Mining System Specific Rules from Change Patterns

WCRE 2013

André Hora, Nicolas Anquetil, Stéphane Ducasse, Marco Tulio Valente

Inria, Lille, France

15/10/2013

Static Analysis Tools

- Ensure source code quality
- Generic rules

Static Analysis Tools

- In general, warnings reported by such tools are **false positives**
- But, rules are not equal in identifying real warnings
  - Some rules perform better than others
  - **PMD rules 100% fixed in Apache Ant:** BrokenNullCheck, CloseResource, FinalizeShouldBeProtected, IdempotentOperations, MisplacedNullCheck, UnnecessaryConversionTemporary
- **How can we provide better rules to the developers?**

How can we provide better rules to the developers?

- 1. Create rules with the help of experts:
  - manually defined, expensive, lack of experts in legacy systems
- 2. Extract rules from code history:
  - comparing major releases, from bug-fixes

- However, software evolves over time, and naturally not just bugs are fixed

- Full code history can be investigated as source to produce better rules

How can we provide better rules to the developers?

- We propose to extract API rules from code history

- We focus on extracting data from (small) invocation changes between revisions: replacement to a better suited API, *e.g.*:
  - **PMD**: Hashtable → Map; StringBuffer → StringBuilder
  - **FindBugs**: Double.Double(arg) → Double.valueOf(arg)
  - **SmallLint**: Object.equals(nil) → Object.isNil()

- In this process:
  - Information is extracted from incremental revisions
  - Rules are mined from predefined patterns that ensure their quality

Mining Changes from History

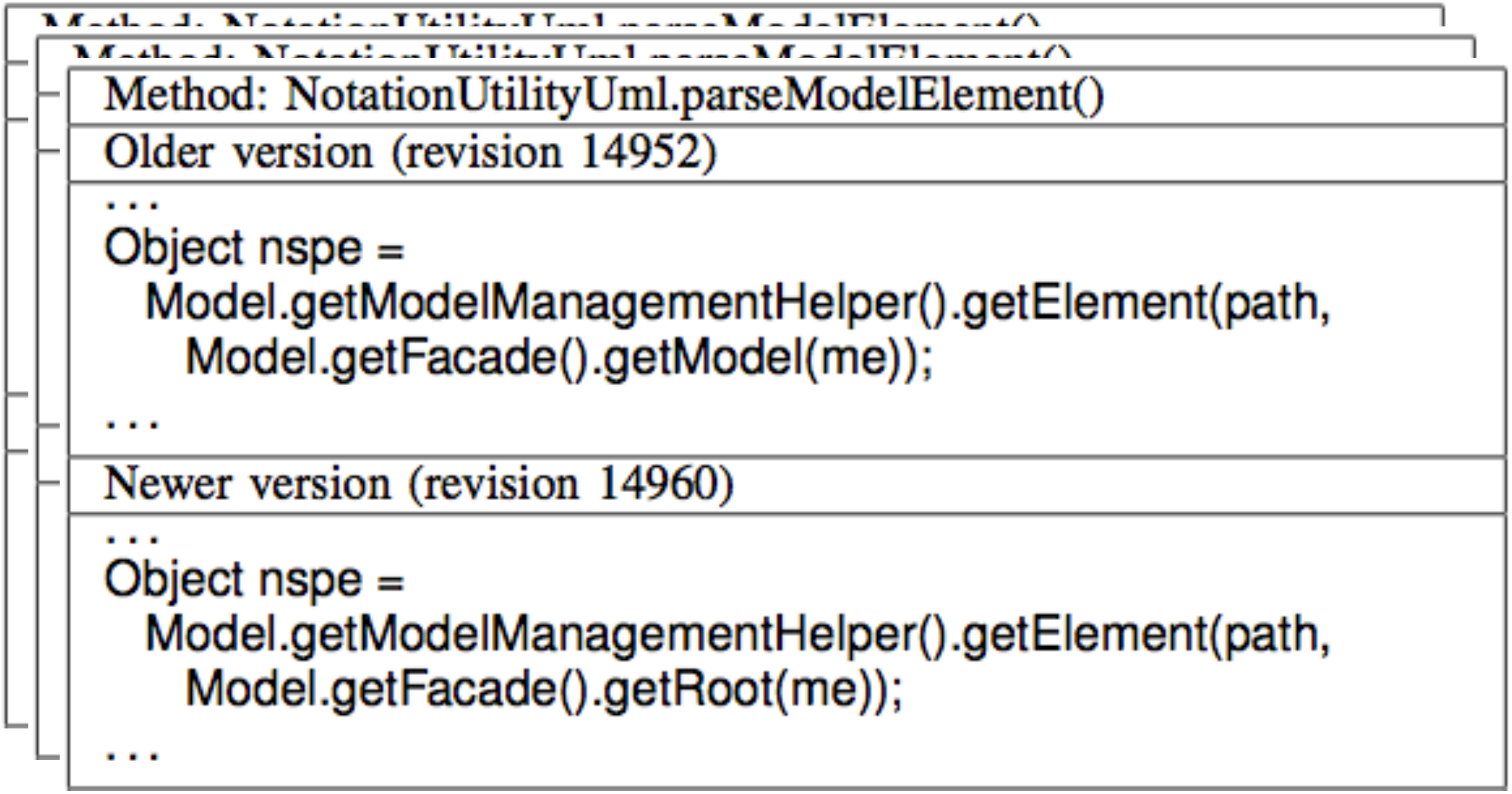
- 1. Extracting Changes from Revisions
- 2. Mining Change Patterns
- 3. Selecting Relevant Rules

Mining Changes from History

- **1. Extracting Changes from Revisions**
- 2. Mining Change Patterns
- 3. Selecting Relevant Rules

Extracting Changes from Revisions

Example: Convention to retrieve the Facade model in ArgoUML



Method: NotationUtilityUml.parseModelElement()

Older version (revision 14952)

```
. . .
Object nspe =
  Model.getModelManagementHelper().getElement(path,
    Model.getFacade().getModel(me));
. . .
```

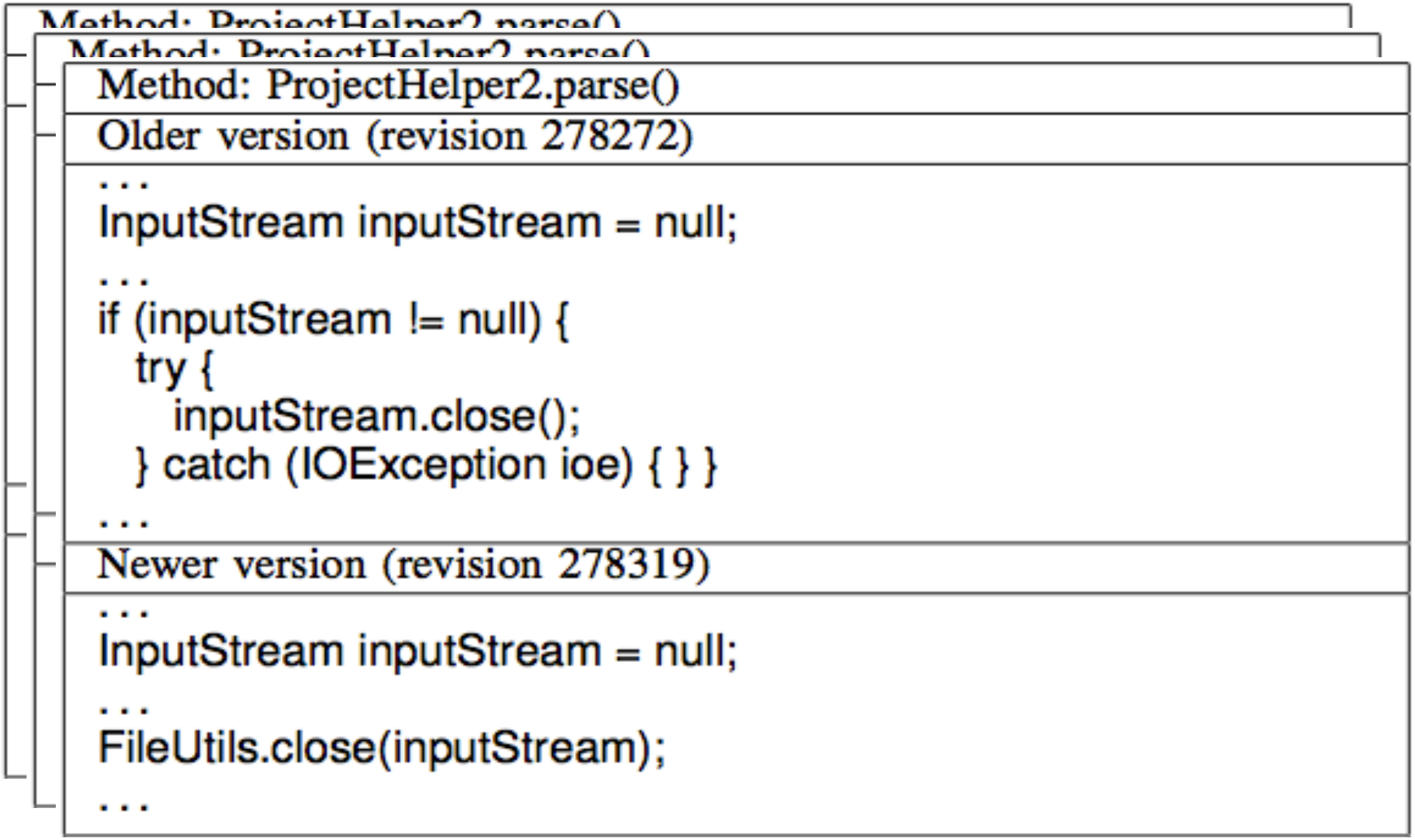
Newer version (revision 14960)

```
. . .
Object nspe =
  Model.getModelManagementHelper().getElement(path,
    Model.getFacade().getRoot(me));
. . .
```

Facade.getModel(arg) → Facade.getRoot(arg)

Extracting Changes from Revisions

Example: Convention to close files in Ant



Method: ProjectHelper2.parse()

Method: ProjectHelper2.parse()

Method: ProjectHelper2.parse()

Older version (revision 278272)

```
. . .
InputStream inputStream = null;
. . .
if (inputStream != null) {
   try {
      inputStream.close();
   } catch (IOException ioe) { } }
. . .
```

Newer version (revision 278319)

```
. . .
InputStream inputStream = null;
. . .
FileUtils.close(inputStream);
. . .
```

InputStream.close() → FileUtils.close(arg)

Extracting Changes from Revisions

- We represent the delta between two revisions of a method with predicates that describe added/deleted invocations:
  - *deleted-invoc(id, receiver, signature)*
  - *added-invoc(id, receiver, signature)*

- Examples:
  - deleted-invoc(1, Facade, getModel(arg))
  - added-invoc(1, Facade, getRoot(arg))

  - deleted-invoc(2, InputStream, close())
  - added-invoc(2, FileUtils, close(arg))

Mining Changes from History

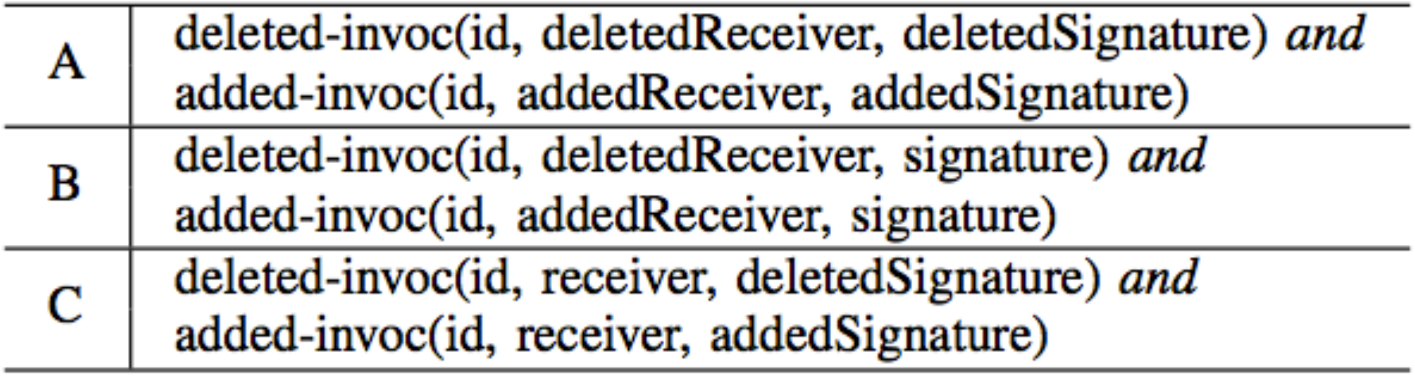
- 1. Extracting Changes from Revisions

- **2. Mining Change Patterns**

- 3. Selecting Relevant Rules

Mining Change Patterns

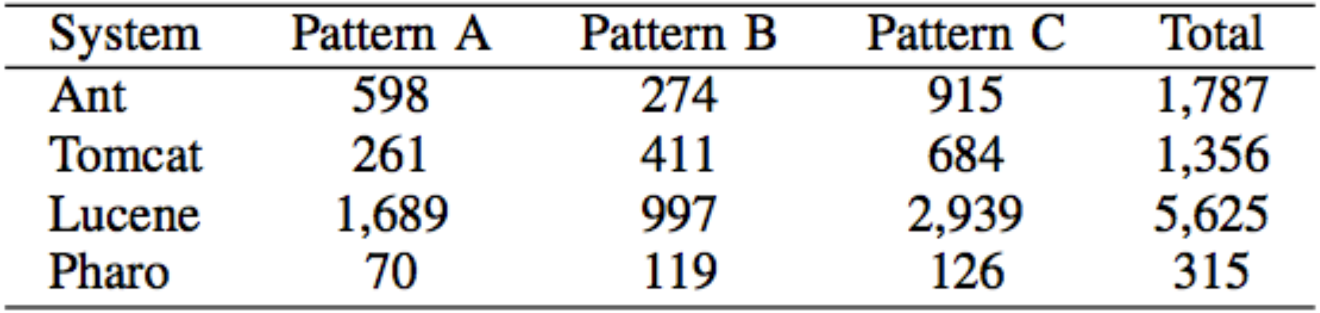
- A: deletedReceiver.deletedSignature → addedReceiver.addedSignature
- B: deletedReceiver.**signature** → deletedReceiver.**signature**
- C: **receiver**. deletedSignature → **receiver**. addedSignature



A	deleted-invoc(id, deletedReceiver, deletedSignature) *and* added-invoc(id, addedReceiver, addedSignature)
B	deleted-invoc(id, deletedReceiver, signature) *and* added-invoc(id, addedReceiver, signature)
C	deleted-invoc(id, receiver, deletedSignature) *and* added-invoc(id, receiver, addedSignature)

Mining Change Patterns

- A: deletedReceiver.deletedSignature → addedReceiver.addedSignature
- B: deletedReceiver.**signature** → deletedReceiver.**signature**
- C: **receiver**. deletedSignature → **receiver**. addedSignature



System	Pattern A	Pattern B	Pattern C	Total
Ant	598	274	915	1,787
Tomcat	261	411	684	1,356
Lucene	1,689	997	2,939	5,625
Pharo	70	119	126	315

Mining Changes from History

- 1. Extracting Changes from Revisions

- 2. Mining Change Patterns

- **3. Selecting Relevant Rules**

Selecting Relevant Rules

- Frequency over time in **different** revisions
  - A rule that occurs in two different revisions is more relevant than another that occurs many times in just one revision
  - They are in fact being incrementally fixed by developers

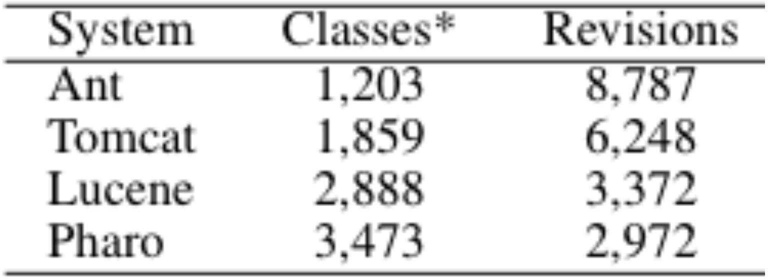
Research Questions

- **How can we provide better rules to the developers?**

- **RQ1**: Are specific warnings more likely to point to real violations than generic warnings?
- **RQ2**: Are specific rules more likely to point to real violations than generic rules?

- **RQ3**: Are best specific warnings more likely to point to real violations than best generic warnings?
- **RQ4**: Are best specific rules more likely to point to real violations than best generic rules?

Experiment Setting

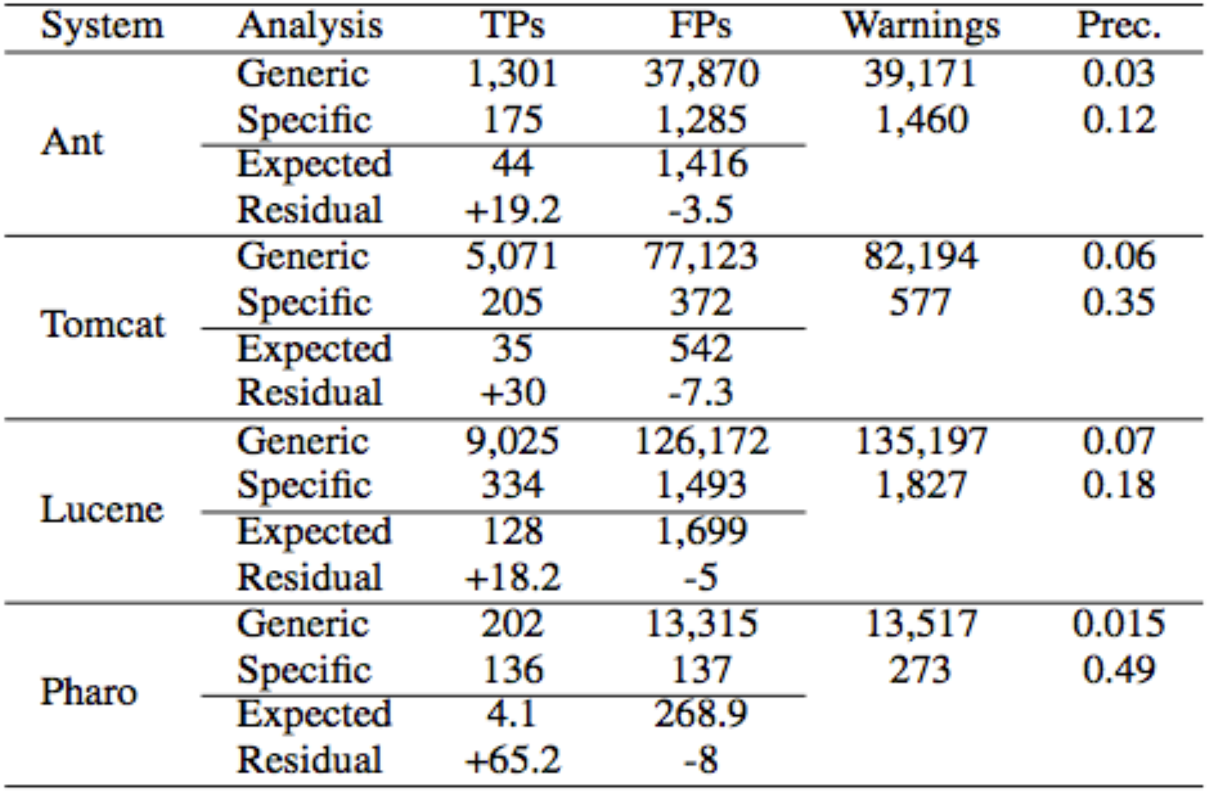


System	Classes*	Revisions
Ant	1,203	8,787
Tomcat	1,859	6,248
Lucene	2,888	3,372
Pharo	3,473	2,972

- Systems
  - Java: Ant, Tomcat , Lucene
  - Smalltalk: Pharo
- Generic rules
  - Java: PMD, 180 rules
  - Smalltalk: SmallLint, 85 rules
  - TP: warning is removed from source code
  - FP: warning remains in source code
- Specific rules
  - We **learn** a rule when it occurs *f* times in **different** revisions (*f* = 2)
  - We **evaluate** at revision *n* the rules learned from revisions 1 to *n* − 1
  - TP: fix at revision *n* matches a rule
  - FP: fix at revision *n* matches the deleted invocation of a rule, but not the added

Experiment Results

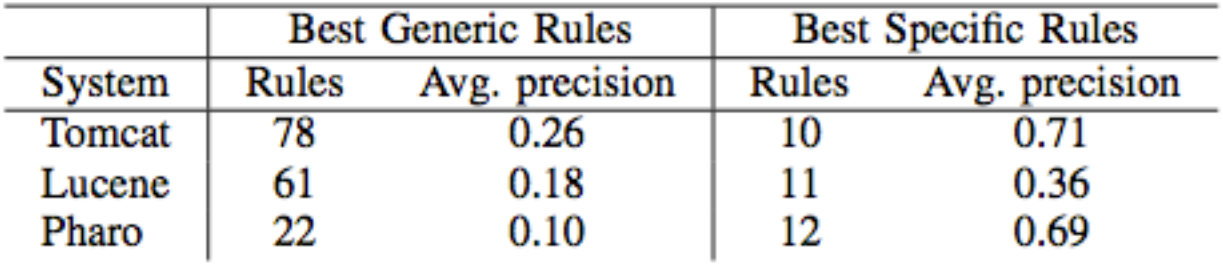
- **RQ1**: Are specific warnings more likely to point to real violations than generic warnings?



System	Analysis	TPs	FPs	Warnings	Prec.	
Ant	Generic	1,301	37,870	39,171	0.03	✔
	Specific	175	1,285	1,460	0.12	
	Expected	44	1,416			
	Residual	+19.2	-3.5			
Tomcat	Generic	5,071	77,123	82,194	0.06	✔
	Specific	205	372	577	0.35	
	Expected	35	542			
	Residual	+30	-7.3			
Lucene	Generic	9,025	126,172	135,197	0.07	✔
	Specific	334	1,493	1,827	0.18	
	Expected	128	1,699			
	Residual	+18.2	-5			
Pharo	Generic	202	13,315	13,517	0.015	✔
	Specific	136	137	273	0.49	
	Expected	4.1	268.9			
	Residual	+65.2	-8			

Experiment Results

- **RQ4**: Are best specific rules more likely to point to real violations than best generic rules?



	Best Generic Rules		Best Specific Rules		
System	Rules	Avg. precision	Rules	Avg. precision	
Tomcat	78	0.26	10	0.71	✓
Lucene	61	0.18	11	0.36	✗
Pharo	22	0.10	12	0.69	✓

Discussion

- Specific warnings are more likely to point to real violations in source code than generic ones (**RQ1**)
- When comparing rules individually it depends of the system (**RQ2**)
- Best specific warnings are the more effective to point to real violation than best generic ones (**RQ3**)
- When grouping warnings by rules, we are able to detect specific rules as good as or even better than the best generic rules (**RQ4**)

Concrete Cases: Java

- **Ant**: convention to close files, convention added in Aug 2004, fixes in 2004, 2007 and 2010 (6 years later), 100 warnings, 37 fixes

- **Tomcat**: invokes inefficient Number constructor, several fixes but last revision still contains warnings, 59 warnings, 35 fixes

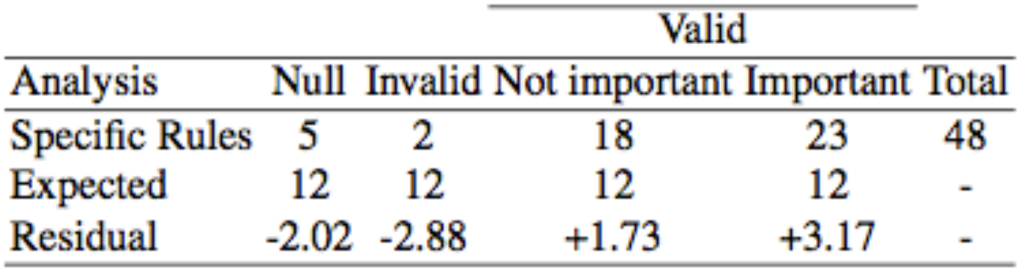
- **Lucene**: internal conventions to have better performance:
  - Analyzer.tokenStream() → Analyzer.reusableTokenStream()
  - Random.nextInt() → SmartRandom.nextInt()

- Java API migration: Vector to ArrayList, Hashtable to Map, and StringBuffer to StringBuilder

Concrete Cases: Pharo

- Migration rules:
  - FileDirectory.default() → FileSystem.workingDirectory()
  - OSPlatform.osVersion() → OSPlatform.version()
  - …

- Are specific rules likely to be classified as valid ones by experts?



			Valid		
Analysis	Null	Invalid	Not important	Important	Total
Specific Rules	5	2	18	23	48
Expected	12	12	12	12	-
Residual	-2.02	-2.88	+1.73	+3.17	-

✔

Future Work

- New rule patterns
  - Collection.findStringStartingAt(*,1) > 0 → Collection.includesSubstring(*)

- On demand rules
  - Generate rules based on provided "evidences"
  - No predefined patterns (data-mining):
    - UserManager.default().currentUser() → Smalltalk.tools().userManager()
    - Character.cr() → ROPlatform.current().newLine()

Mining System Specific Rules from Change Patterns

WCRE 2013

André Hora, Nicolas Anquetil, Stéphane Ducasse, Marco Tulio Valente

Inria, Lille, France

15/10/2013