

# Refactorings of Design Defects using Relational Concept Analysis

**Naouel Moha, Amine Mohamed Rouane Hacene,  
Petko Valtchev, and Yann-Gaël Guéhéneuc**

LORIA, France

DIRO, University of Montréal, Canada

LATECE, Université du Québec à Montréal, Montréal, Canada

**ICFCA'08**

Montréal (Qc), Canada, February 25-28, 2008

**UQÀM**



Université   
de Montréal

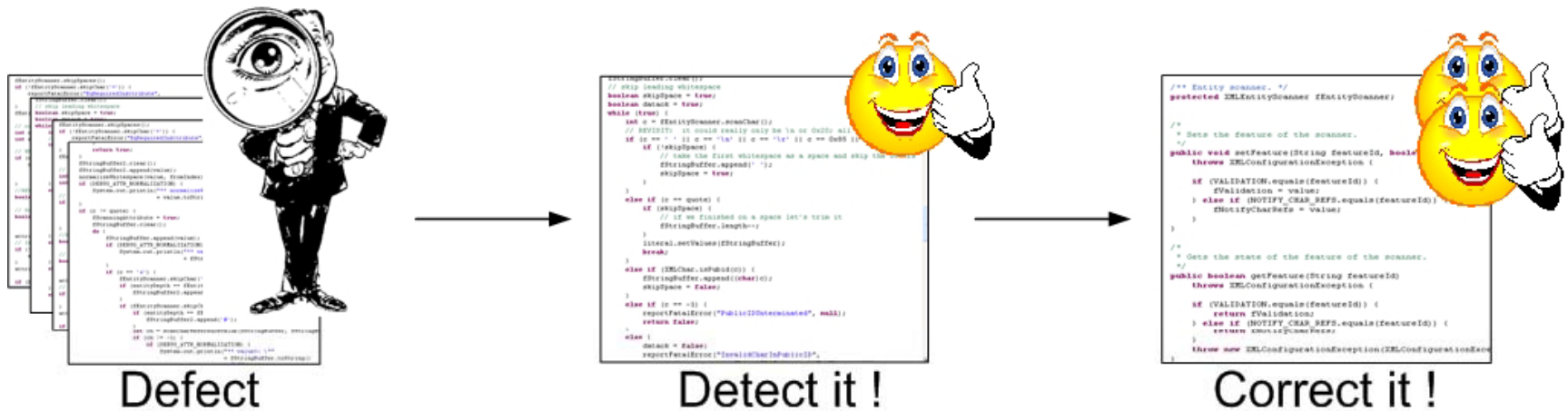
# Context



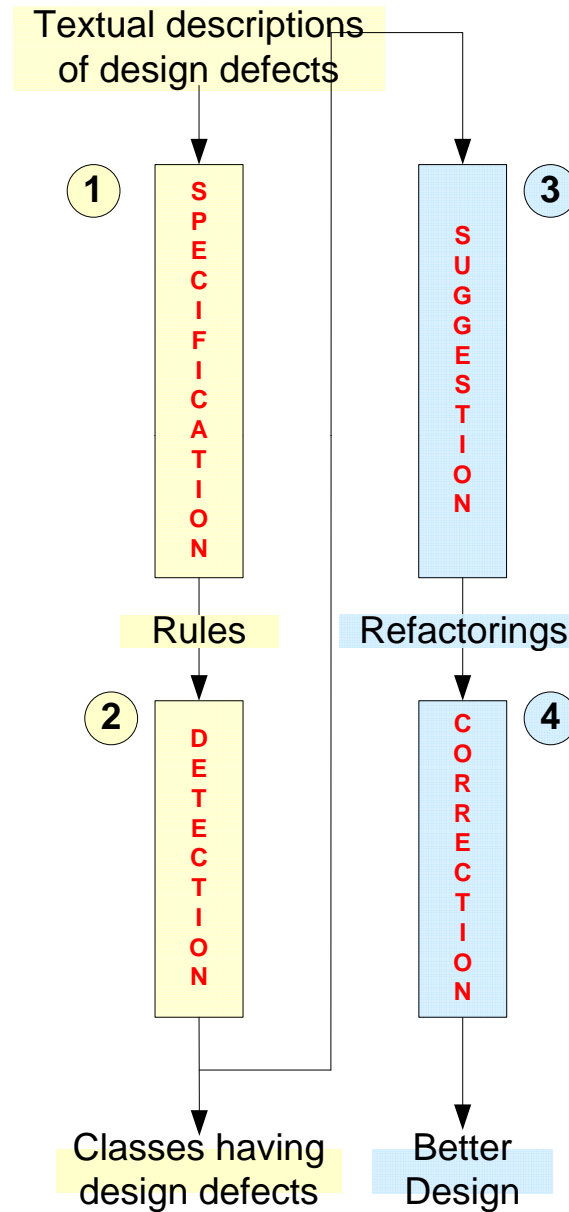
Defect



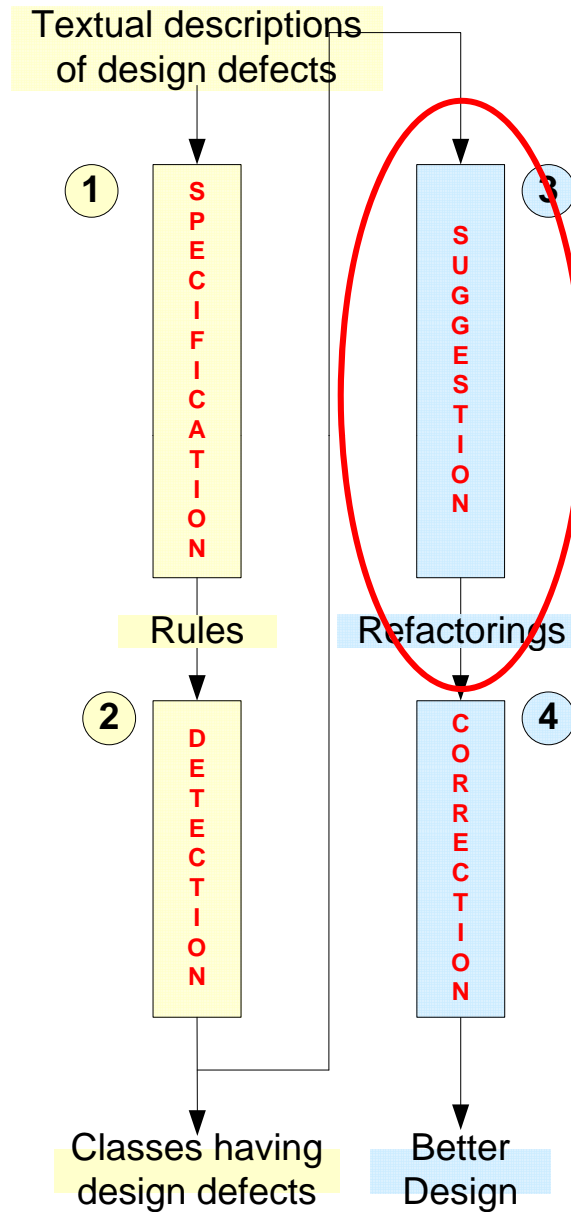
# Context



# Context: Overall Process



# Context: Overall Process



# Context: Overall Process

---

## ■ 1-2. Specification and Detection of DDs

- More and more tools and techniques

- Actively researched area

- Method DECOR (Defect dEtectioN for CORrection) [Moha 06]

## ■ 3. Suggestion

- Manual identification of the modifications

- Time-, resource-consuming, error-prone activity

## ■ 4. Correction

- Refactorings

Technique used to change *“the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior”*

# What Kind of Defects ?

---

- **Design Patterns** are “good” solutions to recurring design problems
- **Design Defects (DDs)**
  - are “bad” solutions to recurring problems, **Antipatterns** [Brown 98]
  - hinder development and maintenance by making programs harder to comprehend and/or evolve
- **DDs of interest:** infected by **low cohesion** and **high coupling**
  - **Cohesion:** how closely the methods are related to the variables in the class
  - **Coupling:** the degree of its reliance on services provided by other classes

# What Kind of Defects ?

- An example of DDs [Brown 98]

```
18     if (fNamespacesEnabled) {
19         fNamespacesScope.increaseDepth();
20         if (attrIndex != -1) {
21             int index = List.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 (p
```

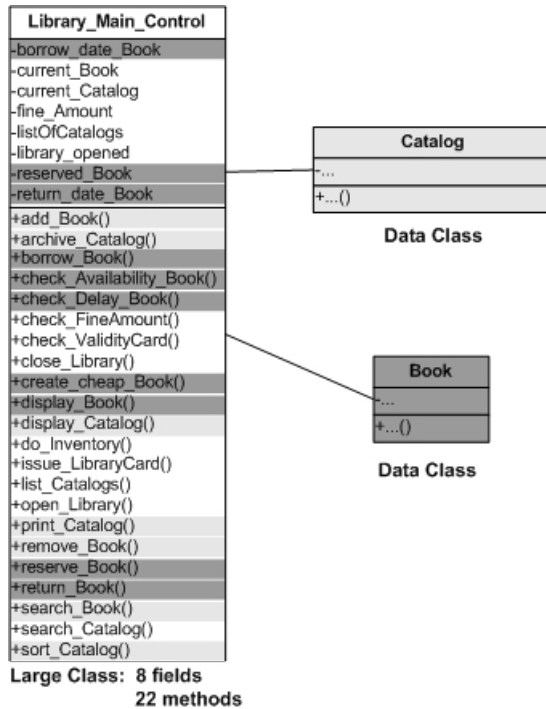
## ■ Blob (God Class)

*“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**
- ❑ **High coupled** with the data stored in associated **data classes**

# An Example

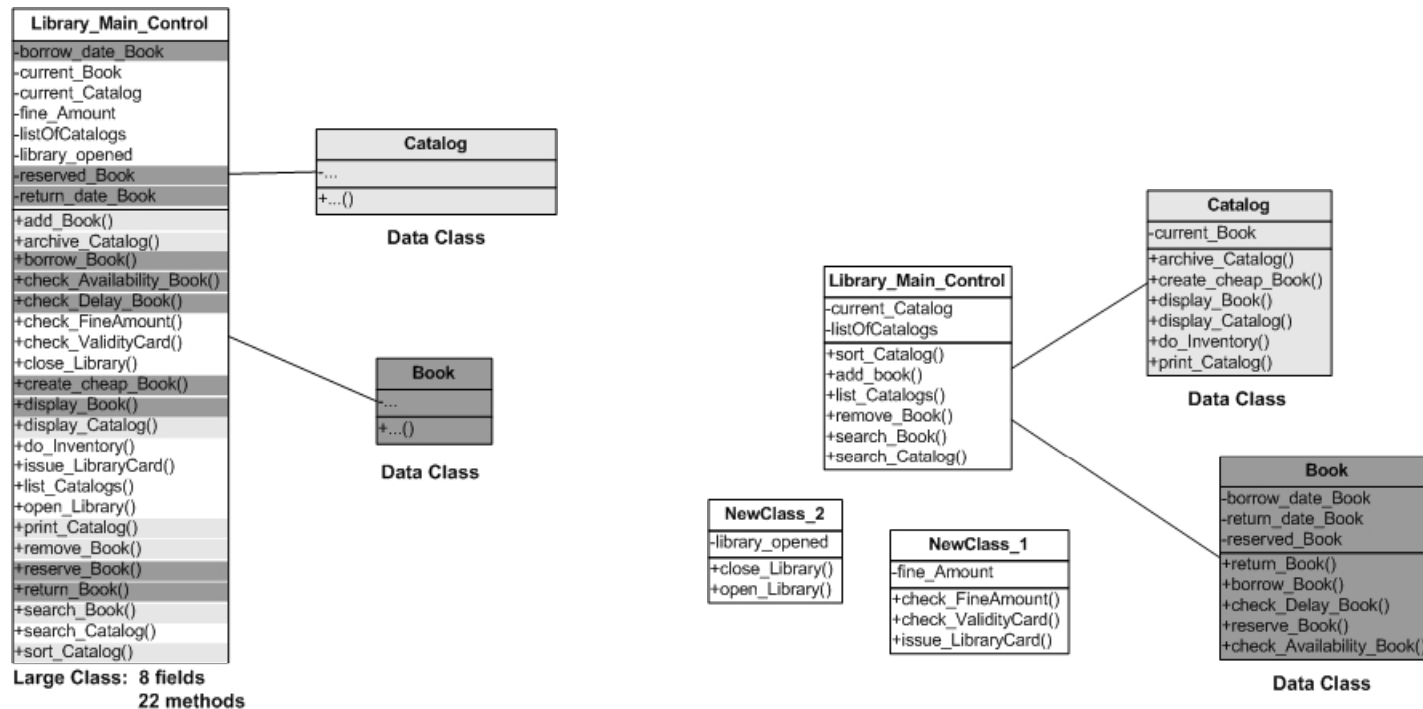
## Blob



## Before

# An Example

## How to correct the defect ?



Before

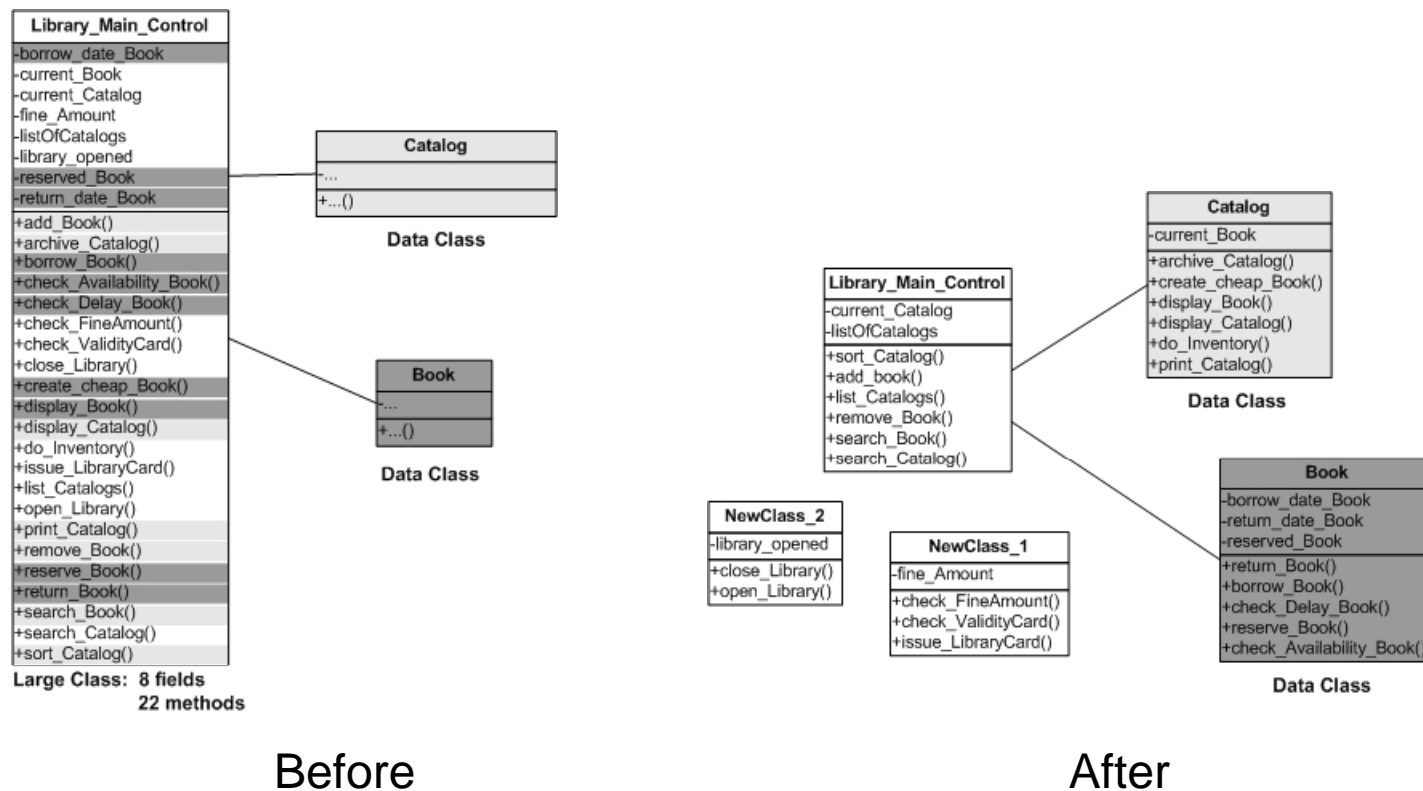
- Large and complex class
- Small data classes
- Not object-oriented

After

- Large class becomes less complex
- Data classes gain more behaviour
- More object-oriented style

# An Example

- Redistribute class members among existing classes (with possibly new classes) to increase cohesion and/or decrease coupling



# Contribution

---

## ■ Suggestion

- **RCA** [Rouane 07, Ph.D. Thesis]
  - Extension of **FCA** to relational data
  - Models the inter-object links and infers description logics role like **relations** between concepts
  - RCA provides a suitable framework for **clustering individuals along the properties they share and their links with other individuals**
  - Identify methods that share common fields and methods that call common methods → **cohesive sets low coupled**

# Contribution

---

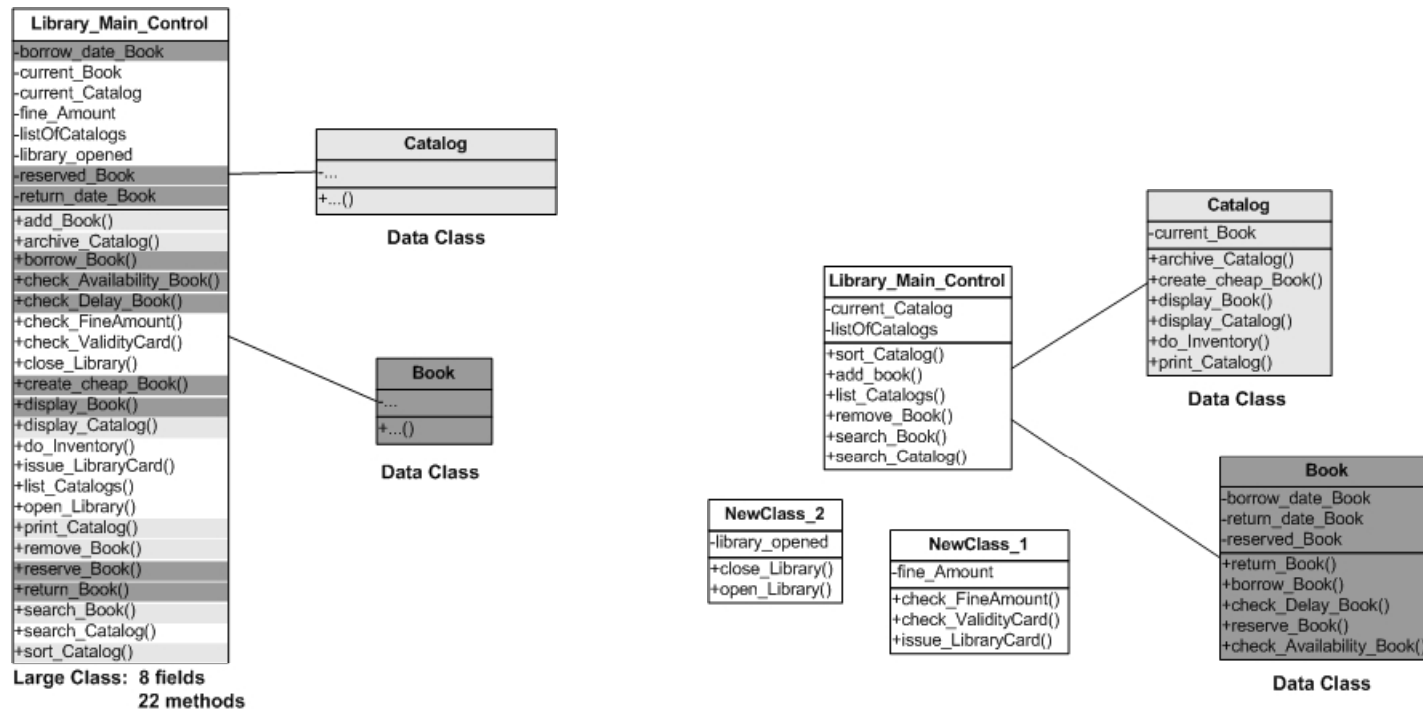
## ■ Suggestion

- **RCA** [Rouane 07, Ph.D. Thesis]
  - Extension of **FCA** to relational data
  - Models the inter-object links and infers description logics role like **relations** between concepts
  - RCA provides a suitable framework for **clustering individuals along the properties they share and their links with other individuals**
  - Identify methods that share common fields and methods that call common methods → **cohesive sets low coupled**

**« Develop an *automated approach* for suggesting *defect-correcting refactorings* using *RCA* »**

# An Example

## How to correct the defect ?



Before

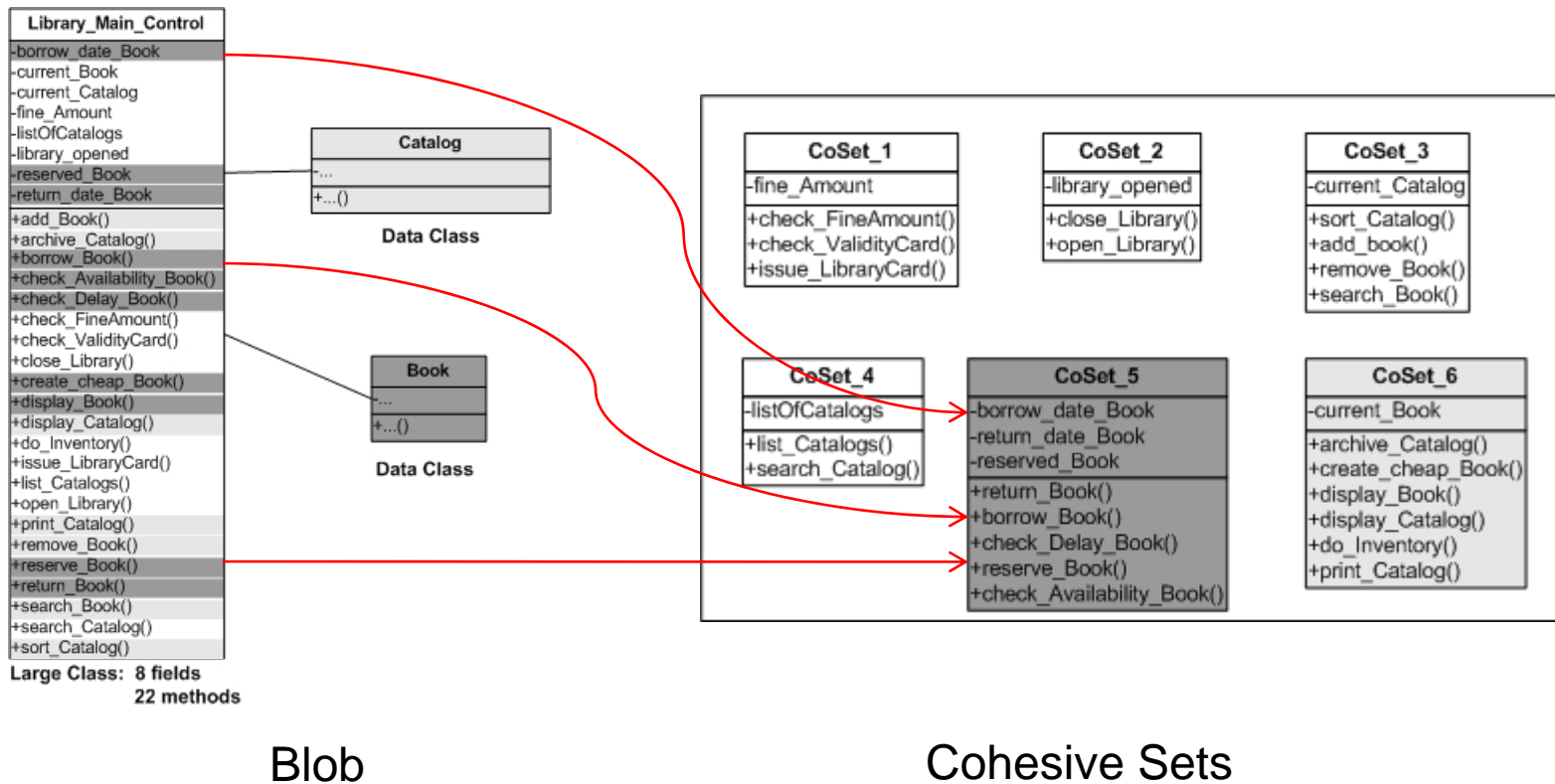
- Large and complex class
- Small data classes
- Not object-oriented

After

- Large class becomes less complex
- Data classes gain more behaviour
- More object-oriented style

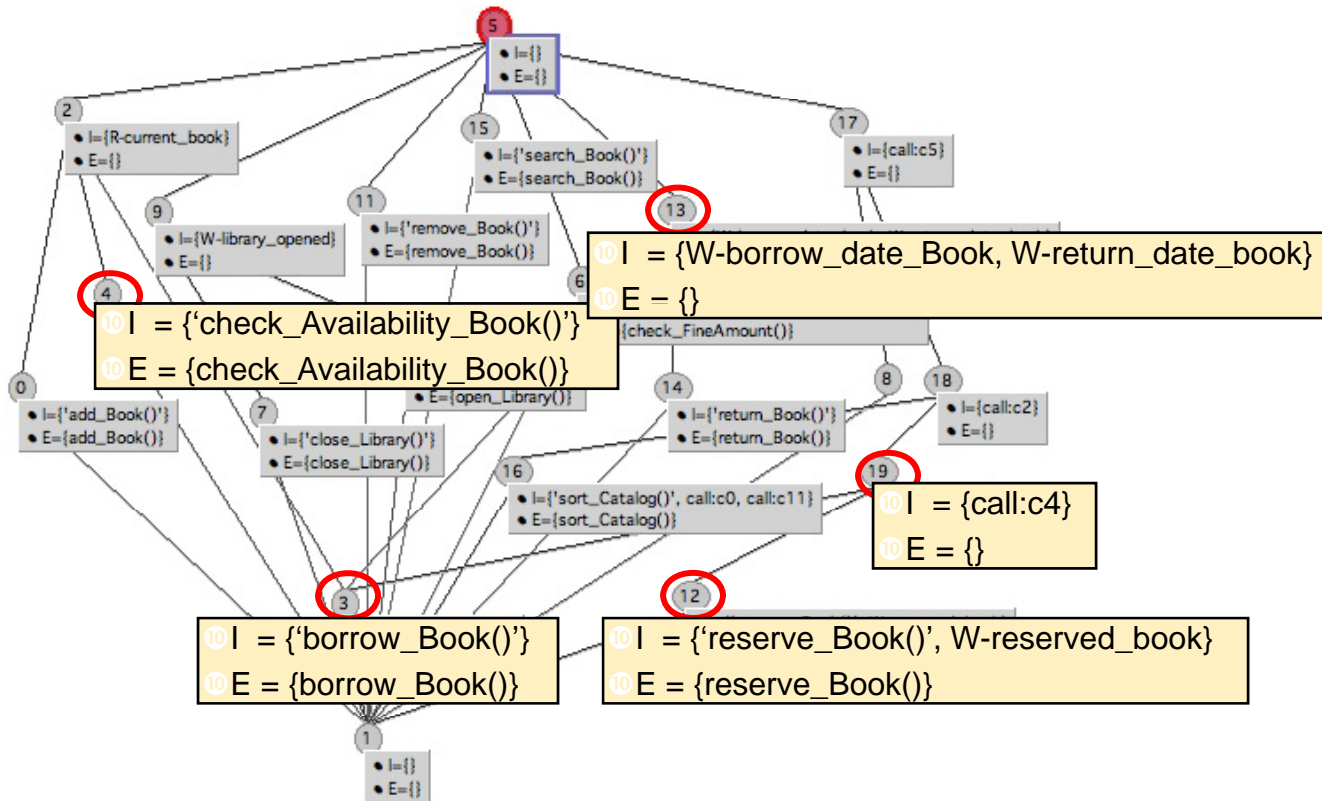
# An Example

## How to correct the defect ?



# An Example

*How to correct the defect ?*



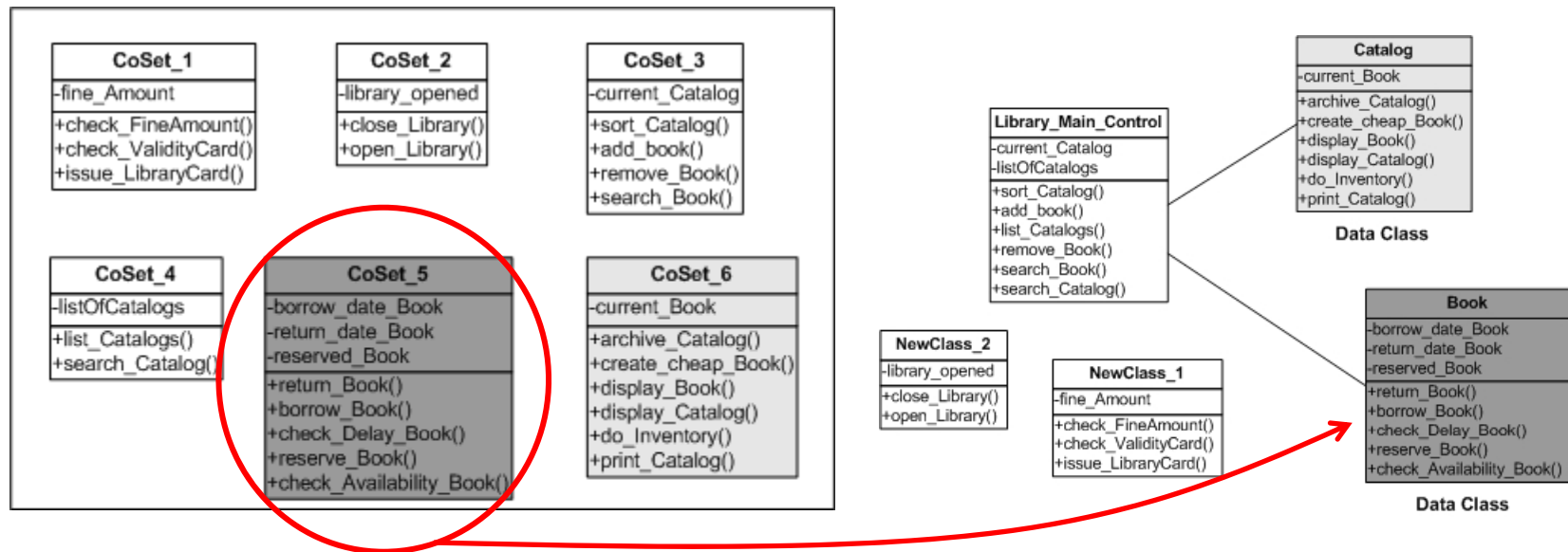
Lattice

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

Cohesive Set

# An Example

*How to correct the defect ?*

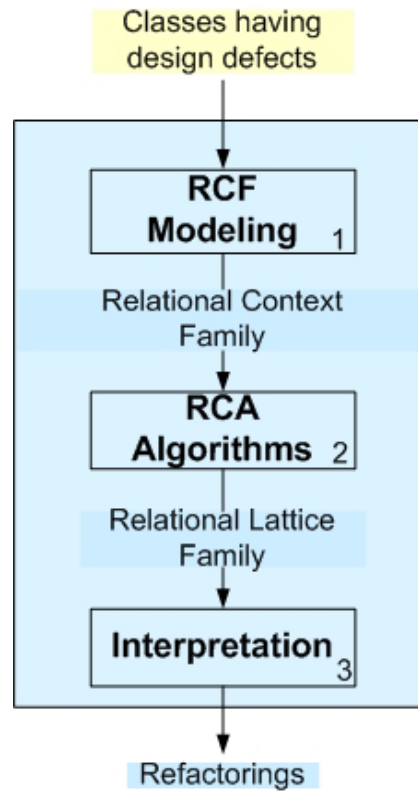


Cohesive Sets

After Refactoring \*

\* Move Method, Move Field, Create Class

# Suggestion

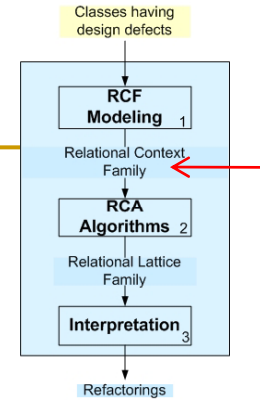


# Suggestion

## 1. RCF Modeling

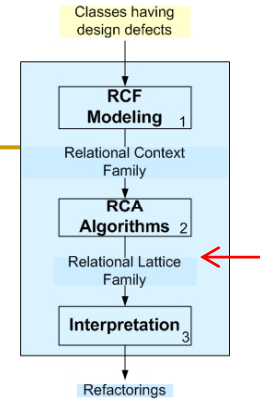
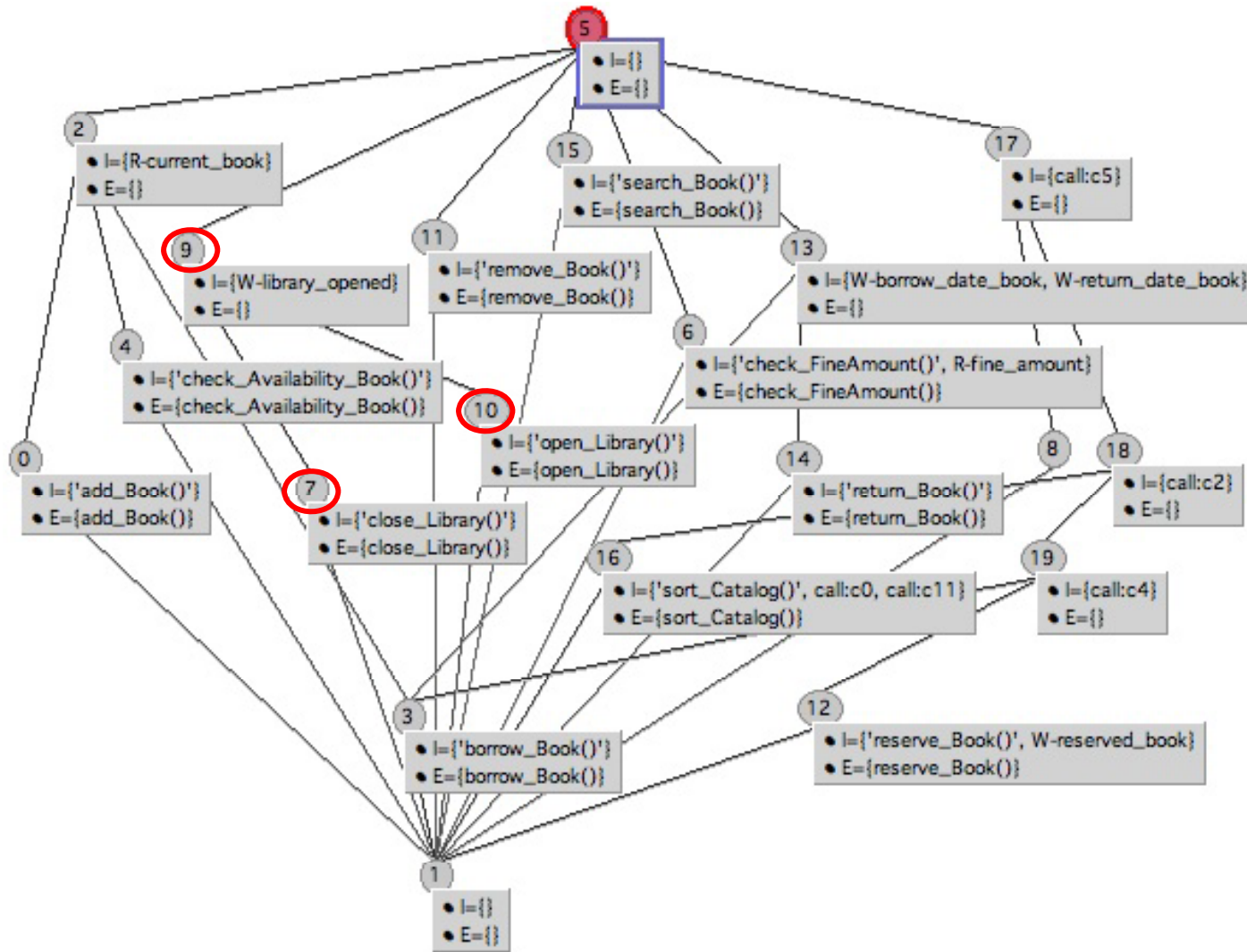
	'add_Book()'	'borrow_Book()'	'check_Availability_Book()'	'check_FineAmount()'	'close_Library()'	'issue_LibraryCard()'	'open_Library()'	'remove_Book()'	'reserve_Book()'	'return_Book()'	'search_Book()'	'sort_Catalog()'	R-current_book	R-fine_amount	W-borrow_date_book	W-library_opened	W-reserved_book	W-return_date_book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

add_Book()	check_Availability_Book()	check_FineAmount()	remove_Book()
	X		
		X	
	X		
X		X	



# Suggestion

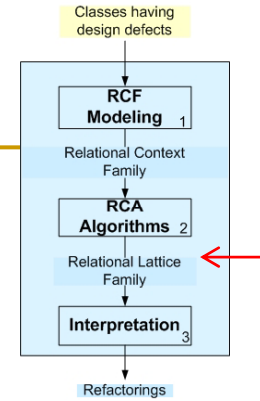
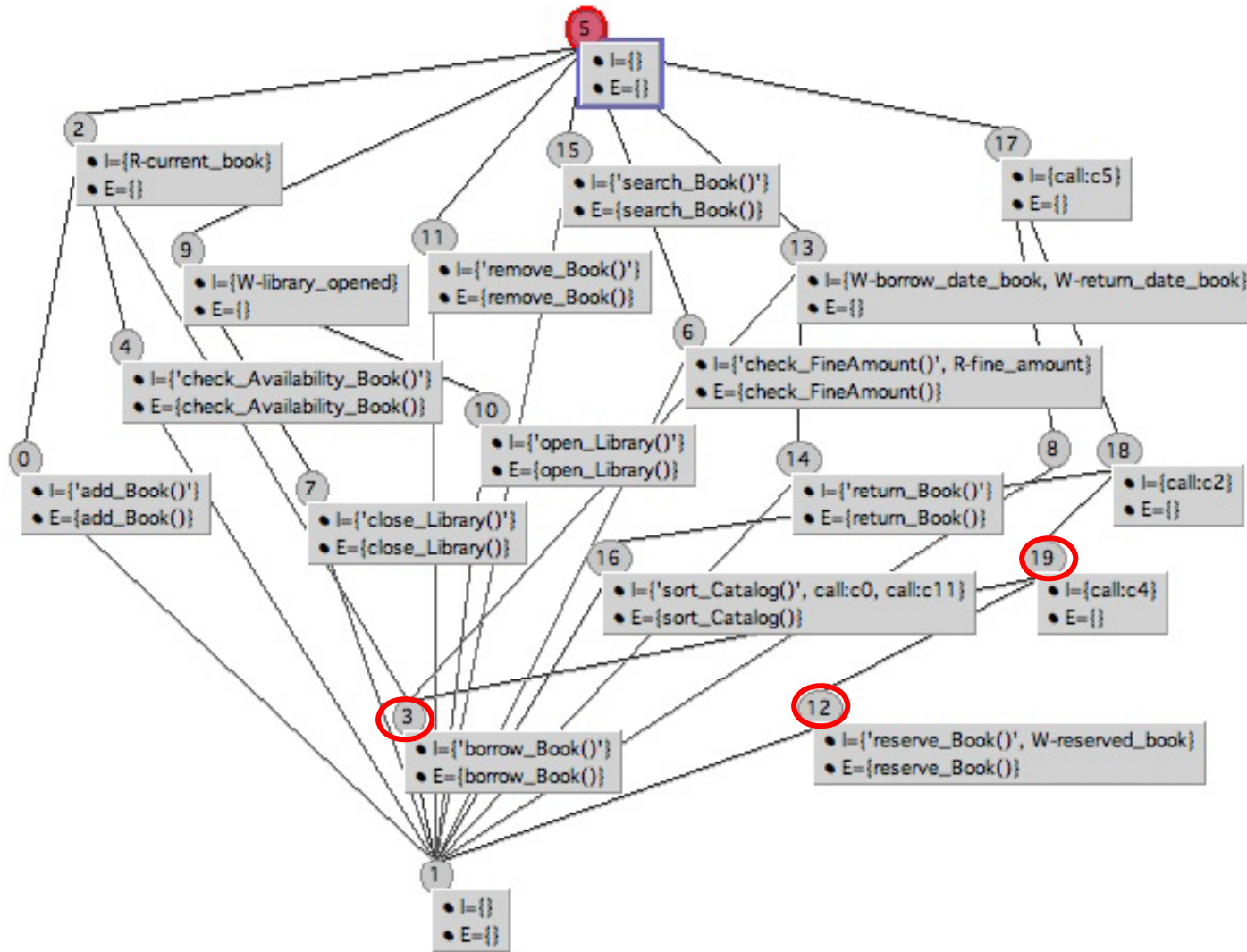
## 2. RCA Algorithms





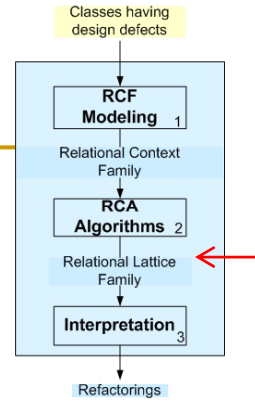
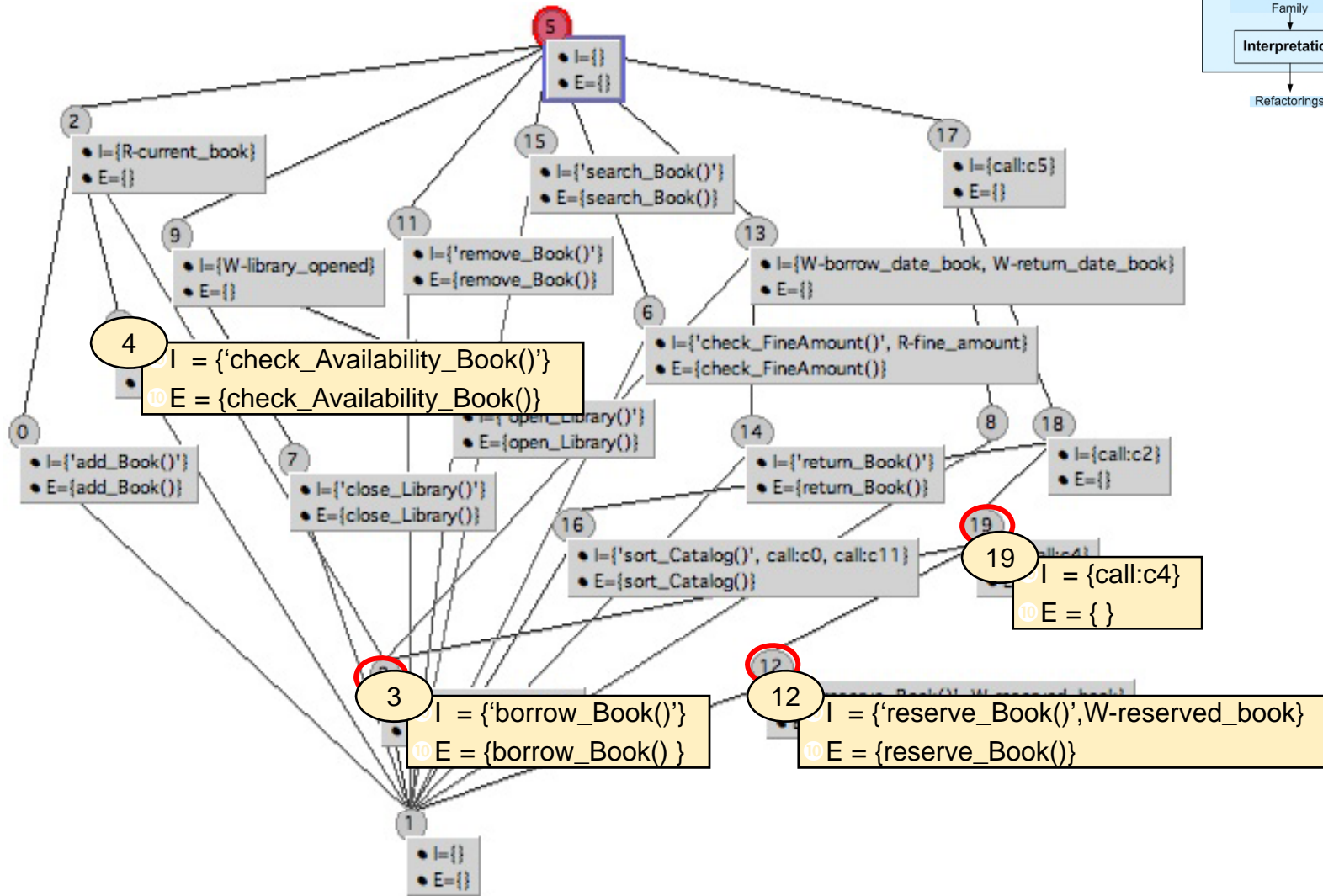
# Suggestion

## 2. RCA Algorithms



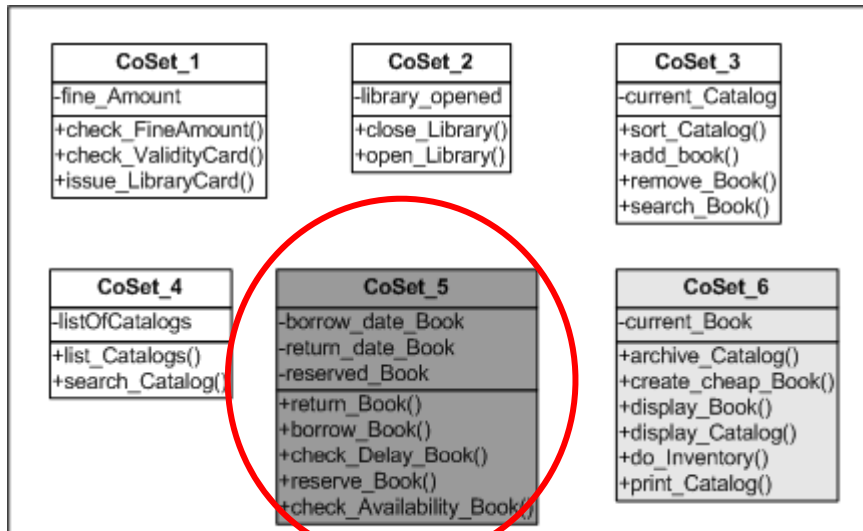
# Suggestion

## 2. RCA Algorithms

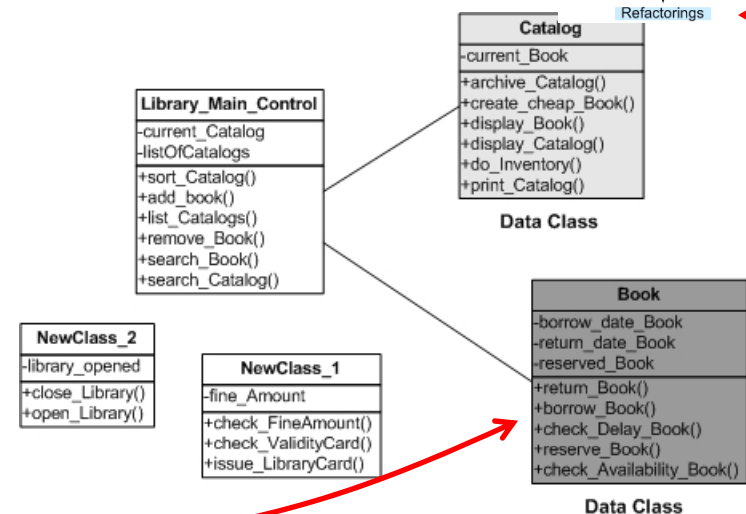


# Suggestion

## 3. Interpretation

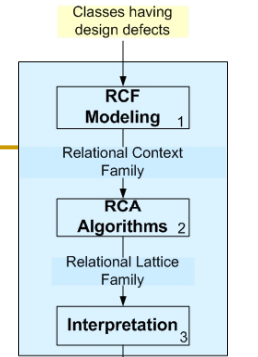


Cohesive Sets



After Refactoring

\* Move Method, Move Field, Create Class



# Experimental Study

---

## ■ Tooling

- **PADL** to model source code and generate contexts
- **Galicia** to construct and visualize the concept lattices

## ■ Experiments

- Goal: evaluate the relevance of the cohesive sets suggested
- 4 different **open-source** programs

# Experimental Study

System	Blob Class	LOC	Size (fields + methods)	Nb of fields/ methods moved	Nb of cohesive sets	Nb of real cohesive sets	Precision
Azureus V2.3.0.6	DHTTransportUDPImpl	2,049	(42+66) 108	(27+32) 59	10	7	70%
	DHTControlImpl	1,868	(47+80) 127	(35+62) 97	19	11	58%
	TRTrackerBTAnnouncerImpl	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%
	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%
	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%
	JobTracker	555	(22+31) 53	(17+18) 35	11	8	73%
<b>Average Precision:</b>							<b>57%</b>

# Conclusion

---

## ■ Contribution

*an approach that uses RCA to suggest refactorings to correct certain DDs, in particular Blob*

## ■ Validation

- 4 different programs
- **Relevant refactorings** to improve programs

## ■ Future Work

- Generalise to other DDs
- Assess more programs via our approach
- Discuss the suggested refactorings with their developers and apply them

# Questions

---

**Contact:** [mohanaou@iro.umontreal.ca](mailto:mohanaou@iro.umontreal.ca)

<http://www-etud.iro.umontreal.ca/~mohanaou>



***Thanks for your attention !***