
Detection and Correction of Design Defects in Object-Oriented Architectures

Naouel MOHA

Under the supervision of:

Yann-Gaël Guéhéneuc, Laurence Duchien, Anne-Françoise Le Meur

Ptidej Team, GEODES, University of Montreal, Canada
LIFL, INRIA project Jacquard, University of Lille, France

MoSART (Montreal Software Analysis Research Talks)

October 10th, 2007



Context

- **Design Patterns** are “good” solutions to recurring design problems
- **Design Defects (DDs)**
 - are “bad” solutions to recurring problems
 - 2 categories:
 - High-level (global) problems: **antipatterns** [Brown 98]
 - Low-level (local) problems: **code smells*** [Fowler 99]

≠ “*deviations from specifications or expectations which might lead to failures in operation*”

* *Some code smells can be considered as high-level DDs*

Context

- Why it is important to detect and correct DDs ?
 - DDs **lessen the quality** of OO architectures and impede their evolution and their maintenance
 - **Maintenance is expensive** because of DDs [PER 92] : adding, debugging, and evolving of features are difficult
 - A **good software architecture without DDs**: easier to understand, change, and thus maintain

- Motivation
 - Reach a **high-quality software**
 - **Reduce the costs** of software development

Context

- 2 examples of high-level DDs [Brown 98]

```
18     if (fNamespacesEnabled) {
19         fNamespacesScope.increaseDepth();
20         if (attrIndex != -1) {
21             int index = fAttrList.getFirstAttr(attrIndex);
22             if (index != -1) {
23                 fPool.equalNames(...) {
24                     ...
25                 }
26             }
27         }
28     }
29 }
30 }
31 int pre...
32 int elem... RI;
33 if (pre... -1) {
```

- **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”

- ❑ Large controller class
- ❑ Many fields and methods with a **low cohesion***
- ❑ Dependent on the data stored in associated **data classes**

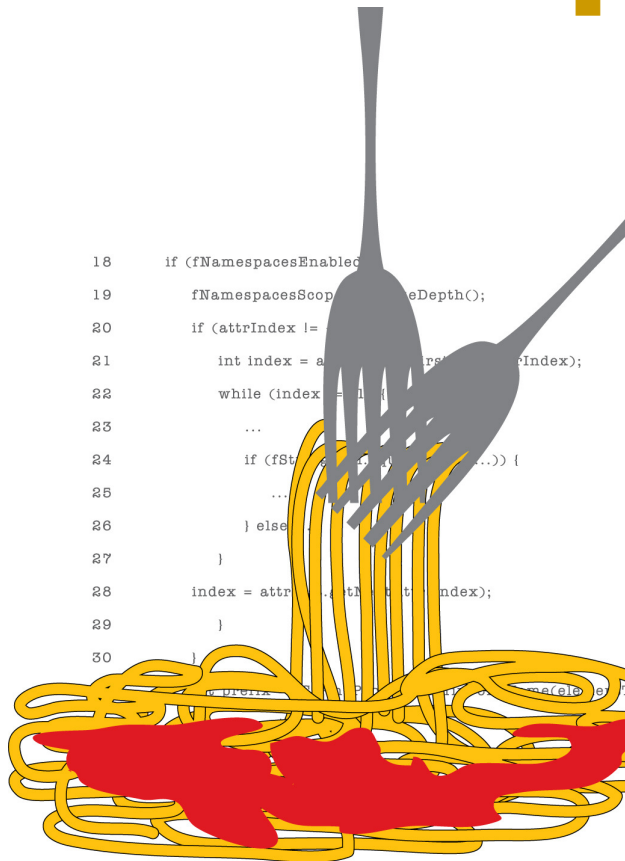
**How closely the methods are related to the instance variables in the class.*

Context

- 2 examples of high-level DDs [Brown 98]

- Spaghetti Code

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



- Procedural thinking in OO programming
- Lack of structure : no inheritance, no reuse, no polymorphism
- Long methods process oriented with no parameters and low cohesion
- Procedural names of classes and methods
- Negligible degree of interaction between objects
- Use of global variables for processing

Overview

- Problem Statement
- Proposed Approach
- Progress
- Conclusion

Problem Statement

- What are the problems ?
 - Textual descriptions of DDs → Misinterpretation
 - Lack of work on high-level DDs → Focus on low-level DDs
 - Detection based mainly on metrics → Lack of precision
 - Manual correction → Time-, resource-consuming, error-prone activity
 - No systematic method
 - Ad hoc way of specifying detection and correction algos

■ Thesis

Develop an approach that allows to automate the detection and correction of high-level DDs

Overview

- Problem Statement
- Proposed Approach
- Progress
- Conclusion

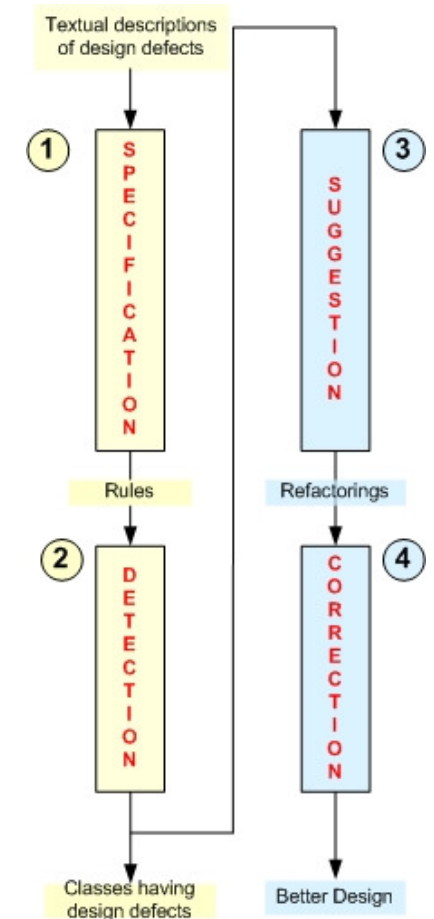
Proposed Approach

■ Our Objectives

- Combine and extend previous works
- Provide a complete method that integrates the best
- Fill the gap between the detection and the correction techniques

■ Our Approach consists in 4 stages :

- 1) Specification
- 2) Detection
- 3) Suggestion
- 4) Correction



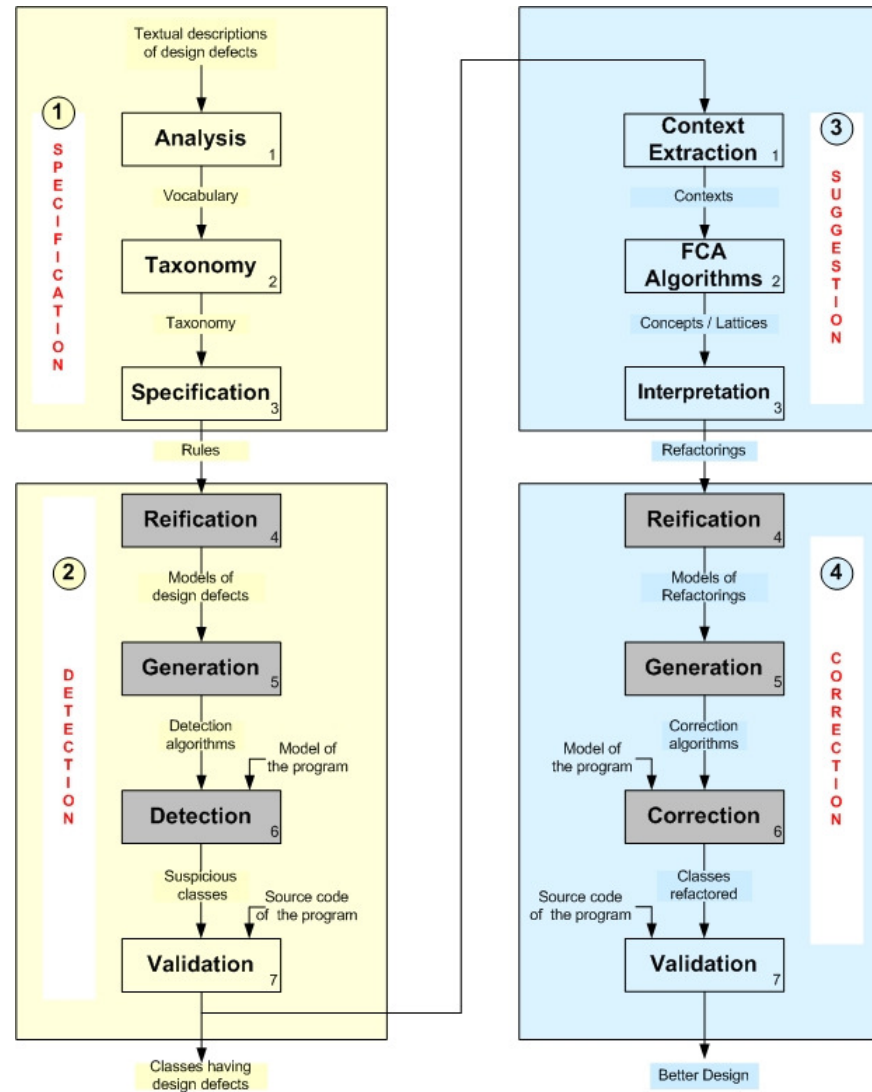
We define a systematic method, called **DECOR**, to specify high-level DDs significantly and to generate detection and correction algorithms from the specifications of DDs



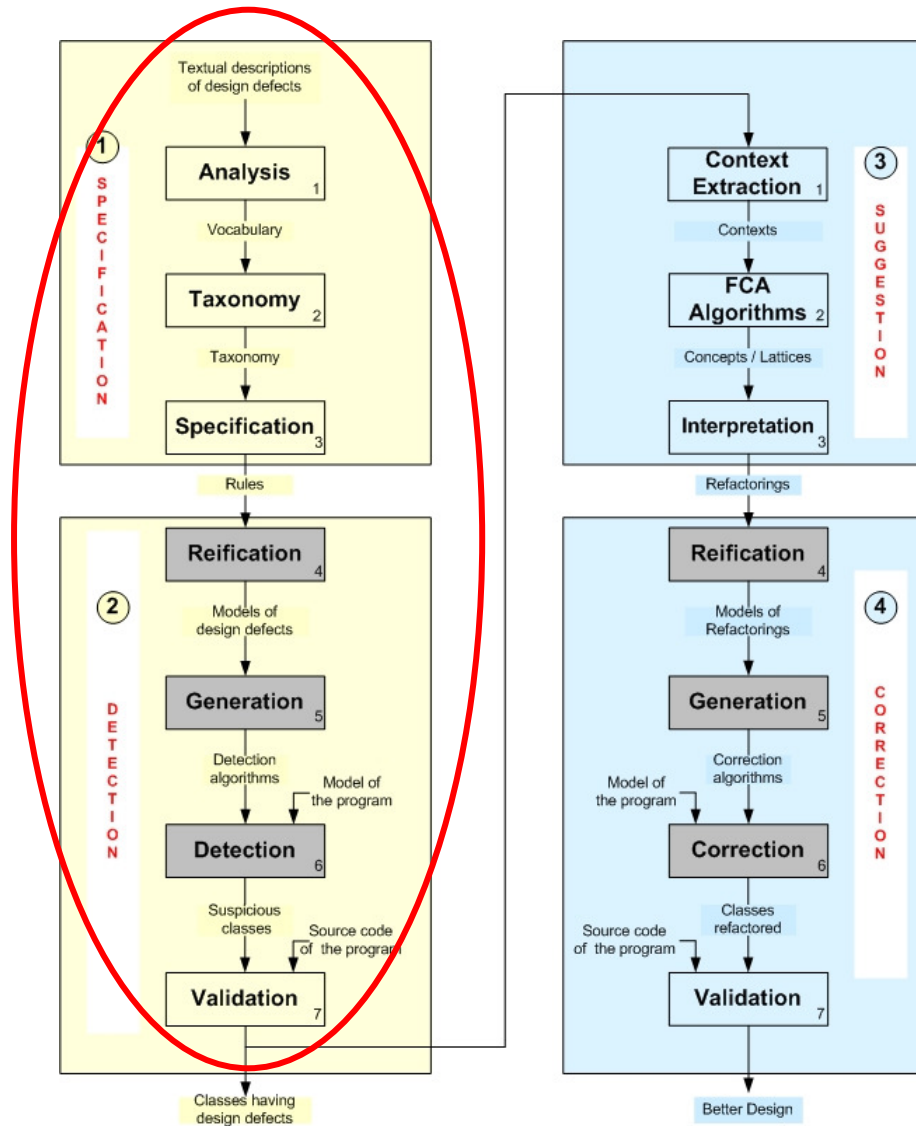
Overview

- Problem Statement
- Proposed Approach
- Progress
- Conclusion

Progress



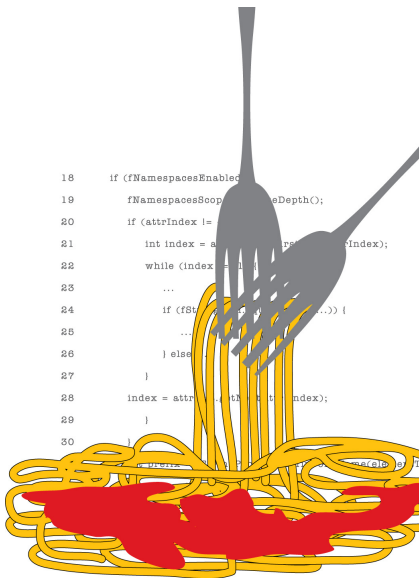
Progress



Progress on Detection

We define a systematic method, called **DECOR**, to specify high-level DDs significantly and to generate detection and correction algorithms from the specifications of DDs

→ Illustration with the Spaghetti Code



- ❑ Procedural thinking in OO programming
- ❑ No inheritance
- ❑ Long methods process oriented with no parameters and low cohesion
- ❑ Procedural names of classes and methods
- ❑ Use of global variables for processing

Progress on Detection

3. Domain-specific Language [Moha 06b]

```
RULE_CARD : SpaghettiCode {  
    ...  
    RULE: LongMethodMethodNoParameter {INTER LongMethod MethodNoParameter};  
    RULE: LongMethod {(METRIC: LOC_METHOD, VERY_HIGH)};  
    RULE: MethodNoParameter {(METRIC: NB_PARAM, 0)};  
    ...  
    RULE: NoInheritance {(METRIC: DIT, 1)} ;  
    RULE: FunctionClassGlobalVariable {UNION FunctionClass GlobalVariable};  
    RULE: FunctionClass {(LEXIC: CLASSNAME, {Make, Create, Creator, Exec}) };  
    RULE: GlobalVariable {(STRUCT: FIELD, CLASS_GLOBAL_VAR)};  
};
```

Progress on Detection

Detection & Validation : We specify and detect DDs in several open-source programs

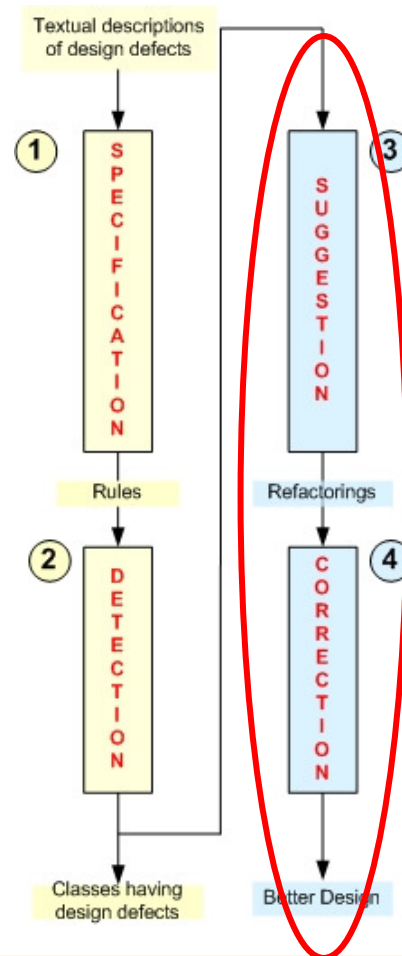
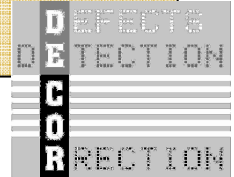
Blob						
	ArgoUML 113 KLOC 1230 classes	Azureus 192 KLOC 1449 classes	GanttProject 22 KLOC 188 classes	PMD 42 KLOC 423 classes	QuickUML 9 KLOC 142 classes	Total
Effectifs	91	143	19	15	3	271
Precisions	70/91 = 76.9%	82/143 = 57.3%	10/19 = 52.6%	3/15 = 20%	1/3 = 33.3%	166/271 = 61.5%
Time	40.5s	2m	5.6s	31.6s	2.1s	3m 19.8s

Spaghetti Code						
	ArgoUML	Azureus	GanttProject	PMD	QuickUML	Total
Effectifs	26	35	8	11	1	81
Precisions	21/26 = 80.7%	29/35 = 82.8%	6/8 = 75%	6/11 = 54.5%	0/1 = 0%	62/81 = 76.6%
Time	39s	1m55s	5.4s	31s	1.9s	3m12s

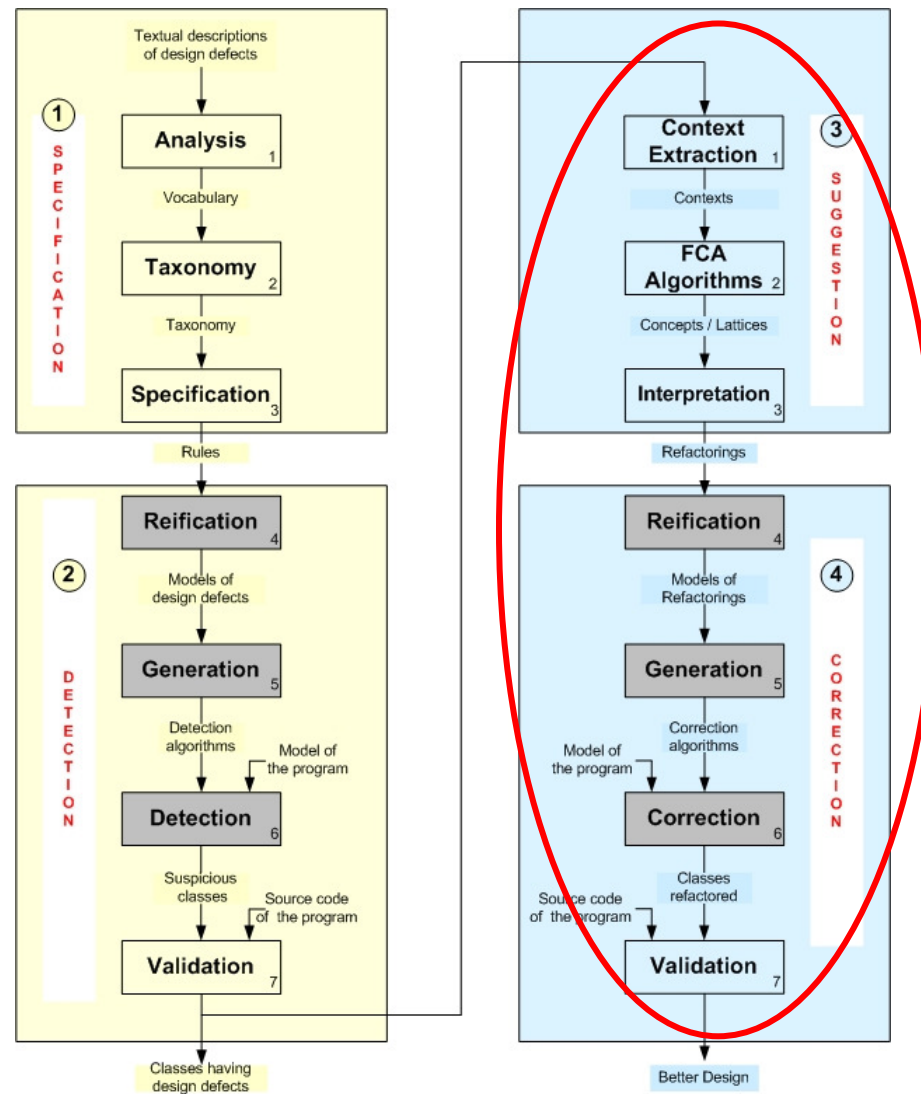
→ **Results are encouraging ! 😊**

Progress on Suggestion

We define a systematic method, called **DECOR**, to specify high-level DDs significantly and to generate detection and correction algorithms from the specifications of DDs



Progress



Progress on Suggestion

→ Illustration with the Blob

```
18     if (fNamespacesEnabled) {
19         fNamespacesScope.increaseDepth();
20         if (attrIndex != -1) {
21             int index = List.getFirstAttr(attrIndex);
22             if (index != -1) {
23                 if (fPool.equalNames(...)) {
24                     ...
25                 }
26             }
27         }
28     }
29 }
30 }
31 int pre...
32 int elem... RI;
33 if (pre... -1) {
```

■ Blob (God Class)

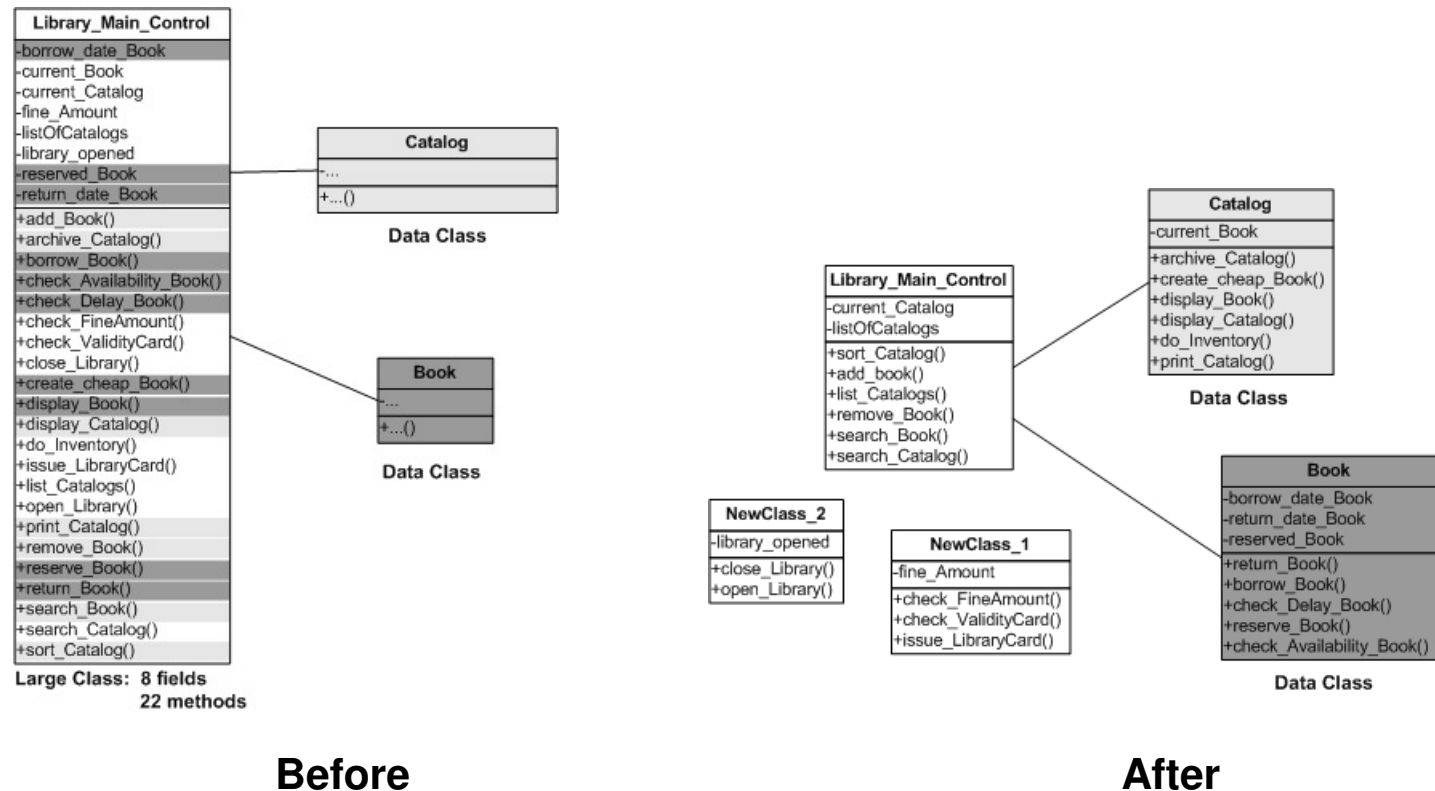
“Procedural-style design leads to one object with a lion’s share of the responsibilities while most other objects only hold data or execute simple processes”

- ❑ Large controller class
- ❑ Many fields and methods with a **low cohesion***
- ❑ Dependent on the data stored in associated **data classes**

**How closely the methods are related to the instance variables in the class.*

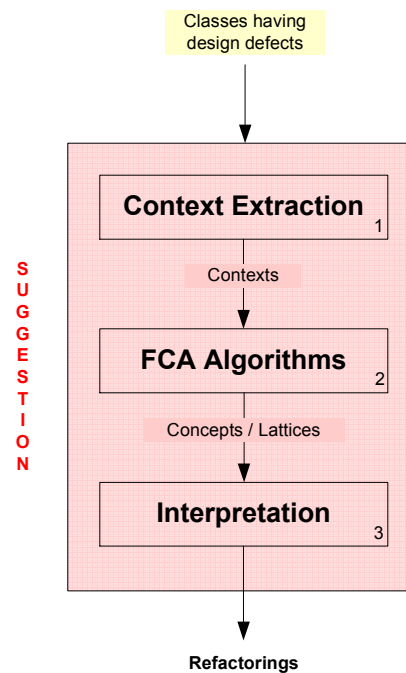
Progress on Suggestion

- DDs resulting in **high coupling and low cohesion** could be improved
- Redistribute class members among existing classes (with possibly new classes) to increase cohesion and/or decrease coupling



Progress on Suggestion

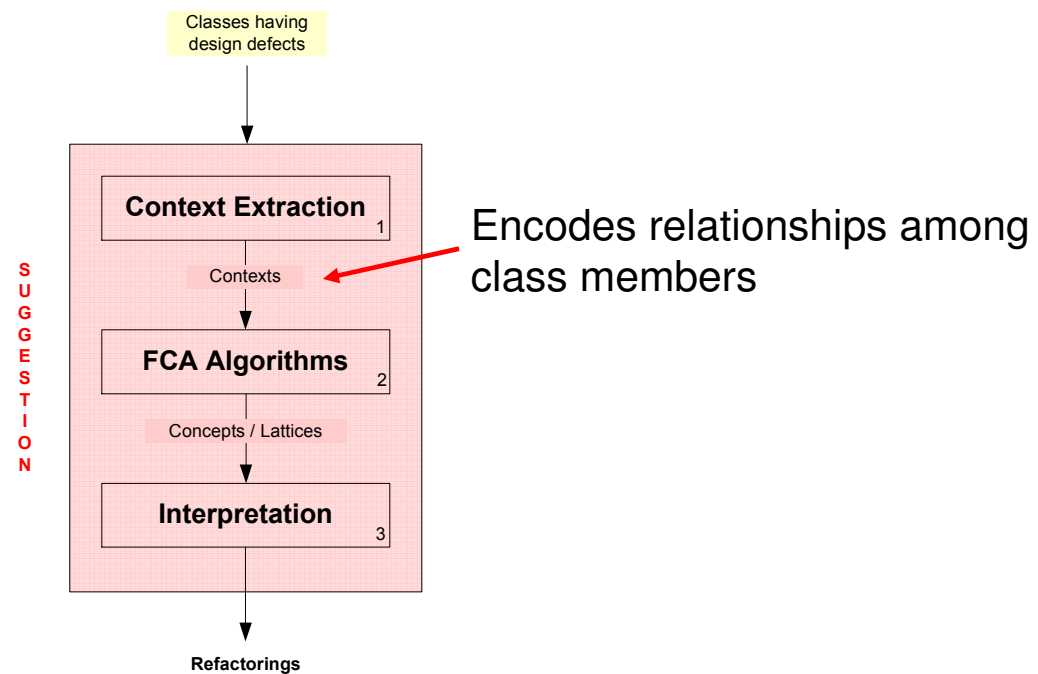
- **Formal Concept Analysis [Moha 06c]**
 - FCA offers a formal framework for clustering individuals along the properties they share
 - Identify methods that share common fields and methods that call common methods → **cohesive sets low coupled**



Progress on Suggestion

■ Formal Concept Analysis [Moha 06c]

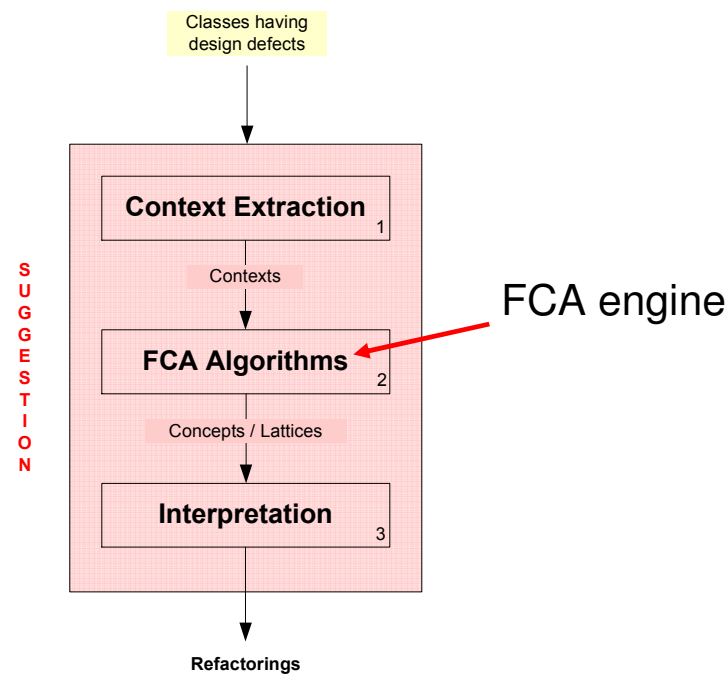
- FCA offers a formal framework for clustering individuals along the properties they share
- Identify methods that share common fields and methods that call common methods → **cohesive sets low coupled**



Progress on Suggestion

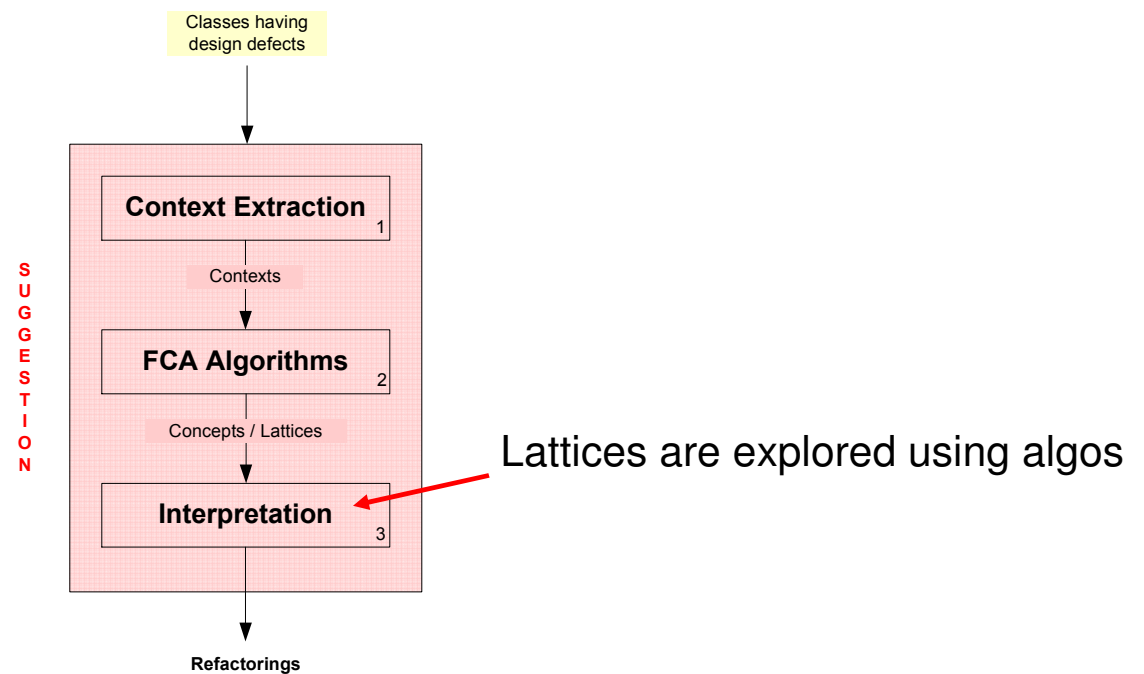
■ Formal Concept Analysis [Moha 06c]

- FCA offers a formal framework for clustering individuals along the properties they share
- Identify methods that share common fields and methods that call common methods → **cohesive sets low coupled**



Progress on Suggestion

- **Formal Concept Analysis [Moha 06c]**
 - FCA offers a formal framework for clustering individuals along the properties they share
 - Identify methods that share common fields and methods that call common methods → **cohesive sets low coupled**



Progress on Suggestion

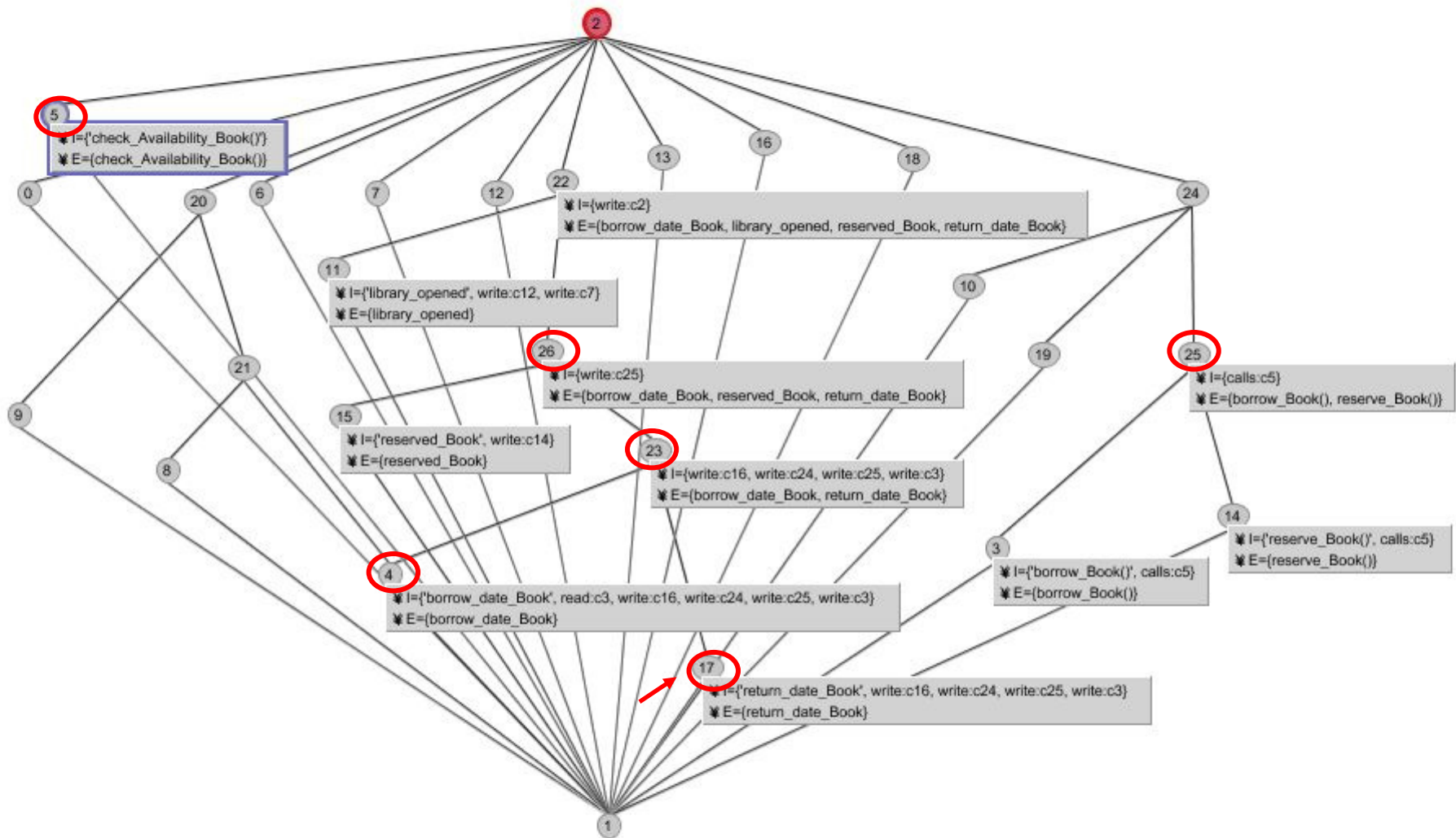
1. Context Extraction

	add_Book()																				
	borrow_Book()	×																			
	check_Availability_Book()																				
	check_FineAmount()																				
	close_Library()																				
	issue_LibraryCard()																				
	open_Library()																				
	remove_Book()																				
	reserve_Book()																				
	return_Book()																				
	search_Book()																				
	sort_Catalog()																				
borrow_date_Book																					
current_Book																					
fine_Amount																					
library_opened																					
reserved_Book																					
return_date_Book																					

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

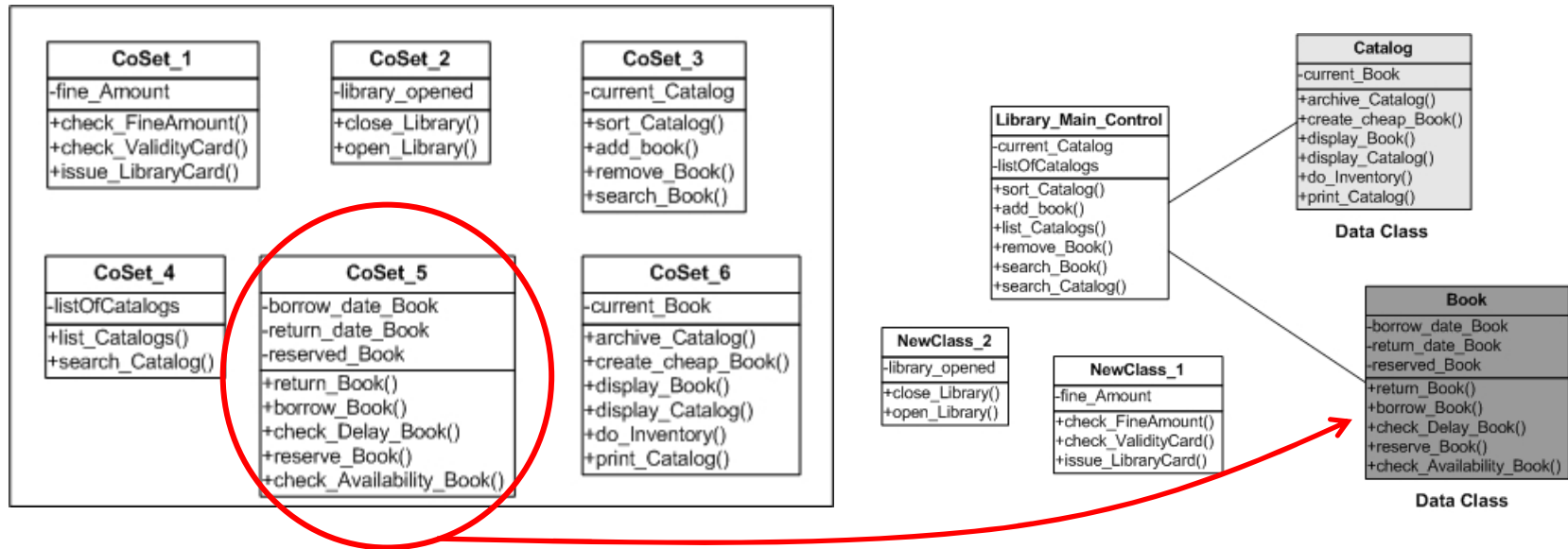
Progress on Suggestion

2. FCA Algorithms



Progress on Suggestion

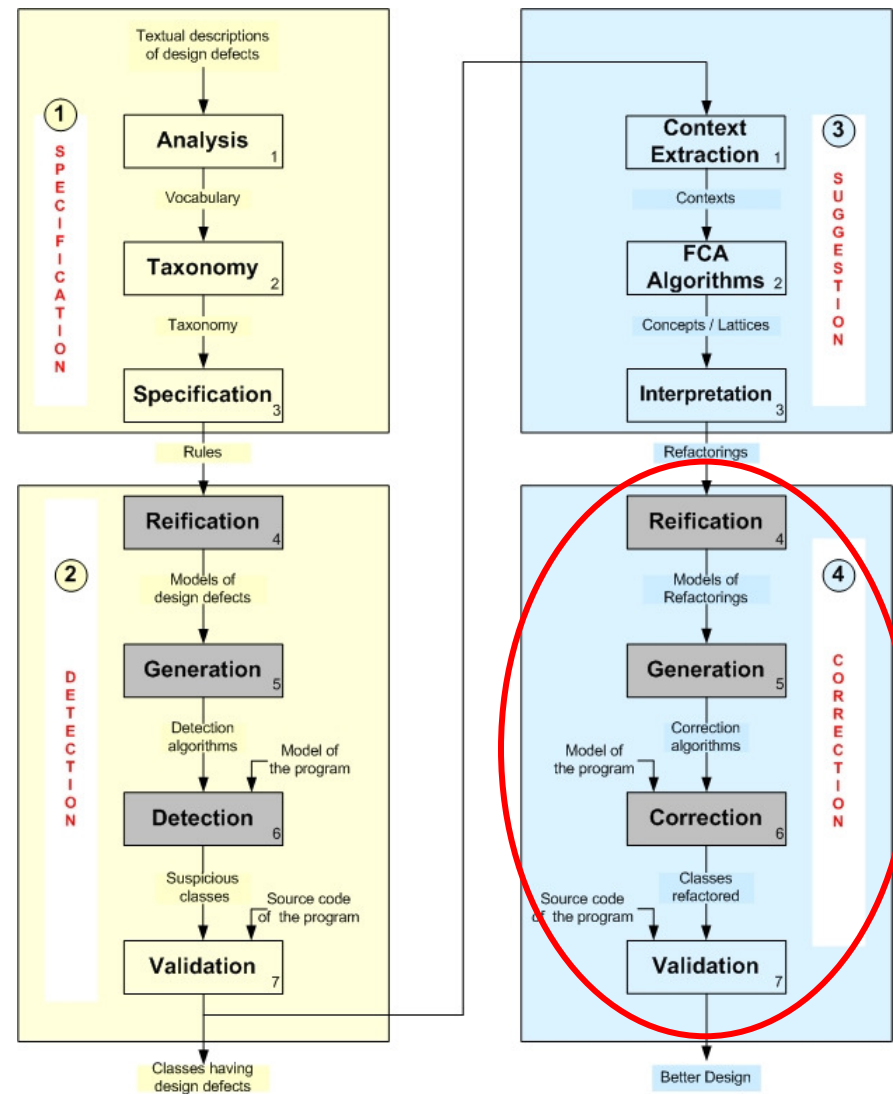
3. Interpretation



Cohesive Sets

After Refactoring

Progress



Overview

- Problem Statement
- Proposed Approach
- Progress
- Conclusion

Conclusion

■ Problem Statement

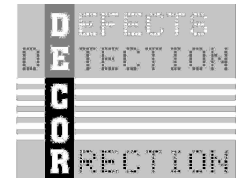
Develop an approach that allows to automate the detection and correction of high-level DDs

■ Our approach

- A systematic method based on a domain-analysis and a rule-based language to generate automatically detection and correction algos of high-level DDs
- Originality on the automatic generation of algos

■ Future Work

- Experiments
- Comparisons with existing approaches

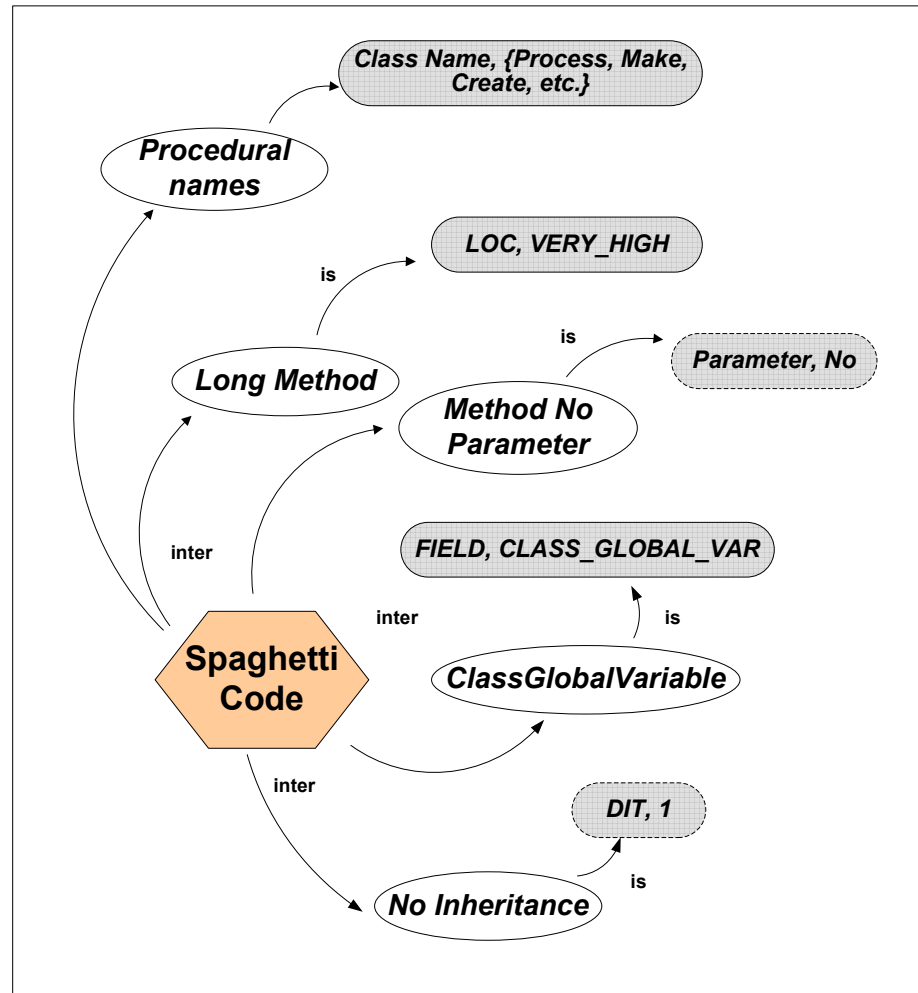


Questions



Progress

2. Taxonomy : Map of DDs



Box-plot

The Box-plot statistical technique

