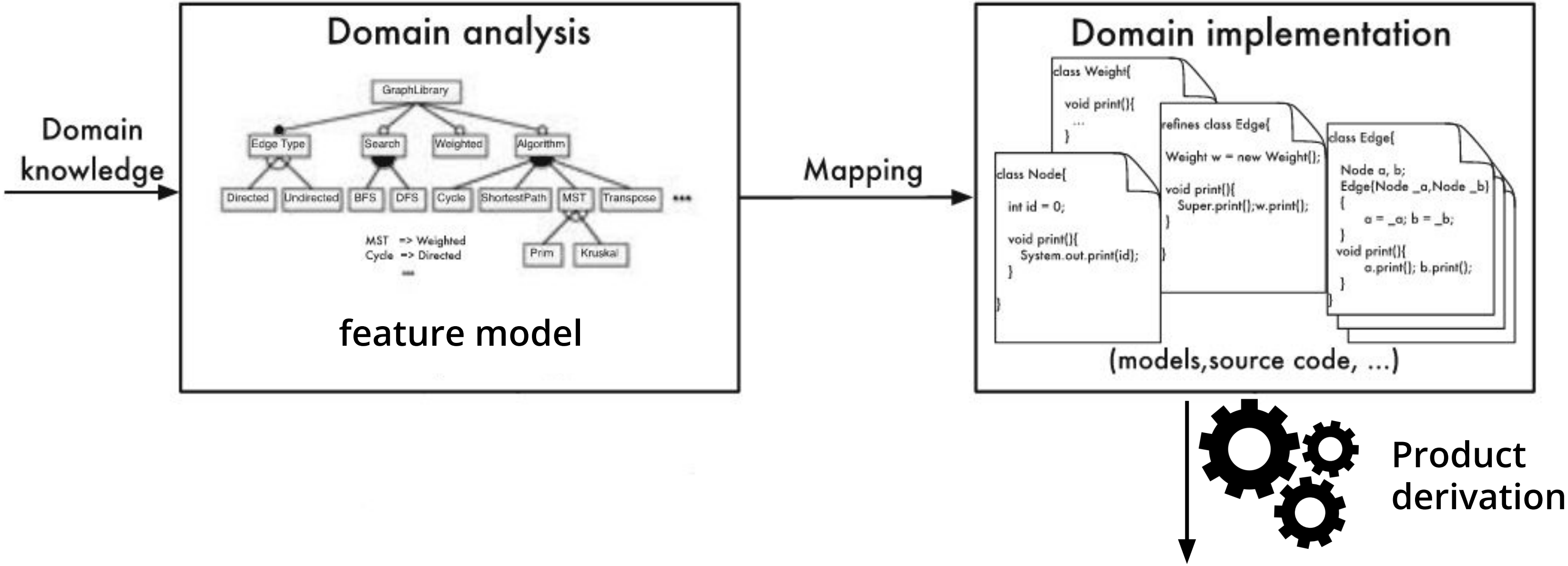
Automatic identification of object-oriented variability implementations

Johann Mortara – Philippe Collet

Université Côte d'Azur, CNRS, I3S, France

Journée de travail du groupe Vélocité Logicielle
du GDR GPL
December 16, 2020

Software Product Lines



Domain knowledge

Domain analysis

MST => Weighted
Cycle => Directed
...

feature model

Mapping

Domain implementation

```
class Weight{

  void print(){
   ...
  }
```

```
class Node{

  int id = 0;

  void print(){
    System.out.print(id);
  }

}
```

```
refines class Edge{

 Weight w = new Weight();

 void print(){
   Super.print();w.print();
 }

}
```

```
class Edge{

  Node a, b;
  Edge(Node _a,Node _b)
  {
       a = _a; b = _b;
  }
 void print(){
       a.print(); b.print();
  }
}
```

(models,source code, ...)

Product derivation

Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

Variability-Rich Systems with a Single Code Base

16.000 options managed in 25M LoC [Acher2018]

#ifdef

24.000 different platforms in 2015 [Open2015]

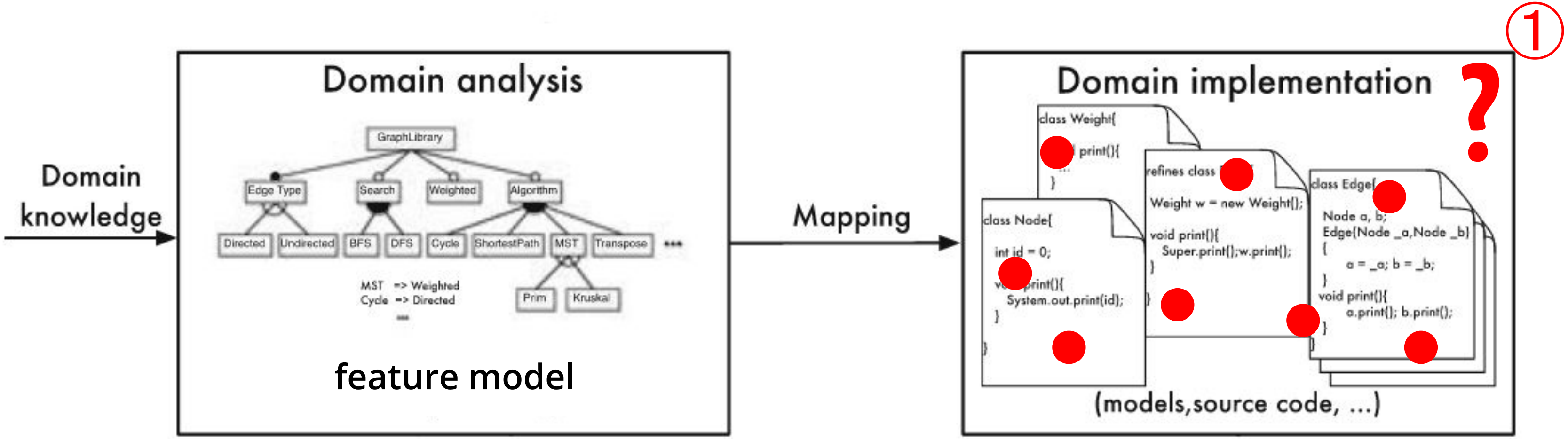
Object-orientation

2.000+ options generating variants for platforms, security levels... [Acher2018]

Object-orientation

and many variability implementation techniques...

Problem 1: How to identify variability implementations in an existing OO codebase?



```
class Weight{
   print(){
   ...
}
```

```
refines class
Weight w = new Weight();
void print(){
   Super.print();w.print();
}
}
```

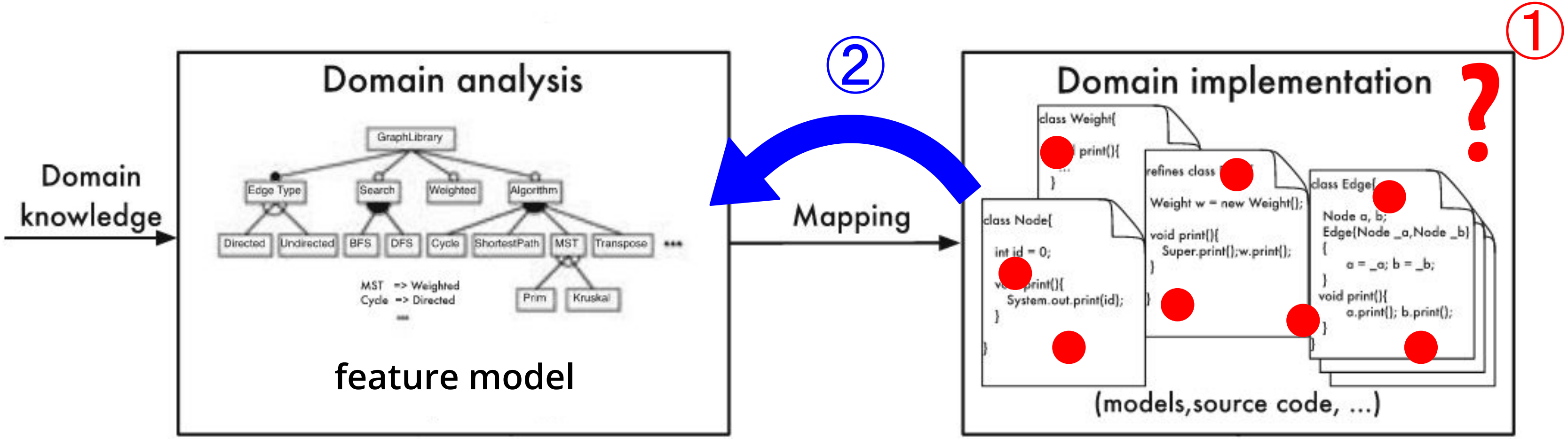
```
class Edge{
 Node a, b;
 Edge(Node _a,Node _b)
 {
     a = _a; b = _b;
 }
void print(){
     a.print(); b.print();
}
}
```

```
class Node{
  int id = 0;
  v    print(){
   System.out.print(id);
  }
}
```

Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

Problem 1: How to identify variability implementations in an existing OO codebase?

Problem 2: How to map these variability implementations to domain features?



```
class Weight{
     print(){
    ...
  }
```

```
class Node{
  int id = 0;
  v    print(){
    System.out.print(id);
  }
}
```

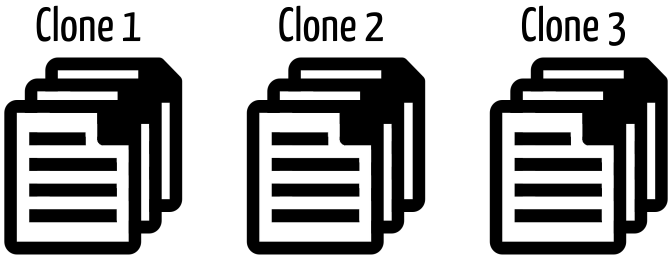
```
refines class
 Weight w = new Weight();
 void print(){
   Super.print();w.print();
 }
}
```

```
class Edge{
  Node a, b;
  Edge(Node _a,Node _b)
  {
      a = _a; b = _b;
  }
 void print(){
      a.print(); b.print();
  }
}
```

Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

State of the art on variability implementations detection

Context: projects clones

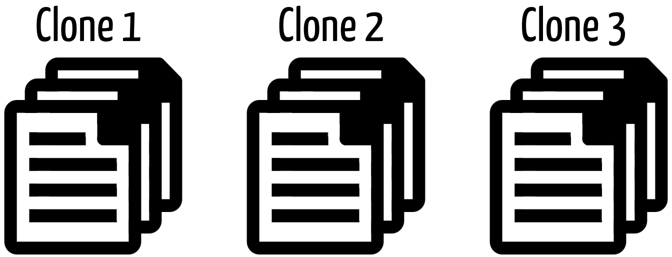


Detection method:

Comparison between clones and mapping with the domain features [Assunção2017]

State of the art on variability implementations detection

Context: projects clones

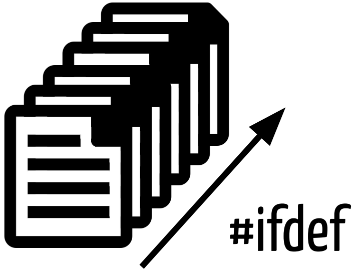


Detection method:

Comparison between clones and mapping with the domain features [Assunção2017]

Context: unique codebase and preprocessing directives

#ifdef ⟶ variant



Detection method:

Determining the consistency of directives [Liebig2010]

State of the art on variability implementations detection

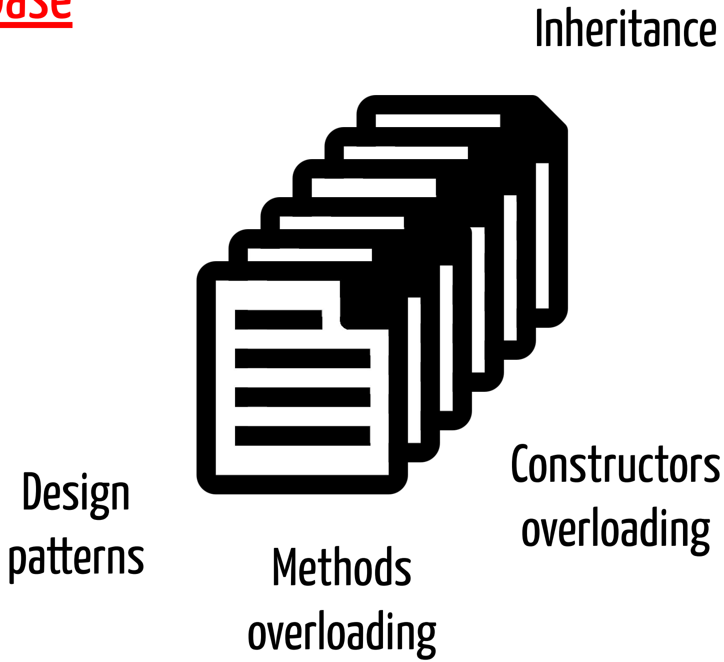
Our context: large and unique object-oriented codebase

- Several implementation mechanisms
- Variability buried in the code (variation points)

Detection method:

Currently no method

[Lozano2011], [Metzger2014], [Tërnava2017]



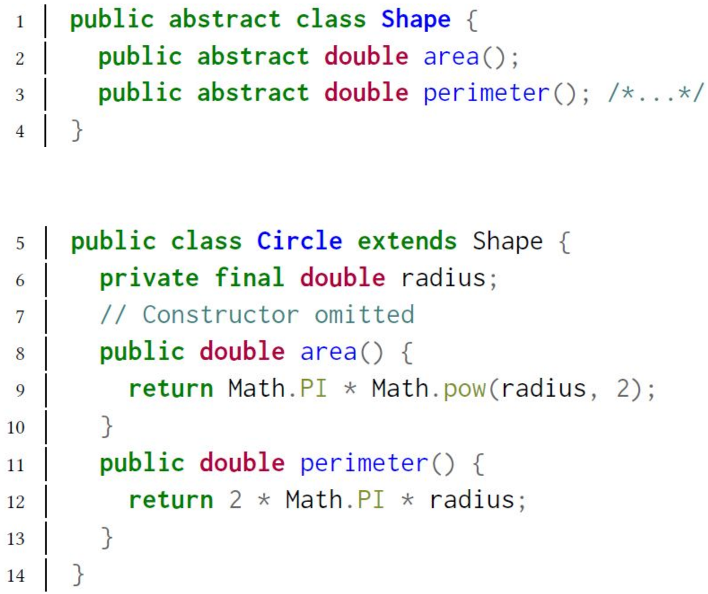
Inheritance

Design patterns

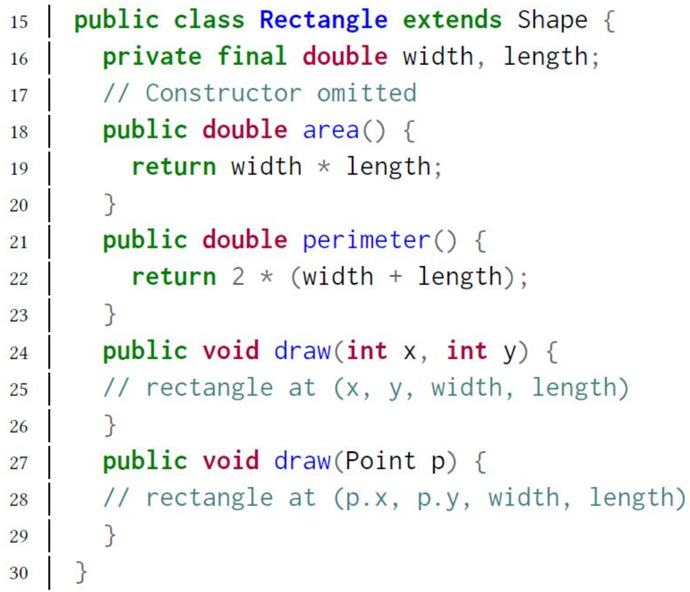
Methods overloading

Constructors overloading

Variation points and variants



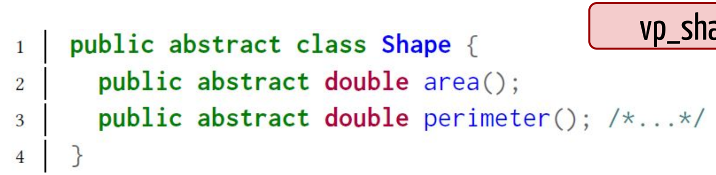
```
public abstract class Shape {
  public abstract double area();
  public abstract double perimeter(); /*...*/
}


public class Circle extends Shape {
  private final double radius;
  // Constructor omitted
  public double area() {
    return Math.PI * Math.pow(radius, 2);
  }
  public double perimeter() {
    return 2 * Math.PI * radius;
  }
}
```



```
public class Rectangle extends Shape {
  private final double width, length;
  // Constructor omitted
  public double area() {
    return width * length;
  }
  public double perimeter() {
    return 2 * (width + length);
  }
  public void draw(int x, int y) {
  // rectangle at (x, y, width, length)
  }
  public void draw(Point p) {
  // rectangle at (p.x, p.y, width, length)
  }
}
```

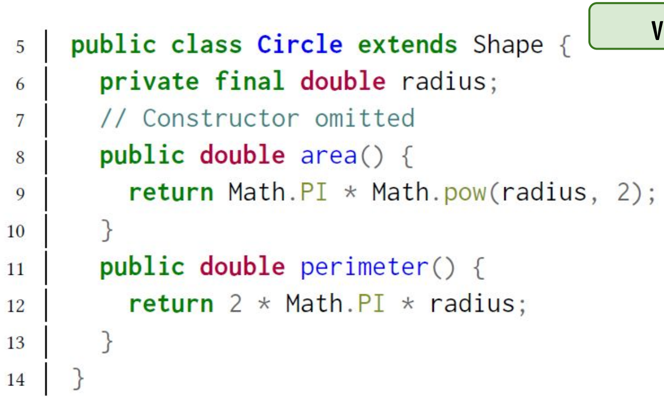
Variation points and variants

vp_shape



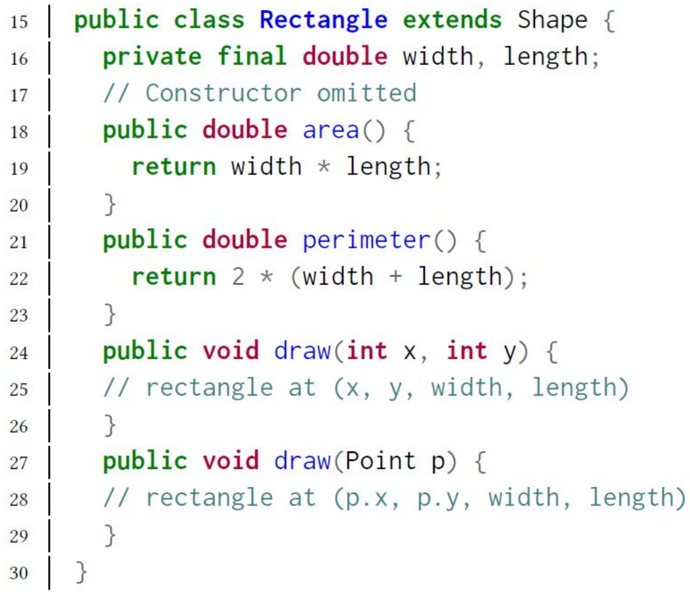
```java
public abstract class Shape {
  public abstract double area();
  public abstract double perimeter(); /*...*/
}
```

v_circle



```java
public class Circle extends Shape {
  private final double radius;
  // Constructor omitted
  public double area() {
    return Math.PI * Math.pow(radius, 2);
  }
  public double perimeter() {
    return 2 * Math.PI * radius;
  }
}
```

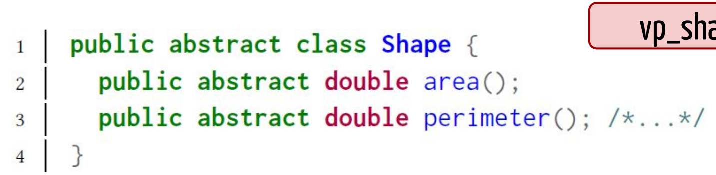
v_rectangle



```java
public class Rectangle extends Shape {
  private final double width, length;
  // Constructor omitted
  public double area() {
    return width * length;
  }
  public double perimeter() {
    return 2 * (width + length);
  }
  public void draw(int x, int y) {
  // rectangle at (x, y, width, length)
  }
  public void draw(Point p) {
  // rectangle at (p.x, p.y, width, length)
  }
}
```

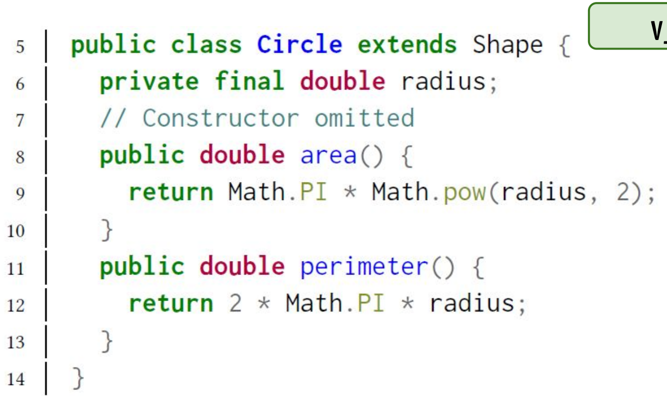
Variation points and variants

vp_shape



```
public abstract class Shape {
  public abstract double area();
  public abstract double perimeter(); /*...*/
}
```

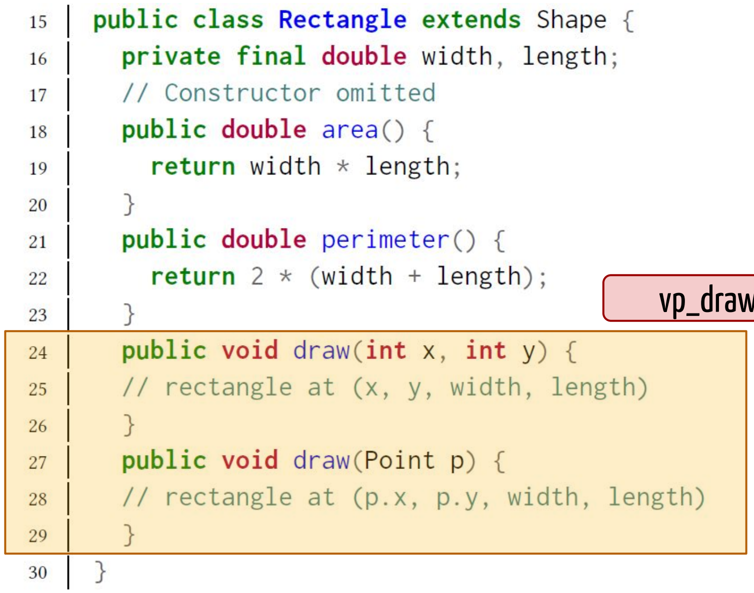
v_circle



```
public class Circle extends Shape {
  private final double radius;
  // Constructor omitted
  public double area() {
    return Math.PI * Math.pow(radius, 2);
  }
  public double perimeter() {
    return 2 * Math.PI * radius;
  }
}
```

v_rectangle

vp_draw



```
public class Rectangle extends Shape {
  private final double width, length;
  // Constructor omitted
  public double area() {
    return width * length;
  }
  public double perimeter() {
    return 2 * (width + length);
  }
  public void draw(int x, int y) {
  // rectangle at (x, y, width, length)
  }
  public void draw(Point p) {
  // rectangle at (p.x, p.y, width, length)
  }
}
```

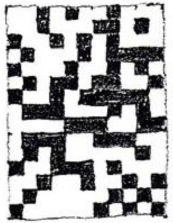
The theory of centres and the notion of symmetry [Alexander2002]

Centre: a field of organized force in an object or part of an object which makes that object or part exhibit centrality.

A centre is commonly formed by a local symmetry.

⇒ The centre is the common part of the symmetric variants.

Random
→ hard to describe



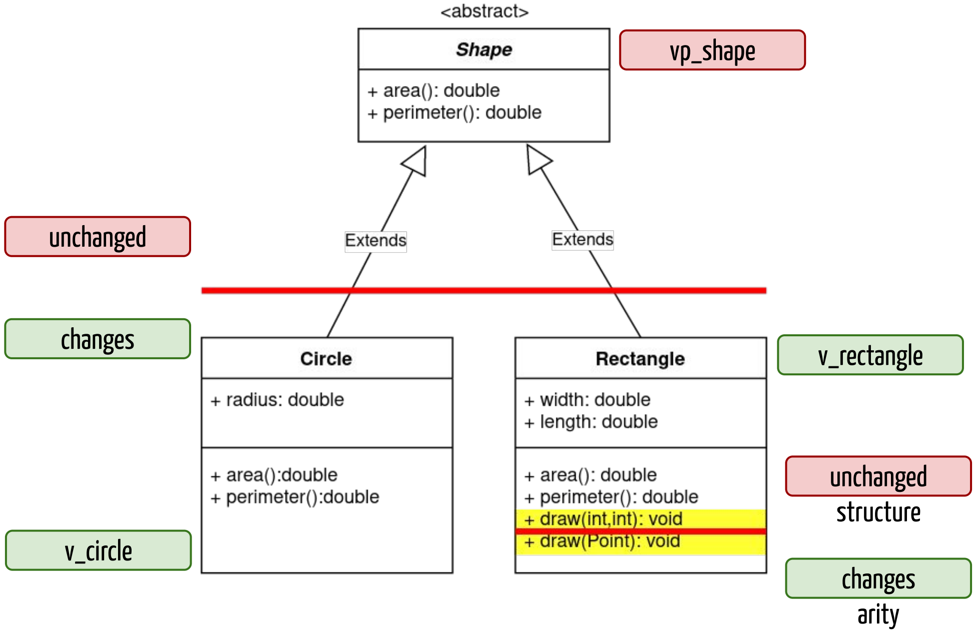
Ordered around a
centre of symmetry
→ easy to describe

Use of symmetries to detect variability implementations?

Intuition:

- Presence of **symmetries in object-oriented codebases** [Coplien2019] inspired from the theory of centres

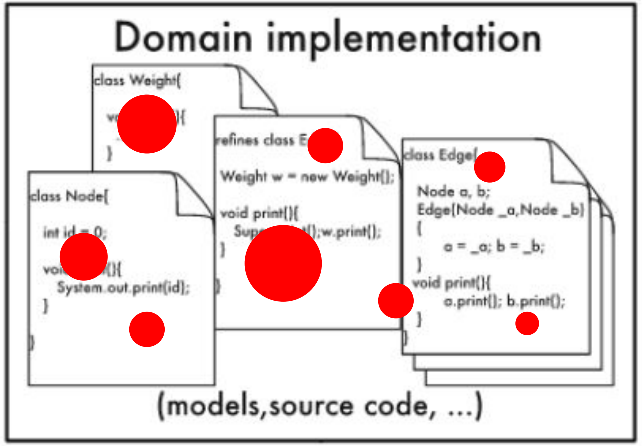
- Symmetries present in **mechanisms implementing variability**



Identifying variation points with variants

Variability implementation technique ⟷ local symmetry

- variation point (commonality) ⟷ unchanged
- variant (variability) ⟷ changes

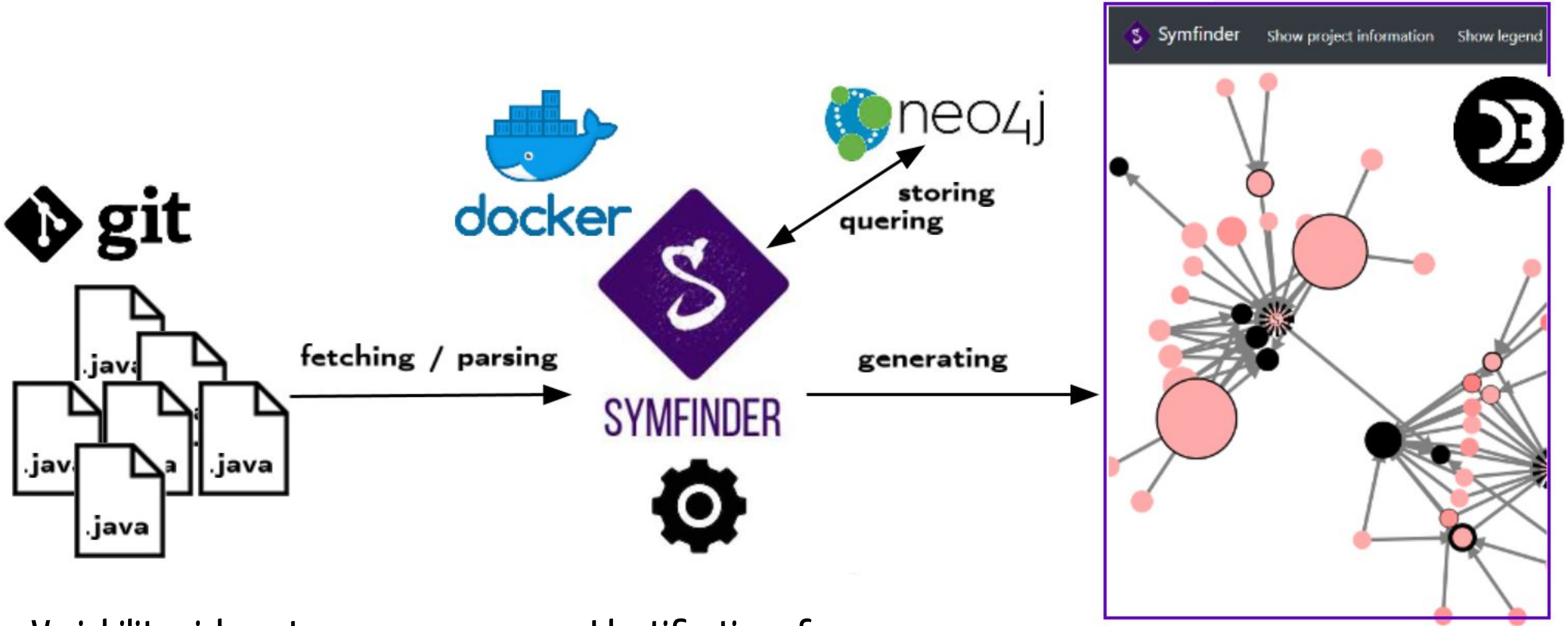


Identification through local symmetries in core assets

High density of symmetries ⟶ variability intense places

Xhevahire Tërnava, Johann Mortara, and Philippe Collet. 2019. Identifying and Visualizing Variability in Object-Oriented Variability-Rich Systems. In 23rd International Systems and Software Product Line Conference - Volume A (SPLC '19), September 9–13, 2019, Paris, France. ACM, New York, NY, USA, 12 pages.

symfinder



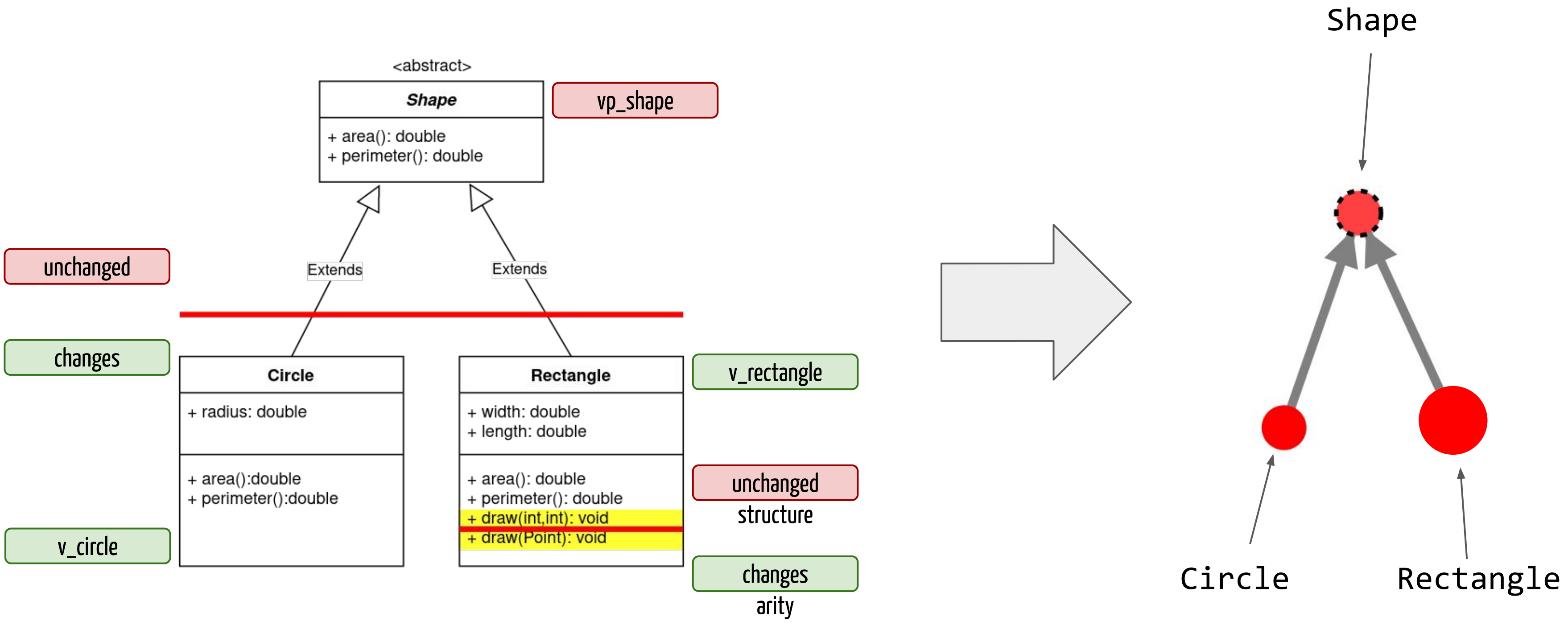
Variability-rich system in a <u>single</u> code base

Identification of symmetries

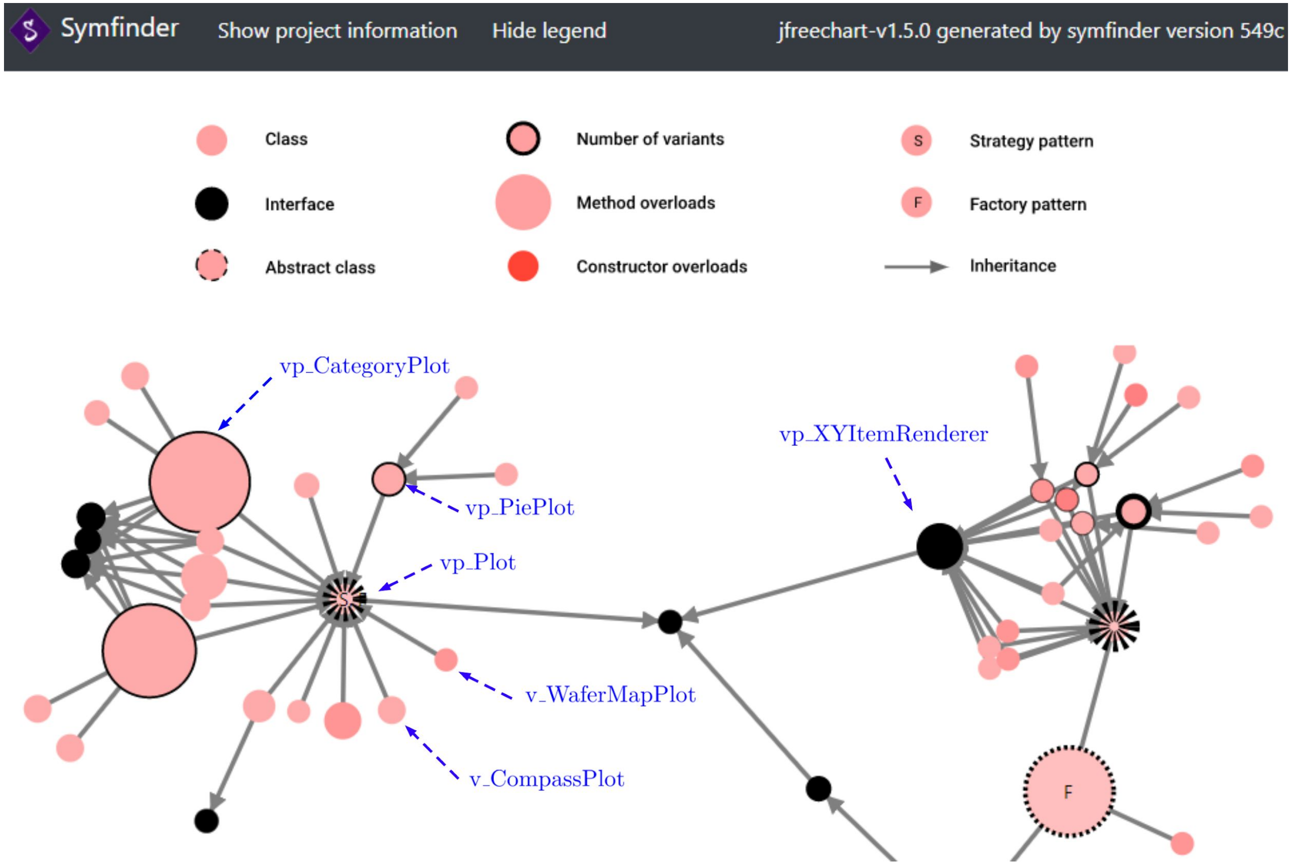
Visualization of *vp-s*

Johann Mortara, Xhevahire Tërnava, and Philippe Collet. 2019. symfinder: A Toolchain for the Identification and Visualization of Object-Oriented Variability Implementations. In 23rd International Systems and Software Product Line Conference - Volume B (SPLC '19), September 9–13, 2019, Paris, France. ACM, New York, NY, USA, 6 pages.

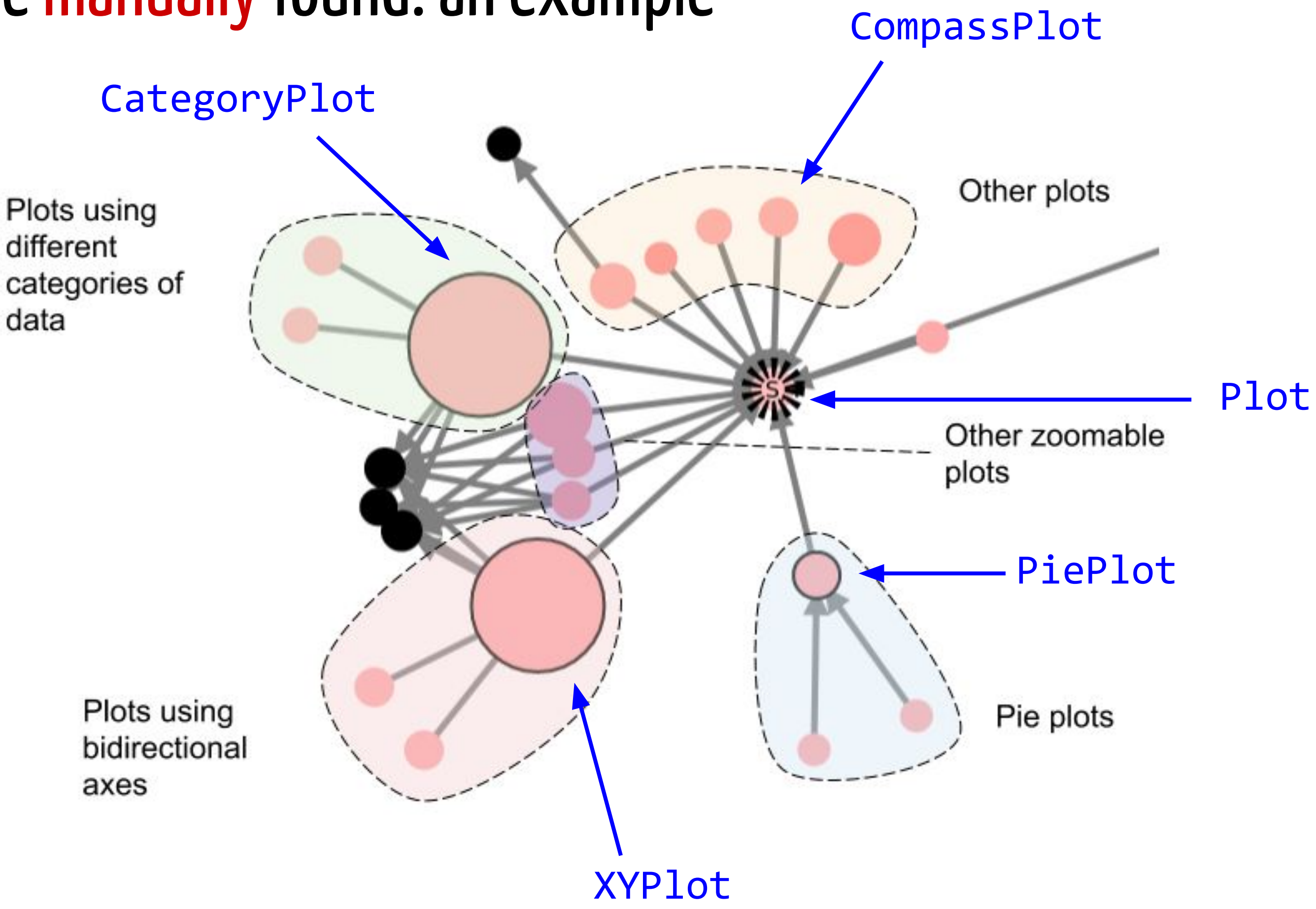
Visualizing a small example



Automatic visualization of *vp-s* with variants



What can be manually found: an example



Subject system	Analysed LoC	#*vp*-s	#variants
Java AWT	69,974	1,221	1,808
Apache CXF 3.2.7	48,655	7,468	9,201
JUnit 4.12	9,317	253	319
Apache Maven 3.6.0	105,342	1,443	1,393
JHipster 2.0.28	2,535	140	115
JFreeChart 1.5.0	94,384	1,415	2,103
JavaGeom	32,755	720	919
ArgoUML	178,906	2,451	3,079

Visualisations et résultats disponibles ici : https://deathstar3.github.io/symfinder-demo/splc2019.html

Synthesis

Goals reached:

✓ Definition of vp-s with variants in implementation relying on the notion of symmetry

✓ Toolchain for automatic identification

✓ Some vp-s and variants can be visually mapped to domain features

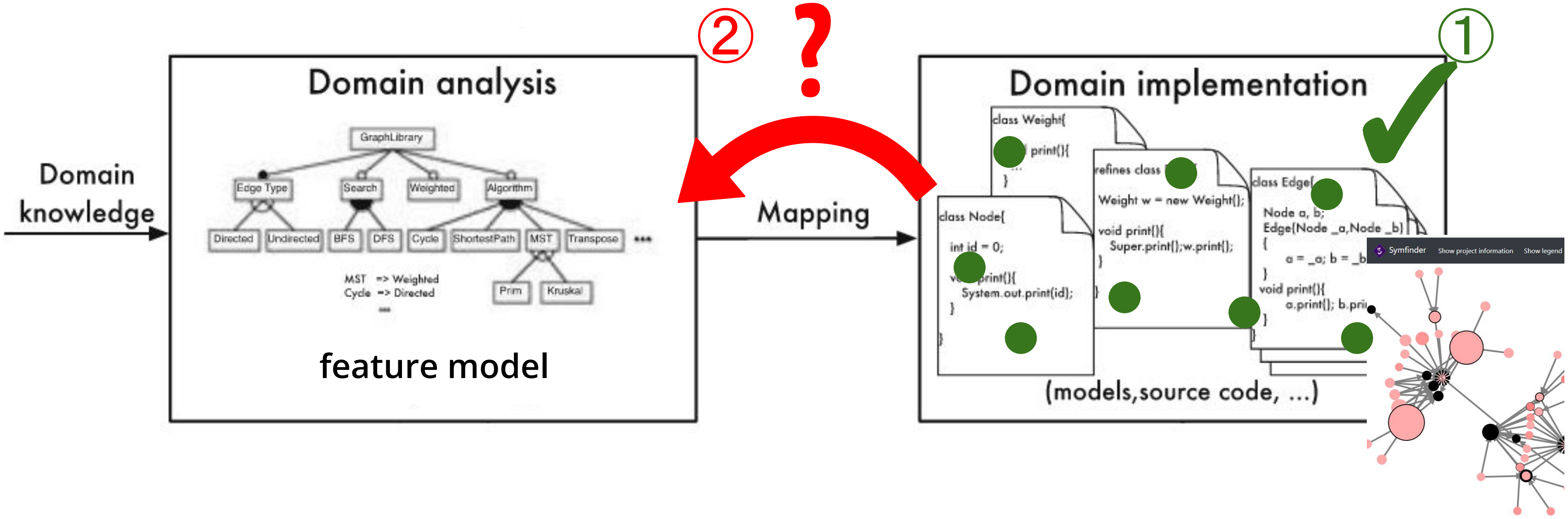
Next step:

Are the identified vp-s with variants valuable?

- Need to measure the quality of our identification method.

Problem 1: How to identify variability implementations in an existing OO codebase?

Problem 2: How to map these variability implementations to domain features?



```
class Weight{
  print(){
  ...
}
```

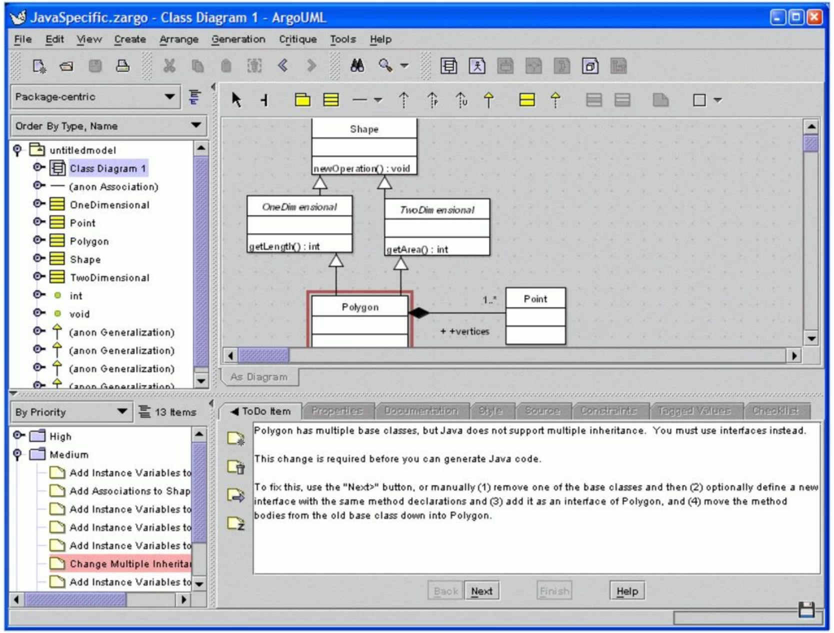
```
refines class
Weight w = new Weight();
void print(){
  Super.print();w.print();
}
}
```

```
class Node{
  int id = 0;
  v  print(){
    System.out.print(id);
  }
}
```

```
class Edge{
  Node a, b;
  Edge(Node _a,Node _b)
  {
    a = _a; b = _b
  }
  void print(){
    a.print(); b.pri
  }
}
```

Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

ArgoUML-SPL [Couto2011]

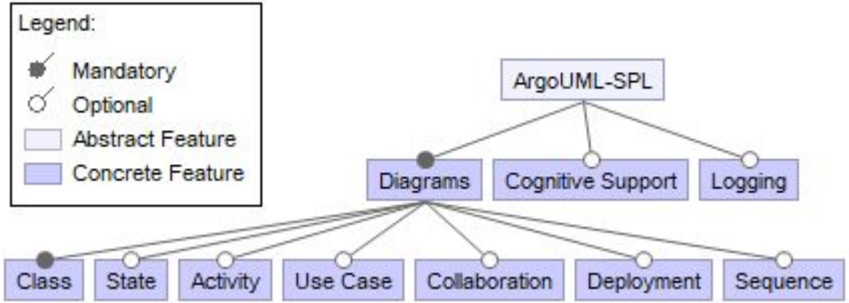


Polygon has multiple base classes, but Java does not support multiple inheritance. You must use interfaces instead.

This change is required before you can generate Java code.

To fix this, use the "Next>" button, or manually (1) remove one of the base classes and then (2) optionally define a new interface with the same method declarations and (3) add it as an interface of Polygon, and (4) move the method bodies from the old base class down into Polygon.

ArgoUML editor

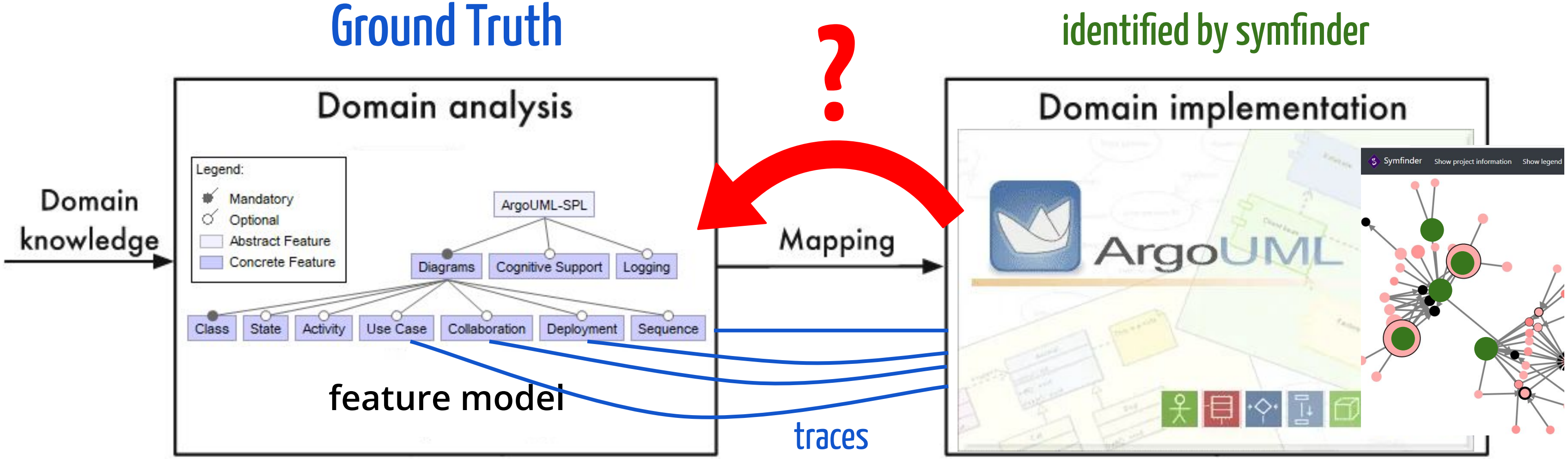


Feature model of ArgoUML-SPL

Question: Are the identified *vp-s* from ArgoUML relevant for a feature mapping?

Ground Truth

vp-s automatically identified by symfinder


Domain knowledge

Domain analysis

Legend:

- Mandatory
- Optional
- Abstract Feature
- Concrete Feature

feature model

?

Mapping

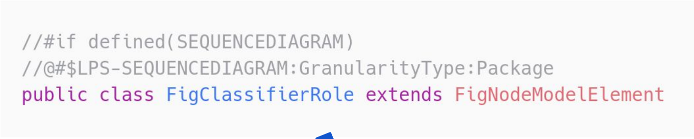
traces

Domain implementation

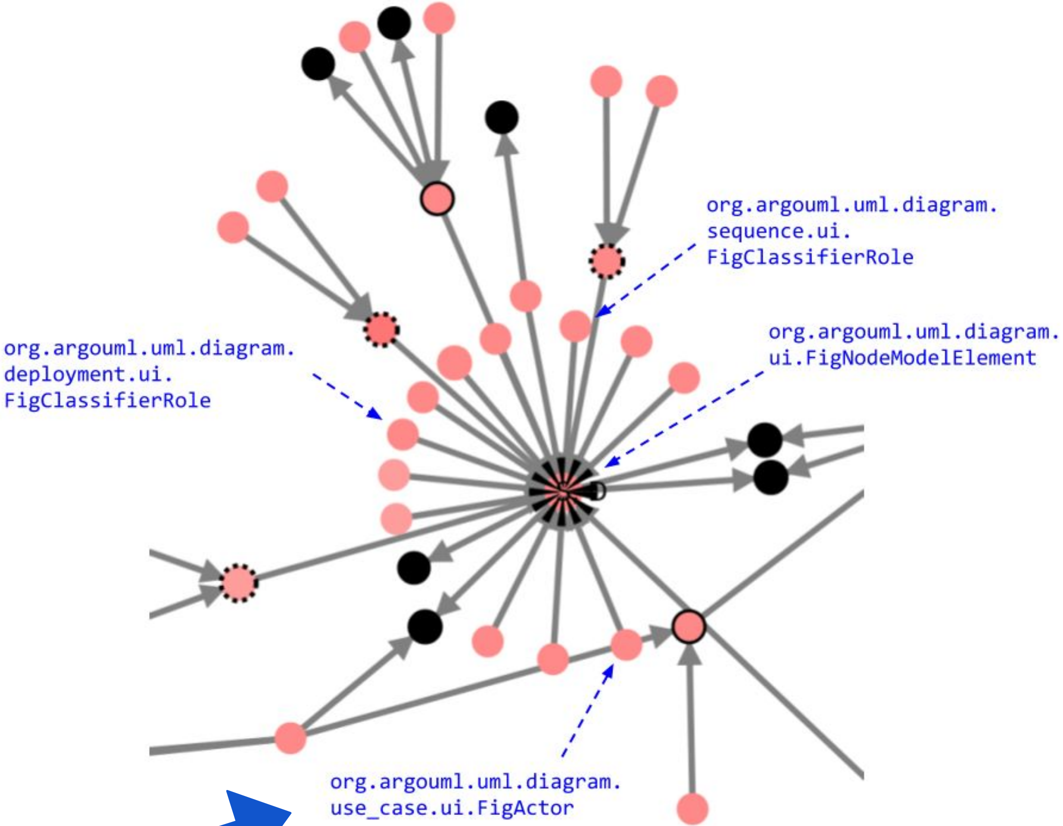
Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

Example on ArgoUML-SPL

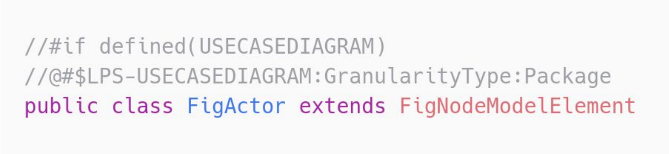
Feature: Sequence



```
//#if defined(SEQUENCEDIAGRAM)
//@#$LPS-SEQUENCEDIAGRAM:GranularityType:Package
public class FigClassifierRole extends FigNodeModelElement
```

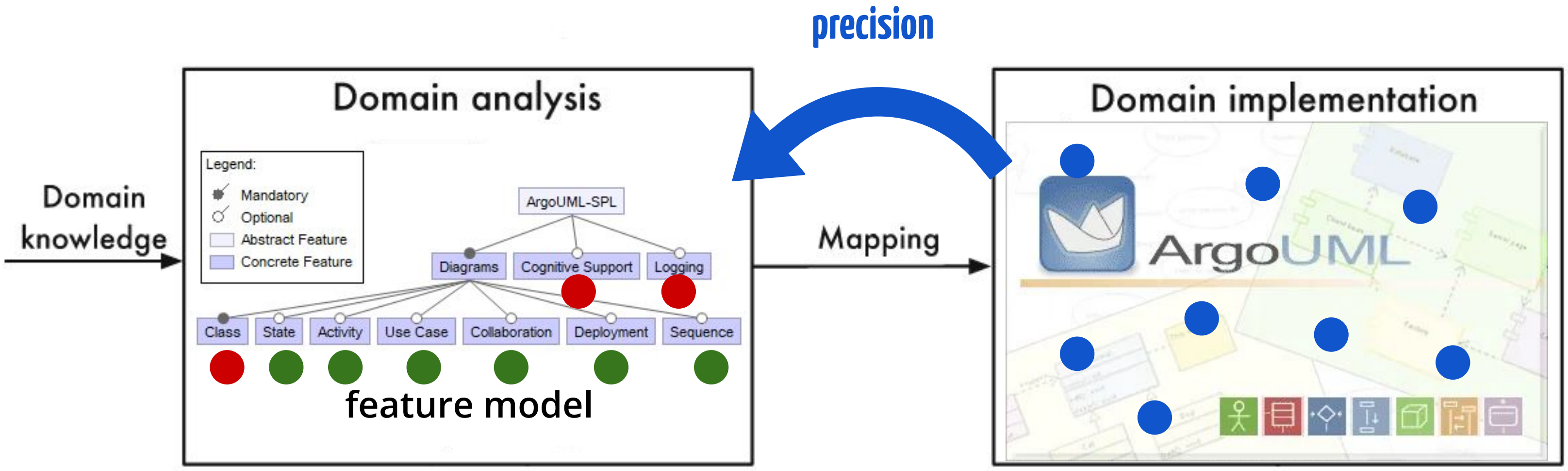


Feature: Use Case



```
//#if defined(USECASEDIAGRAM)
//@#$LPS-USECASEDIAGRAM:GranularityType:Package
public class FigActor extends FigNodeModelElement
```

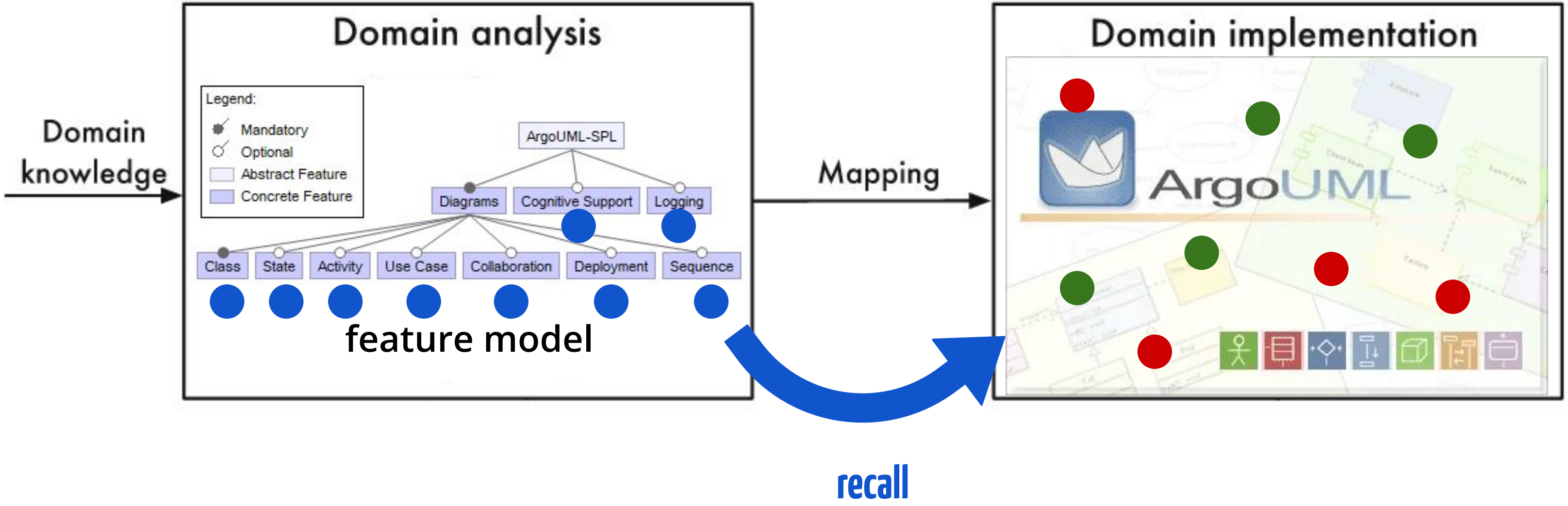
Relevance of the *vp-s*



Precision:

Percentage of identified vp-s and variants that could be mapped to domain features

Relevance of the *vp-s*



Recall:

Percentage of features' traces that could be mapped to identified vp-s and variants

Relevance of the *vp-s*

Calculating precision

$$precision = \frac{TP}{TP + FP} = \frac{|T_{gt} \cap I_{vp-v}|}{|I_{vp-v}|} = \frac{593}{1560} = 38\%$$

Low precision was **expected**:

- coarse grain features based on superficial domain knowledge
- not all identified places with a symmetry are related to variability

⇒ need for a more precise identification

Relevance of the *vp-s*

Calculating recall

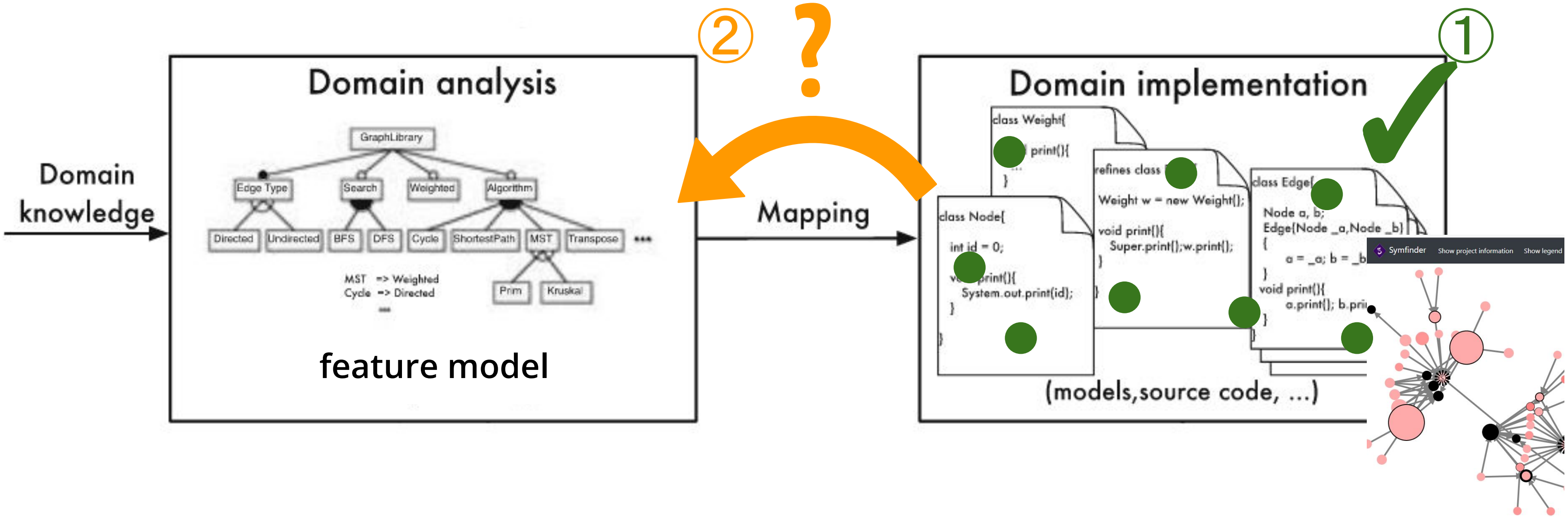
$$recall = \frac{TP}{TP + FN} = \frac{|T_{gt} \cap I_{vp-v}|}{|T_{gt}|} = \frac{593}{712} = 83\%$$

The missing 17% of traces are **not variability related**:

- initialization classes
- external libraries

Problem 1: How to identify variability implementations in an existing OO codebase?

Problem 2: How to map these variability implementations to domain features?



```
class Weight{
    print(){
    ...
  }
```

```
class Node{
  int id = 0;
  v   print(){
    System.out.print(id);
  }
}
```

```
refines class
 Weight w = new Weight();
 void print(){
   Super.print();w.print();
 }
}
```

```
class Edge{
  Node a, b;
  Edge(Node _a,Node _b)
  {
      a = _a; b = _b
  }
 void print(){
      a.print(); b.pri
 }
}
```

Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

Future work

- Formal definition of density to increase the precision
- Map variation points and variant to preprocessor directives
- Analyse multi-components systems and systems of systems
- Ongoing experiment on Sat4j’s codebase with Daniel Le Berre

Automatic identification of object-oriented variability implementations

Johann Mortara – Philippe Collet

✓ Definition of vp-s in implementation relying on the notion of symmetry

✓ Automatic identification and visualization of vp-s and variants, exhibiting zones of high density of symmetries

✓ First mapping shows that some identified vp-s with variants are relevant for feature mapping

⇒ Need for a more precise detection method

References

[Acher2018] Mathieu Acher. Software Variability and Artificial Intelligence. Ecole d'été du GDR GPL - EJCP 2018 https://ejcp2018.sciencesconf.org/file/441457

[Alexander2002] Christopher Alexander. 2002. The nature of order: an essay on the art of building and the nature of the universe. Book 1, The phenomenon of life. Center for Environmental Structure.

[Assunção2017] Wesley KG Assunção, Roberto E Lopez-Herrejon, Lukas Linsbauer, Silvia R Vergilio and Alexander Egyed. 2017. Reengineering legacy applications into software product lines: a systematic mapping. Empirical Software Engineering 22, 6 (2017), 2972–3016.

[Coplien2019] James O. Coplien and Liping Zhao. 2019. Toward a general formal foundation of design. Symmetry and broken symmetry. Technical Report. A VUB Lecture Series Publication. Working draft.

[Couto2011] Marcus Vinicius Couto, Marco Tulio Valente, and Eduardo Figueiredo. Extracting Software Product Lines: A Case Study Using Conditional Compilation. In 15th European Conference on Software Maintenance and Reengineering (CSMR), pages 191-200, 2011.

[Liebig2010] Jörg Liebig, Sven Apel, Christian Lengauer, Christian Kästner and Michael Schulze. 2010. An analysis of the variability in forty preprocessor-based software product lines. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. ACM, 105–114.

[Lozano2011] Angela Lozano. 2011. An overview of techniques for detecting software variability concepts in source code. In International Conference on Conceptual Modeling. Springer, 141–150.

[Metzger2014] Andreas Metzger and Klaus Pohl. 2014. Software product line engineering and variability management: achievements and challenges. In Proceedings of the on Future of Software Engineering (FOSE 2014). ACM, New York, NY, USA, 70-84. DOI: http://dx.doi.org/10.1145/2593882.2593888

[Open2015]OpenSignal. Android Fragmentation Report. August 2015 https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015-08/2015_08_fragmentation_report.pdf

[Tërnava2019] Xhevahire Tërnava, Johann Mortara, and Philippe Collet. 2019. Identifying and Visualizing Variability in Object-Oriented Variability-Rich Systems. In 23rd International Systems and Software Product Line Conference - Volume A (SPLC '19), September 9–13, 2019, Paris, France. ACM, New York, NY, USA, 12 pages.

[Tërnava2018] Xhevahire Tërnava and Philippe Collet. Identifying Variability Implementations with Local Symmetries. unpublished tech report. 2018.

[Tërnava2017] Xhevahire Tërnava and Philippe Collet. 2017. On the Diversity of Capturing Variability at the Implementation Level. In Proceedings of the 21st International Systems and Software Product Line Conference-Volume B. ACM, 81–88.