Mining Source Code^3

Mining Idioms, Usages and Edits

Dario Di Nucci

Research Fellow
dario.di.nucci@vub.be

VRIJE UNIVERSITEIT BRUSSEL

Software Languages Lab

TU Delft

VUB

Mining Software Repositories

Software Repositories?

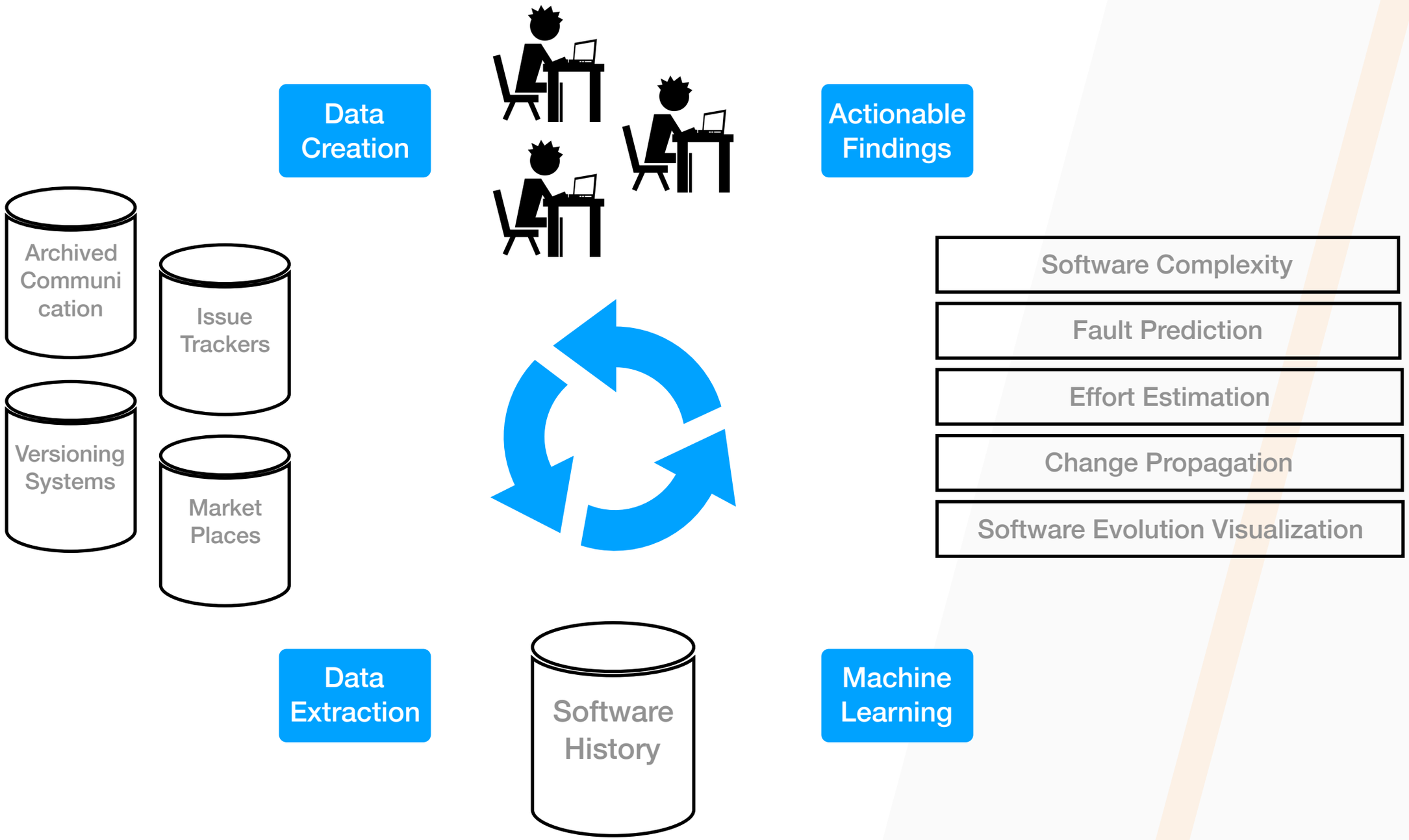
Versioning Systems

Issue Trackers

Archived Communication

Market Places

Why Software Repositories?



Intelligent Modernisation Assistance for Legacy Software

VRIJE UNIVERSITEIT BRUSSEL

raincode LABS
compiler experts

UCL Université catholique de Louvain

teamUP
ACADEMIC RESEARCH & INDUSTRY

innoviris .brussels
empowering research

@intimals_proj

soft.vub.ac.be/intimals

INTiMALS

Problem & context

raincode LABS

compiler experts

migration & maintenance services

modernise legacy software

Increasing demand for such services

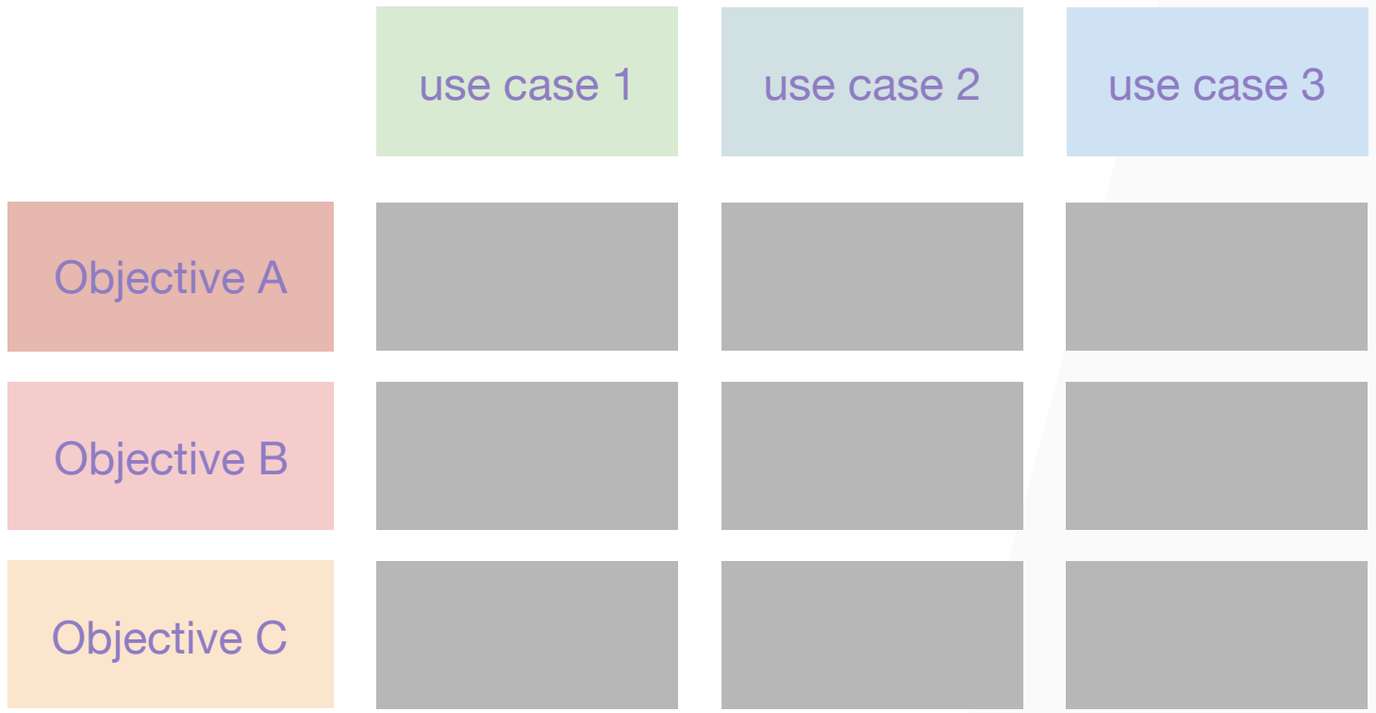
Requires significant **manual work**

Goal

Towards an intelligent modernisation assistant for legacy software

Key ideas:

- **Automating** migration requires **pattern discovery**
- **Mining kinds of patterns in 3 use cases for 3 objectives**



Uses cases

use case 1

Coding idioms and programming **conventions**

Code Syntax

use case 2

Library usage protocols and **violations**

Code Semantic

use case 3

Systematic edits or **repetitive changes**

Code Evolution

Objectives

Objective A
Program comprehension

Provide insights in legacy code

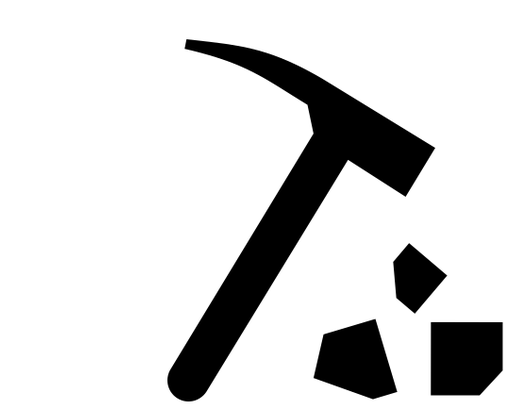
Objective B
Anomaly detection

Detect potential inconsistencies in legacy code

Objective C
Modernisation assistance

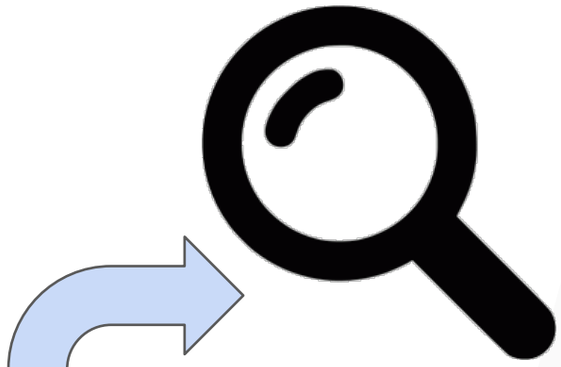
Provide recommendations to engineers for improving legacy code

Approach



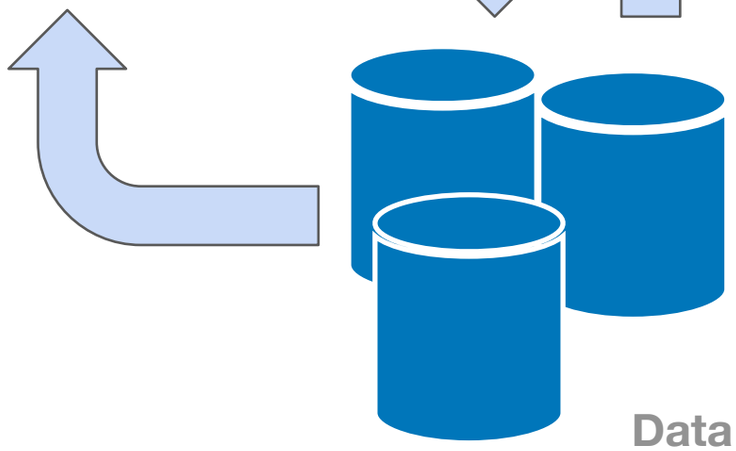
Pattern mining algorithms

previously unknown software patterns

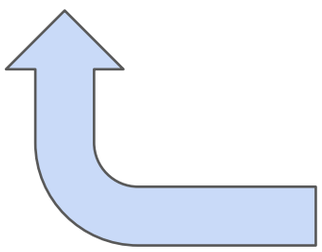


Modernisation assistant

browser for patterns and instances
on-demand partial pattern matching
pro-active recommendations



Data + MetaModel



Code Importers

- **open source software**
- **legacy software**

Mining Code Idioms

Context: Code Idioms

A syntactic fragment that recurs across software projects and serves a single semantic purpose.

M. Allamanis and C. Sutton “Mining idioms from source code" in 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 472-483.

Context: Code Idioms

A syntactic fragment that recurs across software projects and serves a single semantic purpose.

```
…
if (c != null) {
 try {
  if (c.moveToFirst()) {
   number = c.getString(
            c.getColumnIndex(
            phoneColumn));
  }
 } finally {
  c.close();
 }
}
…
```

```
…
try {
 if (c2.moveToFirst()) {
  number = c2.getString(
           c2.getColumnIndex(
           mobilePhoneColumn));
 }
} finally {
 c2.close();
}
…
```

```
…
try {
 if (newCursor.moveToFirst()) {
  number = “-1”
 }
} finally {
 newCursor.close();
}
…
```

M. Allamanis and C. Sutton “Mining idioms from source code" in 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 472-483.

Context: Code Idioms

A syntactic fragment that recurs across software projects and serves a single semantic purpose.

```
…
if (c != null) {
 try {
  if (c.moveToFirst()) {
   number = c.getString(
            c.getColumnIndex(
            phoneColumn));
  }
 } finally {
  c.close();
 }
}
…
```

```
…
try {
 if (c2.moveToFirst()) {
  number = c2.getString(
           c2.getColumnIndex(
           mobilePhoneColumn));
 }
} finally {
 c2.close();
}
…
```

```
…
try {
 if (newCursor.moveToFirst()) {
  number = “-1”
 }
} finally {
 newCursor.close();
}
…
```

```
try {
  if ($(Cursor).moveToFirst()) {
      $BODY$
    }
  } finally {
    $(Cursor).close();
  }
}
```

M. Allamanis and C. Sutton “Mining idioms from source code" in 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 472-483.

Mining for Code Idioms

Recognise code idioms manually can be tedious and error-prone!

Applying frequent itemset algorithms could lead to “boring” idioms.

We are implementing a language-parametric framework to:

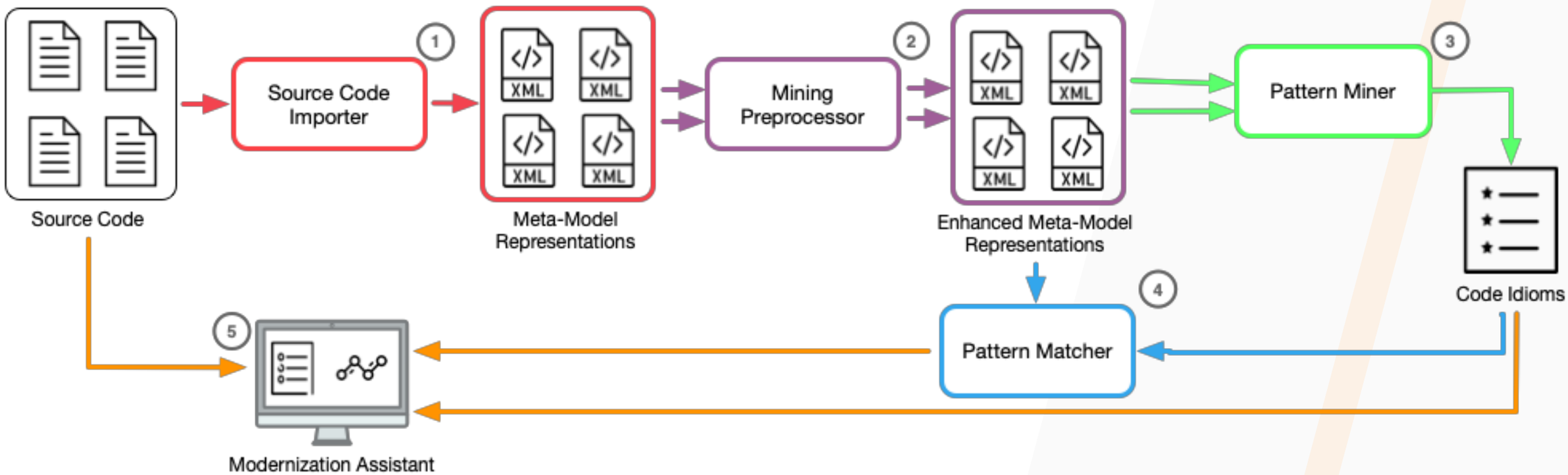
- Explore novel pattern mining algorithms for source code
- Incorporate them in an intelligent software modernisation assistant tool set

Applications:

- Discover syntactic patterns
- Discover code deviating from expected patterns
- Propose actions to improve with respect of idioms

Overview

FREQuent Tree mining algorithm

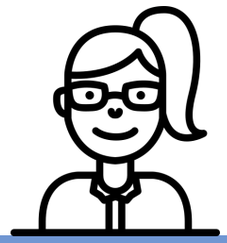


Limitations and Possible Solutions

- Highly time consuming
- Generates a large amount of patterns as well as redundant patterns
- Some patterns are more related to the grammar of the language than to the coding style

Heuristics and constraints could help to reduce the search space!

Setting constraints is not straightforward!



How to evaluate interesting patterns?

Summary

- We developed a language-parametric framework to mine code idioms.
- Currently based on **FREQ**uent **T**ree miner.
- Work in progress:
  - Reducing the search space by applying heuristics and constraints
  - Understanding idioms to improve the mining process

@dardin88

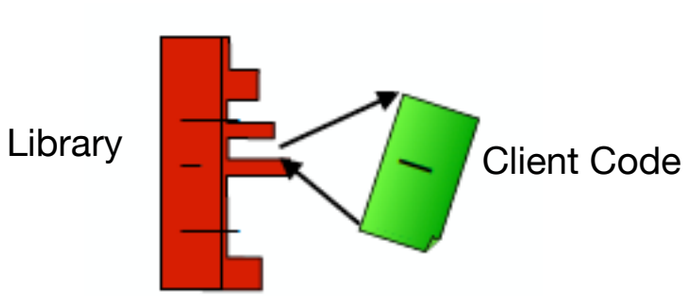
dario.di.nucci@vub.be

Mining Usages

Context: Library Usages

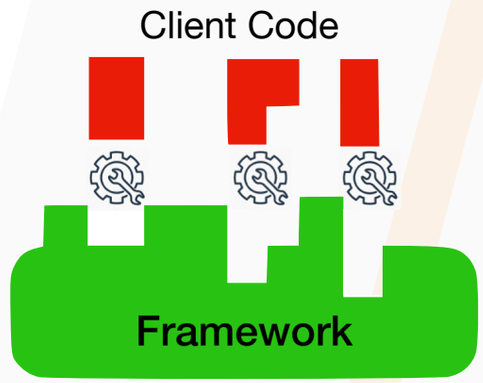
Enable code reuse
Provide high-level abstractions for common tasks

How to use a library?



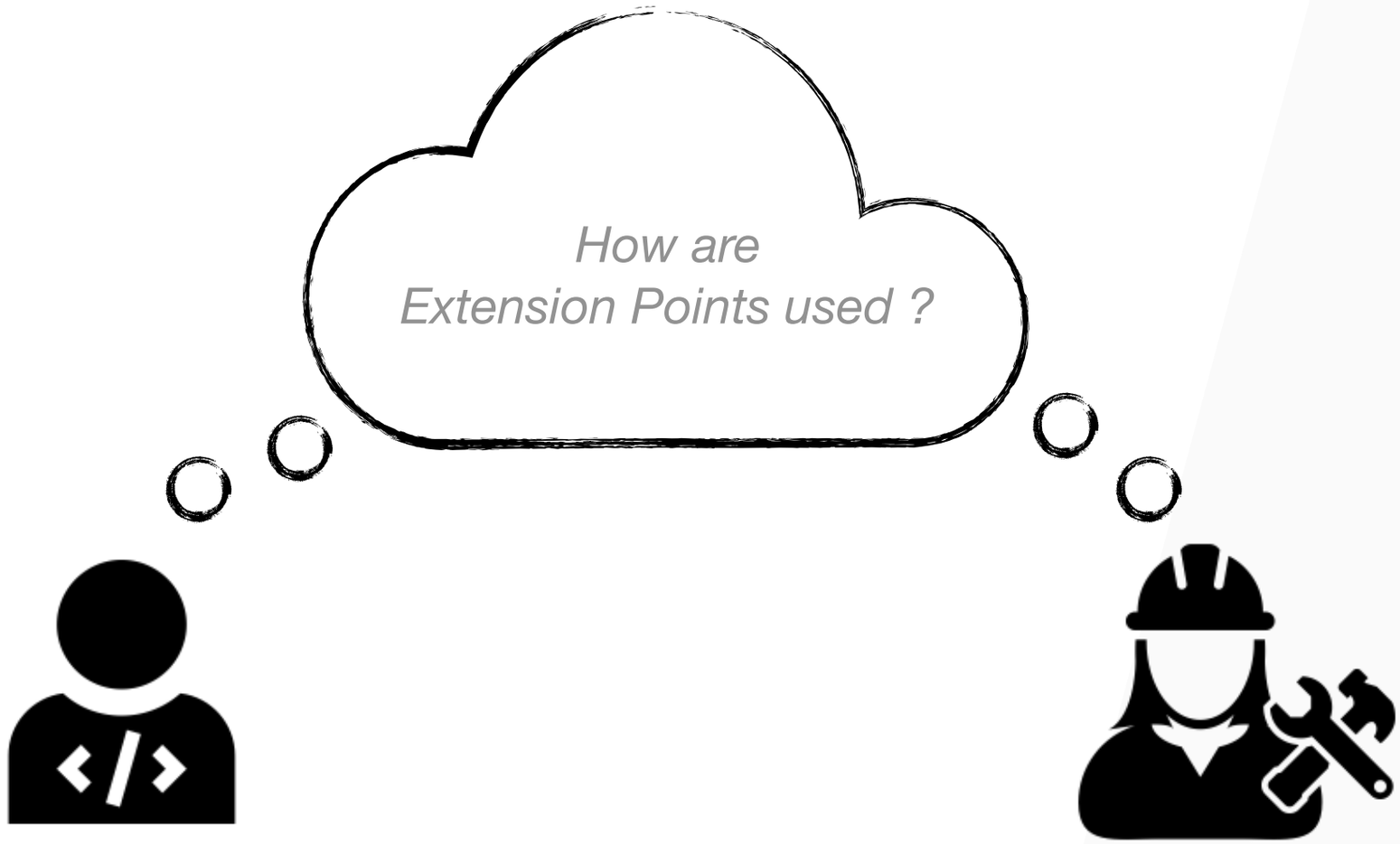
Only the functionalities it provides can be used.

How to use a framework?



Necessary or common to extend or customise its functionality.

Application Code
Third-party Code



What extension point should be used?
How are extension points usually used?

What kinds of extension points are used infrequently?
Which extension points are more error-prone or complex to use?

Extension Patterns

M. Asaduzzaman, C. K. Roy, K. A. Schneider, and D. Hou, “Recommending framework extension examples” in IEEE International Conference on Software Maintenance and Evolution, 2017, pp. 456–466.

Extending a Framework

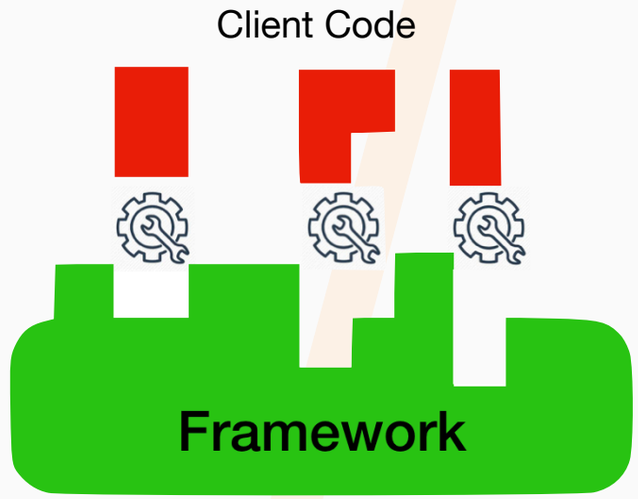
Extension Point

```
package org.apache.spark

class SparkContext(config: SparkConf) extends Logging {
  //…
  def addSparkListener(listener: SparkListenerInterface):{
    //…
  }
  //…
}
```

Extension Point Usage

```
import org.apache.spark._
//…
val listener = new SaveInfoListener
val sc = new SparkContext(“local”, “test”)
sc.addSparkListener(listener)
```



Simple Extension Point Usage

```
package org.apache.spark

class SparkContext(config: SparkConf) extends Logging {
  //…
  def addSparkListener(listener: SparkListenerInterface):{
    //…
  }
  //…
  private class SaveInfoListener extends SparkListener {
    //…
  }
  //…
}
```

Extension Point

```
import org.apache.spark._
//…
val listener = new SaveInfoListener
val sc = new SparkContext(“local”, “test”)
sc.addSparkListener(listener)
```

Extension Point Usage

Customise Extension Point Usage

```
package org.apache.spark

class SparkContext(config: SparkConf) extends Logging {
  //…
  def addSparkListener(listener: SparkListenerInterface):{
    //…
  }
  //…
private class SaveInfoListener extends SparkListener {
  //…
  def awaitNextJobCompletion(): Unit = {
    //…
  }
  //…
}
```

Extension Point

```
import org.apache.spark._
//…
val listener = new SaveInfoListener
listener.awaitNextJobCompletion()
val sc = new SparkContext(“local”, “test”)
sc.addSparkListener(listener)
```

Extension Point Usage

Extend Extension Point Usage

```
package org.apache.spark

class SparkContext(config: SparkConf) extends Logging {
  //…
  def addSparkListener(listener: SparkListenerInterface):{
    //…
  }
  //…
}
```

Extension Point

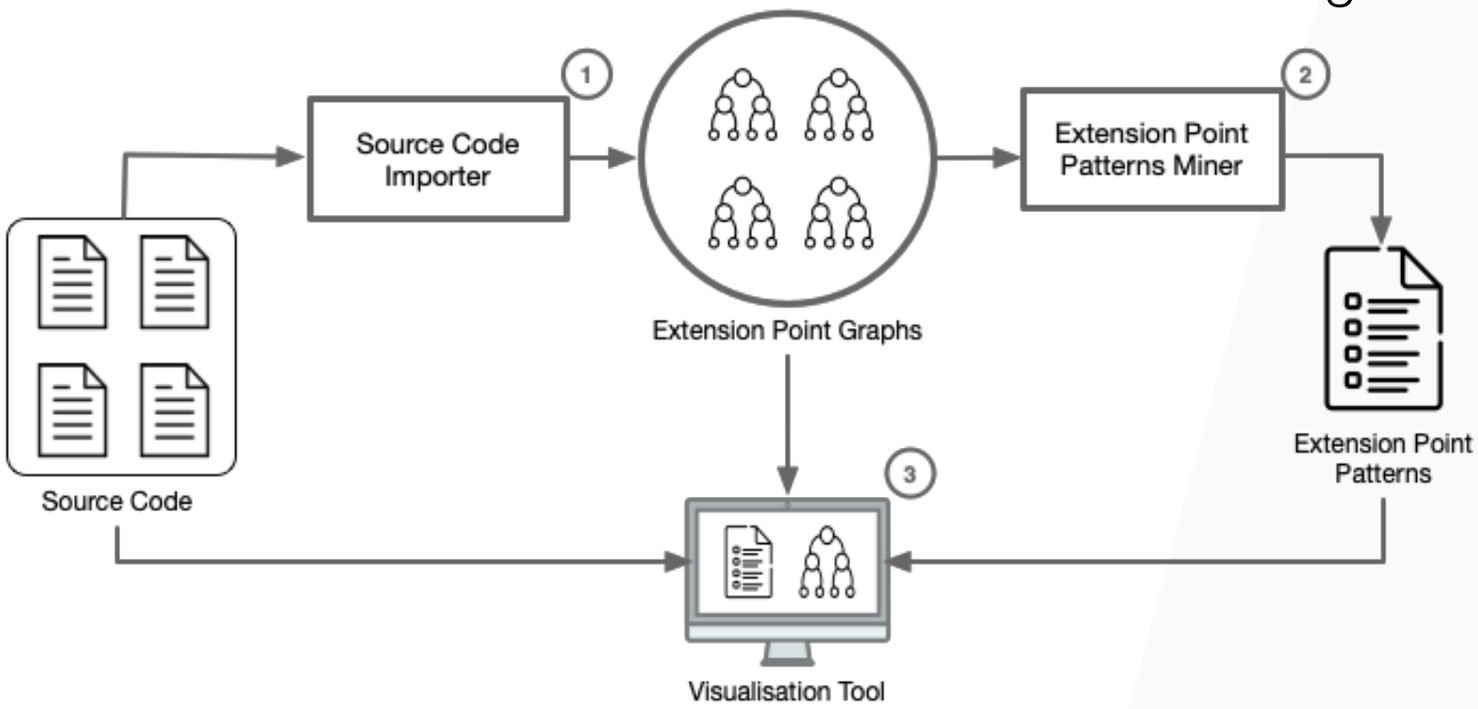
```
import org.apache.spark._
//…
class StageInfoRecorderListener extends SparkListener {
  override def onJobStart(jobStart: SparkListenerJobStart): Unit = {
    //…
  }
  override def onStageCompleted(stageCompleted: SparkListenerStageCompleted): Unit = {
    //…
  }
  //…
}
//…
```

```
import org.apache.spark._
case class StageMetrics(sparkSession: SparkSession) {
  sparkSession.sparkContext.addSparkListener(new StageInfoRecorderListener)
  //…
}
```

Extension Point Usage

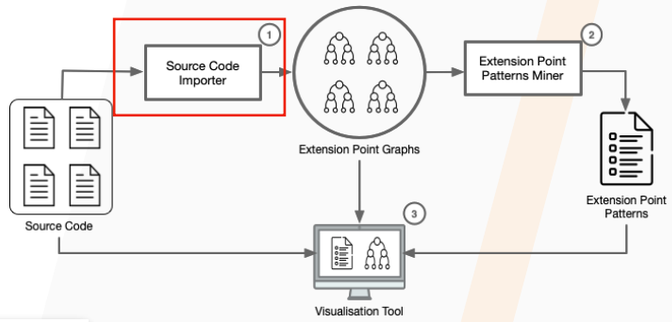
Overview of Scala-XP-Miner

APriori algorithm

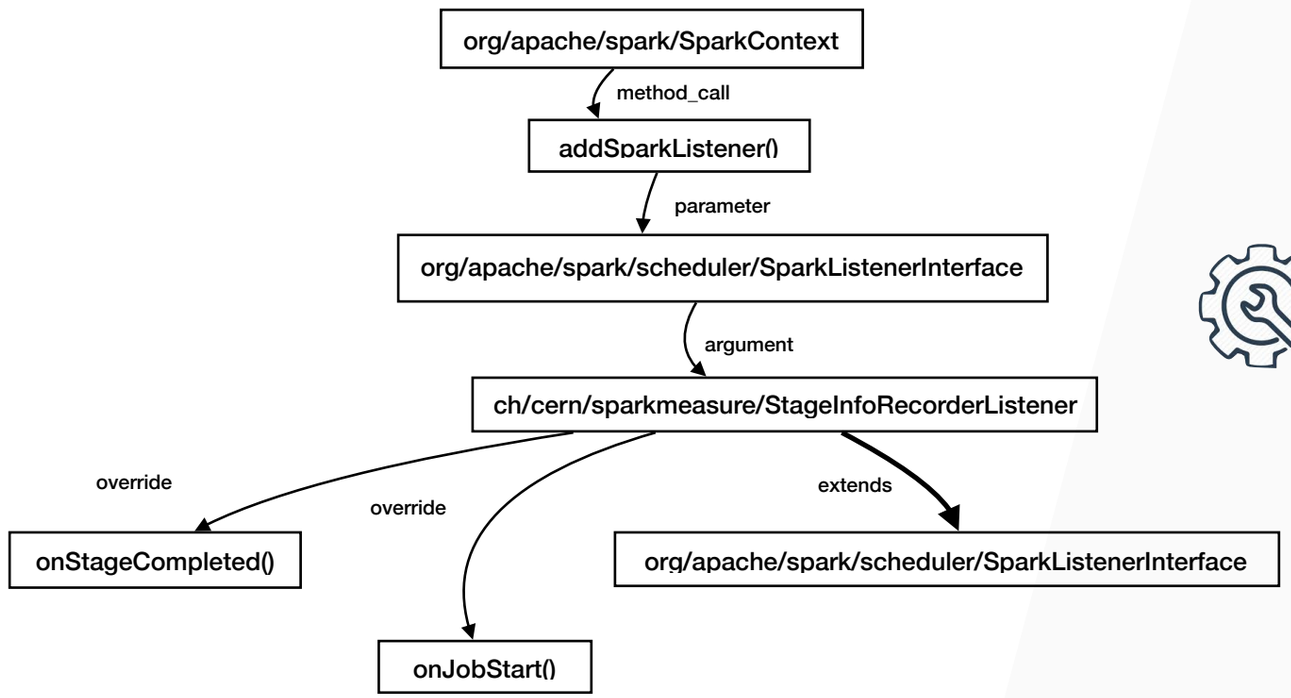


Y. Pacheco, J. De Bleser, T. Molderez, D. Di Nucci, W. De Meuter, and C. De Roover “Mining Scala Framework Extensions for Recommendation Patterns” in IEEE SANER, 2019, to be presented.

Scala-XP-Miner: Importer




```
import org.apache.spark._
case class StageMetrics(sparkSession: SparkSession) {
  sparkSession.sparkContext.addSparkListener(new StageInfoRecorderListener)
  //…
}
```



Extension Graph

RECEIVER TYPE

METHOD CALL

PARAMETER TYPE

ARGUMENT TYPE

OVERRIDING METHOD

IMPLEMENTED INTERFACE

EXTENDED

OTHER METHOD CALLS

FRAMEWORK METHOD CALL.

```scala
package org.apache.spark

class SparkContext(config: SparkConf) extends Logging {
  def addSparkListener(listener: SparkListenerInterface):{
    //…
  }
}
```

```scala
import org.apache.spark._
class StageInfoRecorderListener extends SparkListener {
  override def onJobStart(jobStart: SparkListenerJobStart): Unit = {
    //…
  }
  override def onStageCompleted(stageCompleted: SparkListenerStageCompleted): Unit = {
    //…
  }
}
```

```scala
import org.apache.spark._
case class StageMetrics(sparkSession: SparkSession) {
  sparkSession.sparkContext.addSparkListener(new StageInfoRecorderListener)
}
```

RECEIVER TYPE

METHOD CALL

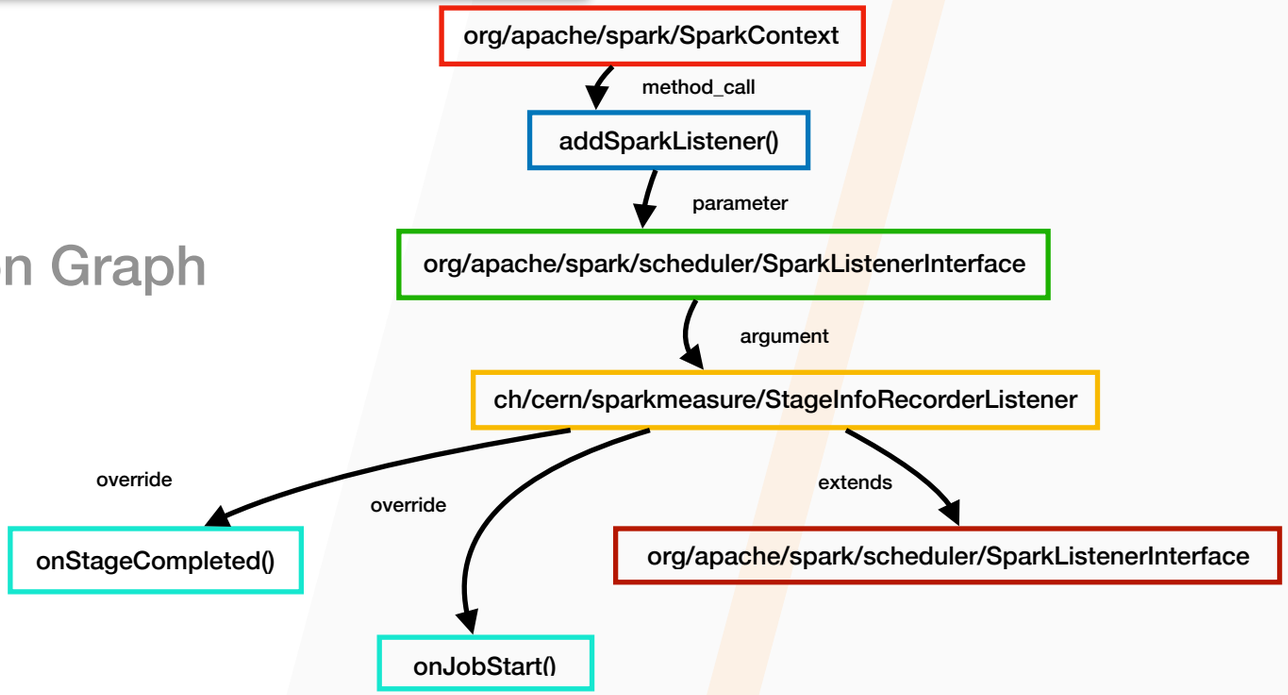
PARAMETER TYPE

ARGUMENT TYPE

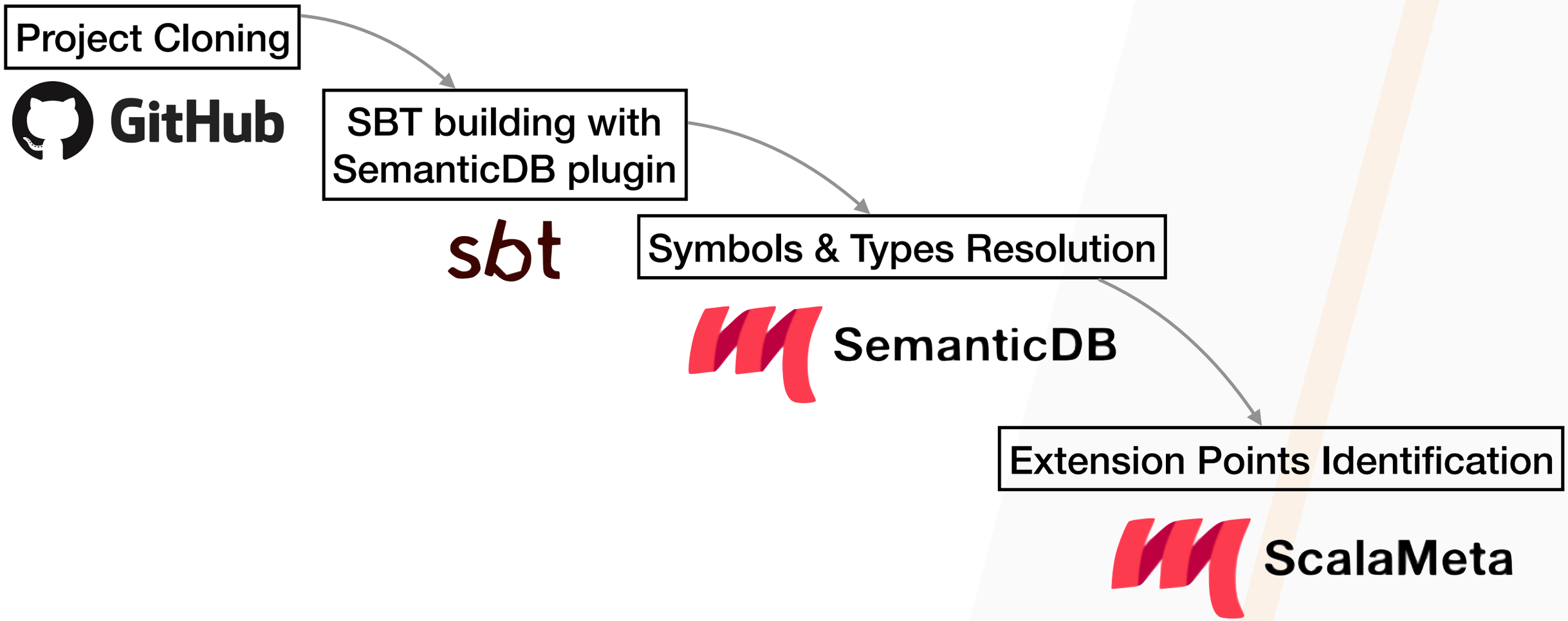
OVERRIDING METHOD

EXTENDED

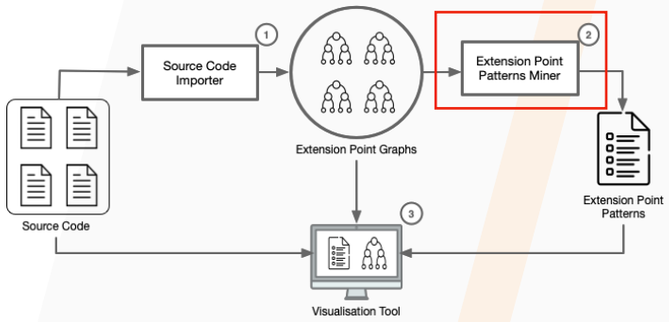
Extension Graph

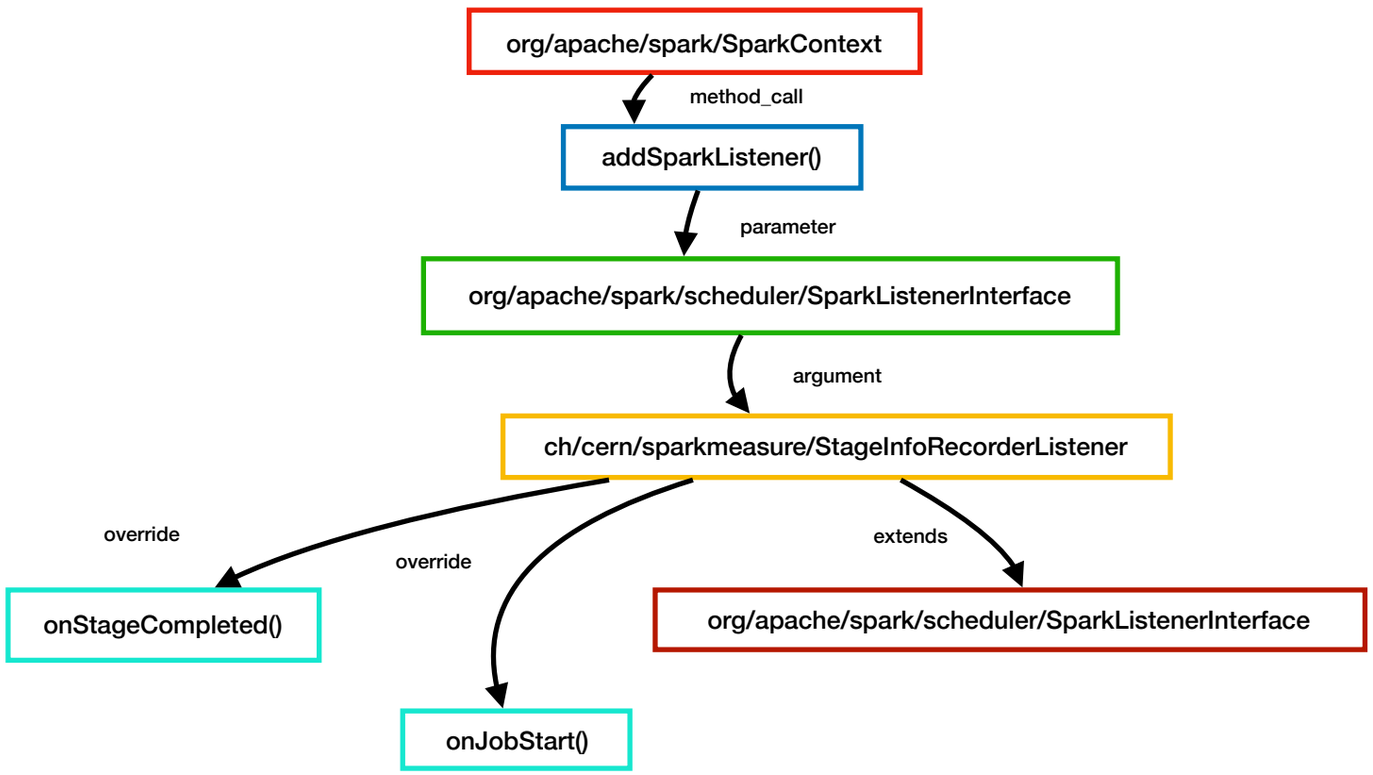


Scala-XP-Miner: Data Extraction



Scala-XP-Miner: Miner





Extension Graph

org/apache/spark/SparkContext

method_call addSparkListener

parameter org/apache/spark/scheduler/SparkListenerInterface

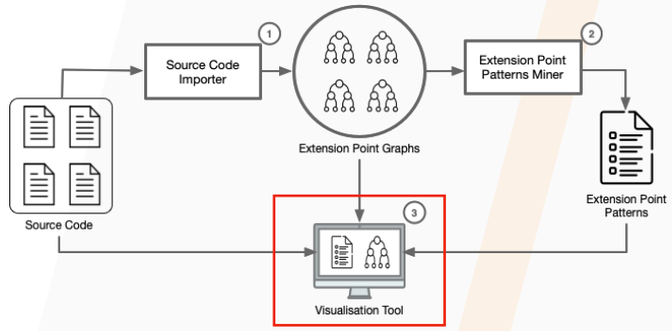
argument Client

extends org/apache/spark/scheduler/SparkListenerInterface

override onJobStart

Extension Pattern

Scala-XP-Miner: Visualization



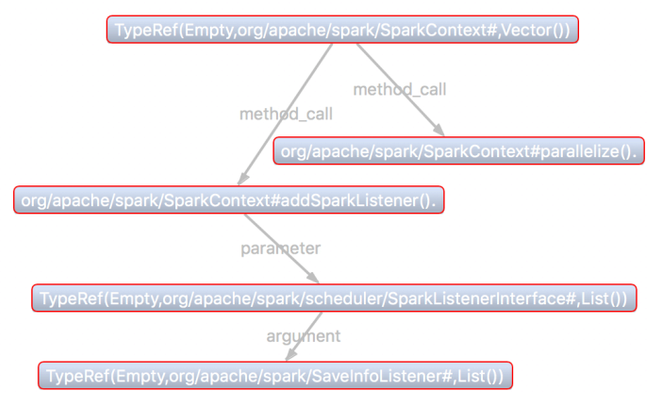
Input

Collect Scala | Collect Single API | Collect All API | Load Collected Input | Save Collected Input | Evaluation

Usages List. Total: 16826

Receiver	Method Call	Parameters
TypeRef(Empty,org/apache/spark/sql/cat	org/apache/spark/sql/catalyst/plans/PlanTest#comparePlans().	TypeRef(Empty,org/apac
TypeRef(Empty,org/apache/spark/sql/cat	org/apache/spark/sql/catalyst/optimizer/EliminateSubqueryAliasesSuite#afterOptimization().	nullTypeRef(Empty,org/a
TypeRef(Empty,org/apache/spark/sql/cat	org/apache/spark/sql/catalyst/plans/PlanTest#comparePlans().	nullTypeRef(Empty,org/a TypeRef(Empty,org/apac
TypeRef(Empty,org/apache/spark/sql/cat	org/apache/spark/sql/catalyst/optimizer/EliminateSubqueryAliasesSuite#afterOptimization().	nullTypeRef(Empty,org/a
TypeRef(Empty,org/apache/spark/sql/cat	org/apache/spark/sql/catalyst/plans/PlanTest#comparePlans().	nullTypeRef(Empty,org/a TypeRef(Empty,org/apac
TypeRef(Empty,org/apache/spark/sql/cat	org/apache/spark/sql/catalyst/rules/RuleExecutor#execute().	nullTypeRef(Empty,org/a

Base Graph Structure



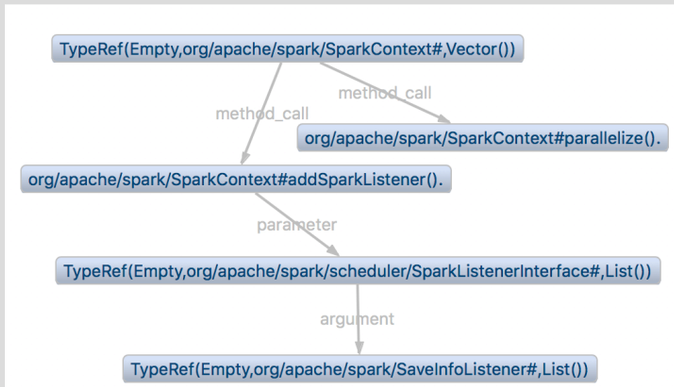
Mine

Mine | Load Mining Results | Save Mining Results

Minned Pattern List. Total: 1842

g Representation	Size	Category	Frequency
ache/spark/sql/Dataset#,Vector())	8	Customization	3
rk/streaming/dstream/DStream#,Vector())	5	Customization	3
/catalyst/parser/DataTypeParserSuite#,Vector())	5	Customization	3
/catalyst/parser/DataTypeParserSuite#,Vector())	5	Customization	3
sql/catalyst/parser/PlanParserSuite#,Vector())	5	Customization	3
sql/catalyst/parser/PlanParserSuite#,Vector())	5	Customization	3
spark/rdd/PairRDDFunctions#,Vector())	11	Customization	3
ql/catalyst/expressions/AttributeSet#,Vector())	5	Customization	3
atalyst/expressions/SpecializedGetters#,Vector())	19	Customization	3
atalyst/expressions/SpecializedGetters#,Vector())	19	Customization	3
xecution/columnar/InMemoryTableScanExec#,V...	5	Customization	3
che/spark/SparkContext#,Vector())	5	Customization	3
che/spark/SparkContext#,Vector())	5	Customization	3
che/spark/SparkContext#,Vector())	5	Customization	3
k/sql/execution/CodegenSupport#,Vector())	5	Customization	3
sql/execution/SparkSqlParserSuite#,Vector())	5	Customization	3
'sql/catalyst/catalog/CatalogTable#,Vector())	5	Customization	3
/memory/UnifiedMemoryManager#,Vector())	5	Customization	3
catalyst/expressions/codegen/CodegenContext...	5	Customization	3
ache/spark/rpc/RpcEnv#,Vector())	5	Customization	3

Pattern Structure



Usages List. Total: 3

Receiver Type	Method Call	Parameter	Argument	Pa
TypeRef(Empty,...	org/apache/spar...	org/apache/spar...	nullTypeRef(Em...	
TypeRef(Empty,...	org/apache/spar...	org/apache/spar...	nullTypeRef(Em...	
TypeRef(Empty,...	org/apache/spar...	org/apache/spar...	nullTypeRef(Em...	

Evaluation

Executed Scala-XP-Miner on 467 Scala projects based on 5 frameworks:

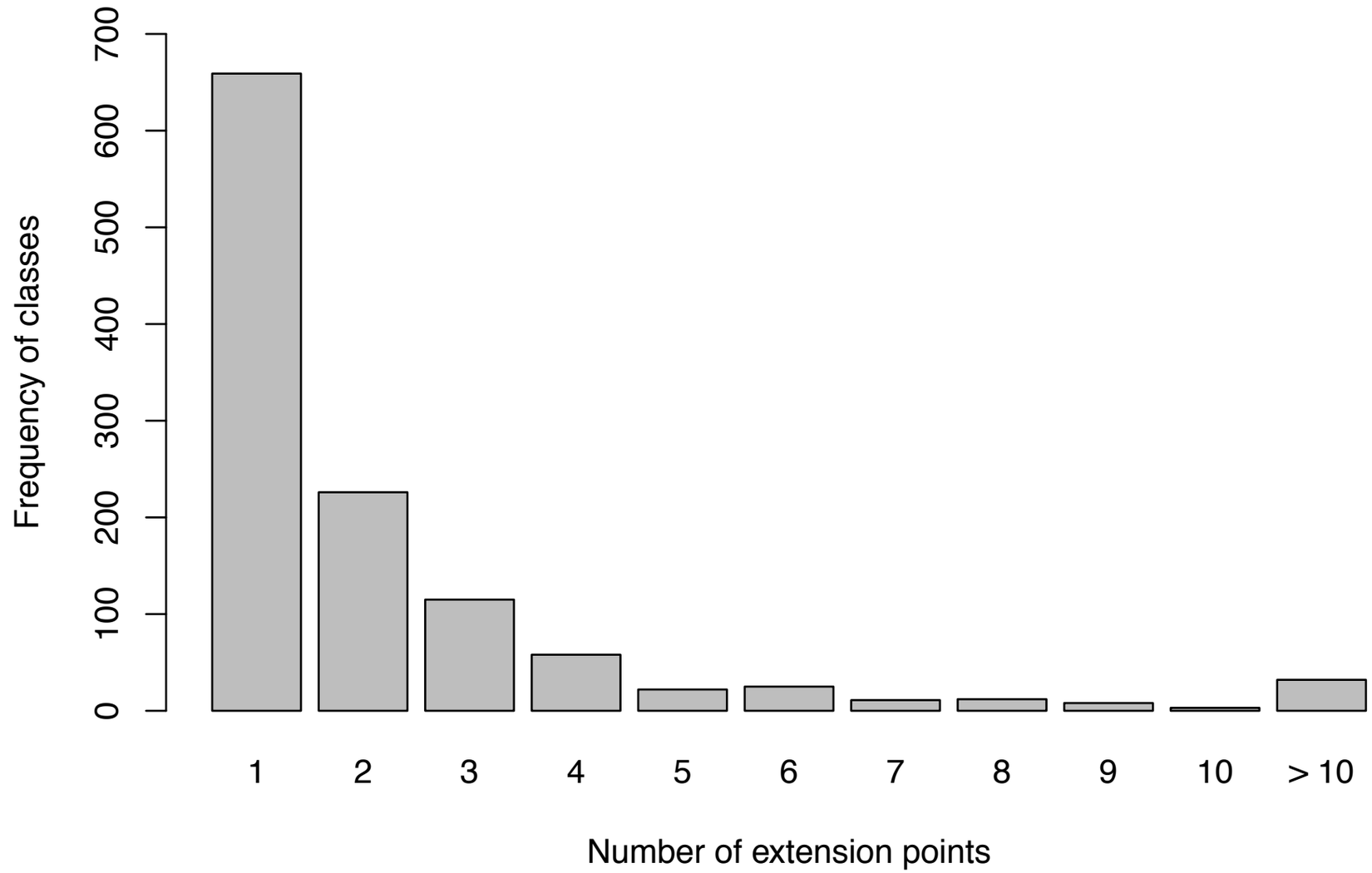
Pattern Accuracy

- Good results in terms of precision, recall, and F-Measure.

Framework	Strategy	RMC AMC	E I	O FMC	Other	ARS	No REC	Precision			Recall			F-Measure		
								Top-1	Top-3	Top-5	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
Spark	Global	9.50%	0.03%	0.47%	90.00%	4	50.83%	0.92	0.95	0.96	0.85	0.89	0.91	0.88	0.91	0.93
	Local					4	53.03%	0.93	0.95	0.97	0.85	0.89	0.91	0.88	0.92	0.93
	D					4	53.18%	0.90	0.94	0.96	0.84	0.88	0.91	0.86	0.91	0.93
Akka	Global	3.96%	0.17%	1.42%	94.45%	3	27.46%	0.96	0.97	0.97	0.88	0.91	0.92	0.91	0.94	0.94
	Local					3	28.93%	0.96	0.97	0.98	0.88	0.92	0.93	0.91	0.94	0.95
	D					3	28.78%	0.95	0.97	0.97	0.88	0.91	0.92	0.91	0.94	0.94
Mockito	Global	11.55%	0.03%	0.03%	88.39%	3	27.31%	0.96	0.97	0.97	0.88	0.91	0.92	0.91	0.94	0.94
	Local					3	0.65%	0.96	0.97	0.97	0.72	0.79	0.82	0.82	0.87	0.88
	D					3	0.65%	0.96	0.97	0.97	0.72	0.77	0.80	0.82	0.85	0.87
Hadoop	Global	3.70%	0.06%	0.03%	96.21%	4	15.06%	0.96	0.96	0.96	0.97	0.98	0.98	0.96	0.97	0.97
	Local					4	23.49%	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98	0.98
	D					4	23.64%	0.98	0.98	0.98	0.97	0.98	0.98	0.98	0.98	0.98
Play	Global	17.02%	0.10%	0.06%	82.82%	4	11.40%	0.92	0.94	0.96	0.85	0.88	0.90	0.87	0.90	0.92
	Local					4	15.36%	0.93	0.95	0.97	0.94	0.88	0.90	0.87	0.90	0.92
	D					4	15.38%	0.92	0.95	0.96	0.85	0.88	0.90	0.88	0.90	0.92

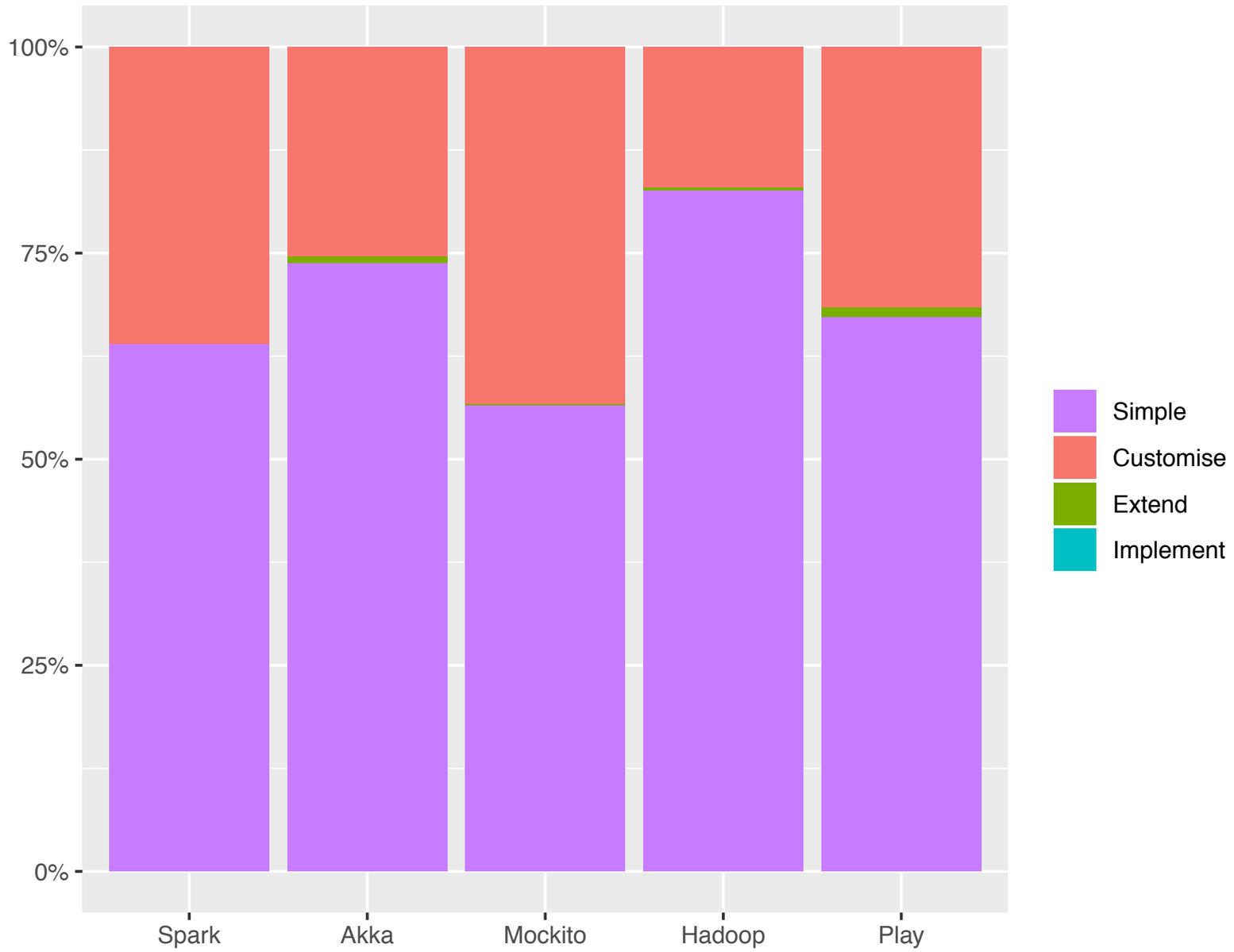
Number of Extension Points

- The majority of classes have less than four extension points.



Extension Point Categories

- Almost all patterns belong to the **Simple** or **Customise** categories.



Summary

- We developed a framework to mine framework usages in Scala projects.
- Subgraph mining based on the **APriori** algorithm.
    - The patterns mined by Scala-XP-Miner are accurate
    - The number of extension patterns per class is not very high (i.e., lower than 4)
    - Most of the patterns are pretty simple
- Future work:
    - add support to language-specific patterns;
    - assess the usefulness of the proposed patterns.

 @dardin88  dario.di.nucci@vub.be

Mining Edits

Context: systematic edits

A group of similar changes

Context: systematic edits

A group of similar changes

```
- Parser p = new XmlParser(registerResponse);
+ Parser p = new XmlParser();
  p.setErrorHandler(eh);
- Document registerRoot = p.parse();
+ Document registerRoot = p.parse(registerResponse);
```

Context: systematic edits

A group of similar changes

```
- Parser p = new XmlParser(registerResponse);
+
-
+
```

```
- Parser p = new XmlParser(updateProfileResponse);
+ Parser p = new XmlParser();
  p.setErrorHandler(eh);
- Document profileRoot = p.parse();
+ Document profileRoot = p.parse(updateProfileResponse);
```

Context: systematic edits

A group of similar changes

```
- Parser p = new XmlParser(registerResponse);
+
```

```
- Parser p = new XmlParser(updateProfileResponse);
+
```

```
- Parser p = new XmlParser(loginResponse);
+  Parser p = new XmlParser();
   p.setErrorHandler(eh);
- Document loginRoot = p.parse();
+ Document loginRoot = p.parse(loginResponse);
```

Context: systematic edits

A group of similar changes

```
- Parser p = new XmlParser(registerResponse);
+
```

```
- Parser p = new XmlParser(updateProfileResponse);
+
```

```
- Parser p = new XmlParser(loginResponse);
+  Parser p = new XmlParser();
   p.setErrorHandler(eh);
- Document loginRoot = p.parse();
+ Document loginRoot = p.parse(loginResponse);
```

- 3 instances, 1 systematic edit

Context: systematic edits

A group of similar changes

```
- Parser p = new XmlParser(registerResponse);
+
```

```
- Parser p = new XmlParser(updateProfileResponse);
+
```

```
- Parser p = new XmlParser(loginResponse);
+  Parser p = new XmlParser();
   p.setErrorHandler(eh);
- Document loginRoot = p.parse();
+ Document loginRoot = p.parse(loginResponse);
```

- 3 instances, 1 systematic edit
- Examples: library migration, refactoring, fixing occurrences of a bug, ...

Mining for Systematic edits

Systematic edits can be tedious and error-prone if done manually!

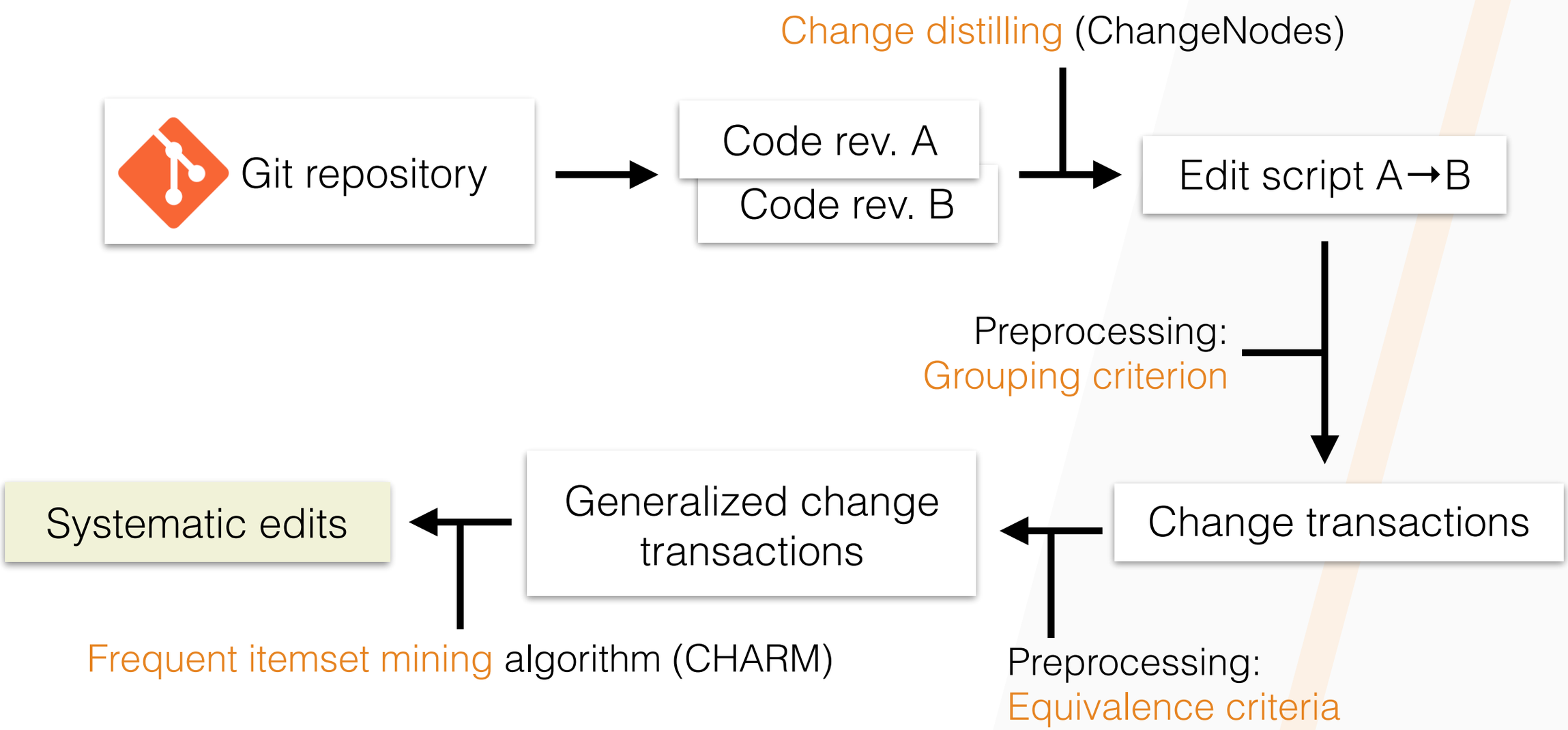
SysEdMiner identifies systematic edits in a Java project's history based on:

- Source code changes expressed as AST nodes
- Frequent itemset mining

Applications:

- Detecting error-prone code
- Assist in refactoring decisions
- Generating transformations based on existing instances

Overview of SysEdMiner



Running example

```
   public class Point {
     private int x;
     private int y;

     public double computeDistance(Point p) {
+      if (this.equals(p)) return 0;
       double dX = this.computeDeltaX(p);
       double dY = this.computeDeltaY(p);
       return Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));
     }

     public double computeDirection(Point point) {
+     if (this.equals(point)) return 0;
       double dX = this.computeDeltaX(point);
       double dY = this.computeDeltaY(point);
       return Math.atan2(dY, dX) * 180 / Math.PI;
     }
    ...
   }
```

Change distilling

```
+  if (this.equals(p)) return 0;
```

Change distilling

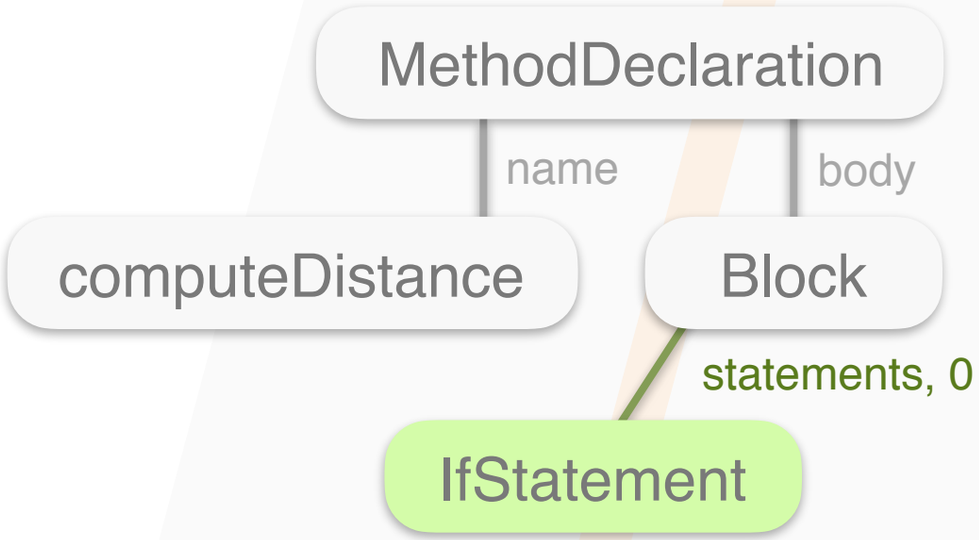
```
+  if (this.equals(p)) return 0;
```

- **C1**: insert(IfStatement if(){}, body of computeDistance, statements-0)

Change distilling

```
+  if (this.equals(p)) return 0;
```

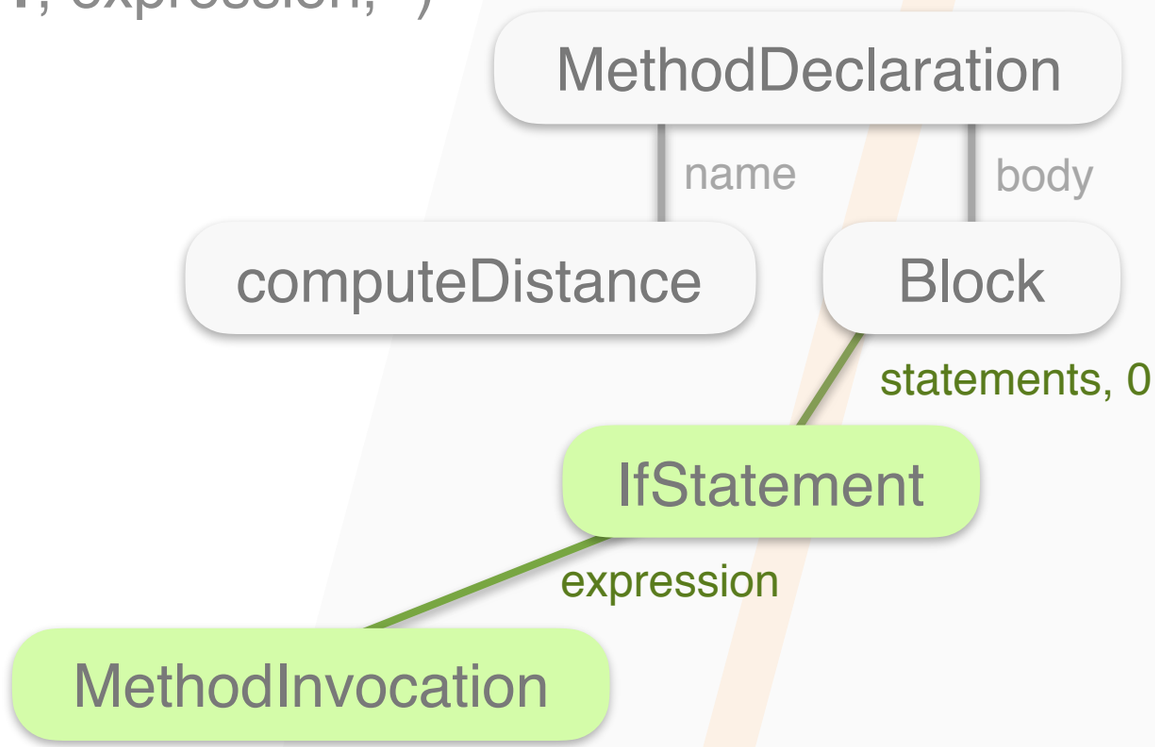
- **C1**: insert(IfStatement if(){}, body of computeDistance, statements-0)



Change distilling

```
+  if (this.equals(p)) return 0;
```

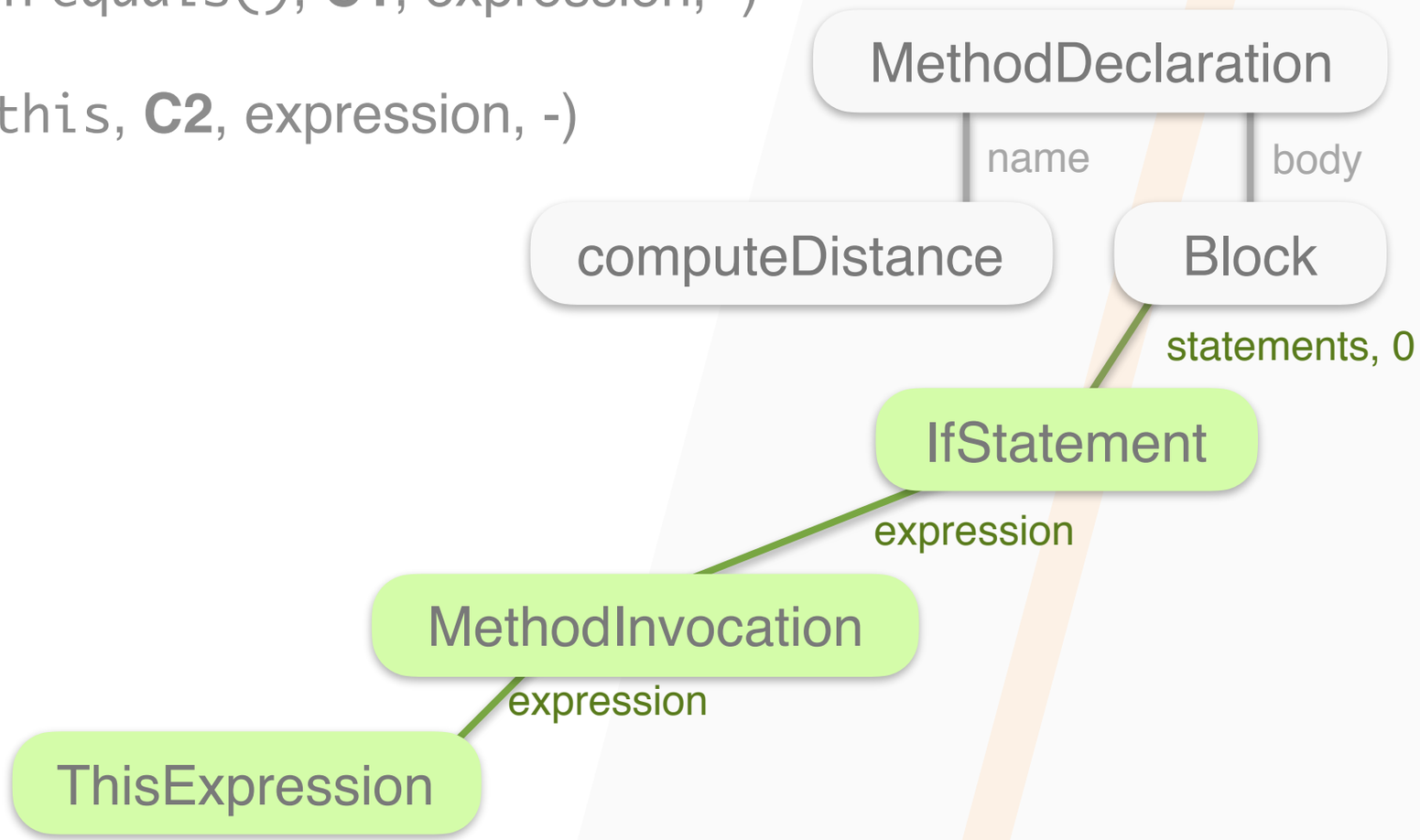
- **C1**: insert(IfStatement if(){}, body of computeDistance, statements-0)
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)



Change distilling

```
+  if (this.equals(p)) return 0;
```

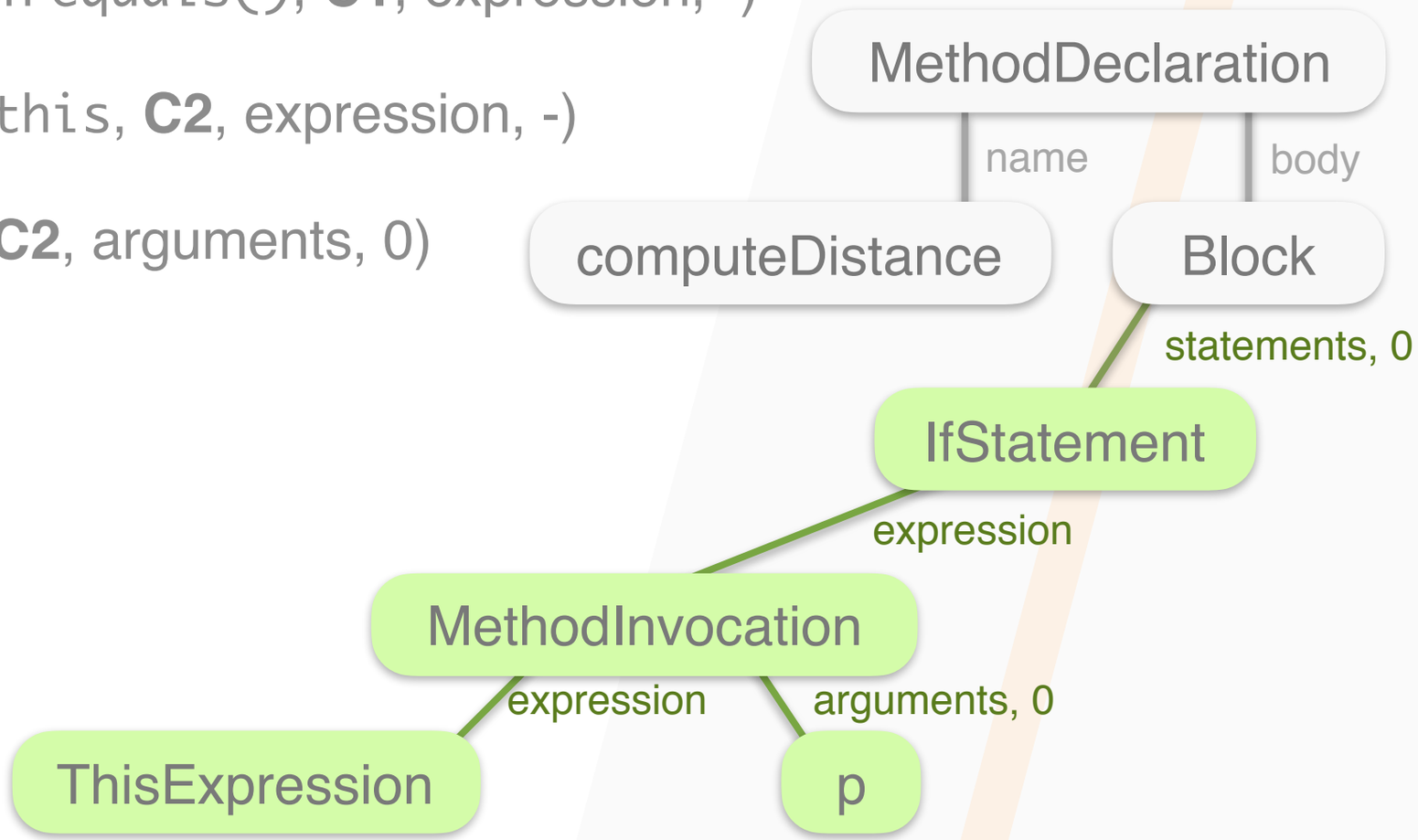
- **C1**: insert(IfStatement if(){}, body of computeDistance, statements-0)
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)
- **C3**: insert(ThisExpression `this`, **C2**, expression, -)



Change distilling

```
+  if (this.equals(p)) return 0;
```

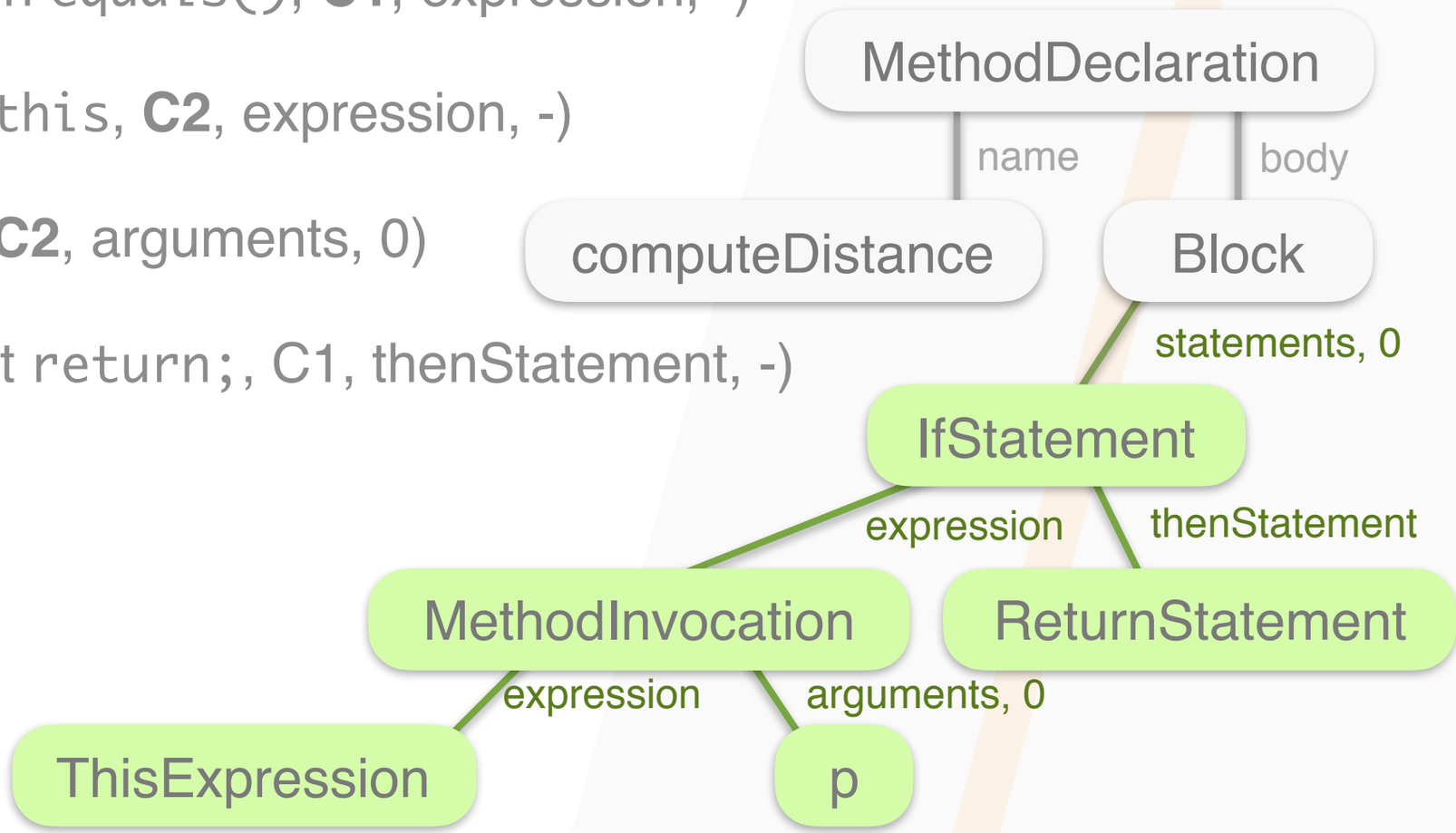
- **C1**: insert(IfStatement if(){}, body of computeDistance, statements-0)
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)
- **C3**: insert(ThisExpression `this`, **C2**, expression, -)
- **C4**: insert(SimpleName p, **C2**, arguments, 0)



Change distilling

```
+  if (this.equals(p)) return 0;
```

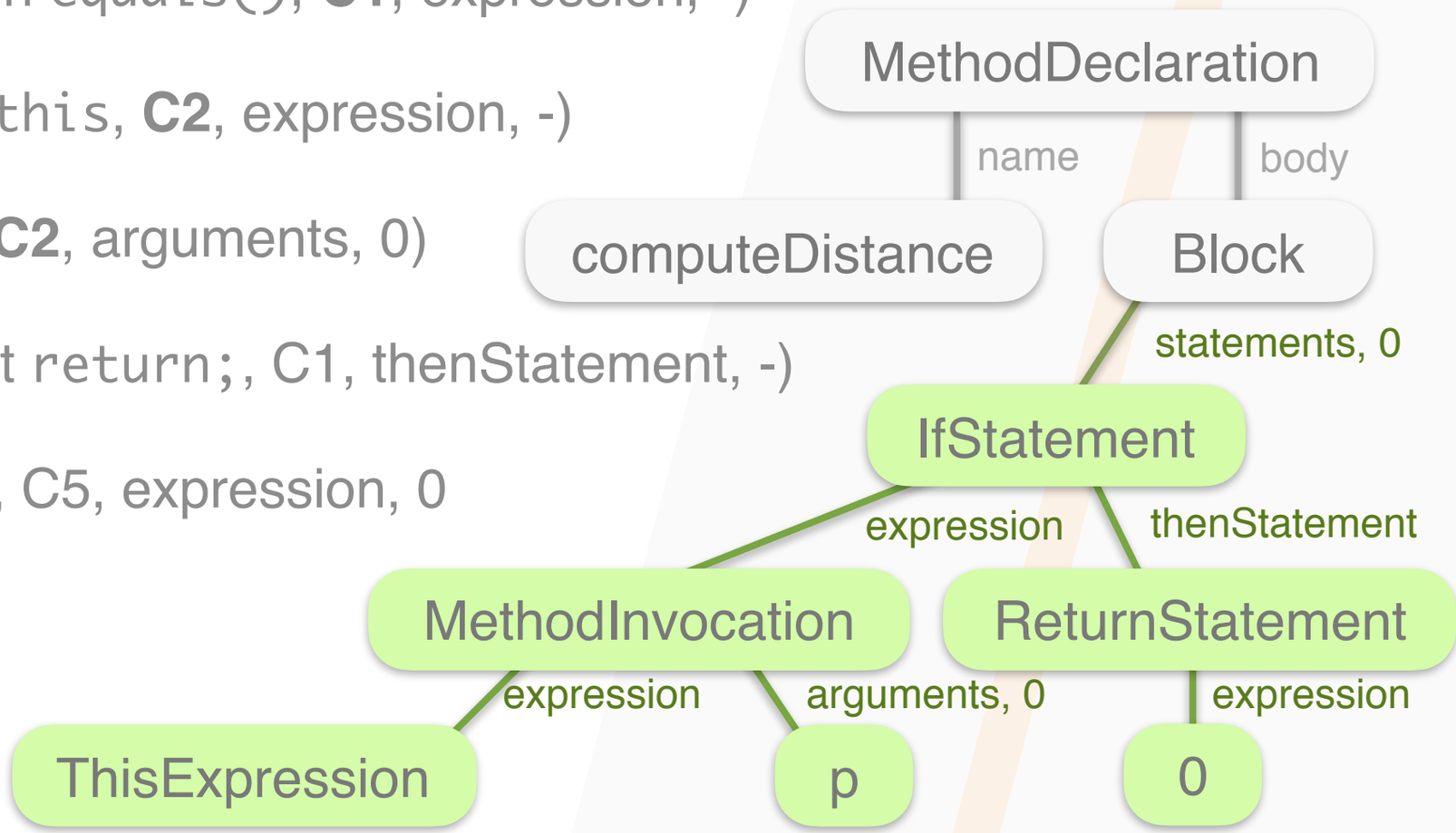
- **C1**: insert(IfStatement if(){}, body of computeDistance, statements-0)
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)
- **C3**: insert(ThisExpression `this`, **C2**, expression, -)
- **C4**: insert(SimpleName `p`, **C2**, arguments, 0)
- **C5**: insert(ReturnStatement `return;`, C1, thenStatement, -)



Change distilling

```
+  if (this.equals(p)) return 0;
```

- **C1**: insert(IfStatement if(){}, body of computeDistance, statements-0)
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)
- **C3**: insert(ThisExpression `this`, **C2**, expression, -)
- **C4**: insert(SimpleName `p`, **C2**, arguments, 0)
- **C5**: insert(ReturnStatement `return;`, C1, thenStatement, -)
- **C6**: insert(NumberLiteral 0, C5, expression, 0



Change distilling

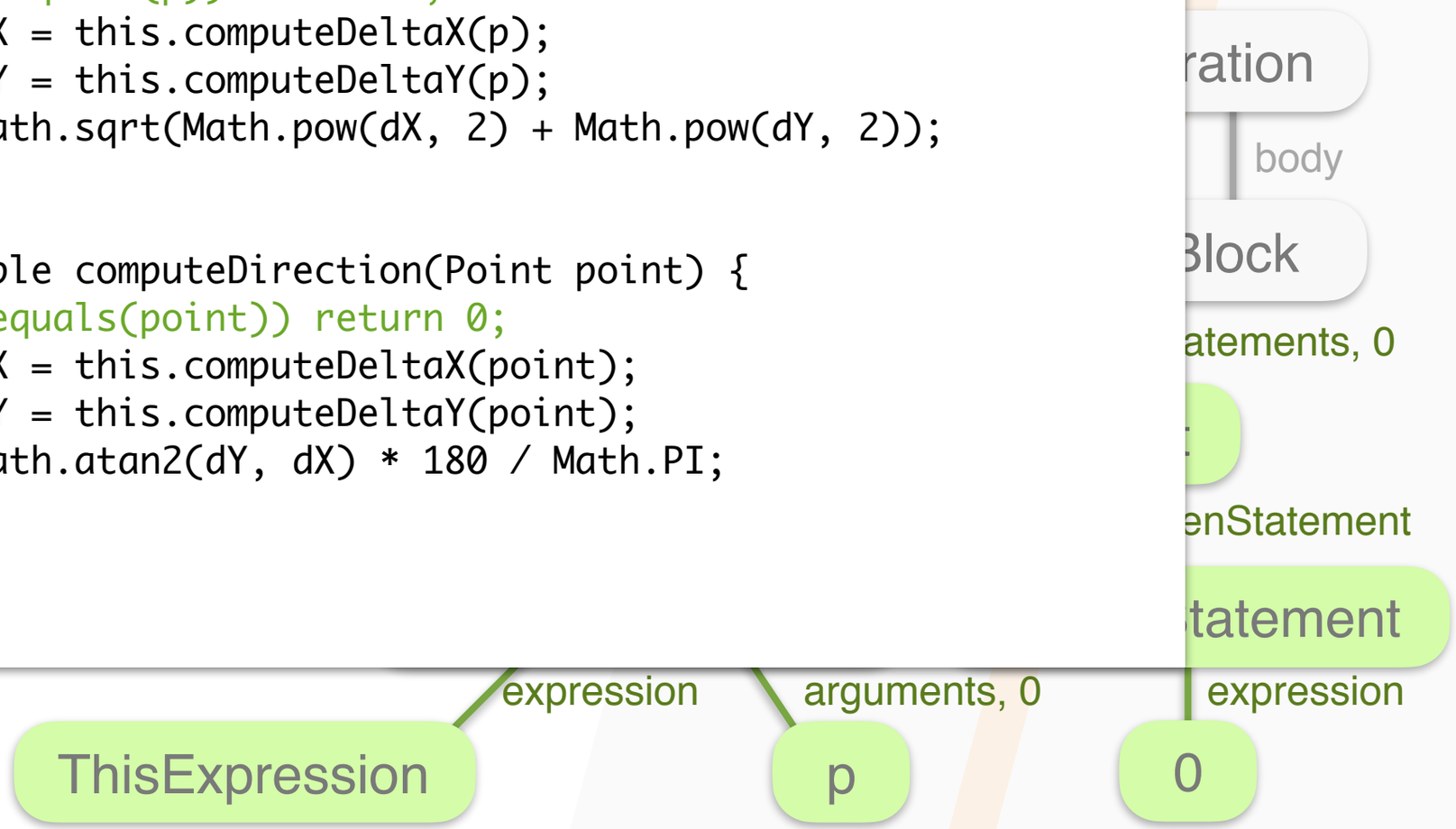
```
+  if (this.
```

- C1: insert(I
- C2: insert(M
- C3: insert(T
- C4: insert(S
- C5: insert(P
- C6: insert(M

```
public class Point {
  private int x;
  private int y;

  public double computeDistance(Point p) {
+   if (this.equals(p)) return 0;
    double dX = this.computeDeltaX(p);
    double dY = this.computeDeltaY(p);
    return Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));
  }

  public double computeDirection(Point point) {
+   if (this.equals(point)) return 0;
    double dX = this.computeDeltaX(point);
    double dY = this.computeDeltaY(point);
    return Math.atan2(dY, dX) * 180 / Math.PI;
  }
  ...
}
```



Change distilling

```
+  if (this.
```

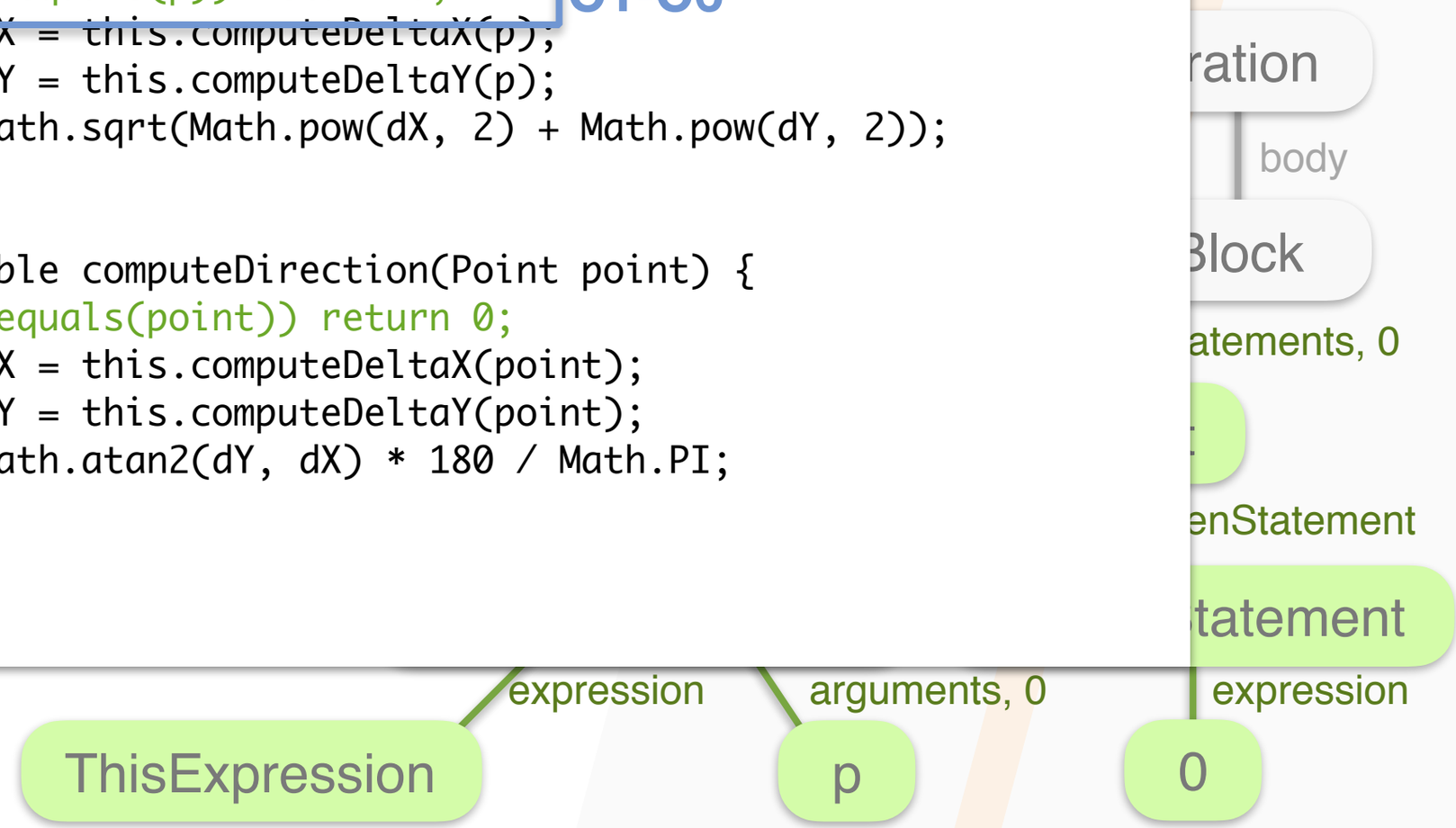
- C1: insert(I
- C2: insert(M
- C3: insert(T
- C4: insert(S
- C5: insert(F
- C6: insert(M

```
public class Point {
  private int x;
  private int y;

  public double computeDistance(Point p) {
+   if (this.equals(p)) return 0;
    double dX = this.computeDeltaX(p);
    double dY = this.computeDeltaY(p);
    return Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));
  }

  public double computeDirection(Point point) {
+   if (this.equals(point)) return 0;
    double dX = this.computeDeltaX(point);
    double dY = this.computeDeltaY(point);
    return Math.atan2(dY, dX) * 180 / Math.PI;
  }
  ...
}
```

C1-C6



Change distilling

+ if (this.

- C1: insert(I
- C2: insert(N
- C3: insert(T
- C4: insert(S
- C5: insert(P
- C6: insert(N

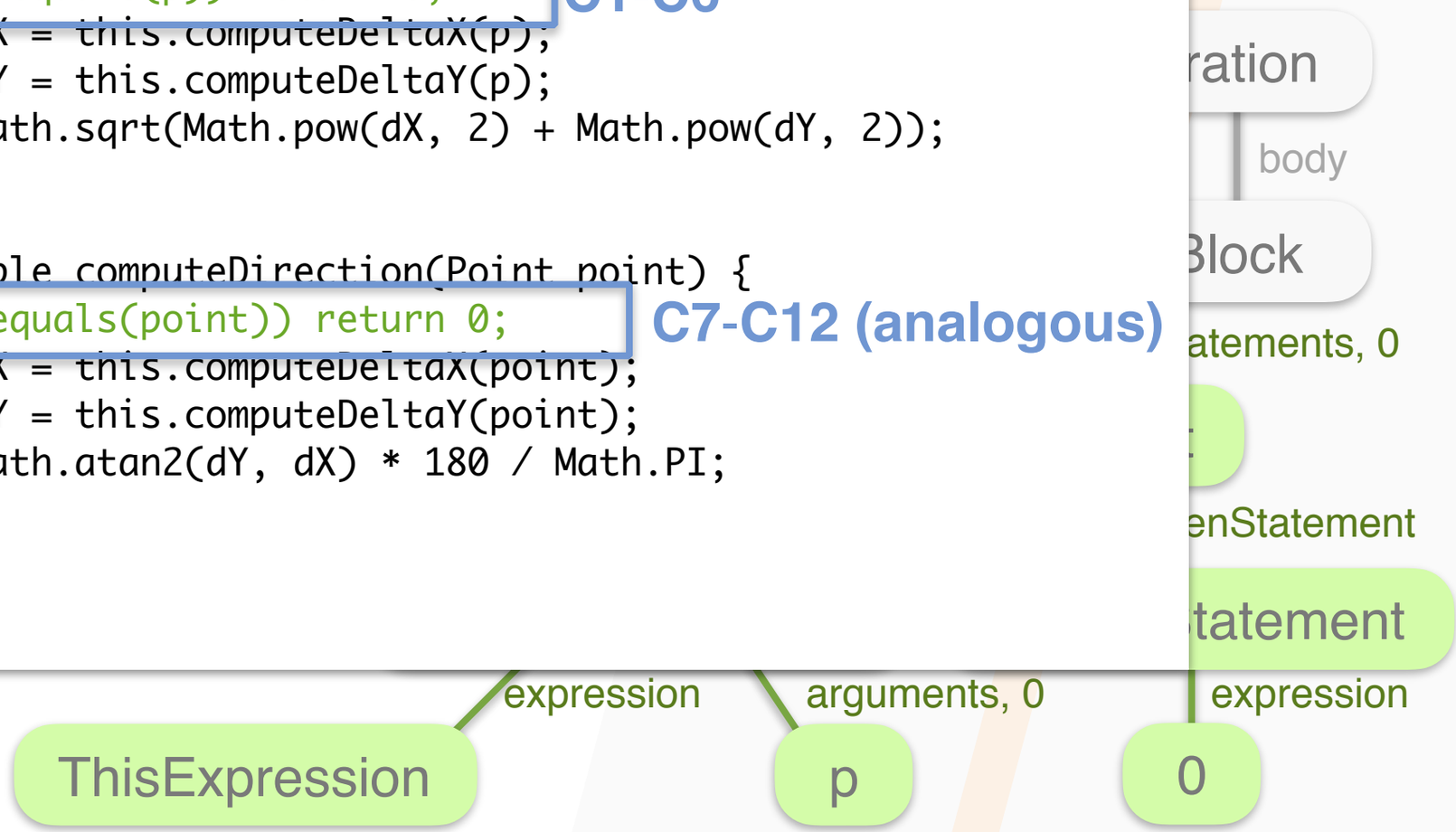
```
public class Point {
  private int x;
  private int y;

  public double computeDistance(Point p) {
+   if (this.equals(p)) return 0;
    double dX = this.computeDeltaX(p);
    double dY = this.computeDeltaY(p);
    return Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));
  }

  public double computeDirection(Point point) {
+   if (this.equals(point)) return 0;
    double dX = this.computeDeltaX(point);
    double dY = this.computeDeltaY(point);
    return Math.atan2(dY, dX) * 180 / Math.PI;
  }
  ...
}
```

C1-C6

C7-C12 (analogous)



Grouping criterion

Grouping criterion

- All changes grouped into transactions

Grouping criterion

- All changes grouped into transactions
- Group changes by the method in which they occur:

Grouping criterion

- All changes grouped into transactions
- Group changes by the method in which they occur:

```
public class Point {
  private int x;
  private int y;

  public double computeDistance(Point p) {
+   if (this.equals(p)) return 0;
    double dX = this.computeDeltaX(p);
    double dY = this.computeDeltaY(p);
    return Math.sqrt(Math.pow(dX, 2) + Math.pow(dY, 2));
  }

  public double computeDirection(Point point) {
+   if (this.equals(point)) return 0;
    double dX = this.computeDeltaX(point);
    double dY = this.computeDeltaY(point);
    return Math.atan2(dY, dX) * 180 / Math.PI;
  }
  ...
}
```

C1-C6

C7-C12

Grouping criterion

- All changes grouped into transactions
- Group changes by the method in which they occur:

Grouping criterion

- All changes grouped into transactions
- Group changes by the method in which they occur:
  - ⟨Point.computeDistance, {**C1**, **C2**, **C3**, **C4**, **C5**, **C6**}⟩
  - ⟨Point.computeDirection, {**C7**, **C8**, **C9**, **C10**, **C11**, **C12**}⟩

Grouping criterion

- All changes grouped into transactions
- Group changes by the method in which they occur:
  - ⟨Point.computeDistance, {**C1**, **C2**, **C3**, **C4**, **C5**, **C6**}⟩
  - ⟨Point.computeDirection, {**C7**, **C8**, **C9**, **C10**, **C11**, **C12**}⟩
- Limitations:
  - Changes outside methods ignored
  - Instances larger than a method are split up
  - Only one instance per method
  - Future work: use multiple grouping criteria

Equivalence criteria

Equivalence criteria

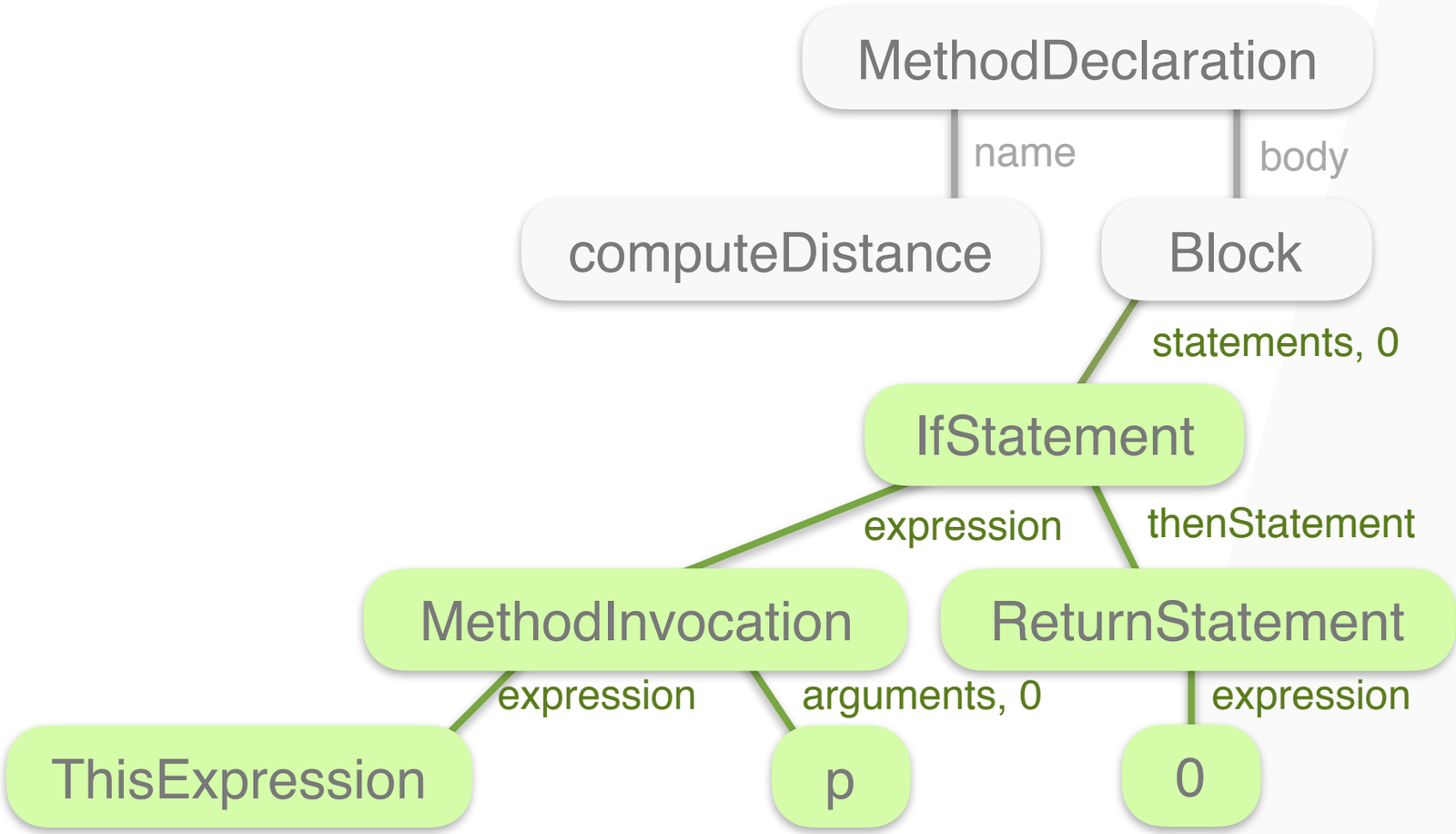
- Change equivalence based on change type, subject & context

Equivalence criteria

- Change equivalence based on change type, subject & context
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)

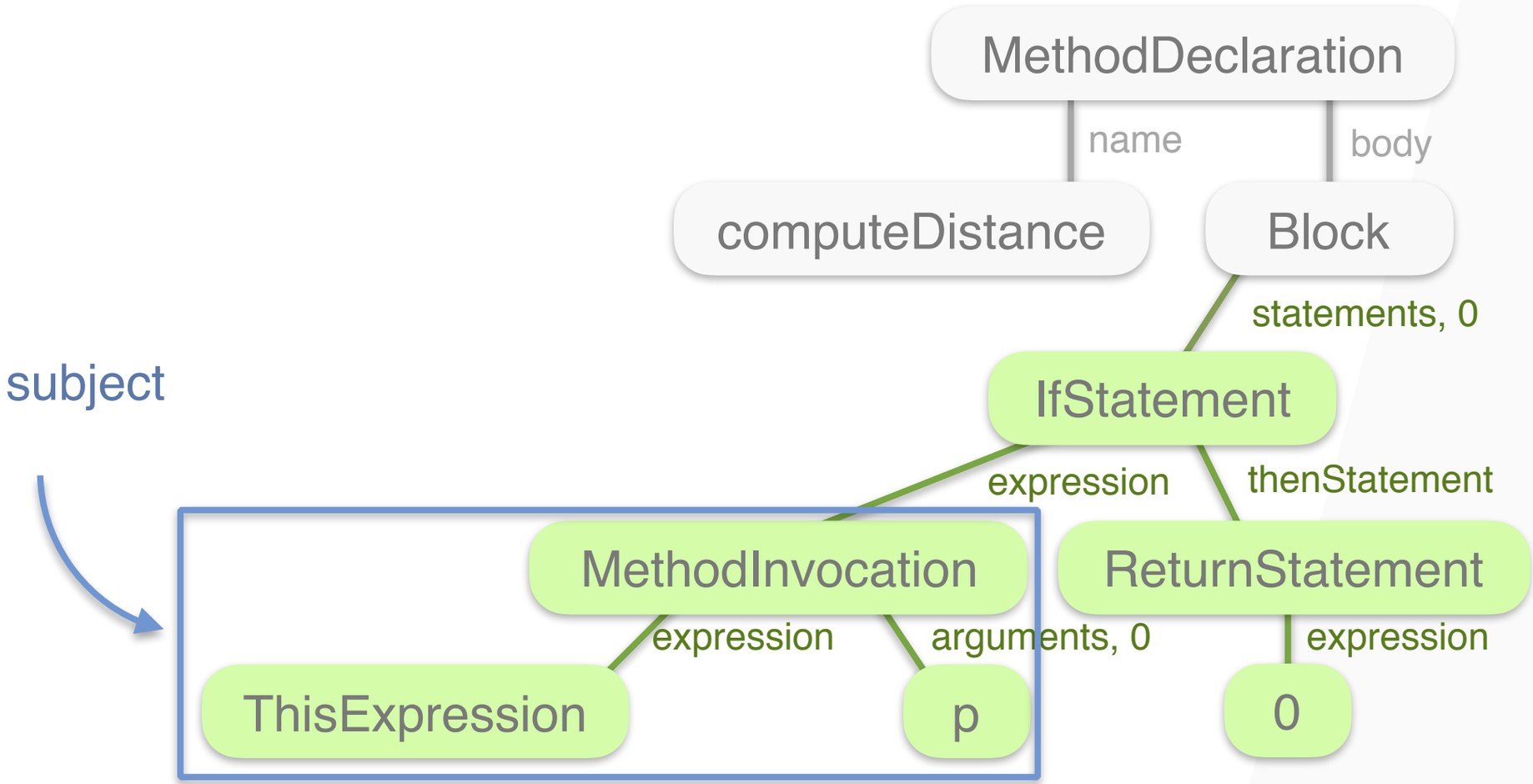
Equivalence criteria

- Change equivalence based on change type, subject & context
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)



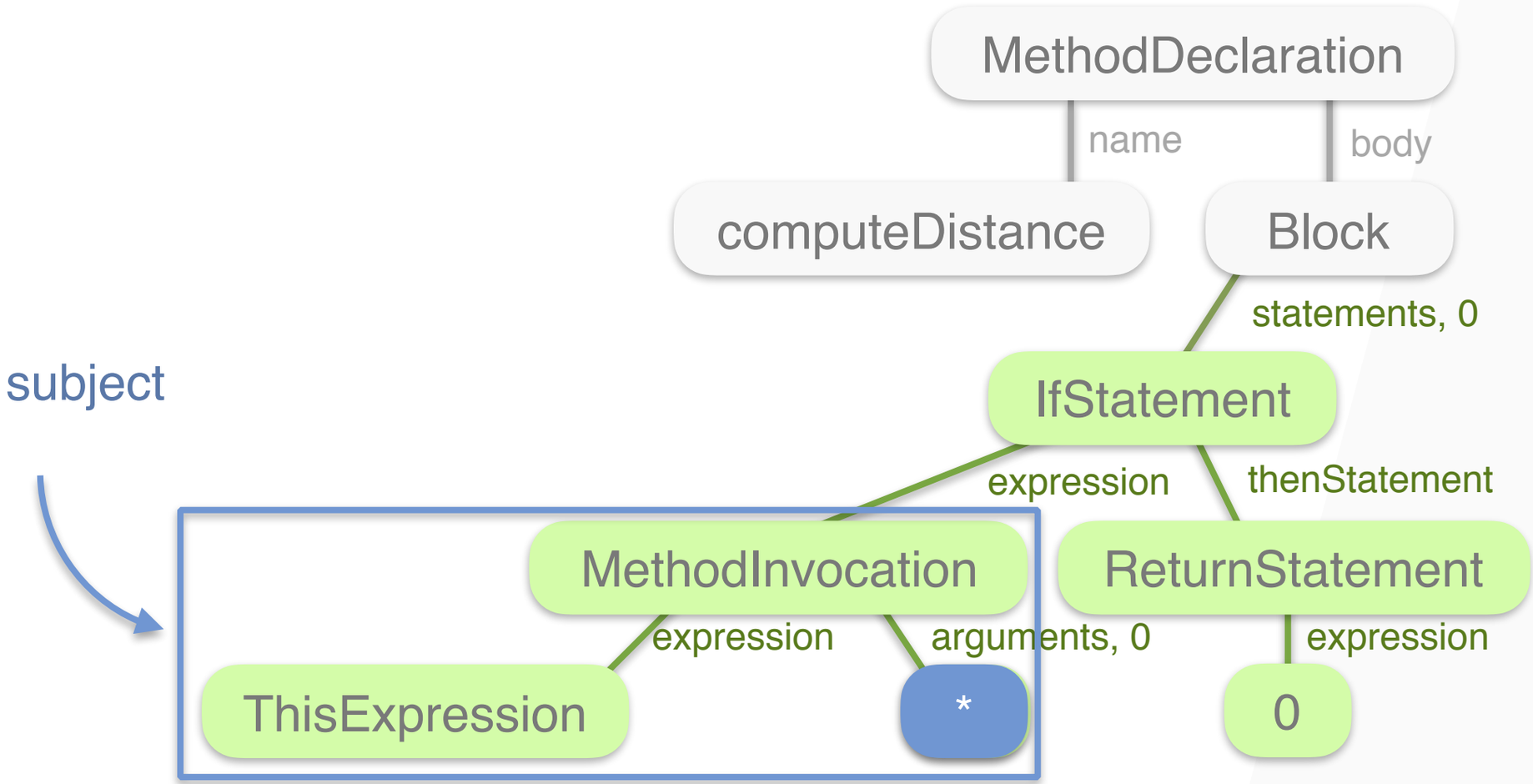
Equivalence criteria

- Change equivalence based on change type, subject & context
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)



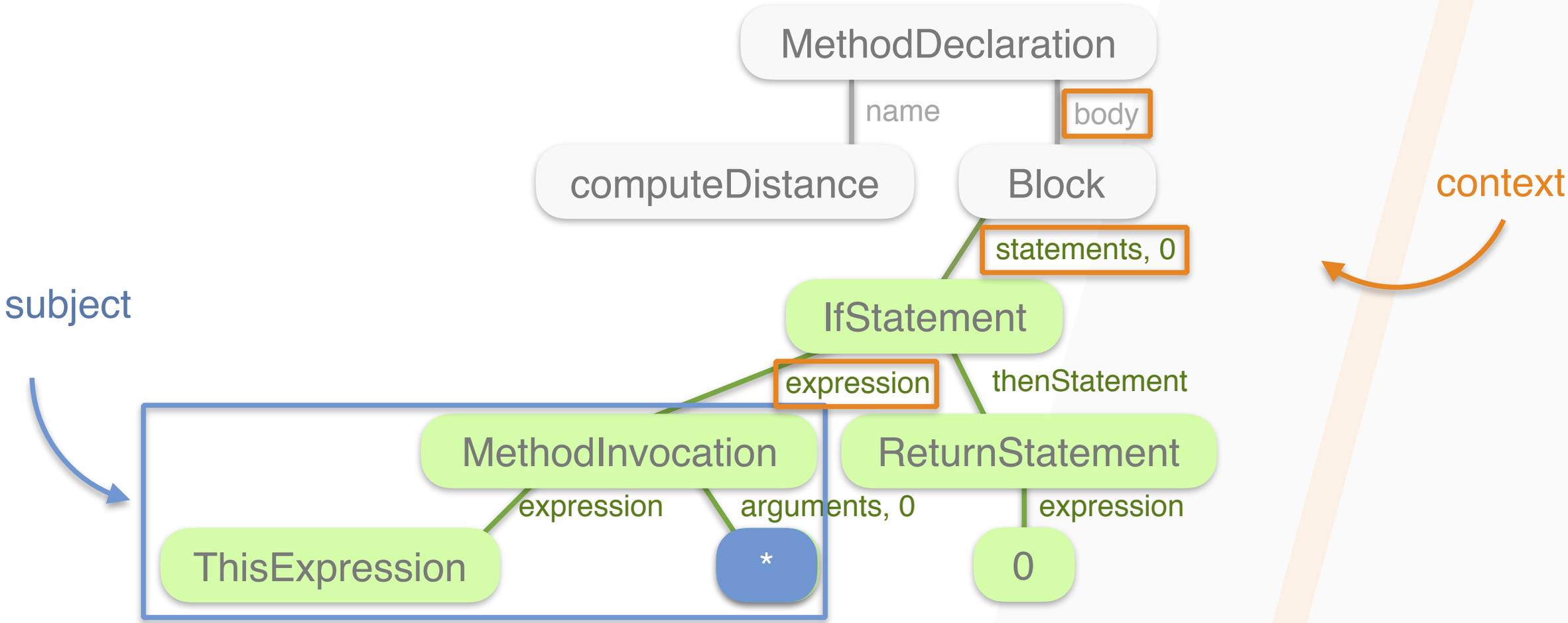
Equivalence criteria

- Change equivalence based on change type, subject & context
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)



Equivalence criteria

- Change equivalence based on change type, subject & context
- **C2**: insert(MethodInvocation `equals()`, **C1**, expression, -)



Generalised transactions

- Equivalence criteria used to generalise transactions:

⟨Point.computeDirection, {

C1: ⟨insert, `if(this.*(*)) return 0;`, body : statements, 0⟩

C2: ⟨insert, `*(*)`, body : statements, 0 : expression⟩

C3: ⟨insert, `this`, body : statements, 0 : expression : expression⟩

C4: ⟨insert, *, body : statements, 0 : expression : arguments, 0⟩

C5: ⟨insert, return *;, body : statements, 0 : thenStatement⟩

C6: ⟨insert, *, body : statements, 0 : thenStatement : expression⟩

}⟩

⟨Point.computeDirection, {

C7: ⟨insert, `if(this.*(*)) return 0;`, body : statements, 0⟩

C8: ⟨insert, `*(*)`, body : statements, 0 : expression⟩

C9: ⟨insert, `this`, body : statements, 0 : expression : expression⟩

C10: ⟨insert, *, body : statements, 0 : expression : arguments, 0⟩

C11: ⟨insert, return *;, body : statements, 0 : thenStatement⟩

C12: ⟨insert, *, body : statements, 0 : thenStatement : expression⟩

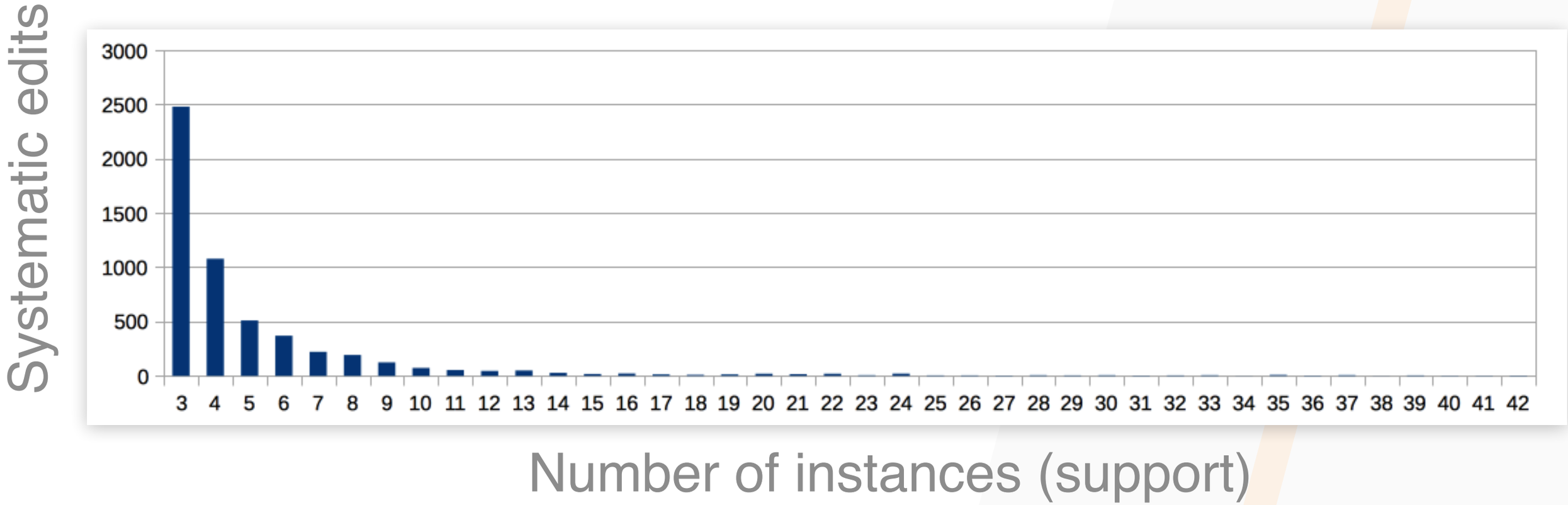
}⟩

Evaluation

- Gauge the tool's correctness, usefulness, and scalability
- Applied to code base of TP Vision Belgium
  - 51 git repositories with  projects
  - 43,756 commits (± 30s per commit)
  - 5,474 systematic edits
  - 78/100 random samples correct (manually inspected)

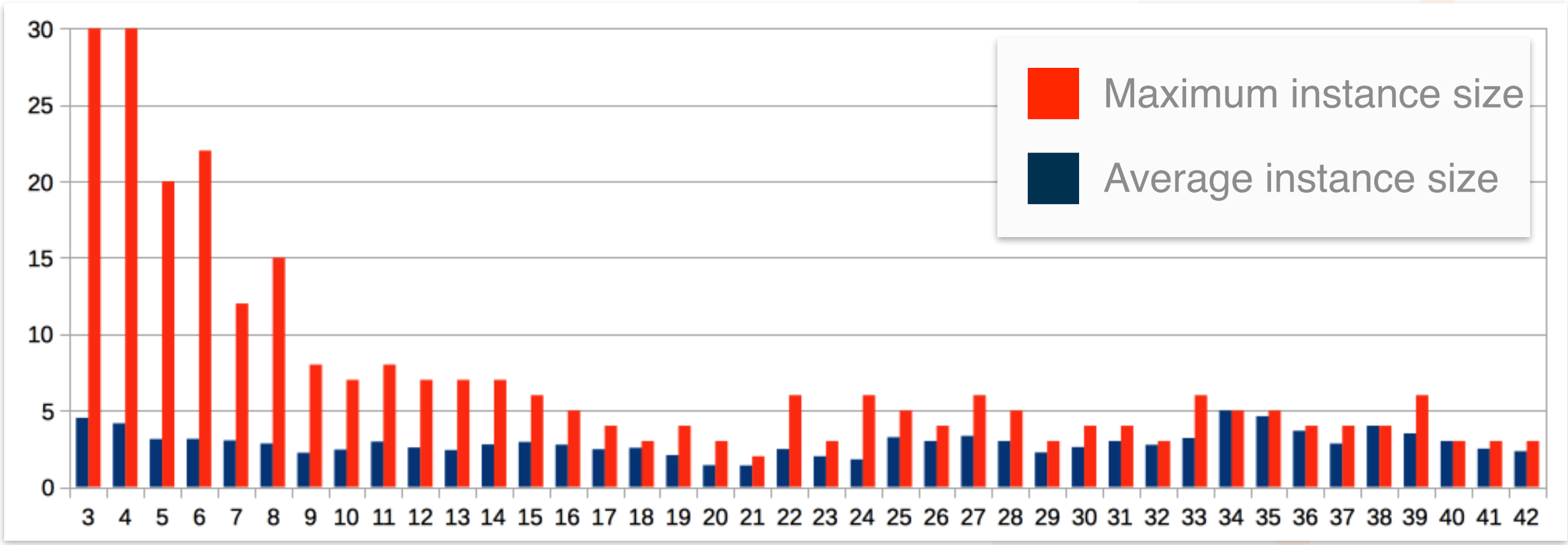
Number of instances

- Majority of systematic edits have few instances



Instance size

- On average, 3-4 AST-level changes, disregarding # of instances
- Larger instance size can occur with small # of instances



Summary

- Technique to identify systematic edits
- ~12.5% of commits contain systematic edits;

  mostly small # of instances, but they can have a large size
- Future work:
  - exploring the configuration space;
  - mining for specific types of systematic edits;
  - mining across commits

timmolderez/SysEdMiner

@timmolderez

tim.molderez@vub.be

Mining Source Code^3

Mining Idioms, Usages and Edits

Dario Di Nucci

Research Fellow
dario.di.nucci@vub.be

VRIJE UNIVERSITEIT BRUSSEL

Software Languages Lab