How I Met Your Implemented Variability: Identification in Object-Oriented Systems with symfinder

Johann Mortara – Philippe Collet

Université Côte d'Azur, CNRS, I3S, France

Tutorial at SPLC '21

September 6, 2021

Thanks to SPLC'21 sponsors!

BT

BOSCH

pure·systems

ELSEVIER

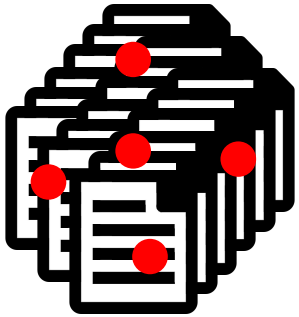
MetaCase

Association for
Computing Machinery

SIGSOFT

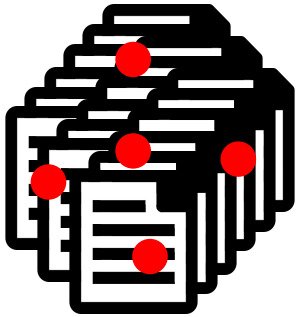
SPECIAL INTEREST GROUP ON SOFTWARE ENGINEERING

What this is all about



Variability-rich system
in a single code base

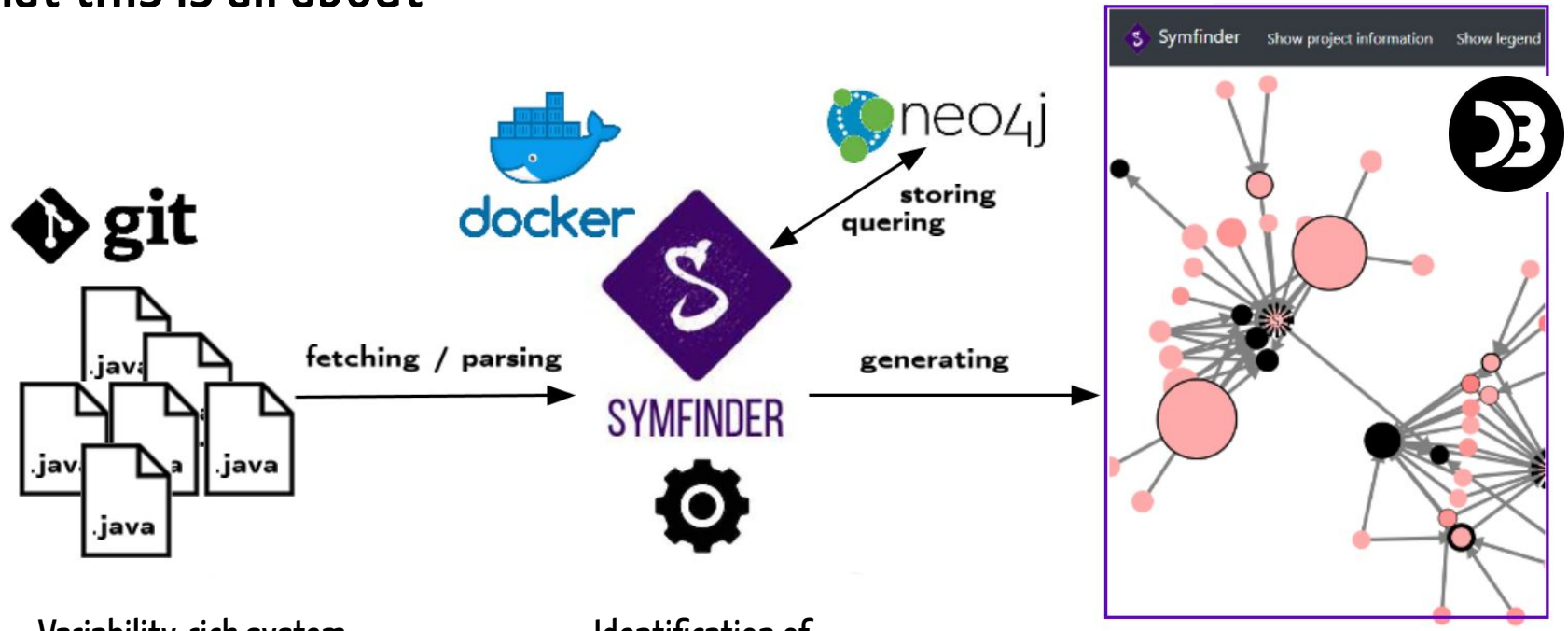
What this is all about



Variability-rich system
in a single code base

How to identify these variability implementations?

What this is all about



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

Identification of symmetries

Visualization of *vp-s* and variants

Table of contents

Motivation and introduction to symfinder

Introduction

Many known software systems are highly-variable

16.000 options managed in 25M LoC [Acher2018]

#ifdef

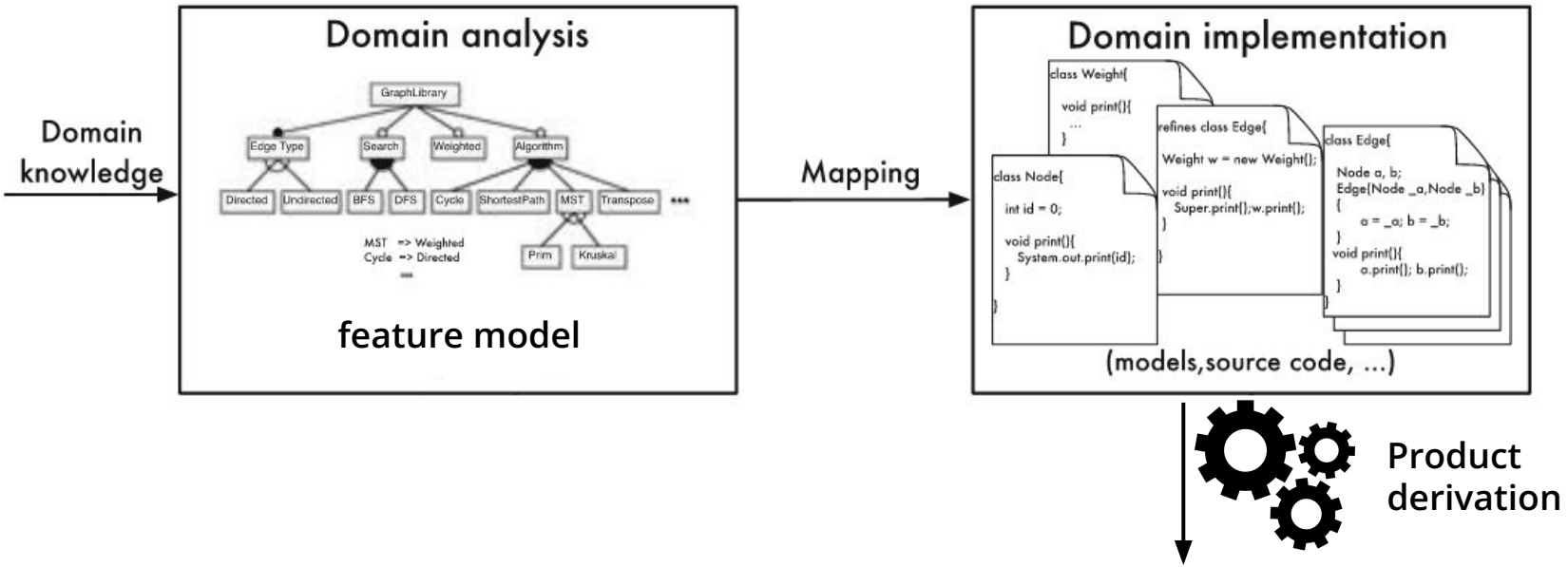
24.000 different platforms in 2015 [Open2015]

Object-orientation

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

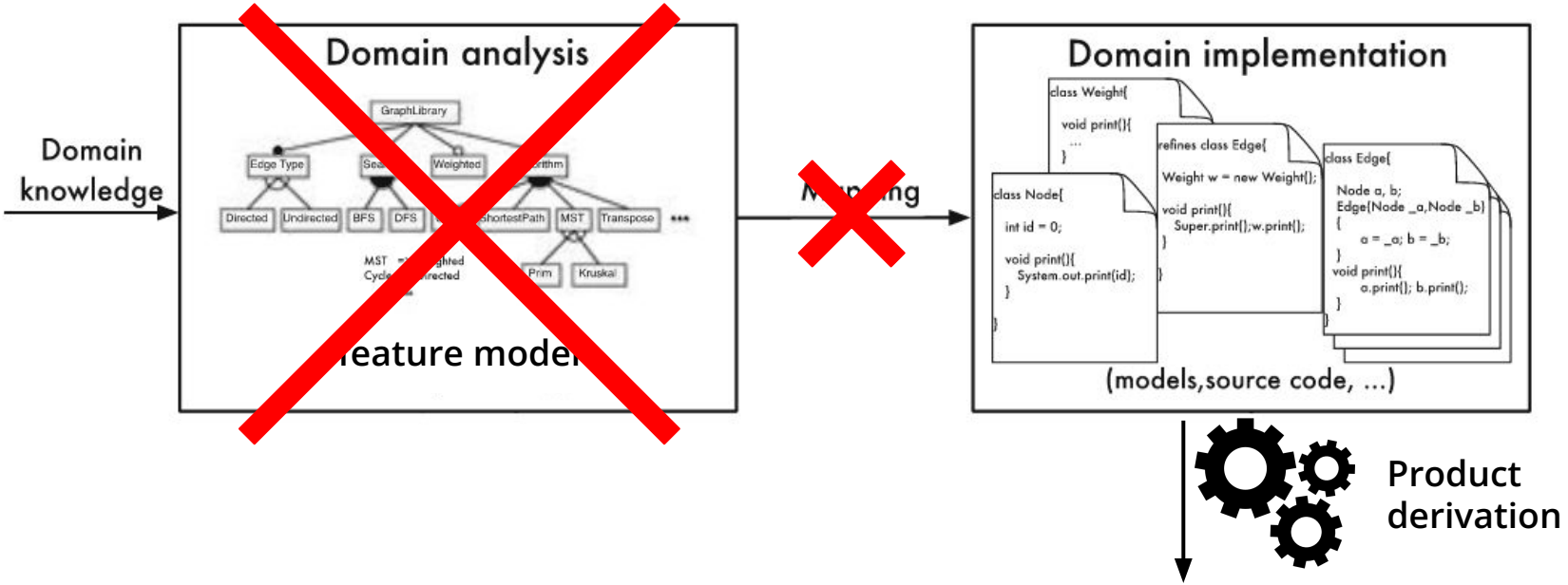
Object-orientation

Software Product Lines: the *classic* (but heavy) variability management chain



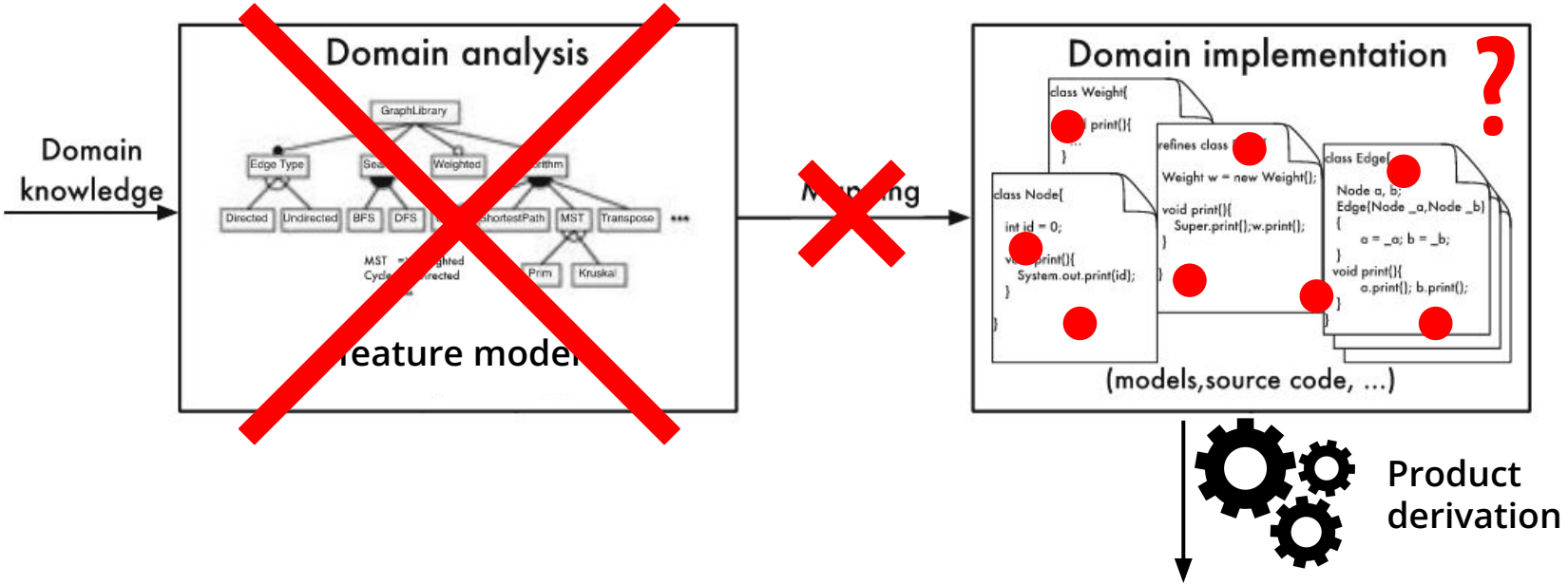
Source: http://stg-tud.github.io/sedc/Lecture/ws16-17/6-SPL.pdf

The reality of variability management



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

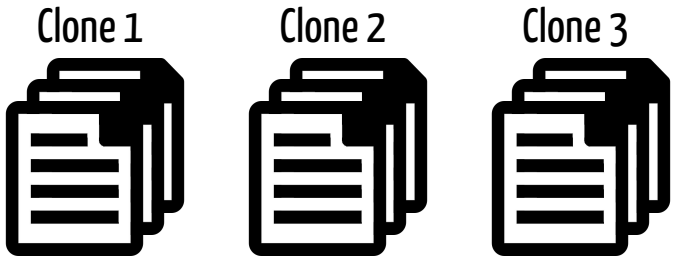
Problem: How to identify variability implementations in an existing OO codebase?



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

Feature location and feature identification: challenges and impact

Context: projects clones

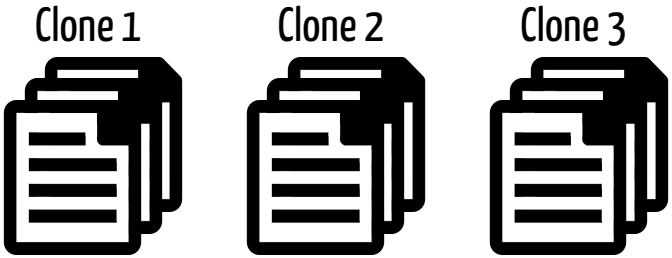


Detection method:

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

Feature location and feature identification: challenges and impact

Context: projects clones

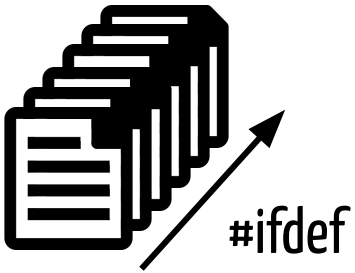


Detection method:

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]

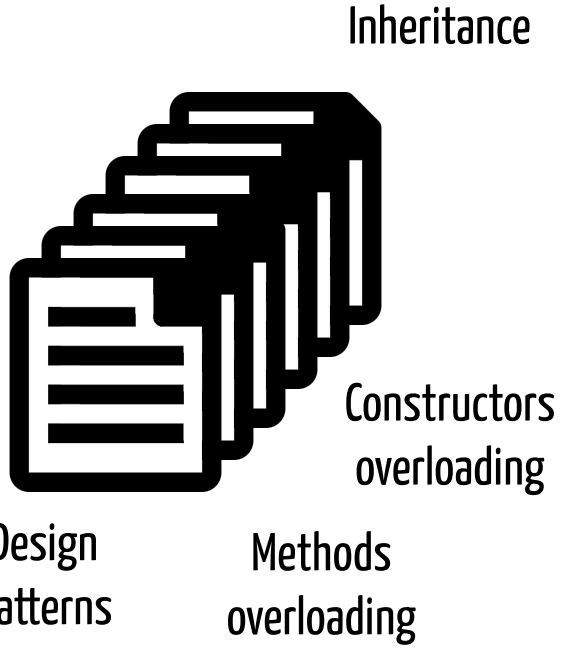
Feature location and feature identification: challenges and impact

Our context: large and unique object-oriented codebase

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

Detection method:

- Techniques coupling static and dynamic analysis [Michelon2021]
- **No technique using only static analysis** [Metzger2014, Tërnava2017]



Features, 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)
  }
}
```

Features, variation points and variants

Inheritance

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)
  }
}
```

Features, variation points and variants

Inheritance

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

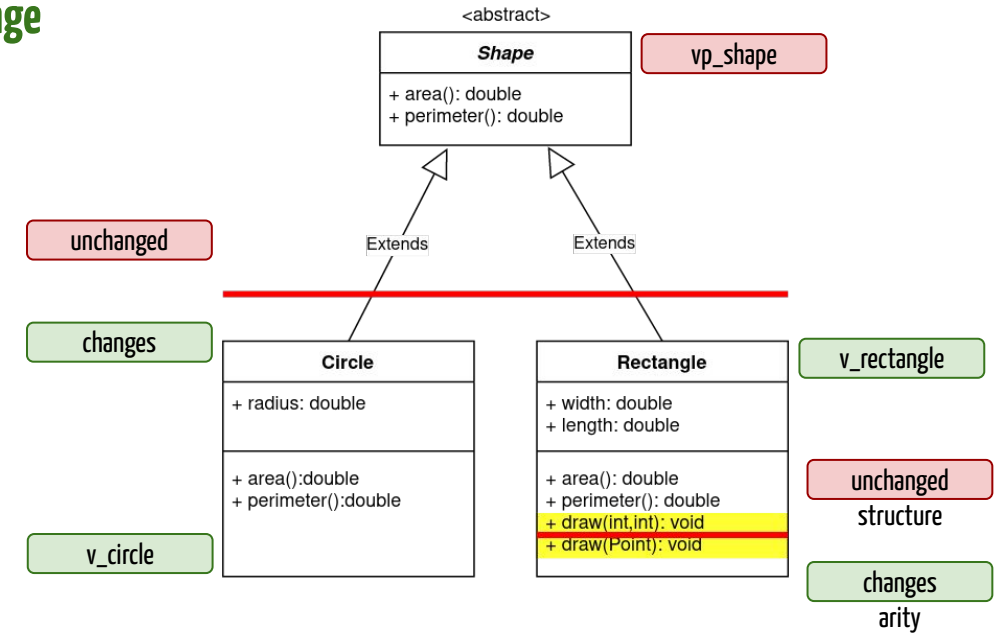
Overloading

Symmetries in nature, human-made artifacts, and OO constructs

Symmetry represents **immunity** to a possible **change**

and is present in object-oriented constructs

including the ones implementing variability!



Some references on symmetry in OO constructs: [Coplien2000, Zhao2003]

Types of symmetries in OO constructs

	Class as type (Not identified by symfinder)	**Class subtyping**	**Method / constructor overriding**	**Method / constructor overloading**	**Strategy**	**Factory**	**Decorator**	**Template**
Commonality (unchange) ⇒ **vp**	Type	Superclass / Interface	Signature	Name	Superclass	Abstract creator and product	Components and decorator interfaces	Method defining the template
Variability (change) ⇒ **variant**	Objects	Subclasses / implementing classes	Implementa-tions in subclasses	Signature	Algorithms	Concrete creators and products	Concrete components and decorators	Steps used in the template

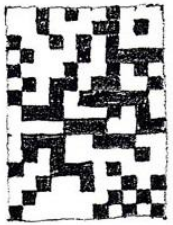
The theory of centres [Alexander2002]

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 one or multiple **local symmetry(ies)**.

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

Random
⟶ hard to describe

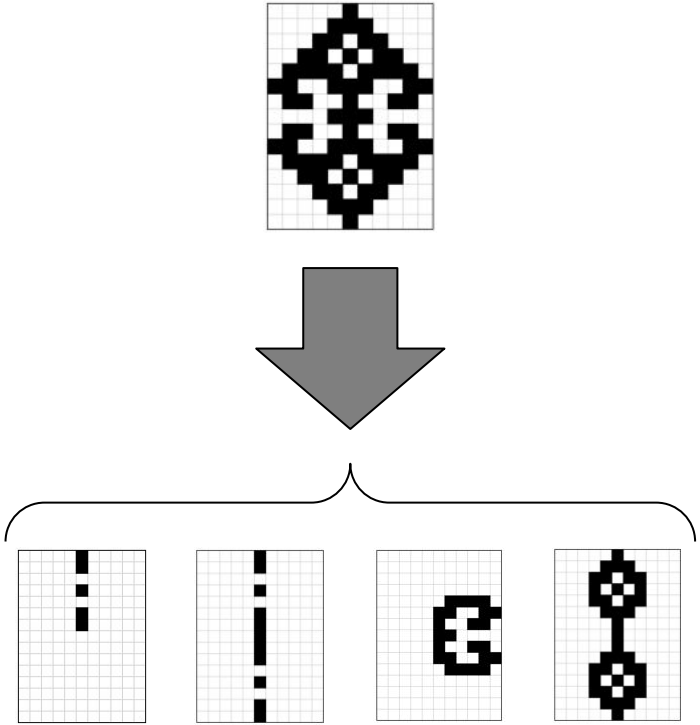


Ordered around a
centre of symmetry
⟶ easy to describe

Centres and density

Local symmetries form a **structure**, whose **coherence** is determined with its **number of symmetries** [Alexander1968]

⇒ remarkable structures aggregate a **density of local symmetries**



Identifying variability implementations

Variability implementation technique ⟷ local symmetry

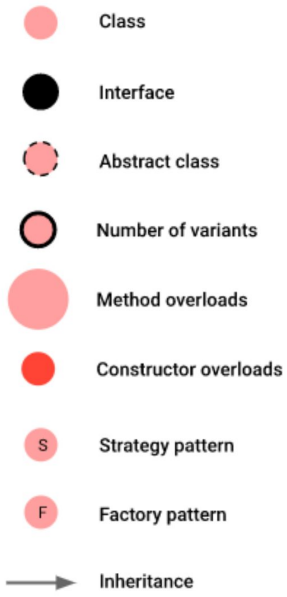
- variation point (commonality) ⟷ unchanged
- variant (variability) ⟷ changes

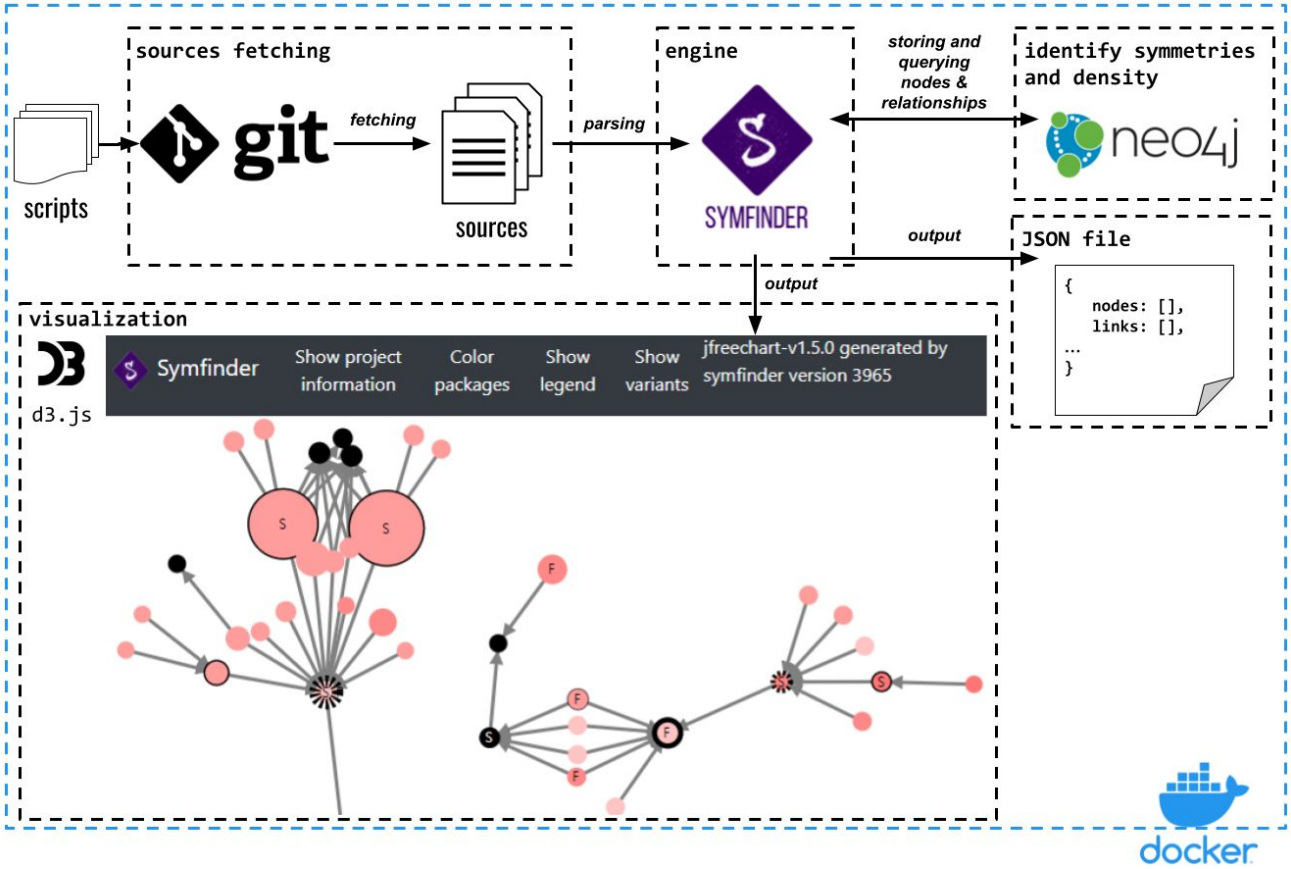
Identification through local symmetries in core assets

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

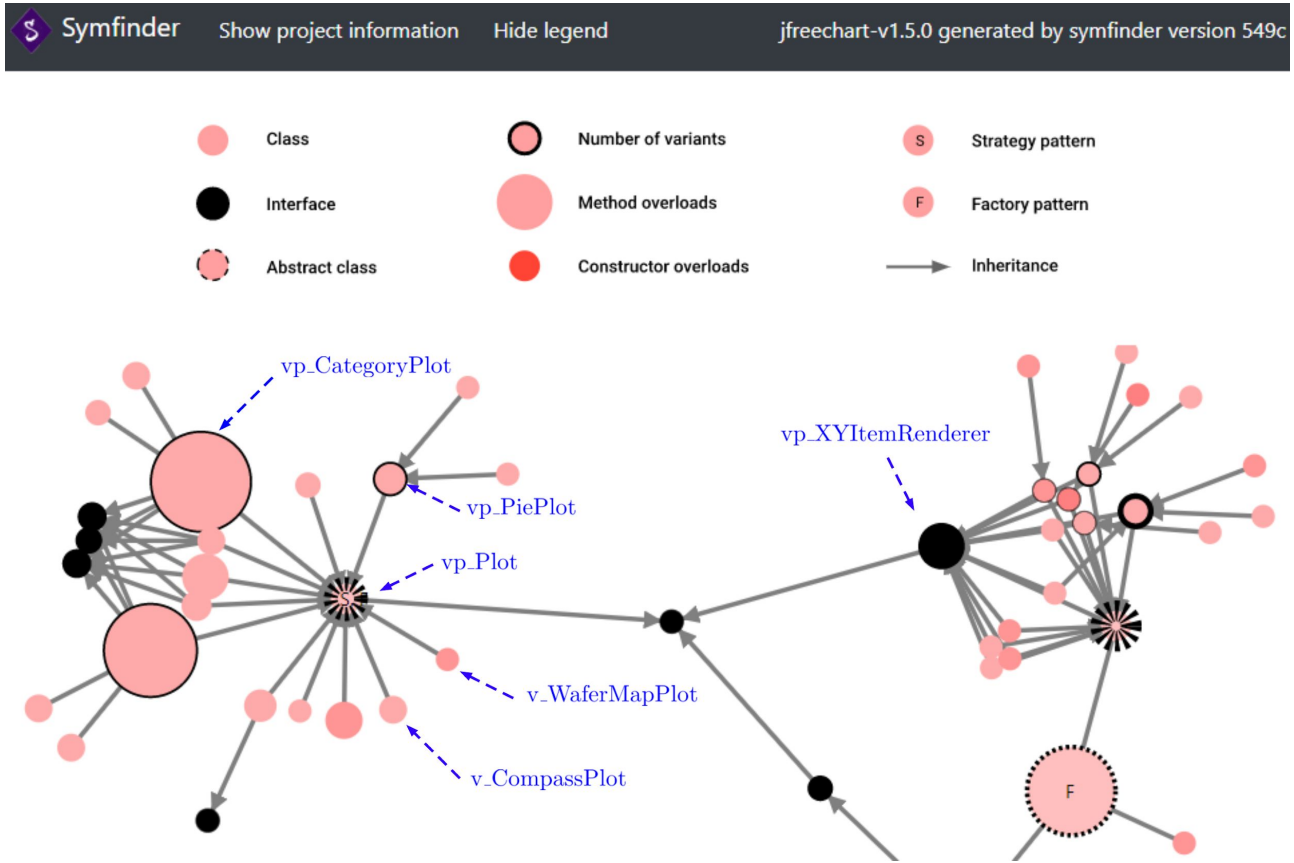




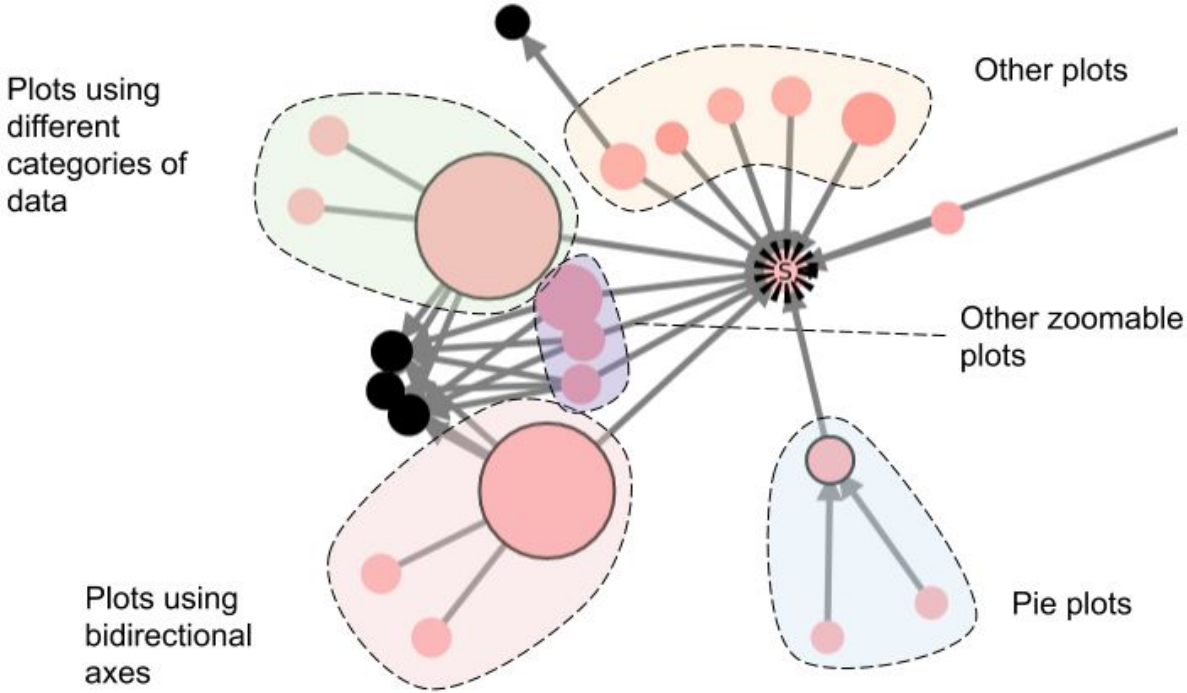
Johann Mortara, Xhevahire Tërnava, and Philippe Collet. “symfinder: A Toolchain for the Identification and Visualization of Object-Oriented Variability Implementations”. In: the 23rd International Systems and Software Product Line Conference. Vol. B. Paris, France: ACM Press, Sept. 2019, pp. 5–8.

Johann Mortara, Philippe Collet, and Xhevahire Tërnava. “Identifying and Mapping Implemented Variabilities in Java and C++ Systems using symfinder”. In: 24th ACM International Systems and Software Product Line Conference (SPLC ’20). Ed. by ACM et al. Virtual Conference. MONTREAL, QC, Canada, Oct. 2020.

Visualization principles



Example of identified variability implementations



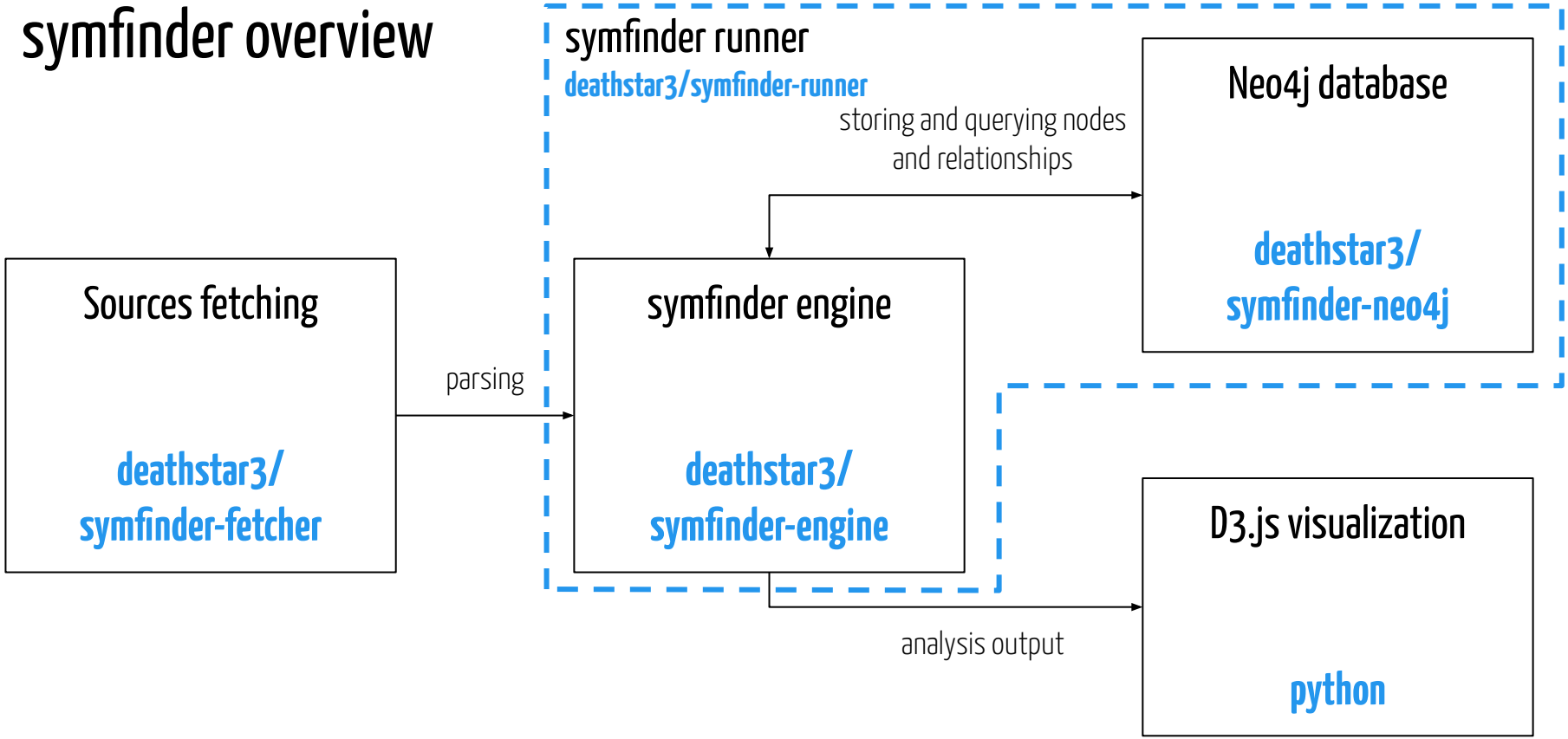
symfinder: first contact

Prerequisites

- Git to clone symfinder's repository, or ZIP download is also possible
- A functional Docker and Docker Compose install
  - Instructions ⟶ https://docs.docker.com/engine/install/
- A web browser to display the visualizations

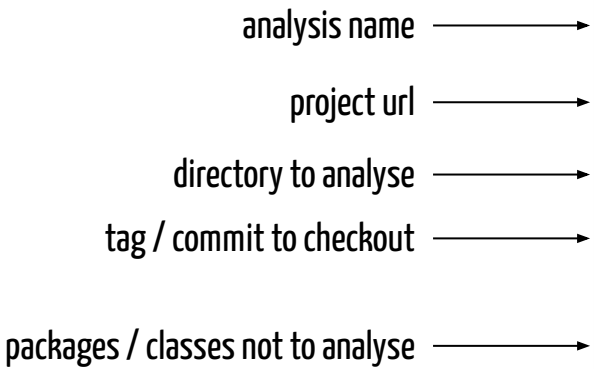
This list is also present in the REQUIREMENTS.md file at the repository's root.

symfinder overview



Used Docker images

Declaring studied projects: the experiments.yaml file




```
jfreechart:

  repositoryUrl: https://github.com/jfree/jfreechart

  sourcePackage: .

  tagIds:
    - v1.5.0

  filters:
    - org.jfree.chart.util.PublicCloneable
    - org.jfree.chart.event
    - org.jfree.data.general
```

symfinder settings: the symfinder.yaml file

Do not touch this file for this tutorial!

address of the Neo4j server →

credentials →

file containing the descriptions of the analysed projects →

```
neo4j:
  boltAddress: bolt://localhost:7687
  user: neo4j
  password: root

experimentsFile: experiments.yaml
```

Running symfinder

Only one script to execute: run.sh

Parameters: projects to analyse

```
$ ./run.sh jfreechart
Creating resources directory
Creating generated_visualizations directory
Using splc2021-tuto images
Cloning into 'resources/jfreechart'...
Note: switching to 'tags/v1.5.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at fd72df7c Prepare for 1.5.0 release.
HEAD is now at fd72df7c Prepare for 1.5.0 release.
```

Docker images tag

Step 1: Fetching the sources

Step 2: symfinder analysis

- the runner creates the containers
- symfinder waits for Neo4j to be started
- the analysis starts
  - visitors parse the code and store information in the Neo4j database
  - **may take some time depending on the project's size!**

```
Creating network "varicity-develop_default" with the default driver
Creating symfinder-runner ... done
Attaching to symfinder-runner
symfinder-runner | Cleaning previous execution...
symfinder-runner | Removing network default_default
symfinder-runner | Network default_default not found.
symfinder-runner | Creating network "default_default" with the default driver
symfinder-runner | Creating symfinder-neo4j ...
symfinder-runner | Creating symfinder-neo4j ... done
symfinder-runner | Creating symfinder       ...
symfinder-runner | Creating symfinder       ... done
symfinder-runner | Attaching to symfinder-neo4j, symfinder
symfinder-runner | symfinder-neo4j | Changed password for user 'neo4j'.
symfinder-runner | symfinder    | WARNING: sun.reflect.Reflection.getCallerClass is not supported. This will impact performance.
symfinder-runner | symfinder    | Jul 30, 2021 9:16:05 AM org.neo4j.driver.internal.logging.JULogger info
symfinder-runner | symfinder    | INFO: Direct driver instance 1489933928 created for server address neo4j:7687
symfinder-runner | symfinder    | 09:16:05.503 [main] MY_LEVEL Symfinder - Symfinder version: 84ef141e60a9414c842e35abdfb5303dc9a204cf
symfinder-runner | symfinder-neo4j | Directories in use:
symfinder-runner | symfinder-neo4j |   home:         /var/lib/neo4j
symfinder-runner | symfinder-neo4j |   config:       /var/lib/neo4j/conf
symfinder-runner | symfinder-neo4j |   logs:         /logs
symfinder-runner | symfinder-neo4j |   plugins:      /plugins
symfinder-runner | symfinder-neo4j |   import:       /var/lib/neo4j/import
symfinder-runner | symfinder-neo4j |   data:         /var/lib/neo4j/data
symfinder-runner | symfinder-neo4j |   certificates: /var/lib/neo4j/certificates
symfinder-runner | symfinder-neo4j |   run:          /var/lib/neo4j/run
symfinder-runner | symfinder-neo4j | Starting Neo4j.
symfinder-runner | symfinder    | Waiting for Neo4j database to be ready...
symfinder-runner | symfinder-neo4j | 2021-07-30 09:16:06.028+0000 INFO  ======== Neo4j 4.0.3 ========
symfinder-runner | symfinder-neo4j | 2021-07-30 09:16:06.033+0000 INFO  Starting...
symfinder-runner | symfinder    | Waiting for Neo4j database to be ready...
symfinder-runner | symfinder-neo4j | 2021-07-30 09:16:13.356+0000 INFO  Called db.clearQueryCaches(): Query cache already empty.
symfinder-runner | symfinder-neo4j | 2021-07-30 09:16:13.404+0000 INFO  Bolt enabled on 0.0.0.0:7687.
symfinder-runner | symfinder-neo4j | 2021-07-30 09:16:13.405+0000 INFO  Started.
symfinder-runner | symfinder-neo4j | 2021-07-30 09:16:13.972+0000 INFO  Remote interface available at http://0.0.0.0:7474/
symfinder-runner | symfinder    | 09:16:15.780 [main] MY_LEVEL Symfinder - ClassesVisitor
symfinder-runner | symfinder    | 09:16:16.300 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ClassesVisitor - Class: org.jfree.chart.ui.StrokeChooserPanel
symfinder-runner | symfinder    | 09:16:17.045 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ClassesVisitor - Class: org.jfree.chart.ui.ApplicationFrame
symfinder-runner | symfinder    | 09:16:17.228 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ClassesVisitor - Class: org.jfree.chart.ui.FontChooserPanel
symfinder-runner | symfinder    | 09:16:17.346 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ClassesVisitor - Class: org.jfree.chart.ui.PaintSample
symfinder-runner | symfinder    | 09:16:17.438 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ClassesVisitor - Class: org.jfree.chart.ui.GradientPaintTransformType
symfinder-runner | symfinder    | 09:16:17.532 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ClassesVisitor - Class: org.jfree.chart.ui.StrokeSample
```

the runner creates the containers

symfinder waits for Neo4j to be started

the analysis starts

End of analysis

- after the visitors parsings are finished, the potential vp-s and variants are identified
- statistics on the analysis are displayed and symfinder stops
- the runner stops the containers

```
symfinder-runner | symfinder    | 09:21:19.953 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class: org.jfree.data.KeyedObjects2D
symfinder-runner | symfinder    | 09:21:20.224 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class:
org.jfree.data.category.IntervalCategoryDataset
symfinder-runner | symfinder    | 09:21:20.299 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class:
org.jfree.data.category.DefaultCategoryDataset
symfinder-runner | symfinder    | 09:21:20.642 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class:
org.jfree.data.category.SlidingCategoryDataset
symfinder-runner | symfinder    | 09:21:20.830 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class:
org.jfree.data.category.CategoryToPieDataset
symfinder-runner | symfinder    | 09:21:20.984 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class: org.jfree.data.category.CategoryDataset
symfinder-runner | symfinder    | 09:21:20.985 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class: org.jfree.data.category.CategoryRangeInfo
symfinder-runner | symfinder    | 09:21:21.011 [main] INFO  visitors.SymfinderVisitor - Visitor: visitors.ComposeTypeVisitor - Class:
org.jfree.data.category.DefaultIntervalCategoryDataset
symfinder-runner | symfinder    | 09:21:21.417 [main] MY_LEVEL Symfinder - visitors.ComposeTypeVisitor execution time: 00:02:17.476
symfinder-runner | symfinder    | 09:21:27.381 [main] MY_LEVEL Symfinder - Number of VPs: 926
symfinder-runner | symfinder    | 09:21:27.385 [main] MY_LEVEL Symfinder - Number of methods VPs: 454
symfinder-runner | symfinder    | 09:21:27.388 [main] MY_LEVEL Symfinder - Number of constructors VPs: 213
symfinder-runner | symfinder    | 09:21:27.395 [main] MY_LEVEL Symfinder - Number of method level VPs: 667
symfinder-runner | symfinder    | 09:21:27.396 [main] MY_LEVEL Symfinder - Number of class level VPs: 259
symfinder-runner | symfinder    | 09:21:27.427 [main] MY_LEVEL Symfinder - Number of variants: 1923
symfinder-runner | symfinder    | 09:21:27.430 [main] MY_LEVEL Symfinder - Number of methods variants: 1061
symfinder-runner | symfinder    | 09:21:27.433 [main] MY_LEVEL Symfinder - Number of constructors variants: 587
symfinder-runner | symfinder    | 09:21:27.439 [main] MY_LEVEL Symfinder - Number of method level variants: 1648
symfinder-runner | symfinder    | 09:21:27.440 [main] MY_LEVEL Symfinder - Number of class level variants: 275
symfinder-runner | symfinder    | 09:21:27.452 [main] MY_LEVEL Symfinder - Number of nodes: 9526
symfinder-runner | symfinder    | 09:21:27.461 [main] MY_LEVEL Symfinder - Number of relationships: 11438
symfinder-runner | symfinder    | 09:21:27.470 [main] MY_LEVEL Symfinder - Number of corrected inheritance relationships: 265/1478
symfinder-runner | symfinder    | Jul 30, 2021 9:21:28 AM org.neo4j.driver.internal.logging.JULogger info
symfinder-runner | symfinder    | INFO: Closing driver instance 1489933928
symfinder-runner | symfinder    | Jul 30, 2021 9:21:28 AM org.neo4j.driver.internal.logging.JULogger info
symfinder-runner | symfinder    | INFO: Closing connection pool towards neo4j:7687
symfinder-runner | symfinder    | 09:21:28.701 [main] MY_LEVEL Symfinder - Total execution time: 00:05:23.198
symfinder-runner | symfinder exited with code 0
symfinder-runner | Stopping symfinder-neo4j ...
symfinder-runner | Stopping symfinder-neo4j ... done
symfinder-runner | Aborting on container exit...
symfinder-runner | Removing symfinder       ...
symfinder-runner | Removing symfinder-neo4j ...
symfinder-runner | Removing symfinder       ... done
symfinder-runner | Removing symfinder-neo4j ... done
```

vp-s and variants identified, statistics on the analysis are displayed and symfinder stops

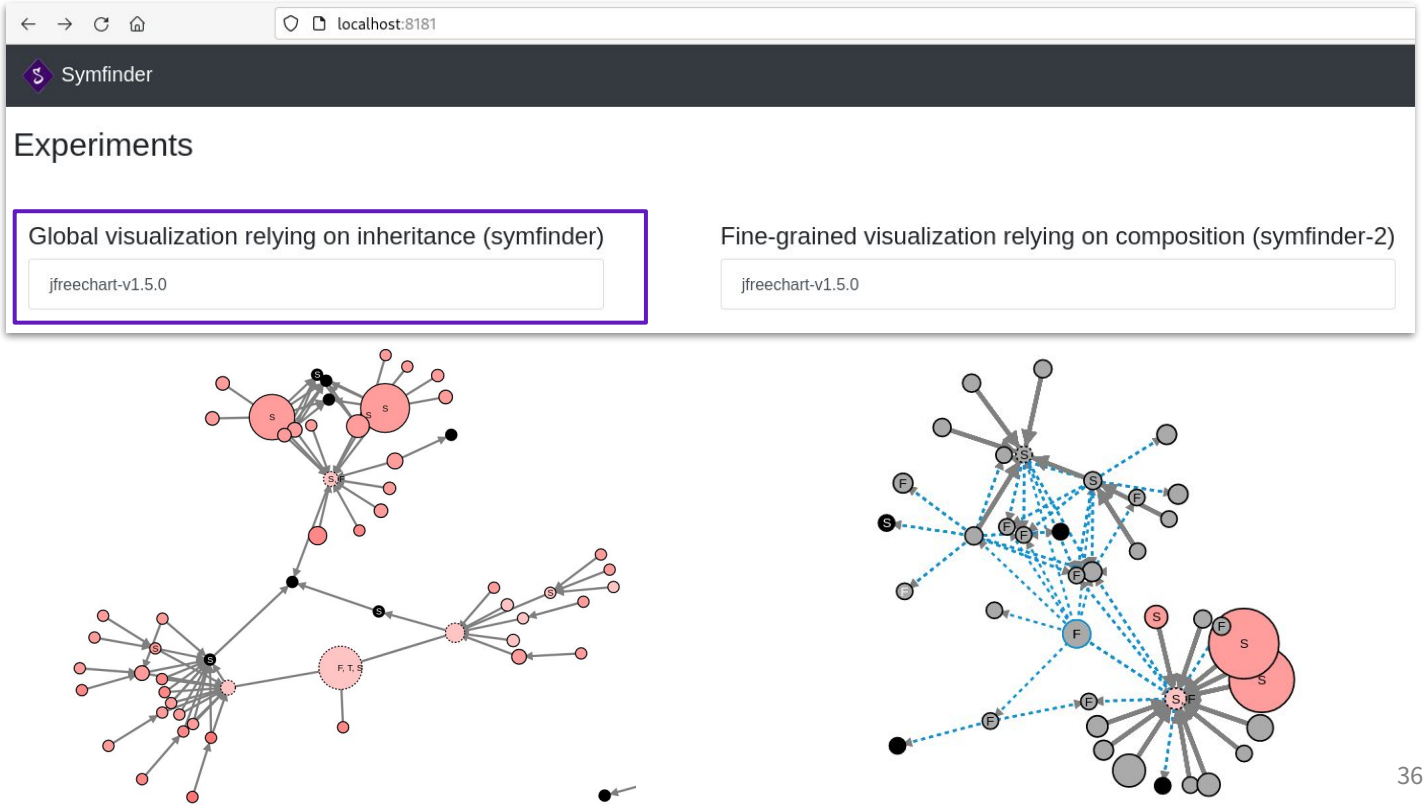
the runner stops the containers

End of analysis

Step 3: accessing the visualization

```
$ ./visualization.sh
Creating network "symfinder-2_default" with the default driver
Creating visualization ... done
Attaching to visualization
```

Focus on this part



First contact: running an existing experiment

Clone symfinder's repository and checkout the "tutorial" tag

Run symfinder on JFreeChart

Display the visualization

```
~ $ git clone https://github.com/DeathStar3/symfinder-SPLC2021-tutorial

~ $ cd symfinder-SPLC2021-tutorial

~/symfinder-SPLC2021-tutorial $ git checkout tutorial

~/symfinder-SPLC2021-tutorial $ ./run.sh jfreechart

~/symfinder-SPLC2021-tutorial $ ./visualization.sh
```

Repository's URL: https://github.com/DeathStar3/symfinder-SPLC2021-tutorial/

Requirements in the REQUIREMENTS.md file

Guided use of symfinder

Adapting symfinder for your project

1. Edit the **experiments/experiments.yaml** to add your project.

```
<project_name>:

  repositoryUrl: <repository_url>

  sourcePackage: <directory_path>

  tagIds:
    - <tag_ids>

  commitIds:
    - <commit_ids>

  filters:
    - <classes>
```

2. Relaunch symfinder on your project

```
~/symfinder $ ./run.sh <project_name>
```

Hall of fame

The hall of fame of the tutorial will contain the visualizations you obtained on **your projects**!

If you wish to send us your visualizations:

- zip the **generated_visualizations** directory;
- send it to us by mail:
  - johann [dot] mortara [at] univ-cotedazur [dot] fr
  - philippe [dot] collet [at] univ-cotedazur [dot] fr

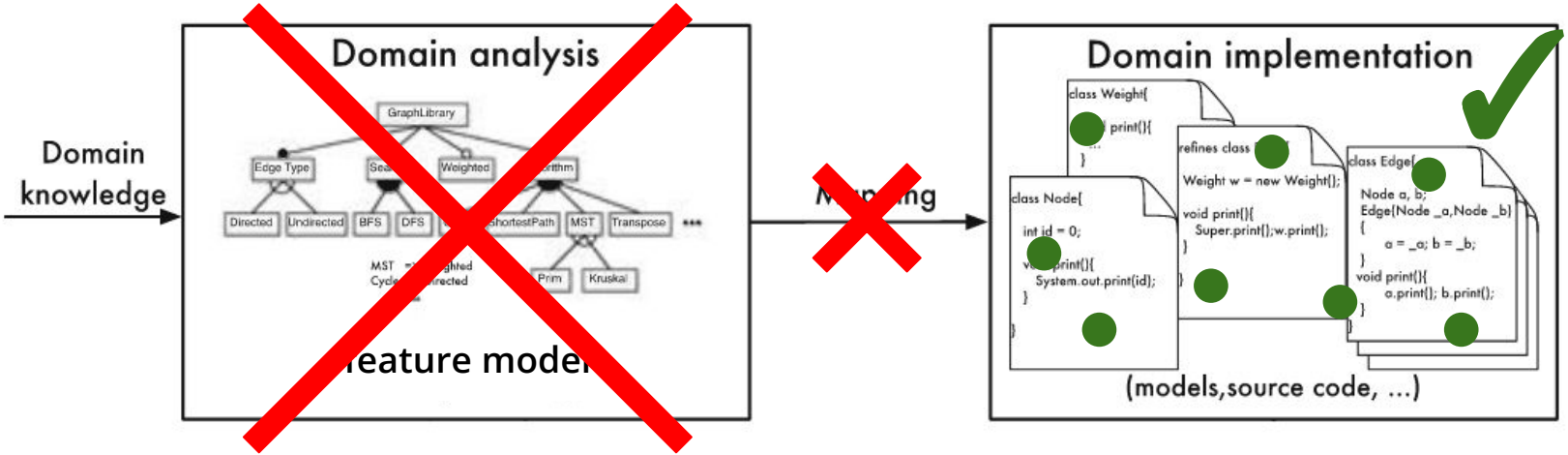
Wrapping up & Exchange time

Wrap up

What you saw in this tutorial:

- Background on variability implementation in OO systems, and challenges around their identification
- Introduction on the notion of symmetry in OO constructs
- The symfinder toolchain
  - how the toolchain uses density of symmetries to identify vp-s and variants in a single Java codebase
  - practical application on your systems

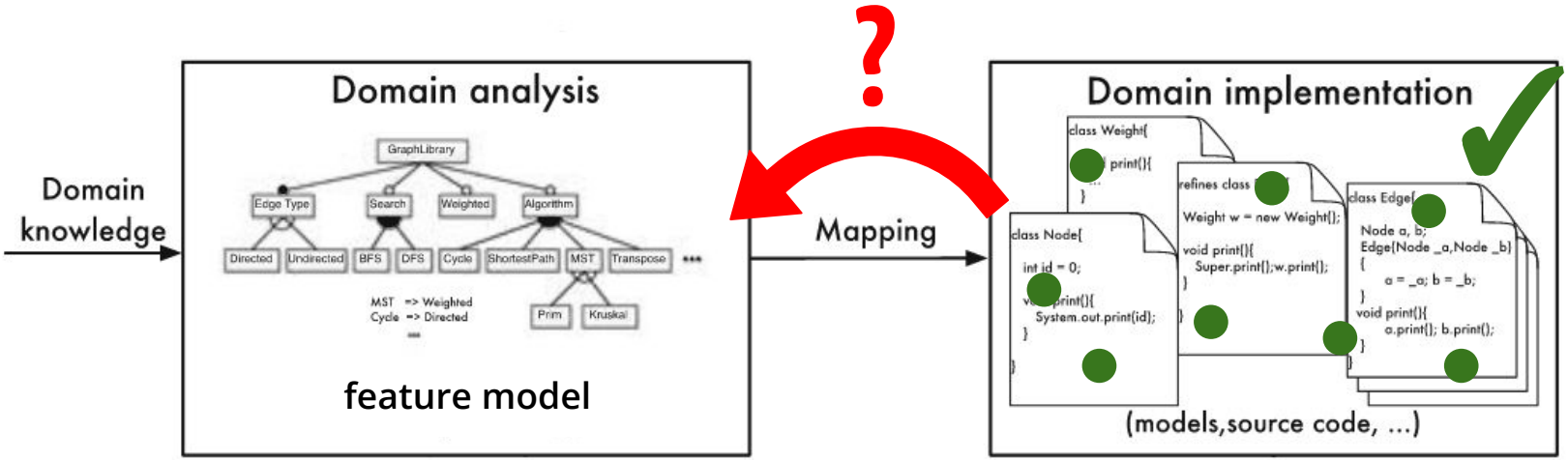
Potential variability implementations identified!



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

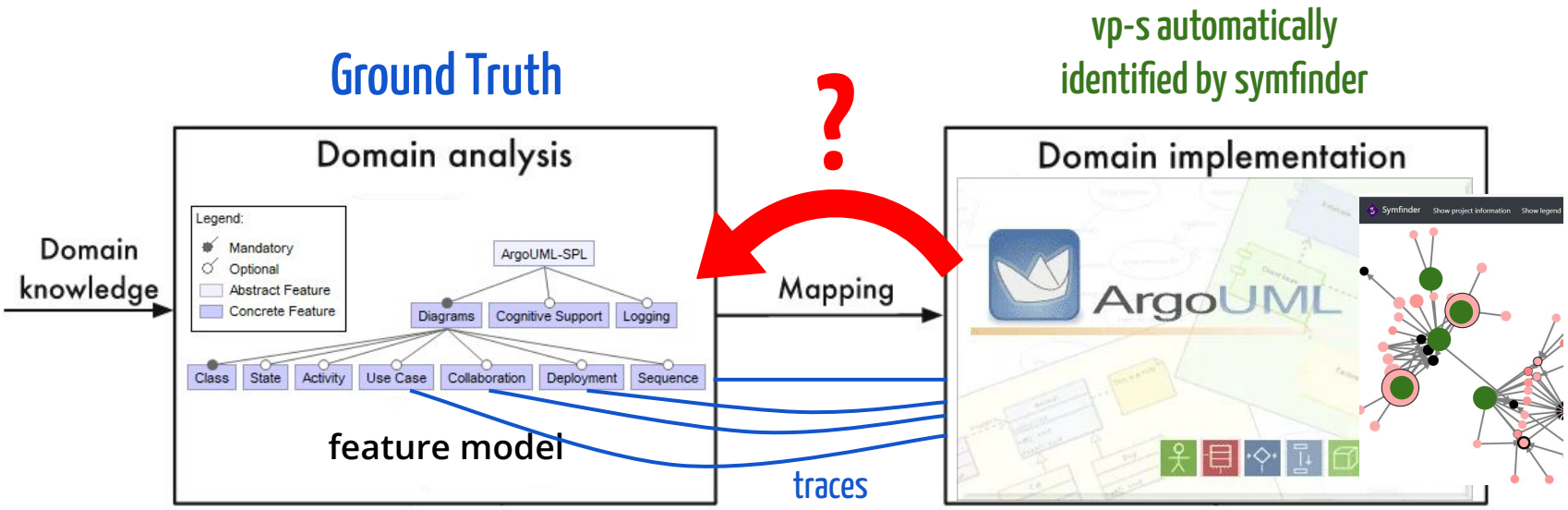
Potential variability implementations identified!

but are they relevant w.r.t. existing features?



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

Application to a case study: ArgoUML



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

Mapping vp-s and variants to features

Precision

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

38%

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

Recall

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

83%

The missing 17% of traces are not variability related (initialization classes, external libraries)

Johann Mortara, Xhevahire Tërnava, and Philippe Collet. "Mapping Features to Automatically Identified Object-Oriented Variability Implementations - The case of ArgoUML-SPL". In: 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS '20). Magdeburg, Germany, Feb. 2020.

Mapping vp-s and variants to features

Precision

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

38%

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

Recall

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

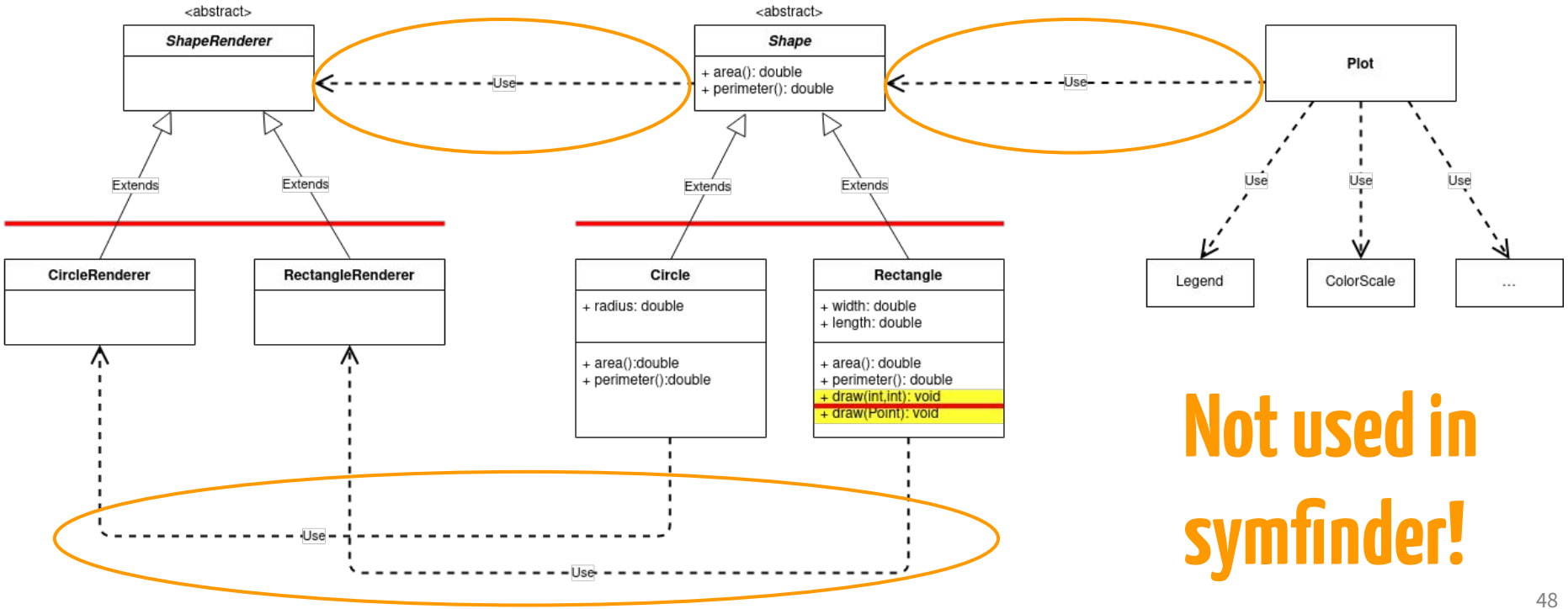
83%

The missing 17% of traces are not variability related (initialization classes, external libraries)

⇒ need for a more precise detection method

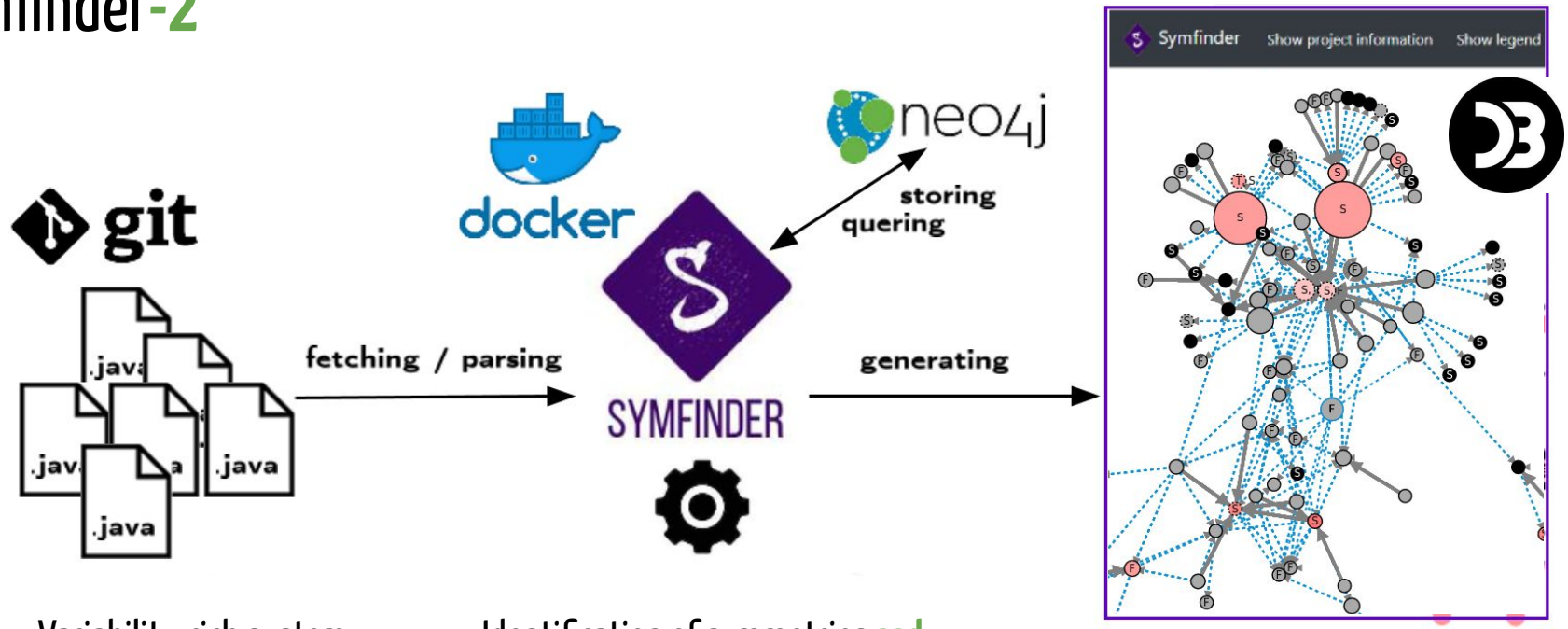
Johann Mortara, Xhevahire Tërnava, and Philippe Collet. "Mapping Features to Automatically Identified Object-Oriented Variability Implementations - The case of ArgoUML-SPL". In: 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS '20). Magdeburg, Germany, Feb. 2020.

Extending the identification method



Not used in symfinder!

symfinder-2

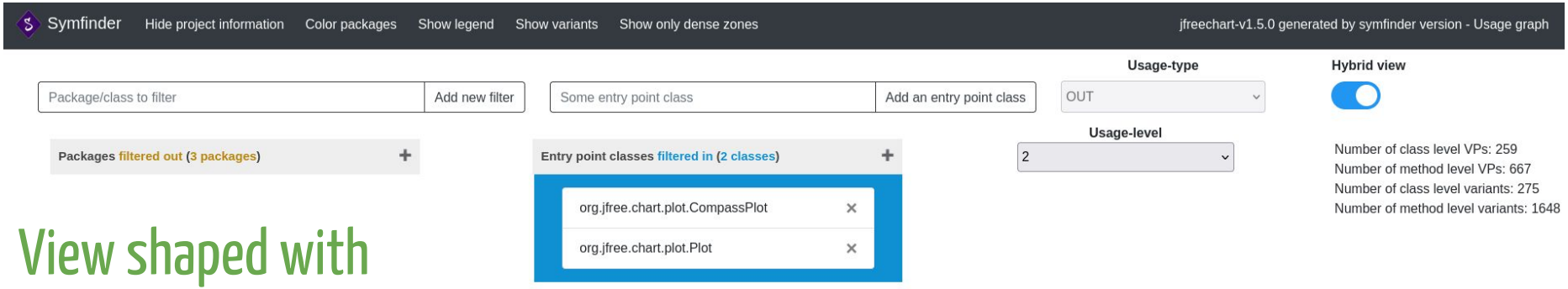


Variability-rich system in a single code base

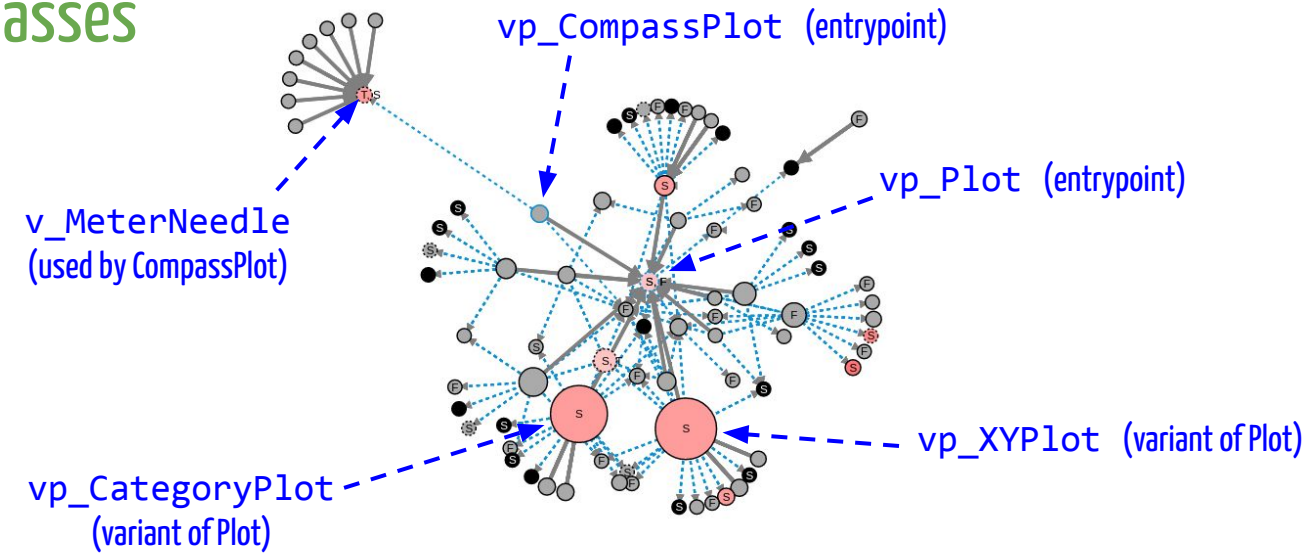
Identification of symmetries and dense zones of variability using usage relationships

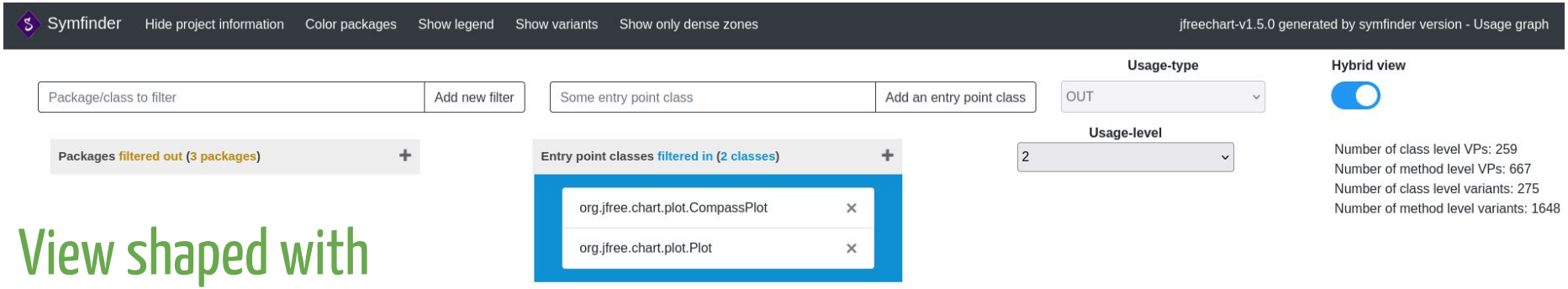
Visualization of *vp-s* and variants with their usage relationships

Johann Mortara, Xhevahire Tërnava, Philippe Collet, Anne-Marie Dery-Pinna. Extending the Identification of Object-Oriented Variability Implementations using Usage Relationships. SPLC 2021 - 25th ACM International Systems and Software Product Line Conference, Sep 2021, Leicester, United Kingdom. pp.1-8

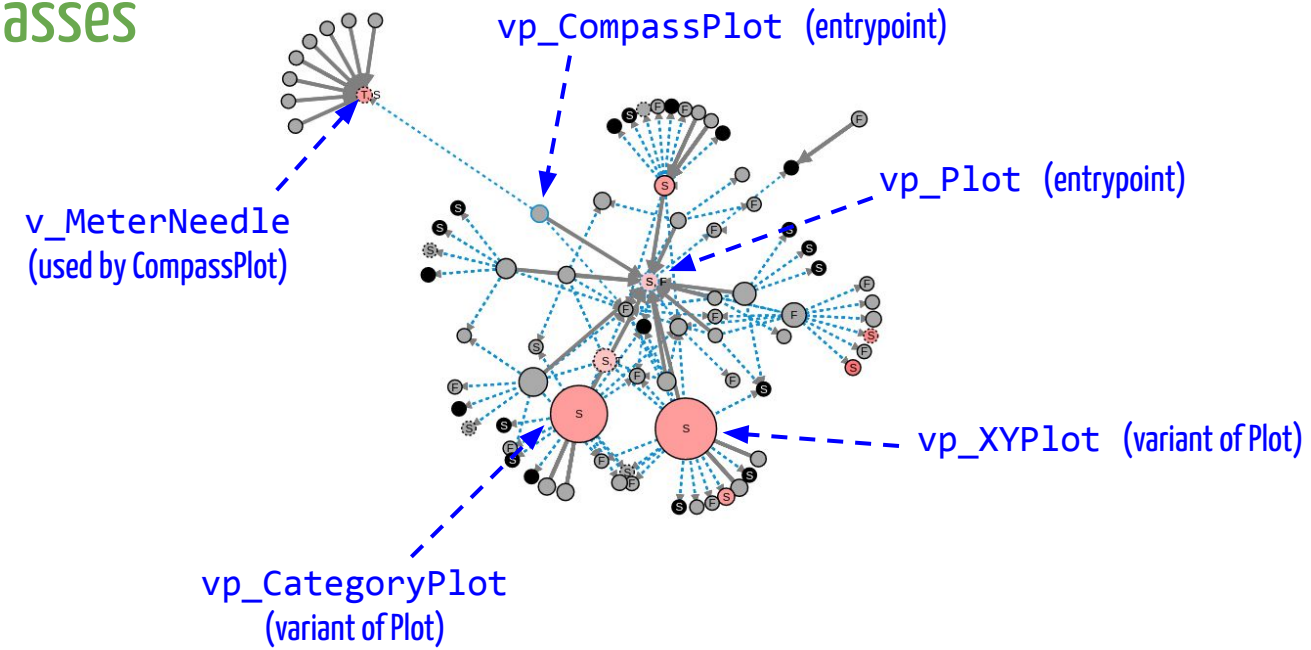


View shaped with entrypoint classes



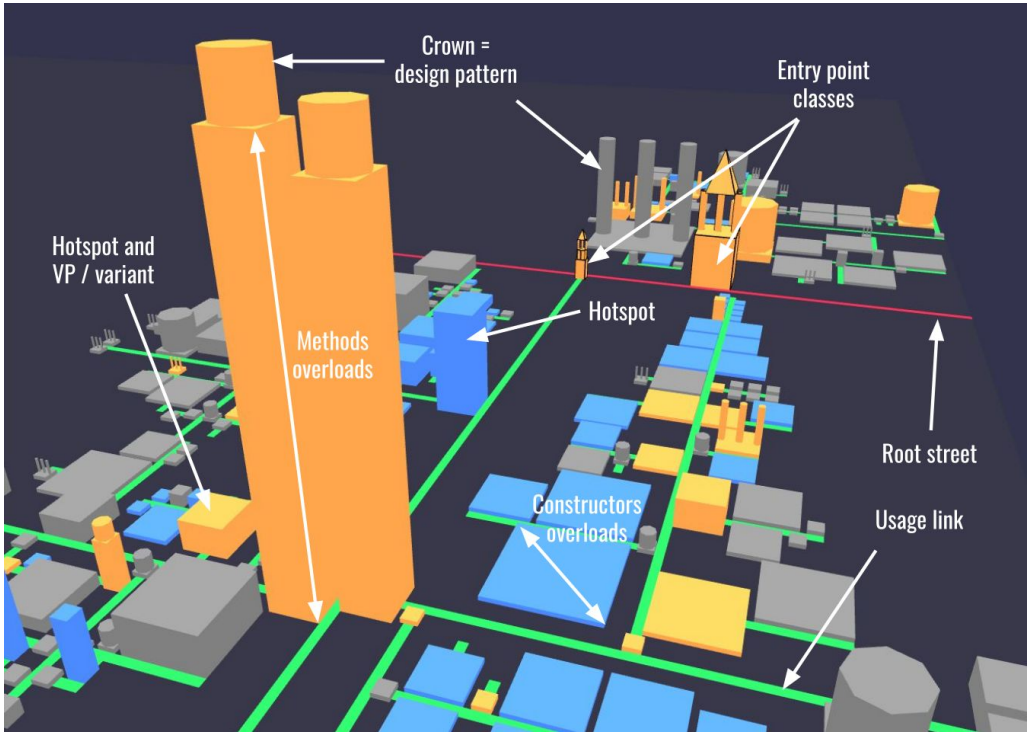


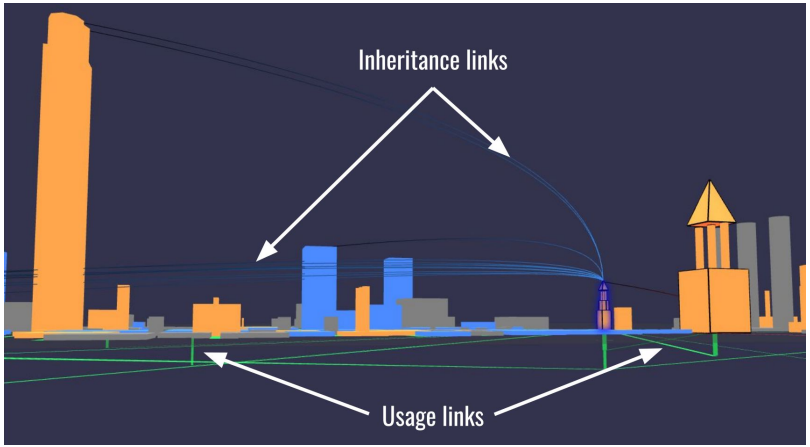
View shaped with entrypoint classes



but

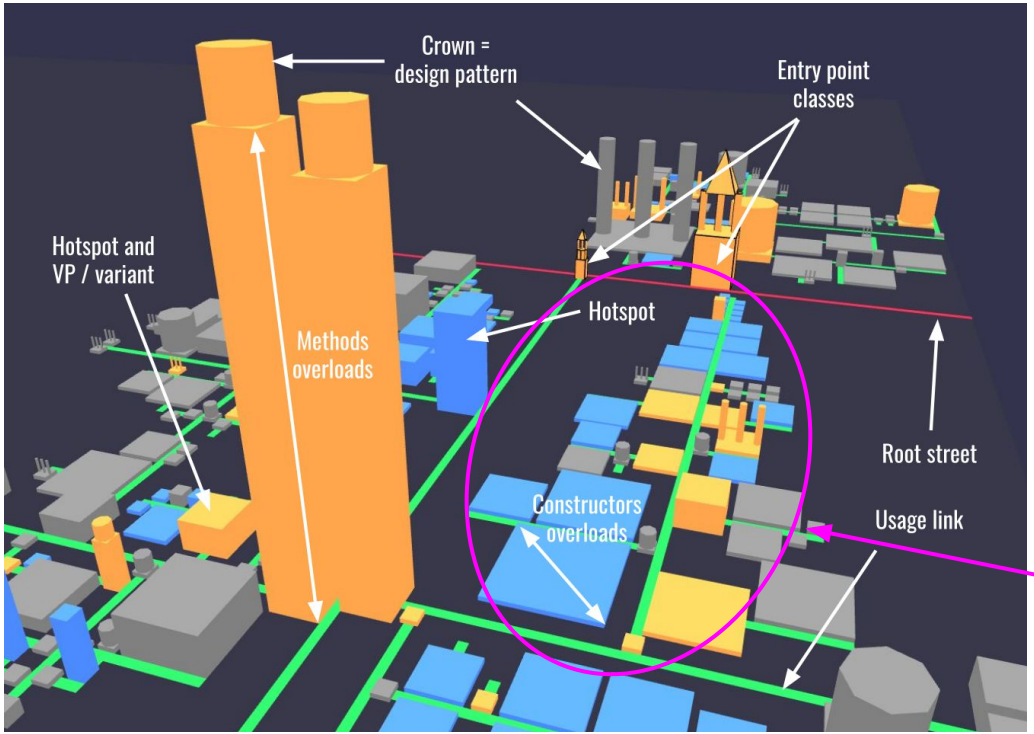
VariCity: visualizing variability implementations as a city

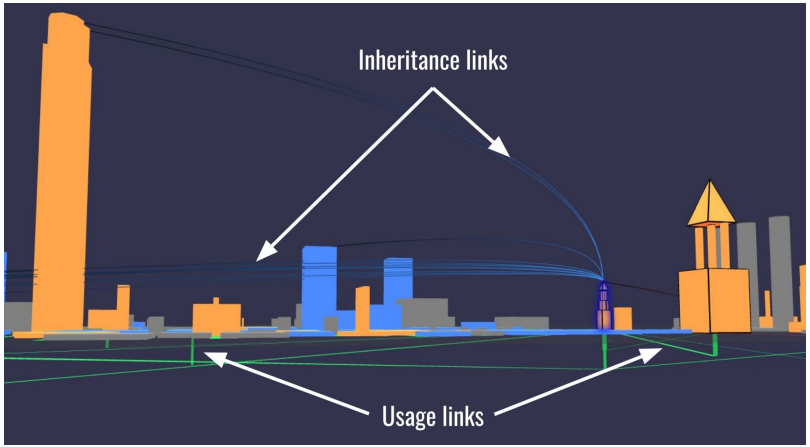




Johann Mortara, Philippe Collet, Anne-Marie Dery-Pinna. Visualization of Object-Oriented Variability Implementations as Cities. 9th IEEE Working Conference on Software Visualization (VISSOFT 2021), Sep 2021, Luxembourg (virtual), Luxembourg.

VariCity: visualizing variability implementations as a city





Density of variability implementations

Johann Mortara, Philippe Collet, Anne-Marie Dery-Pinna. Visualization of Object-Oriented Variability Implementations as Cities. 9th IEEE Working Conference on Software Visualization (VISSOFT 2021), Sep 2021, Luxembourg (virtual), Luxembourg.

Limitations / future work

Toolchain can analyze OO variability implementations projects in Java and C++

but no support for other OO languages (Python, JavaScript…) or implementation techniques

⇒ **extend the toolchain support**

Parameterized density gives first results

but need to understand how to determine these parameters for a project

⇒ **need to analyze projects with ≠ architectures to understand better**

Non exhaustive list of topics

1. Time of the conducted analyses on the different systems
2. Discussions on observed variability implementations and architectures
3. Discussion on the usage of the different features of the symfinder visualization
4. Feedback on the experience and possible improvements

Thanks to SPLC'21 sponsors!

BT

BOSCH

pure·systems

ELSEVIER

MetaCase

acm Association for Computing Machinery

SIGSOFT
SPECIAL INTEREST GROUP ON SOFTWARE ENGINEERING

How I Met Your Implemented Variability: Identification in Object-Oriented Systems with symfinder

Johann Mortara – Philippe Collet

Thank you for attending!

Tutorial's website:
https://deathstar3.github.io/SPLC2021-symfinder-tutorial/

Mail addresses to send your visualizations for the hall of fame:

- johann [dot] mortara [at] univ-cotedazur [dot] fr
- philippe [dot] collet [at] univ-cotedazur [dot] fr

References

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

[Alexander1968] Christopher Alexander and Susan Carey. 1968. Subsymmetries. Perception & Psychophysics 4, 2 (1968), 73–77.

[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.

[Coplien2000] James O Coplien and Liping Zhao. 2000. Symmetry breaking in software patterns. In International Symposium on Generative and Component-Based Software Engineering. Springer, 37–54.

[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.

[Michelon2021] Gabriela K. Michelon, Lukas Linsbauer, Wesley K.G. Assunção, Stefan Fischer, and Alexander Egyed. 2021. A Hybrid Feature Location Technique for Re-engineering Single Systems into Software Product Lines. In 15th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS'21). ACM, New York, NY, USA, Article 11, 1–9.

[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.

[Zhao2002] Liping Zhao and James O Coplien. 2002. Symmetry in class and type hierarchy. In Proceedings of the Fortieth International Conference on Tools Pacific: Objects for internet, mobile and embedded applications. Australian Computer Society, Inc., 181–189.

[Zhao2003] Liping Zhao and James Coplien. 2003. Understanding symmetry in object-oriented languages. Journal of Object Technology 2, 5 (2003), 123–134.