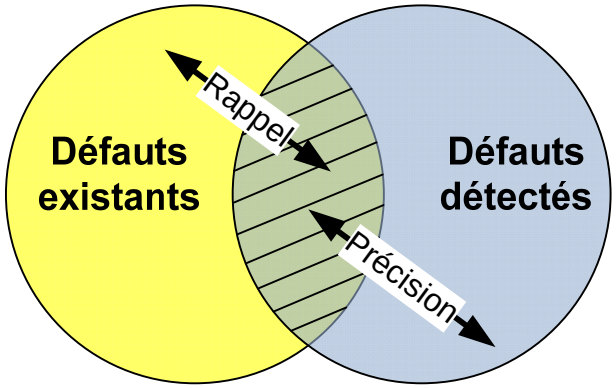
Résultats : XERCES pour les précisions et les rappels

XERCES: 71 217LOC, 513 classes et 162 interfaces

Défauts	Nb de classes	Défauts existants	Défauts détectés	Précision	Rappel	Temps de détection
Blob	513	39 (7,6%)	44 (8,6%)	88,6%	100%	2,45s
Functional Decomp.		15 (2,9%)	29 (5,6%)	51,7%	100%	0,16s
Spaghetti Code		46 (9,0%)	76 (14,8%)	60,5%	100%	0,22s
Swiss Army Knife		23 (4,5%)	56 (10,9%)	41,1%	100%	0,05s
			Précision moyenne	60,5%	100%	0,72s

$$\text{Rappel} = \frac{|\{\text{défauts existants}\} \cap \{\text{défauts détectés}\}|}{|\{\text{défauts existants}\}|}$$

$$\text{Précision} = \frac{|\{\text{défauts existants}\} \cap \{\text{défauts détectés}\}|}{|\{\text{défauts détectés}\}|}$$



Expérimentations

Résultats : Précision sur 10 systèmes

	ArgoUML	Azureus	GanttProject	Log4J	Lucene	Nutch	PMD	QuickUML	Xerces v1.0.1	Xerces v2.7.0
Blob	25/29 (86%) 3.0s	38/41 (93%) 6.4s	9/10 (90%) 2.4s	3/3 (100%) 1.3s	2/3 (67%) 1.8s	4/6 (67%) 3.6s	4/4 (100%) 3.9s	0/0 (100%) 0.4s	10/10 (100%) 2.7s	39/44 (89%) 2.4s
Funct. Dec.	22/37 (59%) 0.4s	17/44 (39%) 0.5s	4/15 (27%) 0.8s	6/11 (54%) 0.05s	0/1 (0%) 0.03s	3/15 (20%) 0.05s	4/13 (31%) 0.06s	3/10 (30%) 0.02s	4/4 (100%) 0.03s	15/29 (52%) 0.16s
Spag. Code	38/44 (86%) 0.3s	125/153 (82%) 2.9s	10/14 (71%) 0.2s	2/3 (67%) 0.08s	6/8 (75%) 0.09s	22/26 (85%) 0.1s	5/9 (57%) 0.06s	0/5 (0%) 0.03s	23/25 (92%) 0.11s	46/76 (60%) 0.2s
SwissArmyK	18/108 (16%) 0.3s	33/145 (23%) 0.13s	3/8 (37%) 0.05s	33/51 (65%) 0.02s	1/9 (11%) 0.02s	13/33 (40%) 0.02s	6/13 (46%) 0.02s	1/6 (17%) 0.02s	5/12 (42%) 0.03s	23/56 (41%) 0.05s
Préci-sion	62%	59%	56%	71%	38%	53%	58%	37%	83%	60%

Expérimentations

Résultats : ECLIPSE pour illustrer le passage à l'échelle

ECLIPSE : 2 538 774LOC, 9 099 classes et 1 850 interfaces

Défauts	Défauts détectés	Précision	Rappel	Temps de détection
Blob	848 (9,3%)	/	/	1h20m
Functional Decomp.	608 (6,7%)	/	/	
Spaghetti Code	436 (4,8%)	/	/	
Swiss Army Knife	520 (5,7%)	/	/	

→ Importance de spécifier les défauts dans le contexte du système

Conclusion sur la détection

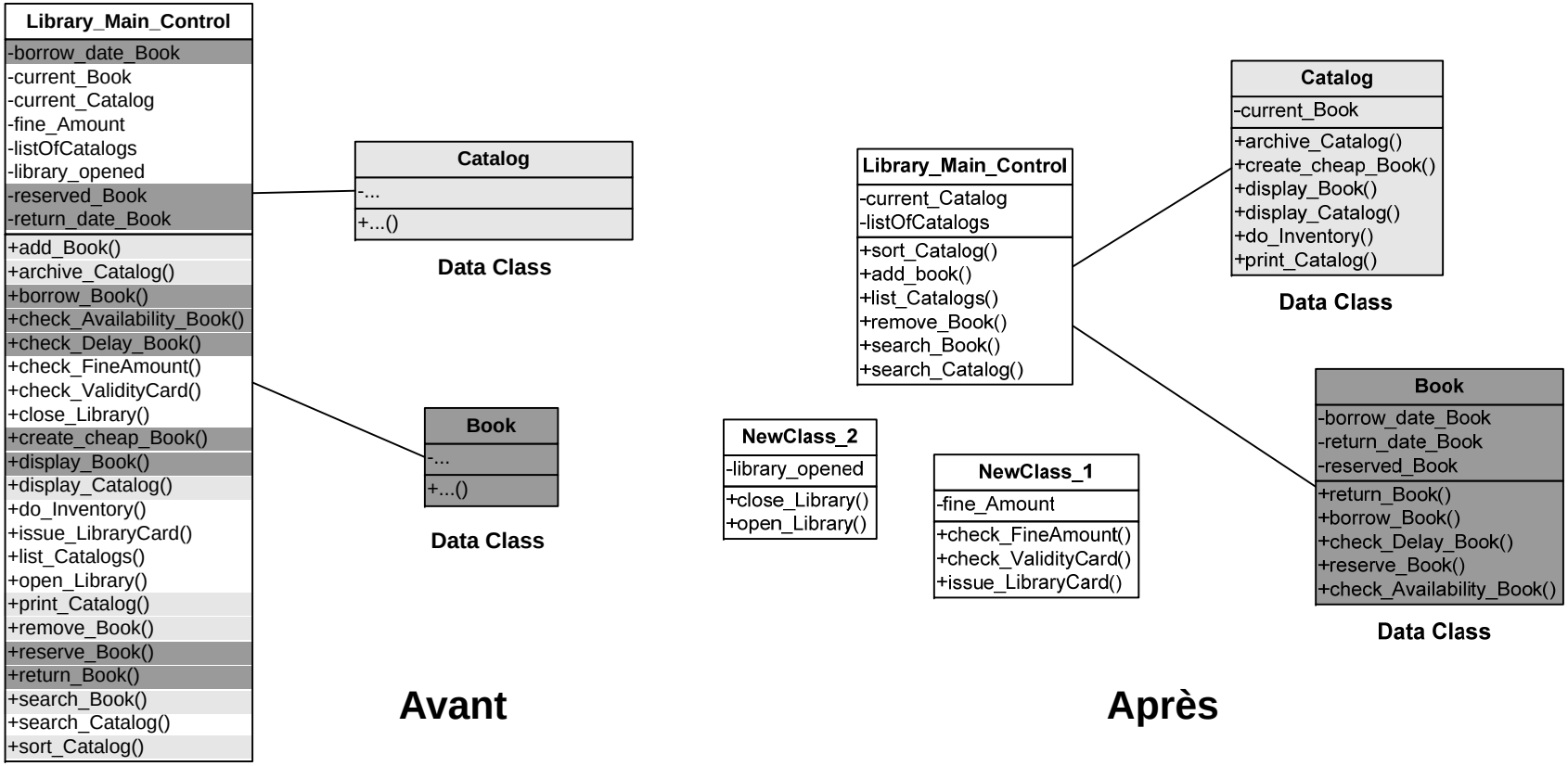
- Technique de détection DETEX en suivant DECOR
  - DSL issu d'une analyse de domaine
  - Génération automatique des algos de détection
- Validation en terme de précision et rappel, passage à l'échelle
- Difficulté de se comparer aux travaux précédents
  - Expérimentations = point de repère pour de futures comparaisons
- http://www.naouelmoha.net/DECOR/DECOR.htm

Plan

- Contexte
- Thèse
- État de l'art et contributions
- Détection
- Correction
- Conclusion et perspectives

L'exemple du Blob

Comment corriger ce défaut ?

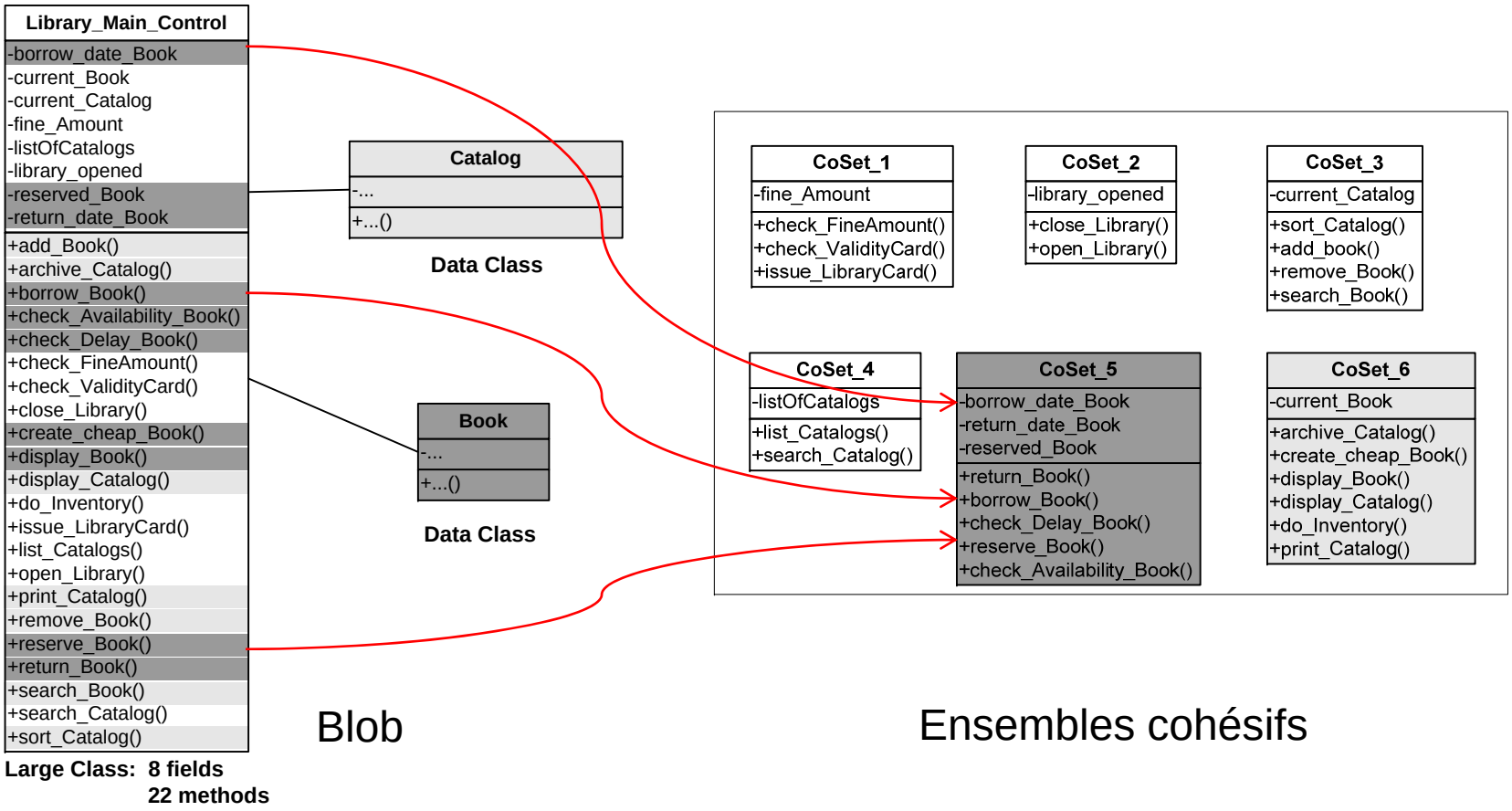


- Large classe complexe
- Petites classes de données
- Non orienté objet

- Large classe devient moins complexe
- Classes de données avec plus de comportement
- Plus orienté objet

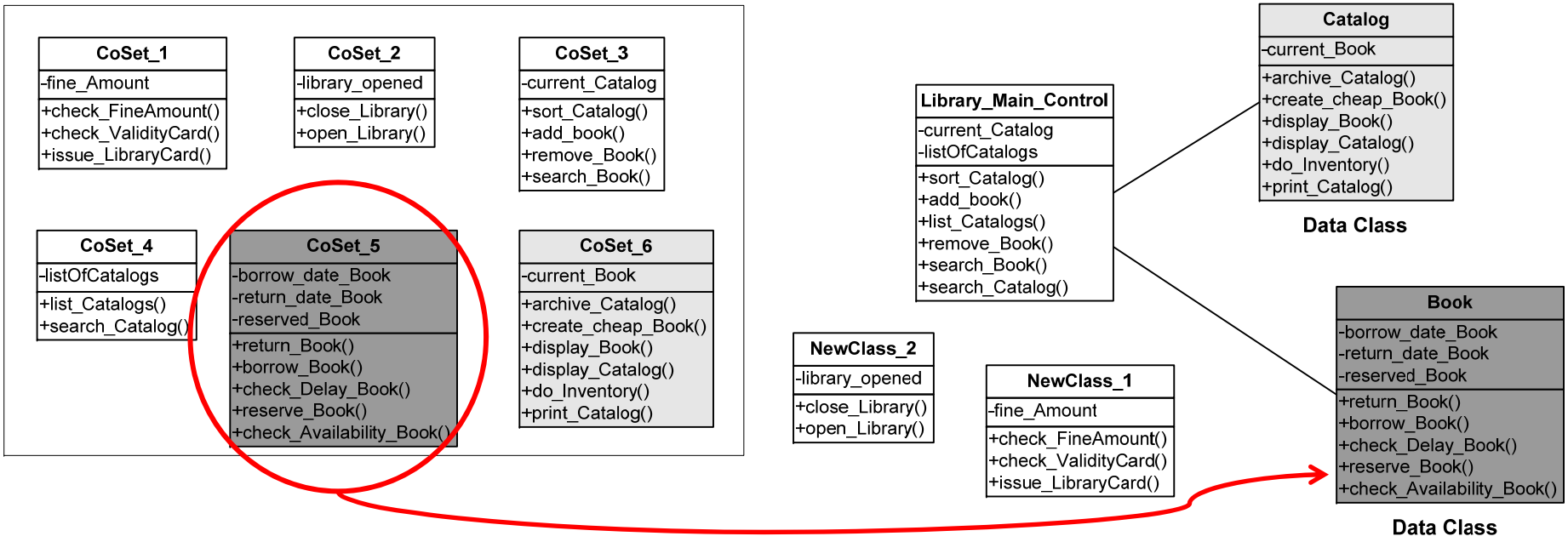
L'exemple du Blob

Comment corriger ce défaut ?



L'exemple du Blob

Comment corriger ce défaut ?



Ensembles cohésifs

Après refactorisation*

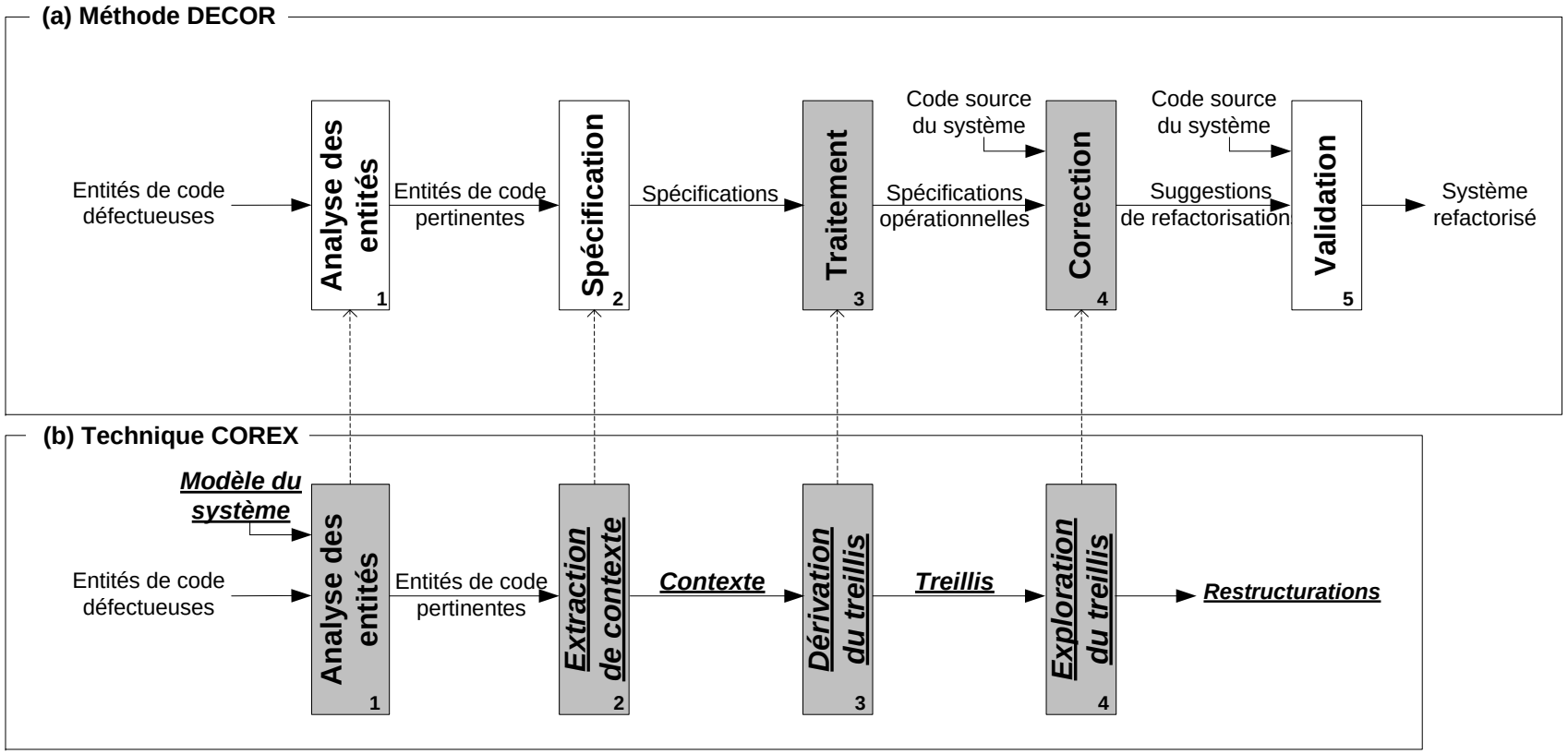
* Déplacer Méthode, Déplacer Attribut, Créer Classe

Correction

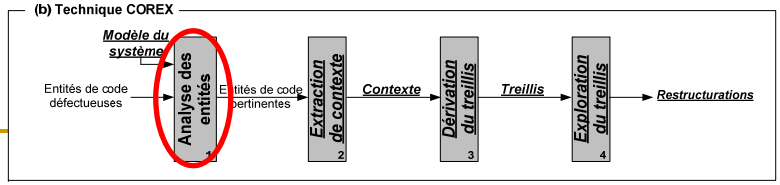
- Analyse Formelle de Concepts (AFC)
  - AFC offre un cadre formel pour regrouper des individus ayant des propriétés communes
- Analyse Relationnelle de Concepts (ARC)
  - Extension de l' AFC à des données relationnelles
  - Liens entre individus
  - Identifier des méthodes qui partagent des attributs et qui appellent des méthodes communes

    → ensembles fortement cohésifs faiblement couplés
- COREX *(CORrection EXpert)*
  - Suggestion de restructurations pour corriger les défauts avec ARC
  - Blob + défauts faible cohésion et fort couplage (Divergent Change, Shotgun Surgery, Feature Envy)

Technique de correction COREX

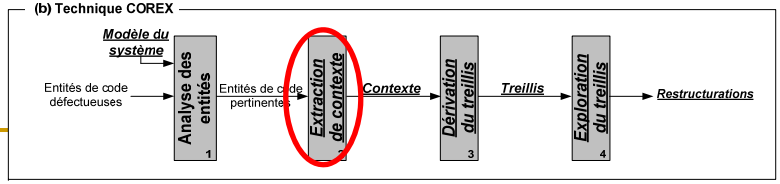


Analyse des entités




```
1.Name = SpaghettiCode
1.Class = org.apache.xerces.xinclude.XIncludeHandler
1.NoInheritance.DIT-0 = 1.0
1.LongMethod.MethodName = handleIncludeElement(XMLAttributes)
1.LongMethod.LOC_METHOD = 759.0
1.LongMethod.LOC_METHOD_Max = 254.5
1.GlobalVariable-0 = SYMBOL_TABLE
1.GlobalVariable-1 = ERROR_REPORTER
1.GlobalVariable-2 = ENTITY_RESOLVER
1.GlobalVariable-3 = BUFFER_SIZE
1.GlobalVariable-4 = PARSER_SETTINGS
...
2.Name = SpaghettiCode
2.Class = org.apache.xerces.impl.xpath.regex.RegularExpression
2.NoInheritance.DIT-0 = 1.0
2.LongMethod.MethodName = matchCharArray(Context,Op,int,int,int)
2.LongMethod.LOC_METHOD = 1246.0
2.LongMethod.LOC_METHOD_Max = 254.5
2.GlobalVariable-0 = WT_OTHER
2.GlobalVariable-1 = WT_IGNORE
2.GlobalVariable-2 = EXTENDED_COMMENT
2.GlobalVariable-3 = CARRIAGE_RETURN
2.GlobalVariable-4 = IGNORE_CASE
...
```

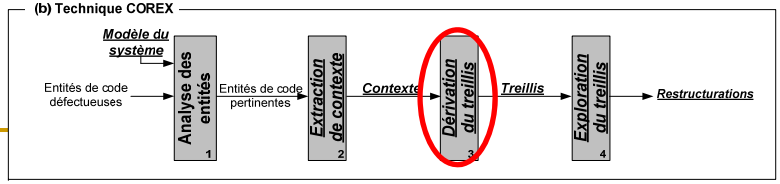
Extraction de la FCR

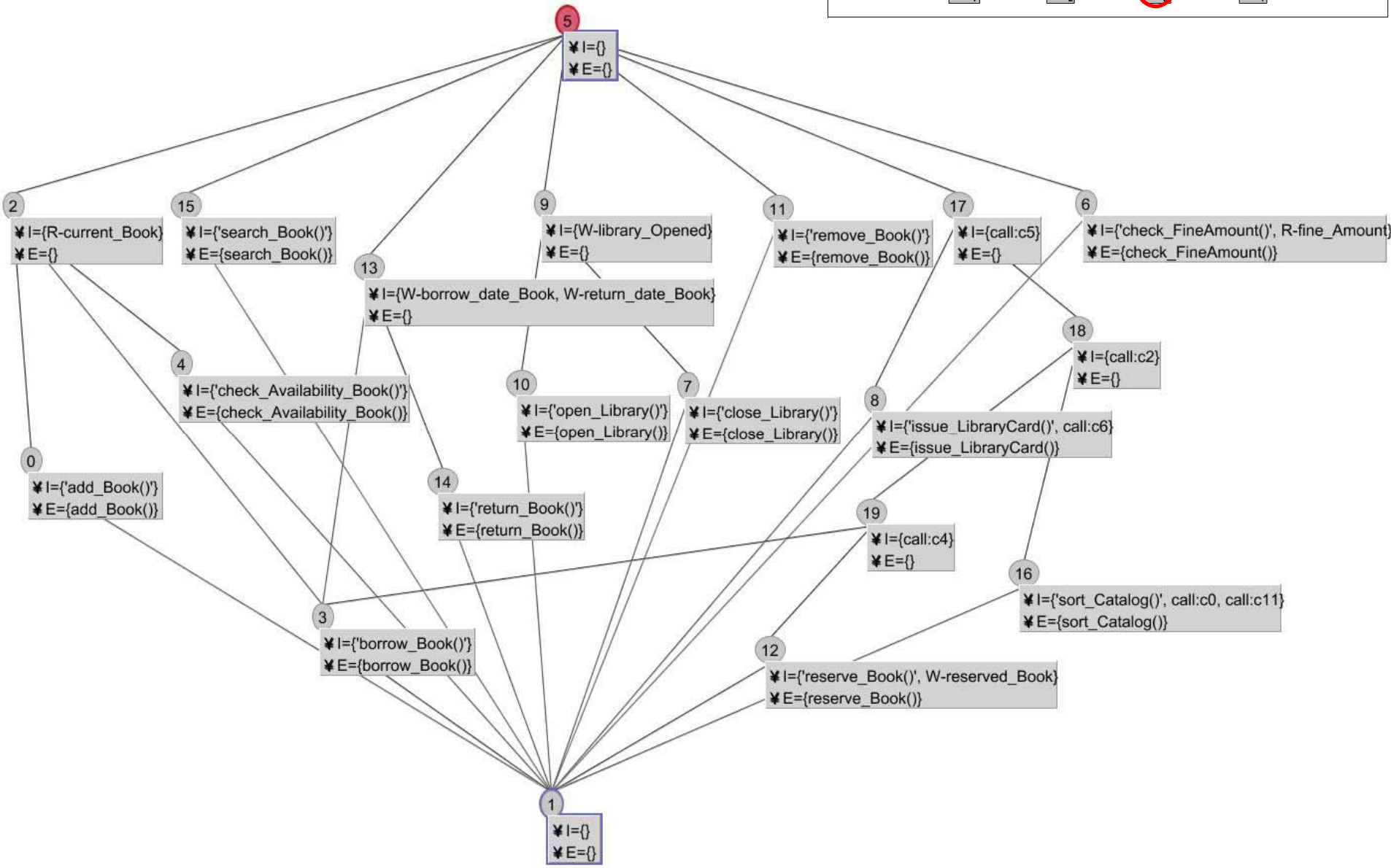


	R-current_book	R-fine_amount	W-borrow_date_book	W-library_opened	W-reserved_book	W-return_date_book
add_Book()	X					
borrow_Book()	X		X			X
check_Availability_Book()	X					
check_FineAmount()		X				
close_Library()				X		
issue_LibraryCard()						
open_Library()				X		
remove_Book()						
reserve_Book()					X	
return_Book()			X			X
search_Book()						
sort_Catalog()						

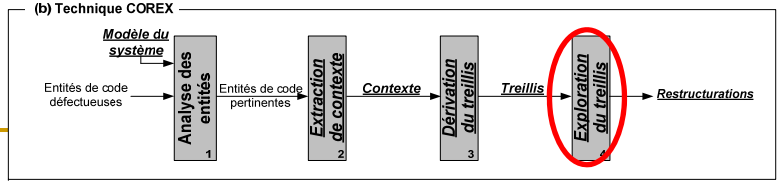
	add_Book()	check_Availability_Book()	check_FineAmount()	remove_Book()
add_Book()				
borrow_Book()		X		
check_Availability_Book()				
check_FineAmount()				
close_Library()				
issue_LibraryCard()			X	
open_Library()				
remove_Book()				
reserve_Book()		X		
return_Book()				
search_Book()				
sort_Catalog()	X			X

Dérivation du treillis





Exploration du treillis



5
I={}
E={}

2
I={R-current_Book}
E={}

15
I={'search_Book()'}
E={search_Book()}

9
I = {W-library_opened}
E = { }

11
emove_Book()'}
emove_Book()}

17
I={call:c5}
E={}

6
I={'check_FineAmount()', R-fine_Amount}
E={check_FineAmount()}

13
I={W-borrow_date_Book, W-return_date_Book}
E={}

18
I={call:c2}
E={}

4
I={'check_Availability_Book()'}
E={check_Availability_Book()}

10
I = {'open_Library()'}
E = {open_Library()}

7
I = {'close_Library()'}
E = {close_Library()}

ibraryCard()', call:c6}
LibraryCard()}

0
I={'add_Book()'}
E={add_Book()}

14
I={'return_Book()'}
E={return_Book()}

19
I={call:c4}
E={}

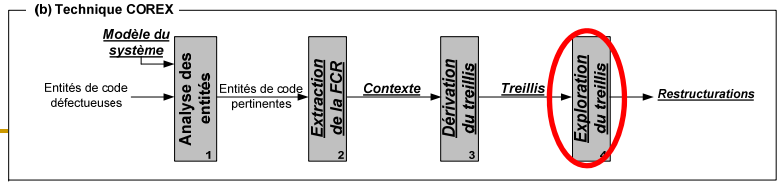
16
I={'sort_Catalog()', call:c0, call:c11}
E={sort_Catalog()}

3
I={'borrow_Book()'}
E={borrow_Book()}

12
I={'reserve_Book()', W-reserved_Book}
E={reserve_Book()}

1
I={}
E={}

Exploration du treillis



5 I={} E={}

2 I={R-current_Book} E={}

15 I={'search_Book()'} E={search_Book()}

9 I={W-library_Opened} E={}

11 I={'remove_Book()'} E={remove_Book()}

17 I={call:c5} E={}

6 I={'check_FineAmount()', R-fine_Amount} E={check_FineAmount()}

13 I={W-borrow_date_Book, W-return_date_Book} E={}

4 I = {'check_Availability_Book()'} E = {check_Availability_Book()}

18 I={call:c2} E={}

7 I={'close_Library()'} E={close_Library()}

8 I={'issue_LibraryCard()', call:c6} E={issue_LibraryCard()}

0 I={'add_Book()'} E={add_Book()}

14 I={'return_Book()'} E={return_Book()}

19 I = {call:c4} E = { }

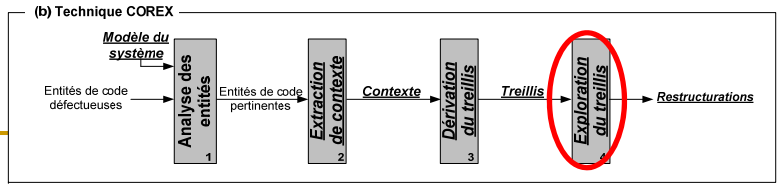
I={'sort_Catalog()', call:c0, call:c11} E={sort_Catalog()}

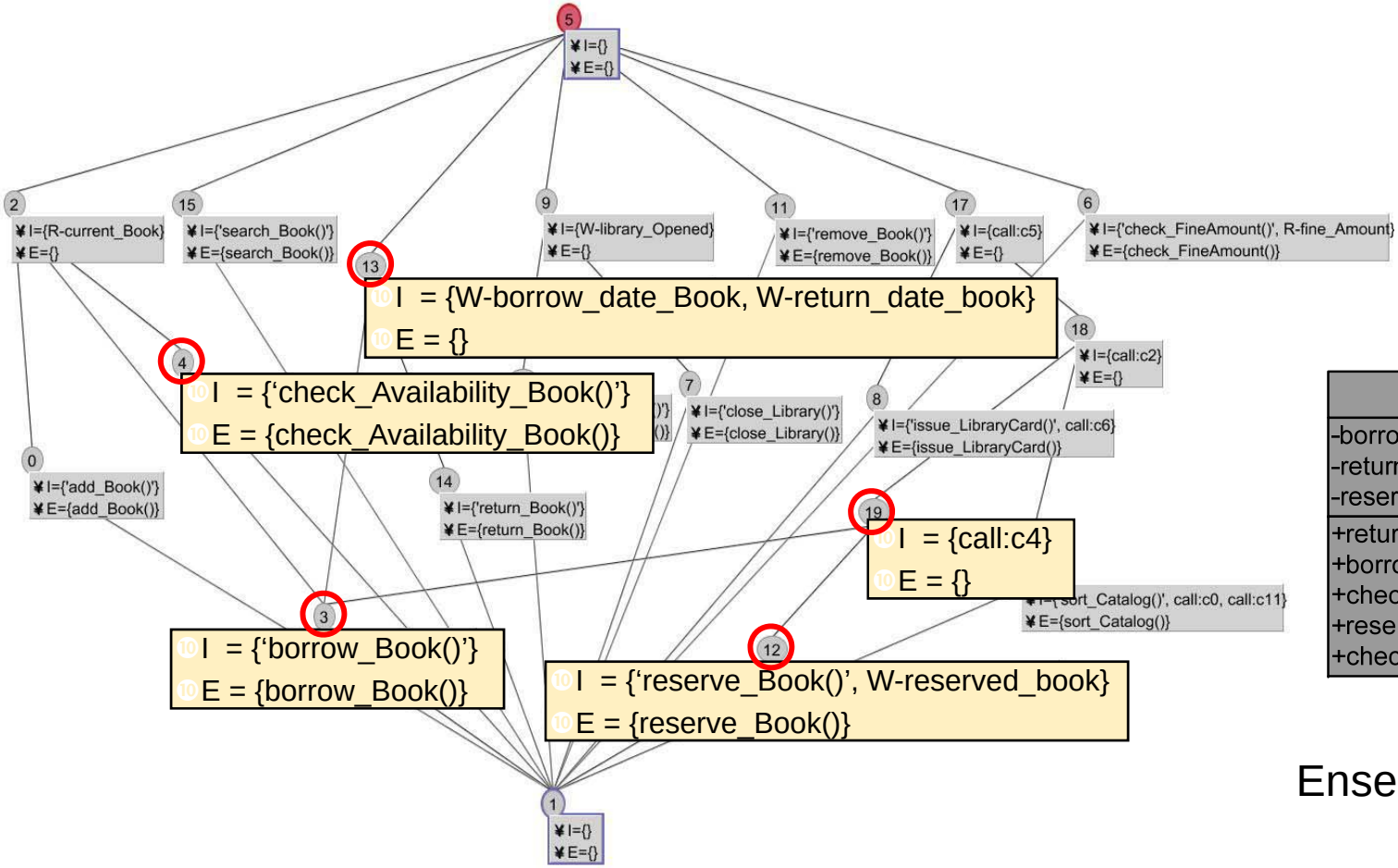
3 I = {'borrow_Book()'} E = {borrow_Book() }

12 I = {'reserve_Book()',W-reserved_book} E = {reserve_Book()}

1 I={} E={}

Exploration du treillis



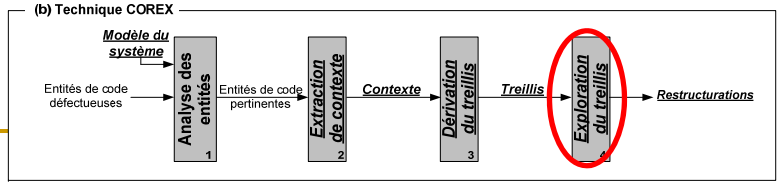


Treillis

CoSet_5
-borrow_date_Book -return_date_Book -reserved_Book
+return_Book() +borrow_Book() +check_Delay_Book() +reserve_Book() +check_Availability_Book()

Ensemble cohésif

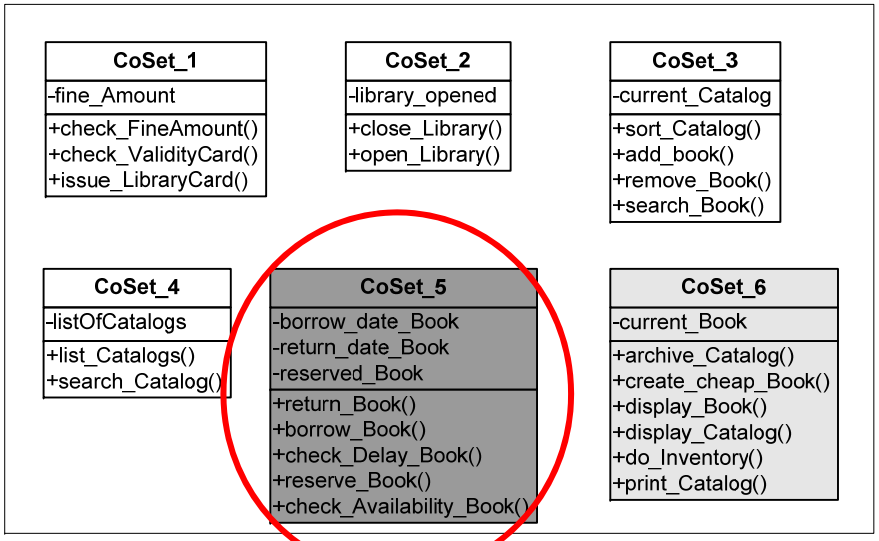
Exploration du treillis

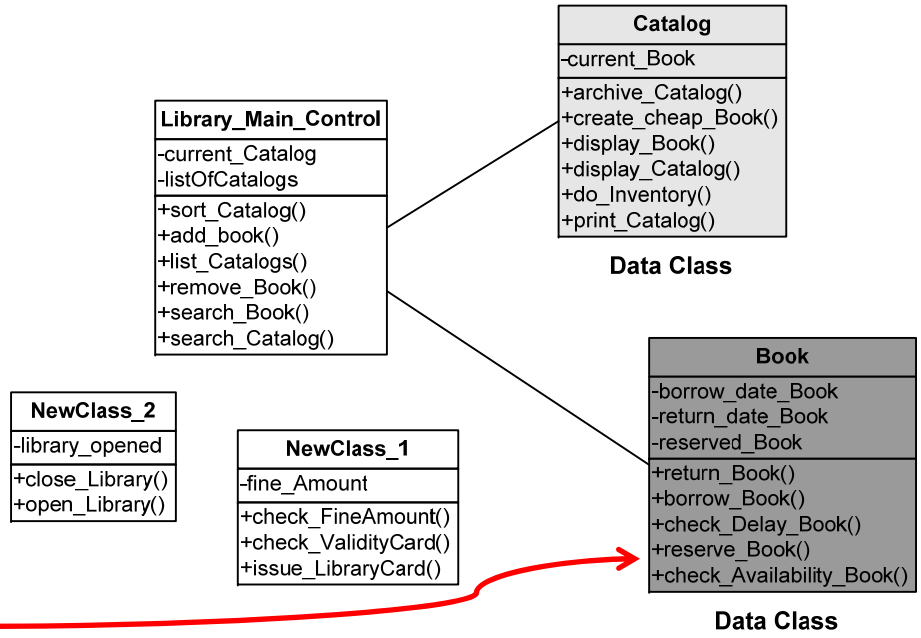


- Règles d'exploration pour construire les ensembles cohésifs
  - Règle 1. Les méthodes accédant en mode écriture au même ensemble d'attributs.
  - Règle 2. Les méthodes accédant en mode lecture au même ensemble d'attributs si le nombre d'attributs communs auxquels elles accèdent est plus grand que le nombre d'attributs auxquels elles accèdent séparément.
  - Règle 3. Les méthodes qui appellent le même ensemble de méthodes si le nombre de méthodes appelées conjointement est supérieur au nombre de méthodes appelées séparément.

L'exemple du Blob

Comment corriger ce défaut ?





Ensembles cohésifs

Après refactorisation*

* Déplacer Méthode, Déplacer Attribut, Créer Classe

Expérimentations

- But: Valider l'efficacité de COREX et indirectement l'utilité de DECOR
- Résultat attendu
  - La précision des restructurations suggérées est supérieure à 50%
- Sujets
  - Blob et applicable sur d'autres défauts (faible cohésion et fort couplage)
- Objets
  - 4 systèmes JAVA libres
- Processus
  - GALICIA
  - Analyse manuelle de la pertinence des restructurations suggérées

Expérimentations

Système	Classe Blob	LOC	Taille (attributs + méthodes)	Nb d'attributs / méthodes déplacés	Nb d'ens. cohésifs	Nb de réels ens. cohésifs	Précision	Précision moyenne
Azureus V2.3.0.6	DHTTransportUDPImpl	2,049	(42+66) 108	(27+32) 59	10	7	70%	53%
	DHTControlImpl	1,868	(47+80) 127	(35+62) 97	19	11	58%	
	TRTrackerBTAnnouncer	1,393	(36+47) 83	(24+33) 57	16	5	31%	
Log4j V1.2.1	LogBrokerMonitor	1,142	(29+105)134	(23+85) 108	31	17	55%	52.5%
	Category	387	(9+53) 62	(8+44) 52	18	9	50%	
Lucene V1.4	IndexReader	236	(7+52) 59	(5+30) 35	4	2	50%	63.5%
	QueryParser	829	(36+48) 84	(24+37) 61	13	10	77%	
Nutch V0.7.1	FSNamesystem	710	(24+35) 59	(17+25) 42	18	9	50%	61.5%
	JobTracker	555	(22+31) 53	(17+18) 35	11	8	73%	

Conclusion sur la correction

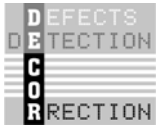
- Technique de correction COREX en suivant DECOR
  - ARC pour suggérer des restructurations
  - Blob + défauts avec faible cohésion et fort couplage
- Validation en terme de précision
- Aspect novateur et précurseur de cette approche

Plan

- Contexte
- Thèse
- État de l’art et contributions
- Détection
- Correction
- Conclusion et perspectives

Conclusion

- Contributions : méthode DECOR



  - Technique de détection DETEX
    - Langage spécifique au domaine issu d'une analyse de domaine
    - Génération automatique des algos de détection à partir de spécifications
    - Validation avec précision > 50% et rappel de 100%
  - Technique de correction COREX
    - Problème de correction des défauts abordé
    - Suggestion automatique des restructurations
    - Validation avec précision > 50%

- Points forts
  - Détection et DSL bien maîtrisés
    → benchmark
  - Automatisation de la phase de suggestion des restructurations
    → Correction reste un problème difficile à maîtriser

Perspectives à court terme

- DECOR
  - Classement des pires défauts détectés et des meilleures suggestions de restructurations
  - Plus de défauts et de systèmes

- DETEX
  - Description plus précise des propriétés dans les spécifications: base de données lexicale WORDNET, grammaire de graphe
  - Ontologie des défauts
  - Flexibilité dans la génération des algos de détection

- COREX
  - Règles additionnelles : impliquant plusieurs classes
  - Autres critères de qualité que le couplage et la cohésion tels que la complexité

Perspectives : extensions

- Évolution des défauts
  - Lien avec les bogues
  - Impact des défauts sur les coûts de maintenance
- Impact des défauts sur la qualité logicielle
  - Impact des patrons de conception sur la qualité [Khomh *et al.* 2008]
  - Modèle de qualité basé sur la présence de défauts
- Cadre de comparaison
  - Critères pour classifier les approches de détection
  - Étude sur les outils de détection REVJAVA, FINDBUGS, PMD, etc

Perspectives : pistes exploratoires

- Étude de nouvelles techniques
  - Analyse dynamique : défauts comportementaux
- Intégration à de nouveaux environnements
  - Appliquer concrètement les restructurations: transformation de programmes KERMETA, SPOON, ECLIPSE

Liste des publications

Conférences internationales

N. Moha, Y.-G. Guéhéneuc, A.-F. Le Meur, L. Duchien. *A Domain Analysis to Specify Design Defects and Generate Detection Algorithms*. Proceedings of the 11th International Conference on Fundamental Approaches to Software Engineering, pp. 276-291, March-April 2008, LNCS, Springer-Verlag.

N. Moha, A. M. Rouane Hacene, P. Valtchev, Y.-G. Guéhéneuc. *Refactorings of Design Defects using Relational Concept Analysis*. Proceedings of the 4th International Conference on Formal Concept Analysis, February 2008. Springer-Verlag.

N. Moha, J. Rezgui, Y.-G. Guéhéneuc, P. Valtchev, G. El Boussaidi. *Using FCA to Suggest Refactorings to Correct Design Defects.* Proceedings of the 4th International Conference On Concept Lattices and Their Applications (CLA 2006), pp. 269-275, October 30-November 1st, 2006, Tunisia. LNAI, Springer-Verlag.

N. Moha, Y.-G. Guéhéneuc, P. Leduc. *Automatic Generation of Detection Algorithms for Design Defects.* Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE 2006), pp. 297-300, September 18-22, 2006, Tokyo, Japan.

Conférences nationales

N. Moha, F. Khomh, Y.-G. Guéhéneuc, L. Duchien, A.-F. Le Meur. *Génération automatique d'algorithmes de détection des défauts de conception*. In Mireille Blay-Fornarino, editor, Actes du 14e colloque Langages et Modèles à Objets, mars 2008.

N. Moha, D.-L. Huynh, Y.-G. Guéhéneuc. *Une taxonomie et un métamodèle pour la détection des défauts de conception.* In Roger Rousseau, editor, actes du 12e colloque Langages et Modèles à Objets, Hermès Science Publications, pp.201-216, March 22-24, 2006, Nîmes, France.

Démonstrations d'outils

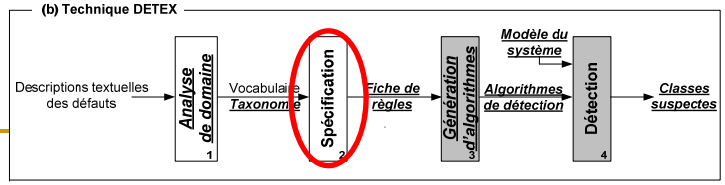
N. Moha, Y.-G. Guéhéneuc, *Ptidej and DECOR: Identification of Design Patterns and Design Defects*. 21st International Conference on Object-Oriented Programming, Systems, Languages and Applications, October 2007.

N. Moha and Y.-G. Guéhéneuc. *Ptidej and DECOR: Identification of Design Patterns and Design Defects*. 22nd International Conference on Automated Software Engineering, November 2007.

Questions

Merci pour votre attention !

Specifications

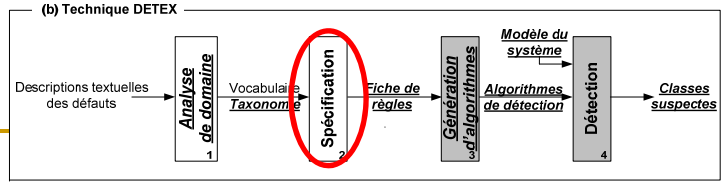


DSL

- Language based on a meta-model for specifying DDs

```
CODESMELL  define  LongMethod     as  METRIC  LOC_METHOD with  VERY HIGH and 10.0 ;
CODESMELL  define  NoParameter    as  METRIC  NMNOPARAM  with  VERY HIGH and 5.0 ;
CODESMELL  define  NoInheritance  as  METRIC  DIT        with  1 and 0.0 ;
CODESMELL  define  NoPolymorphism as  STRUC   NO_POLYMORPHISM ;
CODESMELL  define  ProceduralName as  LEXIC   CLASS_NAME with  (Make, Create, Exec) ;
CODESMELL  define  GlobalVariable as  STRUC   USE_GLOBAL_VARIABLE ;

ANTIPATTERN define SpaghettiCode  as {
       ((LongMethod INTER NoParameter) INTER (NoInheritance UNION NoPolymorphism))
   INTER
       (ProceduralName UNION UseGlobalVariable) } ;
```

Specifications

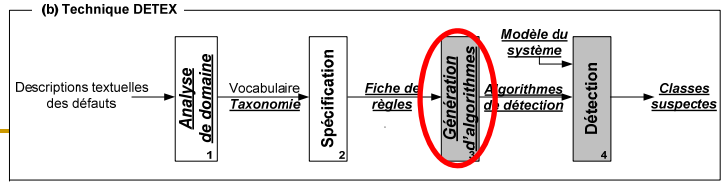


DSL

- Language based on a meta-model for specifying DDs

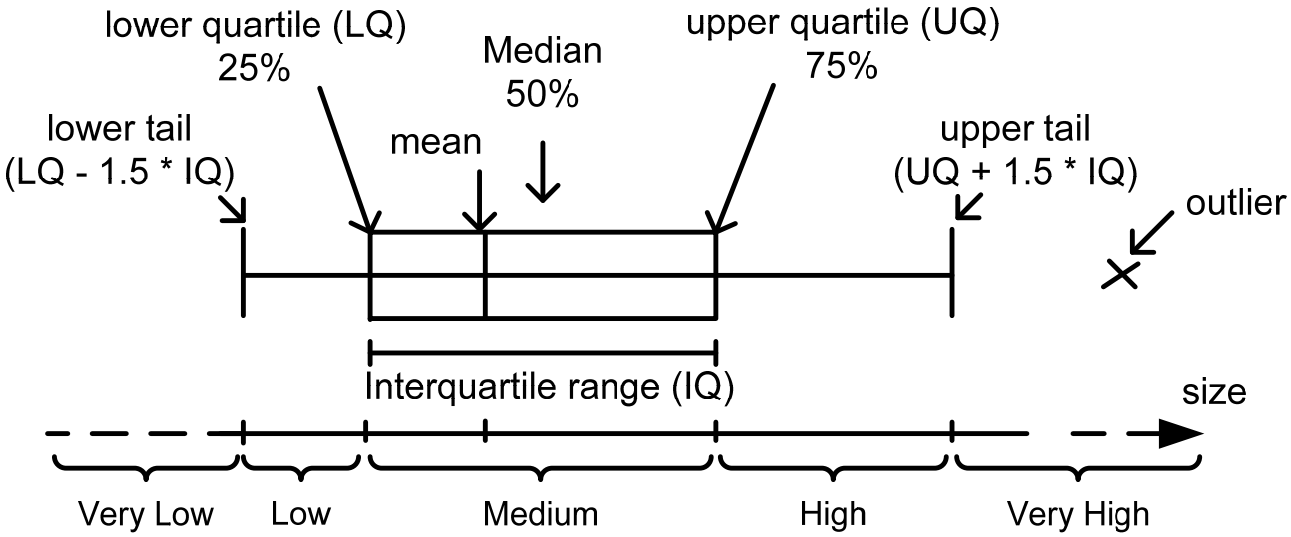
```
CODESMELL  define  LongMethod     as  METRIC  LOC_METHOD with  VERY HIGH and 10.0 ;
CODESMELL  define  NoParameter    as  METRIC  NMNOPARAM  with  VERY HIGH and 5.0 ;
CODESMELL  define  NoInheritance  as  METRIC  DIT        with  1 and 0.0 ;
CODESMELL  define  NoPolymorphism as  STRUC   NO_POLYMORPHISM ;
CODESMELL  define  ProceduralName as  LEXIC   CLASS_NAME with  (Make, Create, Exec) ;
CODESMELL  define  GlobalVariable as  STRUC   USE_GLOBAL_VARIABLE ;

ANTIPATTERN define SpaghettiCode  as {
       ((LongMethod INTER NoParameter) INTER (NoInheritance UNION NoPolymorphism))
   INTER
       (ProceduralName UNION UseGlobalVariable) } ;
```

Génération des algos




```
RULE: LongMethod  { METRIC  LOC_METHOD  VERY_HIGH  10.0 };
```

Box-plot pour les métriques de type ordinal



Pot de thèse

Vous êtes les bienvenus à mon pot de thèse!

Pavillon André Aisenstadt, 6ième étage, salon Maurice-Labbé

Directions : prenez à droite en sortant puis la 1ère sortie, à droite, bâtiment en face