Mapping Features to Automatically Identified Object-Oriented Variability Implementations

The case of ArgoUML-SPL

Johann Mortara [1], Xhevahire Tërnava [2], Philippe Collet [1]

[1] Université Côte d'Azur, CNRS, I3S, France
[2] Sorbonne Université, Paris, France

VaMoS '20, Magdeburg – February 6, 2020

Variability-Rich Systems with a Single Code Base

16.000 options managed in 25M LoC [Acher2018]

#ifdef

24.000 different platforms in 2015 [Open2015]

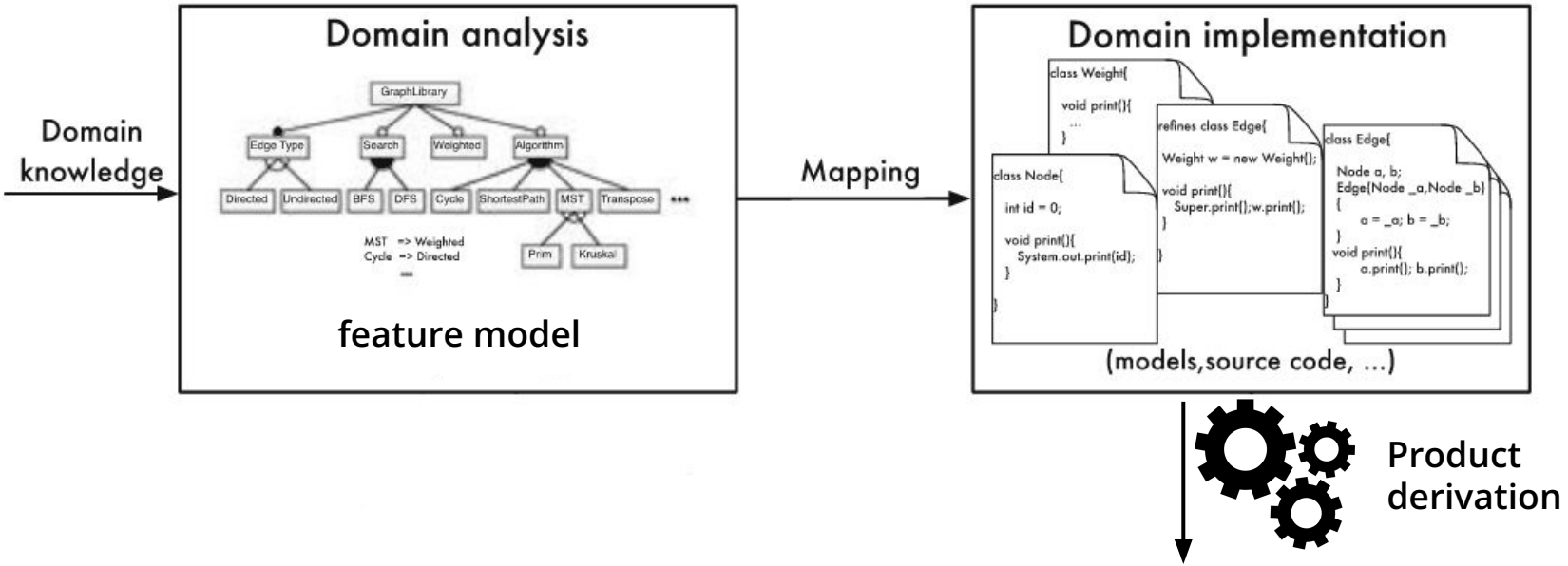
Object-orientation

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

Object-orientation

and many variability implementation techniques...

Problem: How to master them as SPL?



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

How to engineer an SPL?

Forward-engineering:

Feature model ⟶ Domain implementation

Mapping between feature model and features is done **during the implementation**

How to engineer an SPL?

Forward-engineering:

Feature model ⟶ Domain implementation

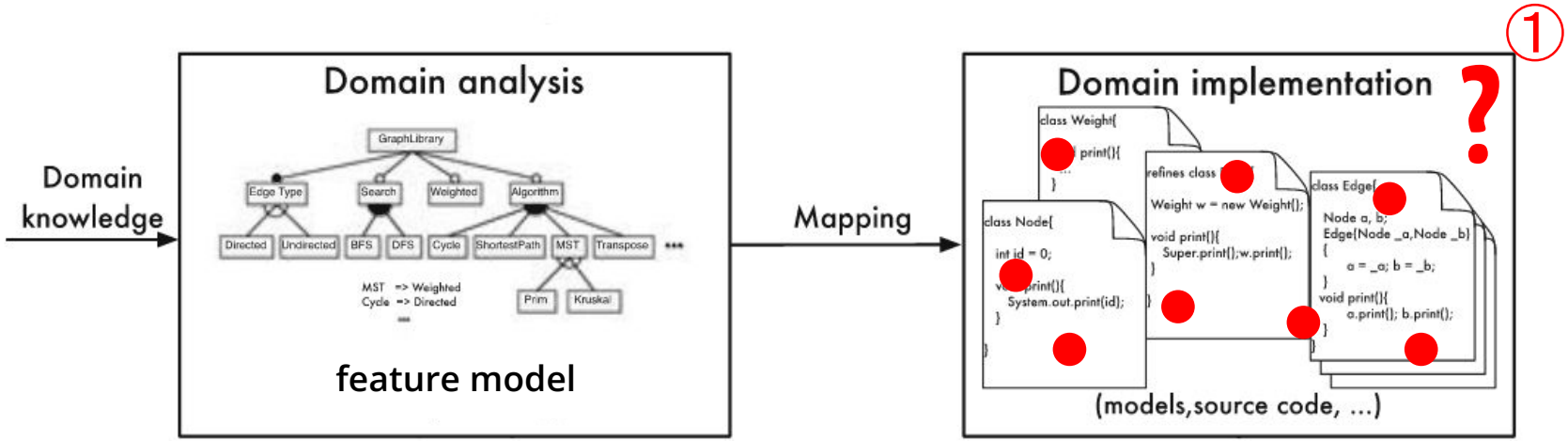
Mapping between feature model and features is done **during the implementation**

Reverse-engineering:

Feature model ⟵ Domain implementation

Need to extract the features and build a mapping with the feature model, or build it

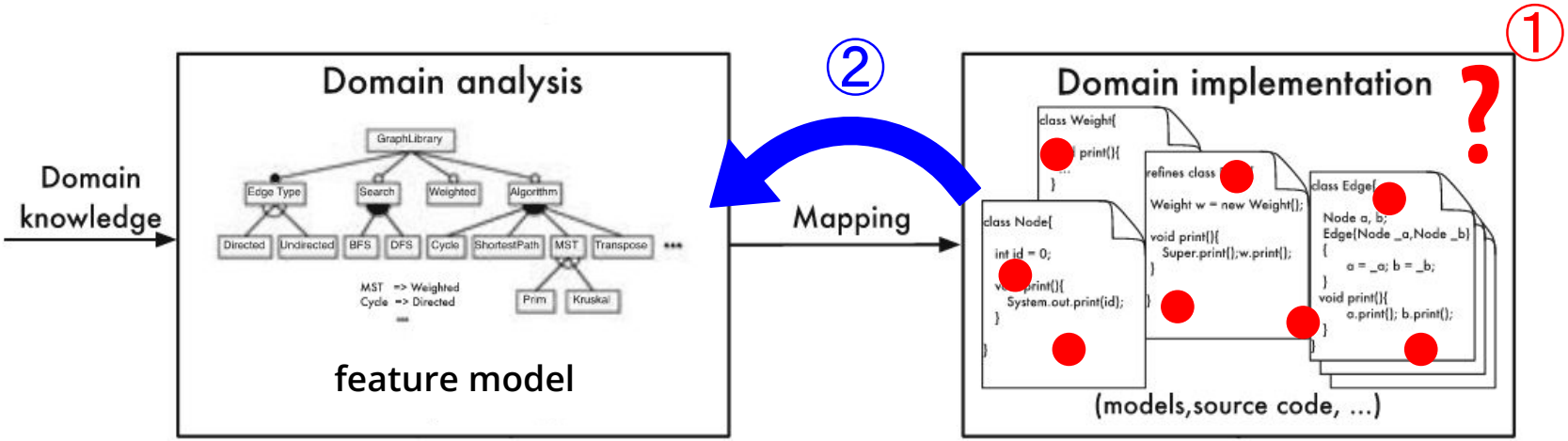
Problem 1: How to identify variability implementations in an existing codebase?



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

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

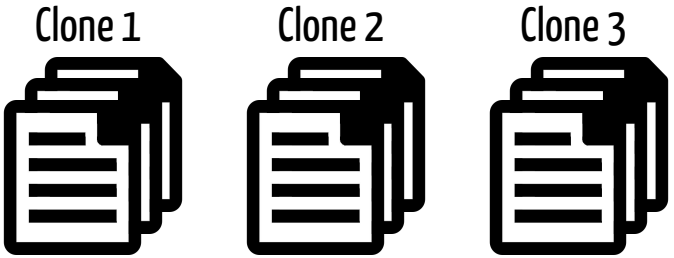
Problem 2: How to map these variability implementations to domain features?



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

State of the art on variability implementations detection

Context: projects clones

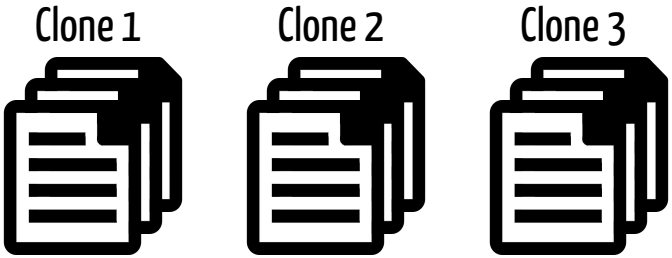


Detection method:

Comparison between clones and mapping with the domain features [Wesley2017]

State of the art on variability implementations detection

Context: projects clones

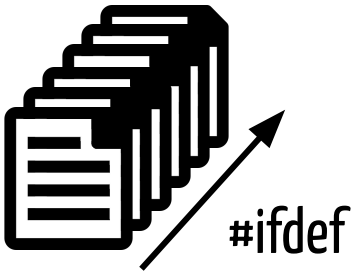


Detection method:

Comparison between clones and mapping with the domain features [Wesley2017]

Context: unique codebase and preprocessing directives

#ifdef ⟶ variant



Detection method:

Determining the consistency of directives [Liebig2010]

State of the art on variability implementations detection

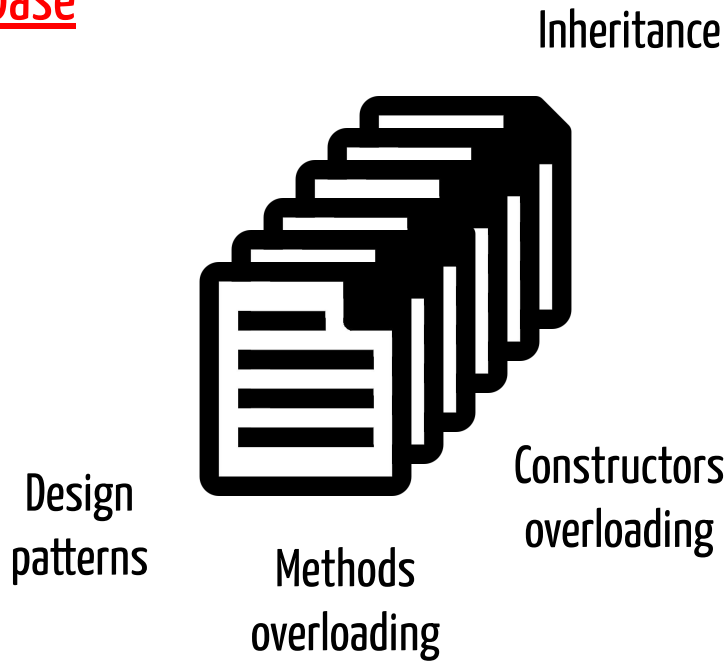
Our context: large and unique object-oriented codebase

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

Detection method:

Currently no method

[Lozano2011], [Metzger2014], [Tërnava2017]



Variation points and variants

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

```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;
  }
}
```

```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

```
1  public abstract class Shape {
2    public abstract double area();
3    public abstract double perimeter(); /*...*/
4  }
```

v_circle

```
5  public class Circle extends Shape {
6    private final double radius;
7    // Constructor omitted
8    public double area() {
9      return Math.PI * Math.pow(radius, 2);
10   }
11   public double perimeter() {
12     return 2 * Math.PI * radius;
13   }
14 }
```

v_rectangle

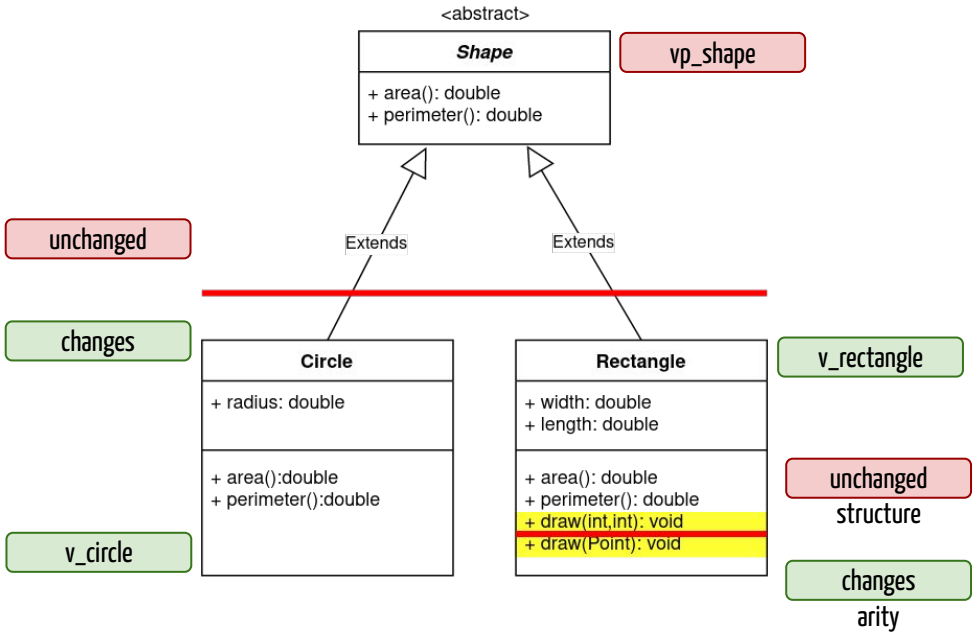
```
15 public class Rectangle extends Shape {
16   private final double width, length;
17   // Constructor omitted
18   public double area() {
19     return width * length;
20   }
21   public double perimeter() {
22     return 2 * (width + length);
23   }
24   public void draw(int x, int y) {
25   // rectangle at (x, y, width, length)
26   }
27   public void draw(Point p) {
28   // rectangle at (p.x, p.y, width, length)
29   }
30 }
```

vp_draw

Use of symmetries to detect variability implementations?

Intuition:

- Presence of **symmetries in object-oriented codebases** [Coplien2019] inspired from the theory of centres [Alexander2002]
- Symmetries present in **mechanisms implementing variability**



Identifying variation points with variants

Variability implementation technique ⟷ local symmetry

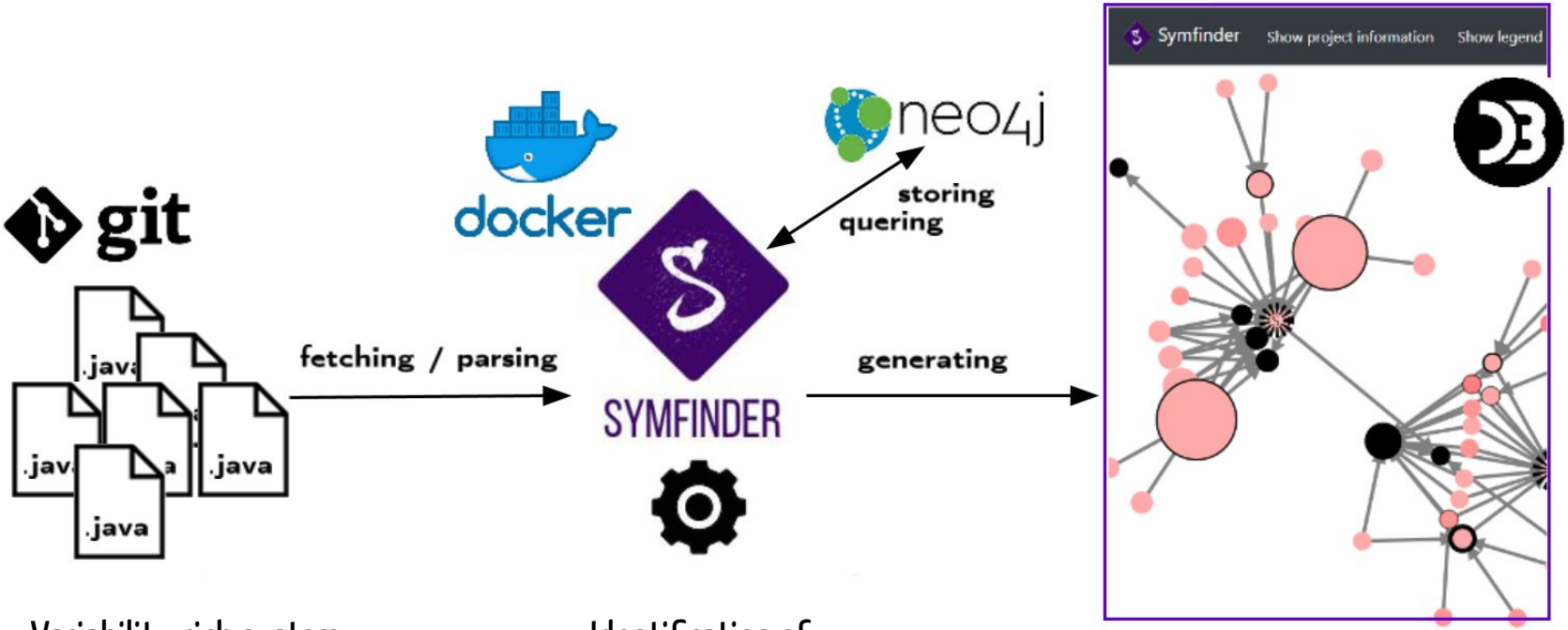
- variation point (commonality) ⟷ unchanged
- variant (variability) ⟷ changes

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



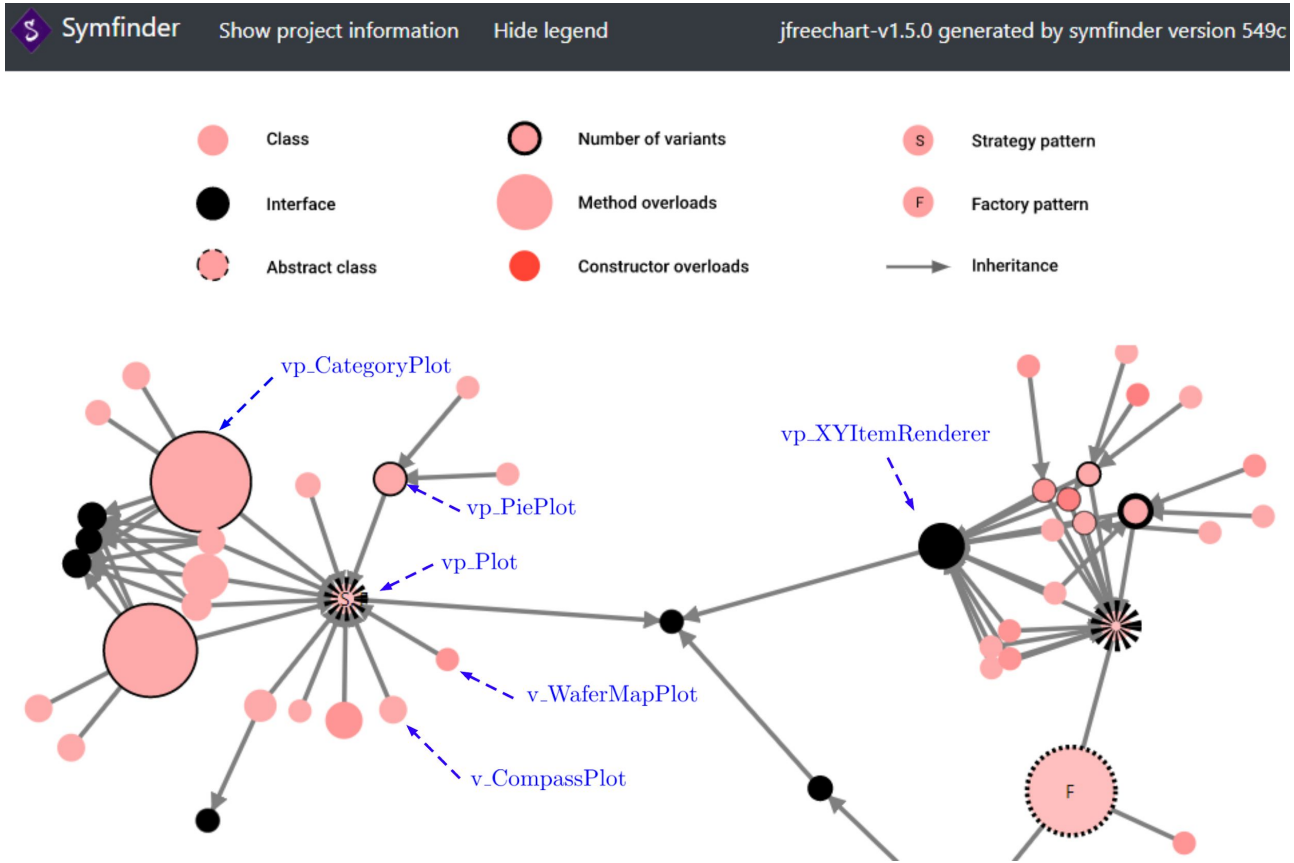
Variability-rich system in a single code base

Identification of symmetries

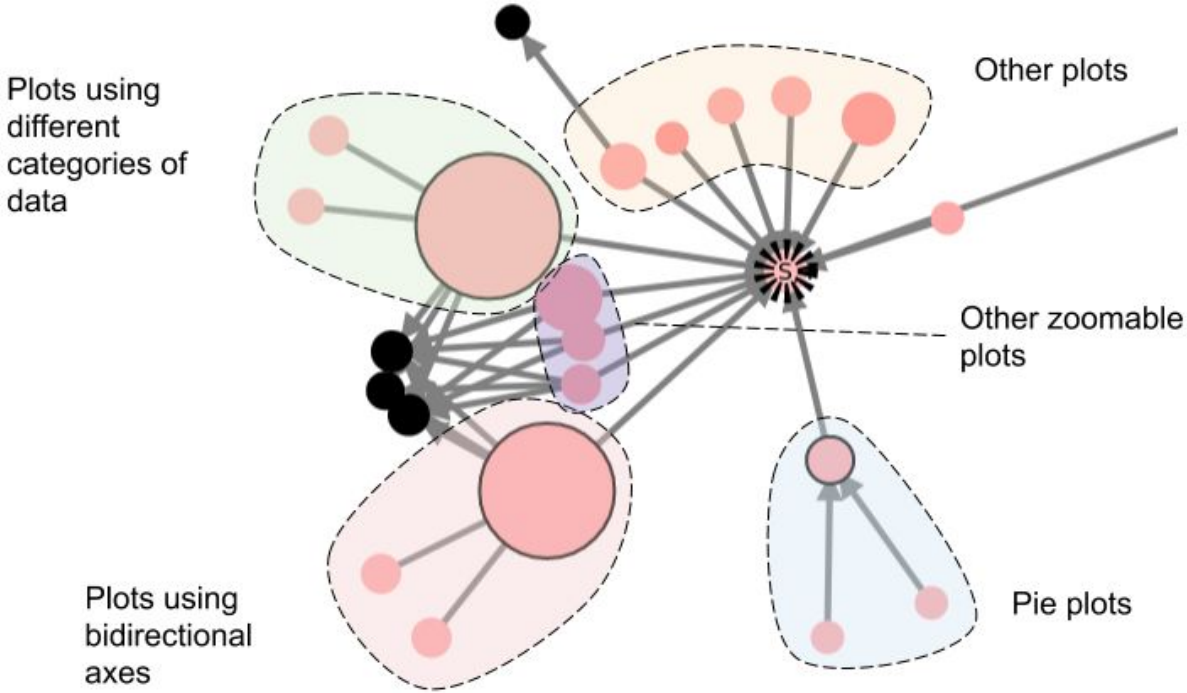
Visualization of *vp-s*

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.

Automatic visualization of *vp-s* with variants

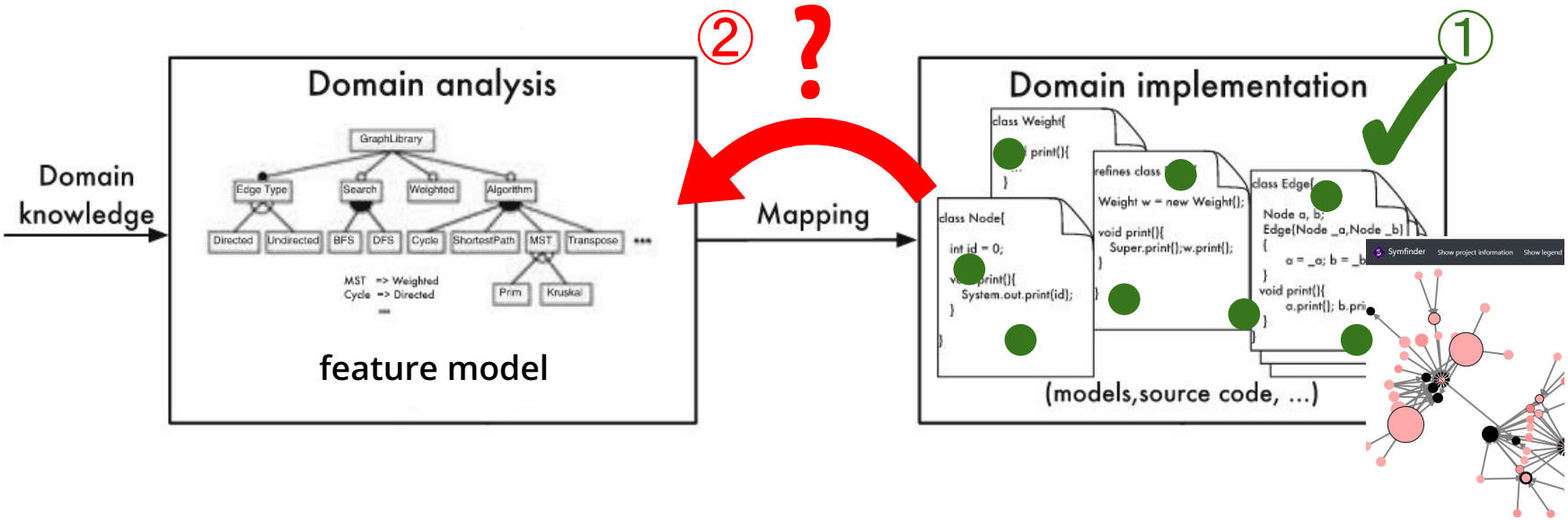


What can be manually found: an example



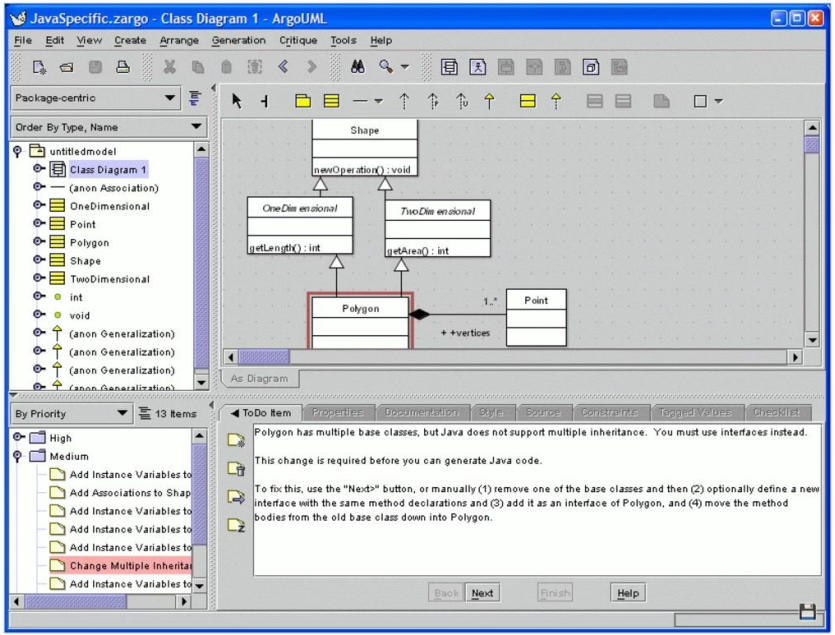
Problem 1: How to identify variability implementations in an existing codebase?

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

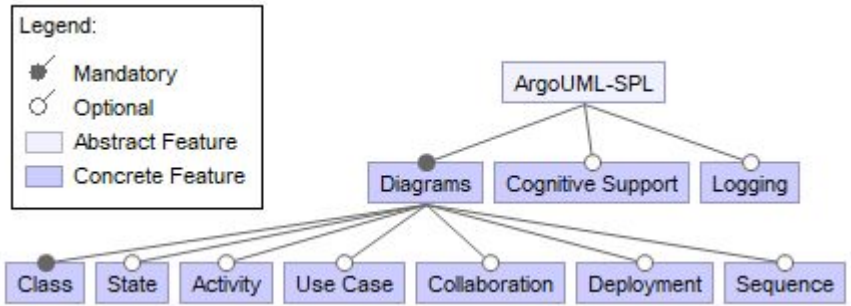


Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

ArgoUML-SPL [Couto2011]

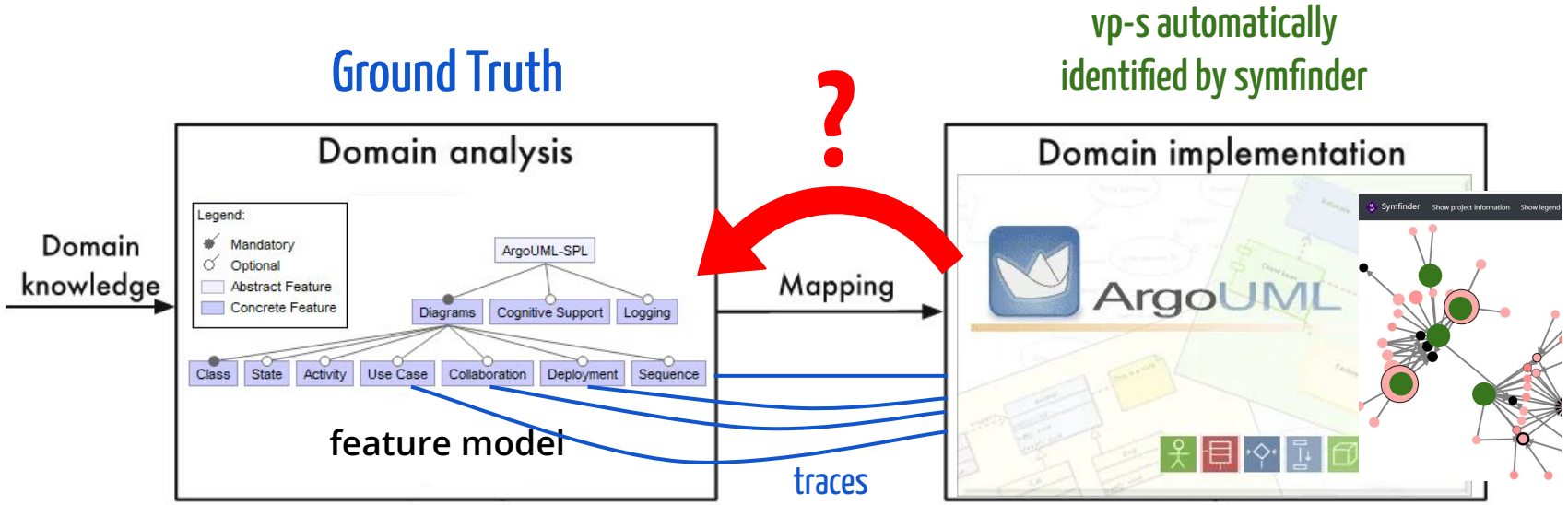


ArgoUML editor



Feature model of ArgoUML-SPL

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



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

Experimental setup

Ground Truth

Excerpt of traces for USECASE feature

org.argouml.uml.diagram.use_case.ui.FigActor

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

org.argouml.uml.diagram.use_case.ui.FigClassifierRole

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

.

.

.

Experimental setup

Ground Truth

Excerpt of traces for USECASE feature

org.argouml.uml.diagram.use_case.ui.FigActor

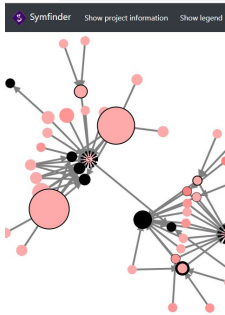
```
//#if defined(USECASEDIAGRAM)
//@#$LPS-USECASEDIAGRAM:GranularityType:Package
public class FigActor extends FigNodeModelElement
```

org.argouml.uml.diagram.use_case.ui.FigClassifierRole

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

.
.
.

Excerpt of *symfinder* JSON output




```
{
  "nodes": [
    {
      "types": [
        "CLASS", "METHOD_LEVEL_VP", "VARIANT"
      ],
      "constructorVPs": 1,
      "methodVariants": 0,
      "classVariants": 0,
      "methodVPs": 0,
      "constructorVariants": 3,
      "name":
"org.argouml.uml.diagram.use_case.ui.FigActor"
    },...
  ],
  "links": [
    {
      "type": "EXTENDS",
      "source":
"org.argouml.uml.diagram.ui.FigNodeModelElement",
      "target":
"org.argouml.uml.diagram.use_case.ui.FigActor"
    },...
  ]
}
```

Experimental setup

Ground Truth

Excerpt of traces for USECASE feature

org.argouml.uml.diagram.use_case.ui.FigActor

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

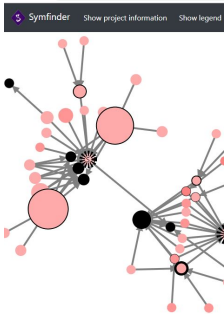
org.argouml.uml.diagram.use_case.ui.FigClassifierRole

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

.
.
.

Manual mapping using Excel formulae

Excerpt of *symfinder* JSON output




```
{
  "nodes": [
    {
      "types": [
        "CLASS", "METHOD_LEVEL_VP", "VARIANT"
      ],
      "constructorVPs": 1,
      "methodVariants": 0,
      "classVariants": 0,
      "methodVPs": 0,
      "constructorVariants": 3,
      "name":
"org.argouml.uml.diagram.use_case.ui.FigActor"
    },...
  ],
  "links": [
    {
      "type": "EXTENDS",
      "source":
"org.argouml.uml.diagram.ui.FigNodeModelElement",
      "target":
"org.argouml.uml.diagram.use_case.ui.FigActor"
    },...
  ]
}
```

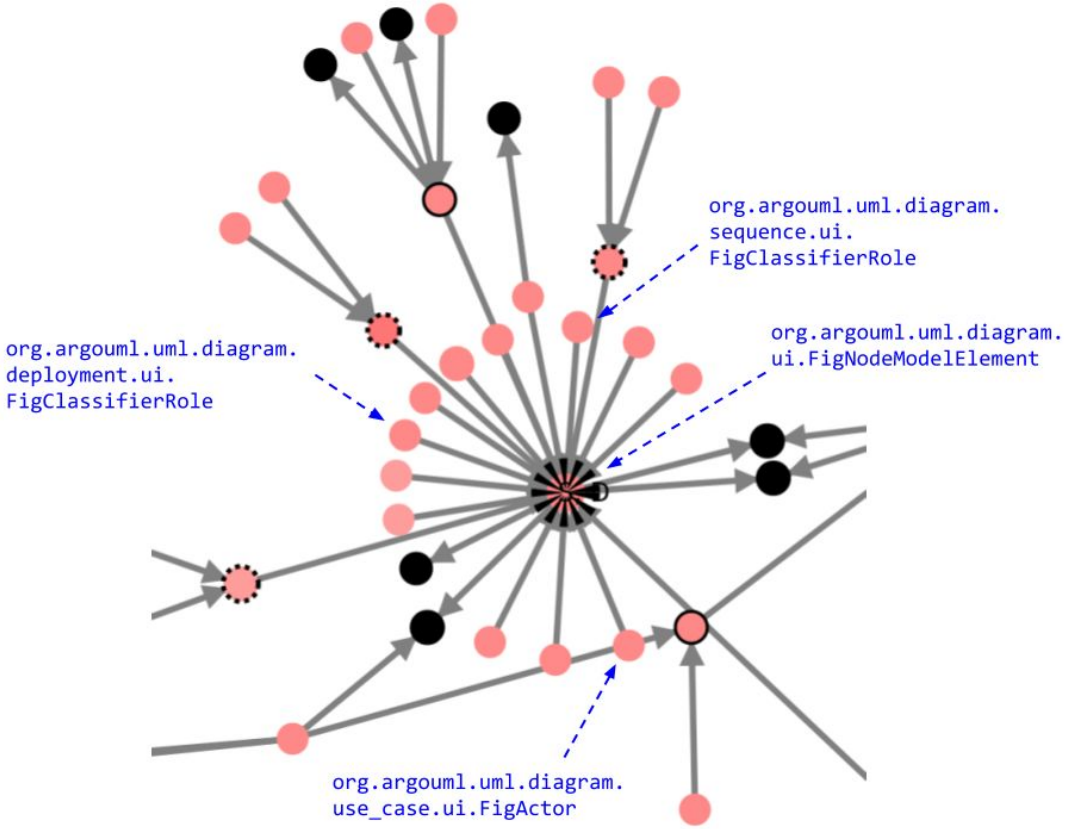
Validation

Feature: Use Case

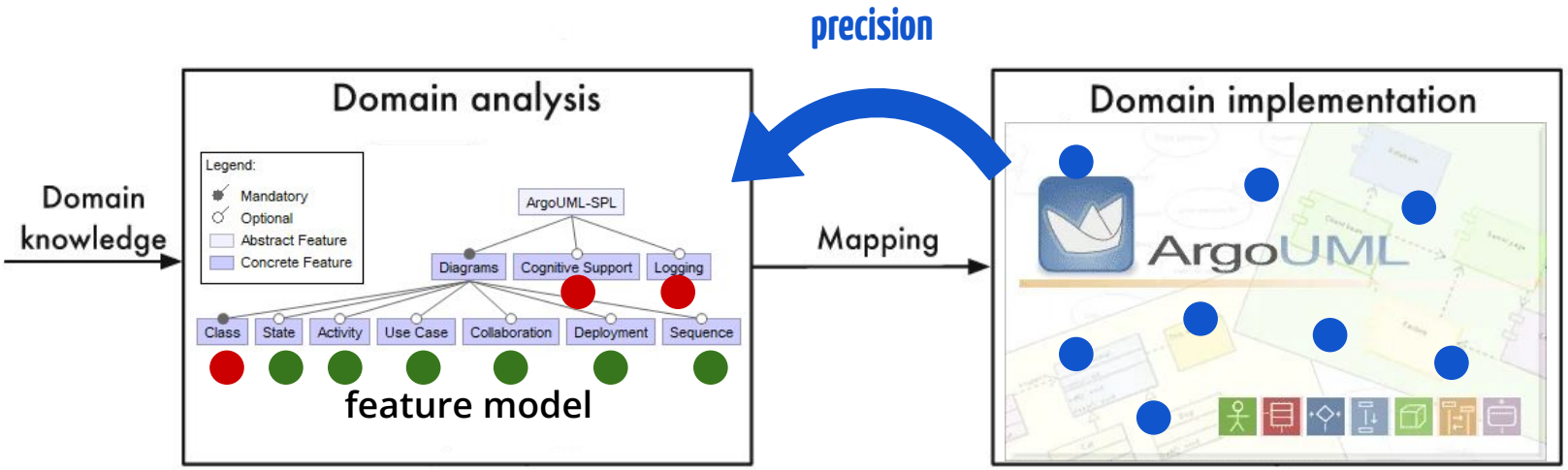
```
//#if defined(USECASEDIAGRAM)
//@#$LPS-USECASEDIAGRAM:GranularityType:Package
public class FigActor extends FigNodeModelElement
```

Feature: Sequence

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



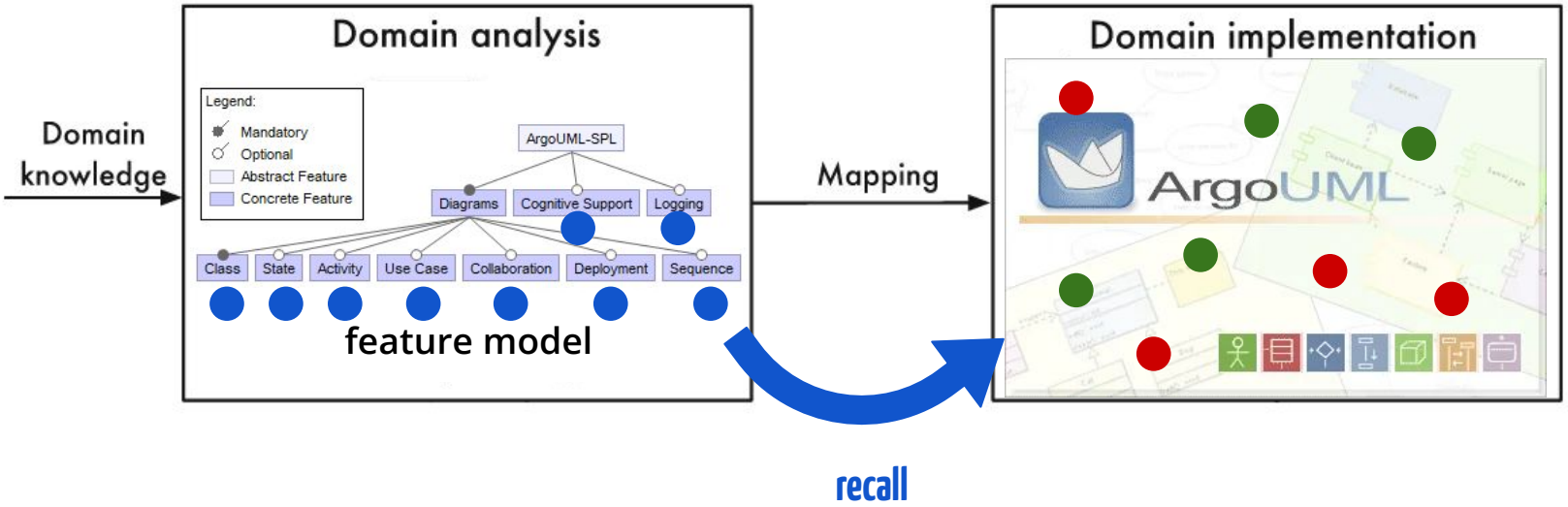
Relevance of the *vp-s*



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

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

Future work

Map the identified *vp-s* with variants to #ifdef directives

Take into account *vp-s* with variants at method level

Extend symfinder to be able to analyse projects in other languages

Mapping Features to Automatically Identified Object-Oriented Variability Implementations

The case of ArgoUML-SPL

Successful mapping to preexisting domain features

vp-s detection method is **little precise** but **highly robust** on ArgoUML-SPL

symfinder identifies ***vp-s*** **with variants relevant** for feature mapping

Availability:

- Public release: tag **vamos2020**
  https://github.com/DeathStar3/symfinder
- symfinder demonstration
  https://deathstar3.github.io/symfinder-demo/

Get the paper:

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.

[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.

[Wesley2017] 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.