

Fingerprinting Design Patterns

Yann-Gaël Guéhéneuc
Houari Sahraoui

{guehene,sahraouh}@iro.umontreal.ca

2004/11/10





Motivations

- Design patterns propose standard solutions (design motifs) to recurrent design problems
- Detecting partial/complete occurrences of design motifs (DM) helps
 - Understanding solved design problems
 - Changing/refactoring programs



Motivations

- DM define roles
- Occurrence of a DM = Combination of classes that matches the roles
- Detecting occurrences of DMs = Finding among all possible combinations of classes those that match the roles
 - Complex and time- and resource-consuming task



Example

- Detecting occurrences of Abstract Factory in a medium size program
 - JREFACTORY (575 classes)
 - 5 roles (AbstractFactory, ConcreteFactory, AbstractProduct, Product, and *Client*)
 - $\sim 575^5 = 6.28549\text{E}+13$ possible combinations



Idea

- Reduce the search space using heuristics
- Efficient heuristic
 - Quantitative characterization of DM roles
- Not really new
 - For example, [Antoniol *et al.*, 98]



Idea

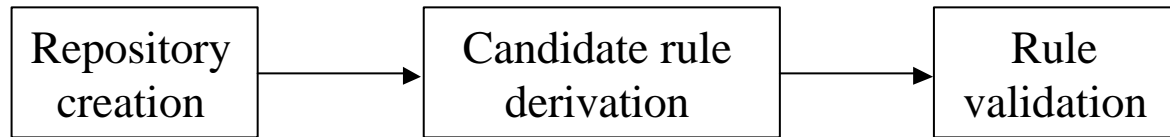
- Limitations of current approaches
 - Quantitative characterization extracted from design pattern description manually
 - Strong assumption: Implementation is conform to theory
- Solution
 - Automatic quantitative characterization from past-discovered occurrences



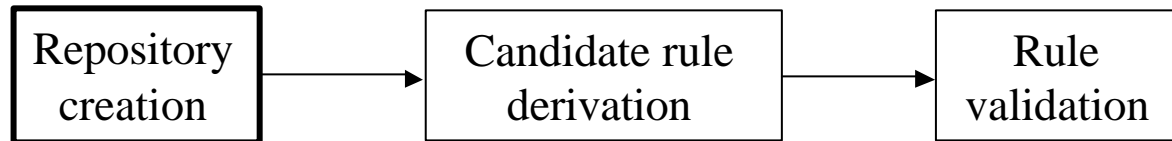
Fingerprinting Principle

- Use internal attributes of classes
 - Size, inheritance, coupling, cohesion
- Values of internal attributes are not unique
 - A class can play more than one role
 - Two or more classes can play a same role
- Attributes can be used to reduce the search space
 - Eliminating classes that do not play a particular role *obviously*

Building Fingerprints



Building Fingerprints



■ Repository creation

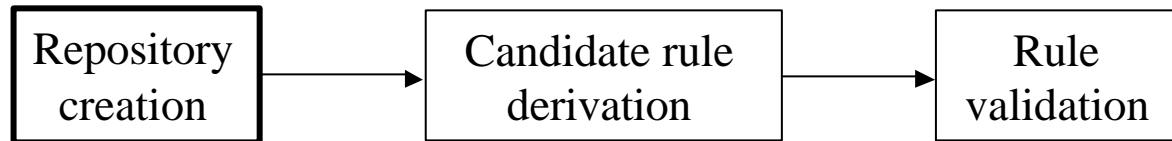
– Sources

- Studies in the literature (cf., [Bieman *et al.*, 03])
- Home-made tool for structural detection [Guéhéneuc, 01]
- Assignments in a graduate course

– Structure

- <program><designmotif><occurrence><role>

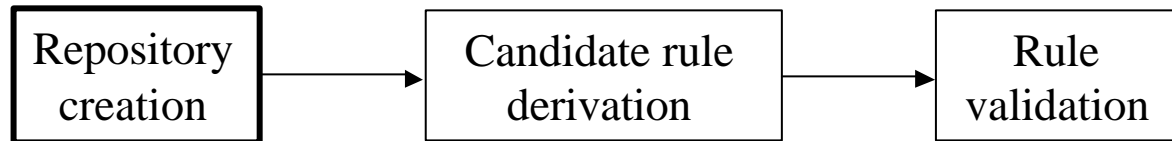
Building Fingerprints



■ Repository creation

	JHOTDRAW v5.1	JREFACTORY v2.6.24	JUNIT v3.7	LEXI v0.0.1a	NETBEANS v1.0.x	QUICKUML 2001	Total	Number of roles [Gamma, 1994]	Number of classes playing a role per design motif	
Number of classes	173	575	157	127	5812	224	7 068			
Design motif	Number of DM occurrences per program									
Abstract Factory					12	1	13	5	217	
Adapter	1	17			8		26	4	230	
Builder		2		1		1	4	4	24	
Command	1				1	1	3	5	67	
Composite	1		1			2	4	4	107	
Decorator	1		1				2	4	64	
Factory Method	3	1					4	4	67	
Iterator			1		5		6	5	30	
Observer	2		3	2		1	8	4	93	
Prototype	2						2	3	32	
Singleton	2	2	2	2		1	9	1	9	
State	2	2					4	3	32	
Strategy	4						4	3	36	
Template Method	2						2	2	36	
Visitor		2					2	4	138	
							Total	93	55	1 182

Building Fingerprints

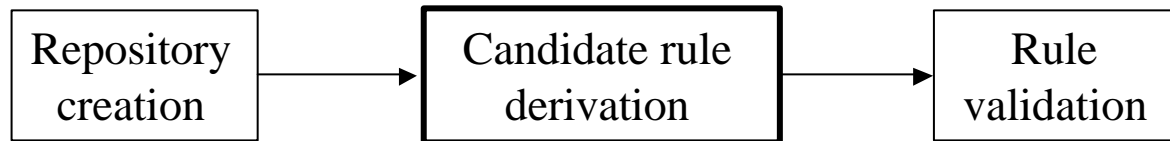


■ Repository creation

- Metric extraction for each class in the repository

	Acronyms	Descriptions	References
Size	NM	Number of methods	[Lorenz, 94]
	NMA	Number of new methods	[Lorenz, 94]
	NMI	Number of inherited methods	[Lorenz, 94]
	NMO	Number of overridden methods	[Lorenz, 94]
	WMC	Weighted methods per class	[Chidamber, 93]
Inheritance	CLD	Class to leaf depth	[Tegarden, 95]
	DIT	Depth of inheritance tree	[Chidamber, 93]
	NOC	Number of children	[Chidamber, 93]
Cohesion	C	Connectivity	[Hitz, 95]
	LCOM5	Lack of cohesion in methods	[Briand 97]
Coupling	ACMIC	Antecedent class-method import coupling	[Briand 97]
	CBO	Coupling between objects	[Chidamber, 93]
	DCMEC	Descendent class-method import coupling	[Briand 97]

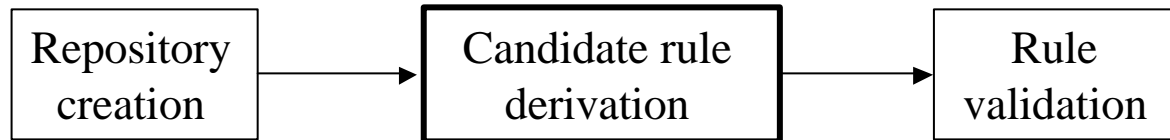
Building Fingerprints



■ Candidate rule derivation

- Rule learner, JRIP (from Weka tool)
- Learning dataset (for each role): $4 \cdot n$ cases
 - n classes playing the role
 - $3 \cdot n$ classes chosen randomly
- Each case
 - $\langle \text{metric}_1, \text{metric}_2, \dots, \text{metric}_m, \text{role} \rangle$

Building Fingerprints



■ Candidate rule derivation

– Rules shape

Rules for <Role>:

Fingerprint₁, confidence

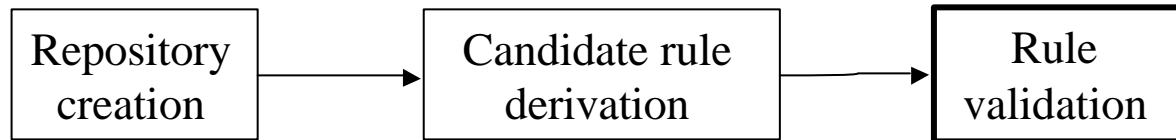
...

Fingerprint_k, confidence

where

Fingerprint₁: $\{metric_1 \hat{I} D_{11}; \dots ; metric_m \hat{I} D_{m1}\}$

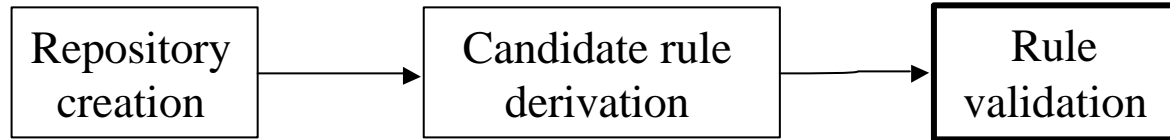
Building Fingerprints



■ Candidate rule validation

- Roles without a significant number of occurrences are eliminated (20 over 55)
- Cross-validation
 - Leave-one-out method
 - Roles with a recall less than 75% are eliminated (15 over 35)
 - For the 20 remaining roles, error < 10%

Building Fingerprints



■ Rule validation

Design motifs	Roles	Error (%)	Recall (%)
Iterator	Client	0,00	100,00
Observer	Subject	0,00	100,00
Observer	Observer	2,38	100,00
Template Method	Concrete Class	0,00	97,06
Prototype	Concrete Prototype	0,00	96,30
Decorator	Concrete Component	4,17	89,58
Visitor	Concrete Visitor	0,00	88,89
Strategy	Context	3,70	88,89
Visitor	Concrete Element	2,04	88,78
Singleton	Singleton	8,33	87,50
Factory Method	Concrete Creator	4,30	87,10
Factory Method	Concrete Product	3,45	86,21
Adapter	Target	4,00	84,00
Composite	Leaf	6,47	82,09
Decorator	Concrete Decorator	0,00	80,00
Iterator	Iterator	0,00	80,00
Command	Receiver	6,67	80,00
State	Concrete State	6,67	80,00
Strategy	Concrete Strategy	2,38	78,57
Command	Concrete Command	3,23	77,42

Using Fingerprints

- Structural detection with constraints
 - $\{V, C, D\}$
 - Variables represent roles in a DM
 - Domains contain all classes of a program
- Fingerprints reduce the domains
- Fingerprints reduce the number of possible combinations

Composite DM

■ Without fingerprints

- $S = \text{All classes}$
- Variables
 - client, $D_{cl} = S$
 - component, $D_{cn} = S$
 - composite, $D_{co} = S$
 - leaf, $D_{le} = S$

■ With fingerprints

- $S_{cl}, S_{cn}, S_{co}, \text{ and } S_{le} \subseteq S$
- Variables
 - client, $D_{cl} = S_{cl}$
 - component, $D_{cn} = S_{cn}$
 - composite, $D_{co} = S_{co}$
 - leaf, $D_{le} = S_{le}$

$S_{cl}, S_{cn}, S_{co}, \text{ and } S_{le} = \text{Classes matching fingerprints of the four roles}$

– Constraints

- association(client, component), inheritance(component, composite), inheritance(component, leaf), composition(composite, component)

Composite DM

■ Rules for “Leaf”

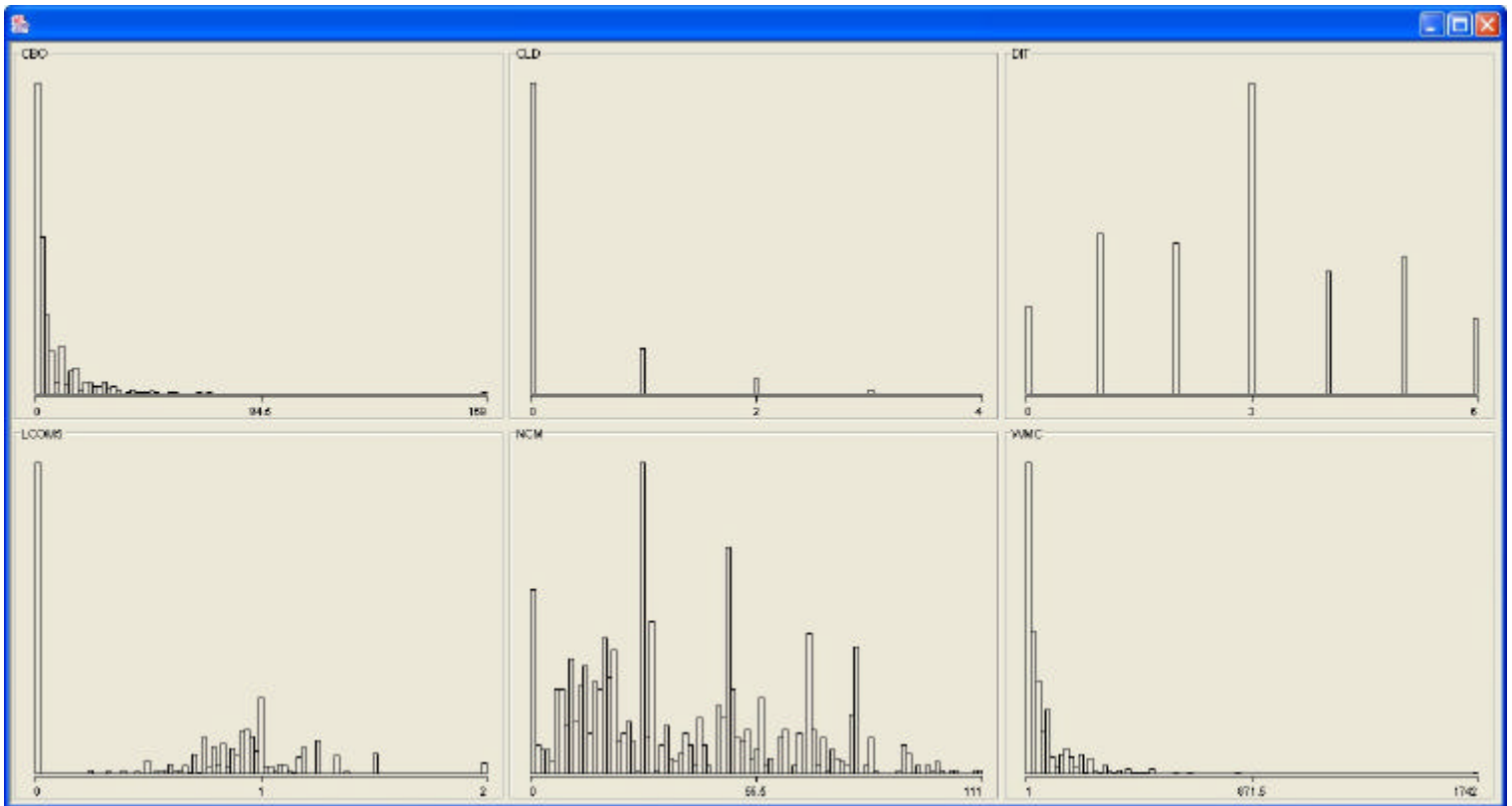
- $NMI = 26 \wedge DIT = 5, 23/67$
- $NMI = 25 \wedge NMO = 2, 45/67$
- $DIT = 3 \wedge NM = 12, 9/67$

■ Search space reduction

Fingerprints	Number of classes	Reduction (%)
$NMI = 26 \wedge DIT = 5$	20	69,00
$NIM = 25 \wedge NMO = 2$	7	89,15
$DIT = 3 \wedge NM = 12$	10	84,50

Side Effects

■ DM vs. Quality





Conclusion

- Experimental study for mining role fingerprints (quantitative characterization)
- Role fingerprints defined using metrics
- Role fingerprints used for DM occurrences detection
- Significant reduction in time and resources
- Applicable to partial occurrence detection



Future work (just a joke)



Future work (just a joke)

- Fingerprinting components



Future work (just a joke)

- Fingerprinting components
- Fingerprinting frameworks



Future work (just a joke)

- Fingerprinting components
- Fingerprinting frameworks
- Fingerprinting aspects



Future work (just a joke)

- Fingerprinting components
- Fingerprinting frameworks
- Fingerprinting aspects
- Fingerprinting refactorings



Future work (just a joke)

- Fingerprinting components
- Fingerprinting frameworks
- Fingerprinting aspects
- Fingerprinting refactorings
- Fingerprinting services



Future work (just a joke)

- Fingerprinting components
- Fingerprinting frameworks
- Fingerprinting aspects
- Fingerprinting refactorings
- Fingerprinting services
- Fingerprinting metrics



Future work (just a joke)

- Fingerprinting components
- Fingerprinting frameworks
- Fingerprinting aspects
- Fingerprinting refactorings
- Fingerprinting services
- Fingerprinting metrics
- ...



Data

- Accuracy depends on the repository
 - P-MARt (Pattern-like Micro-Architecture Repository) is a repository of micro-architectures similar to design motifs
 - P-MARt and its associated tools are freely available under the LPGL

Please contribute!

- Information, downloads, and contributions at <http://www2.iro.umontreal.ca/~labgelo/p-mart/>