Identifying and Mapping Implemented Variabilities in Java and C++ Systems using symfinder

Johann Mortara [1] – Philippe Collet [1] – Xhevahire Tërnava

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

SPLC '20

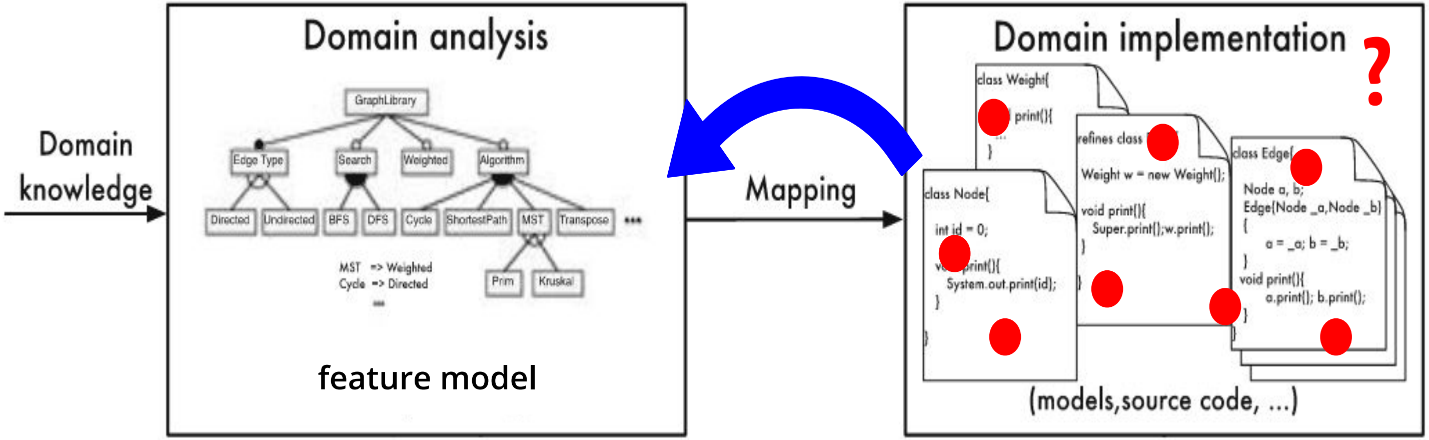
Montréal – October 21, 2020

Managing Large Variability-Rich Systems

Firefox Developer Edition

ESR

BETA



```
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();
  }
}
```

android

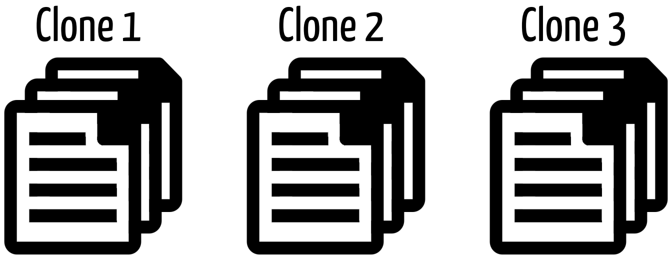
Apache NetBeans IDE

How to identify variability implementations in an existing codebase?

How to map these variability implementations to domain features?

Variability implementations techniques

Project clones

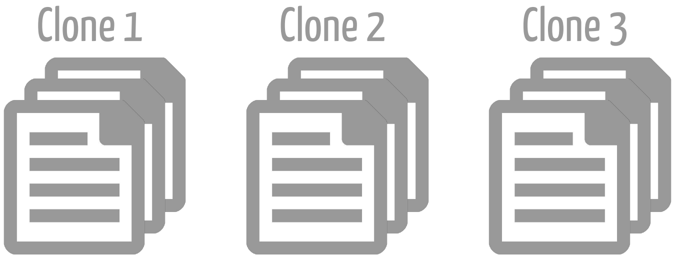


Detection method:

Comparison between clones and mapping with the domain features

Variability implementations techniques

Project clones

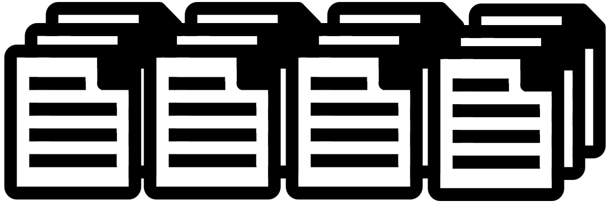


Detection method:

Comparison between clones and mapping with the domain features

Our focus: Single OO codebase

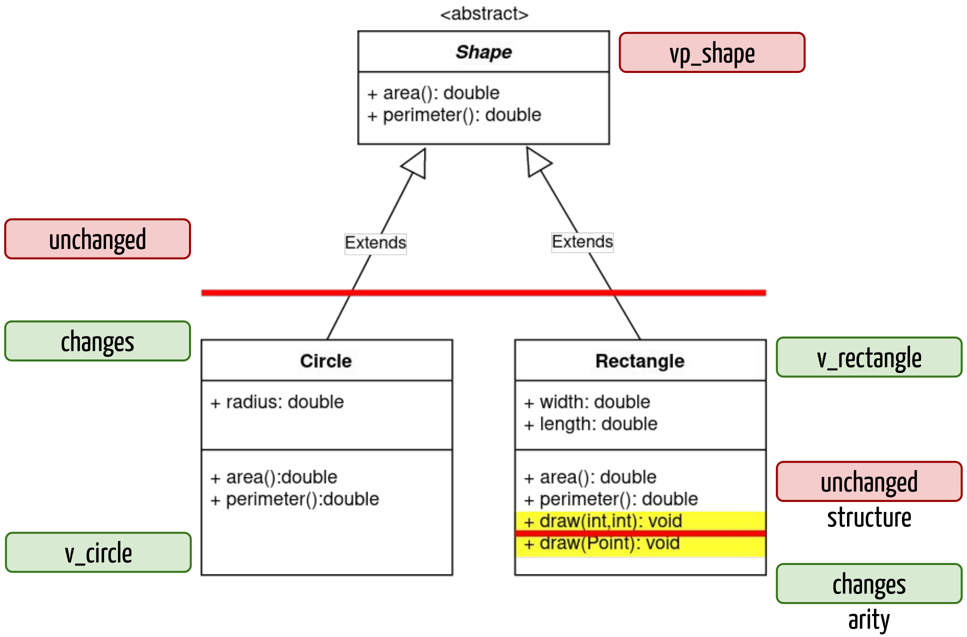
(no preprocessing directives)



- Several implementation mechanisms (inheritance, overloading, design patterns)

⟶ variability buried in the code

Use of symmetries to detect variability implementations



vp_shape

Shape

+ area(): double
+ perimeter(): double

Extends

Extends

unchanged

changes

Circle

+ radius: double

+ area():double
+ perimeter():double

v_circle

Rectangle

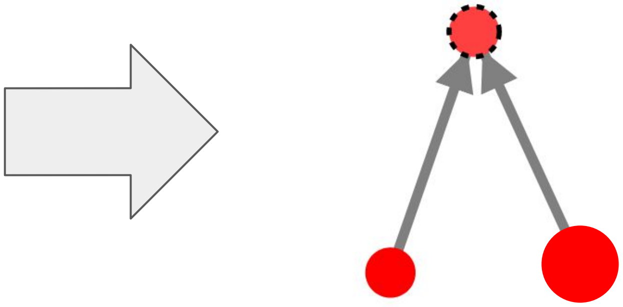
+ width: double
+ length: double

+ area(): double
+ perimeter(): double
+ draw(int,int): void
+ draw(Point): void

v_rectangle

unchanged
structure

changes
arity

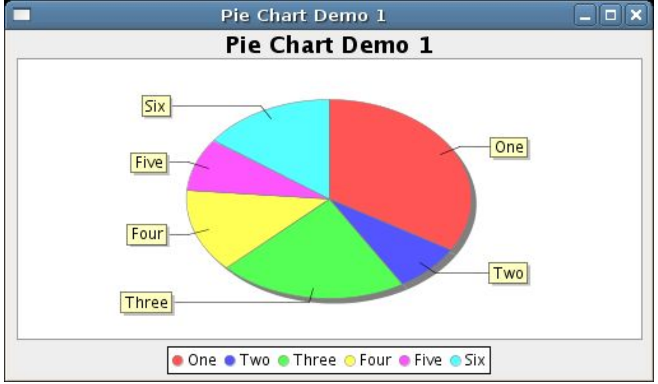


Identification through local symmetries in core assets

High density of symmetries ⇒ variability intense places

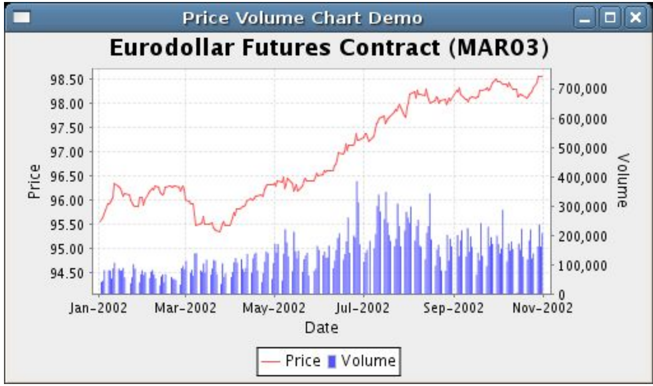
Xhevahire Tërnava, Johann Mortara, and Philippe Collet. 2019. Identifying and Visualizing Variability in Object-Oriented Variability-Rich Systems. In 23rd International Systems and Software Product Line Conference - Volume A (SPLC '19), September 9–13, 2019, Paris, France. ACM, New York, NY, USA, 12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

Project ID card - JFreeChart



©JFree

Description: A 2D chart library for Java applications.



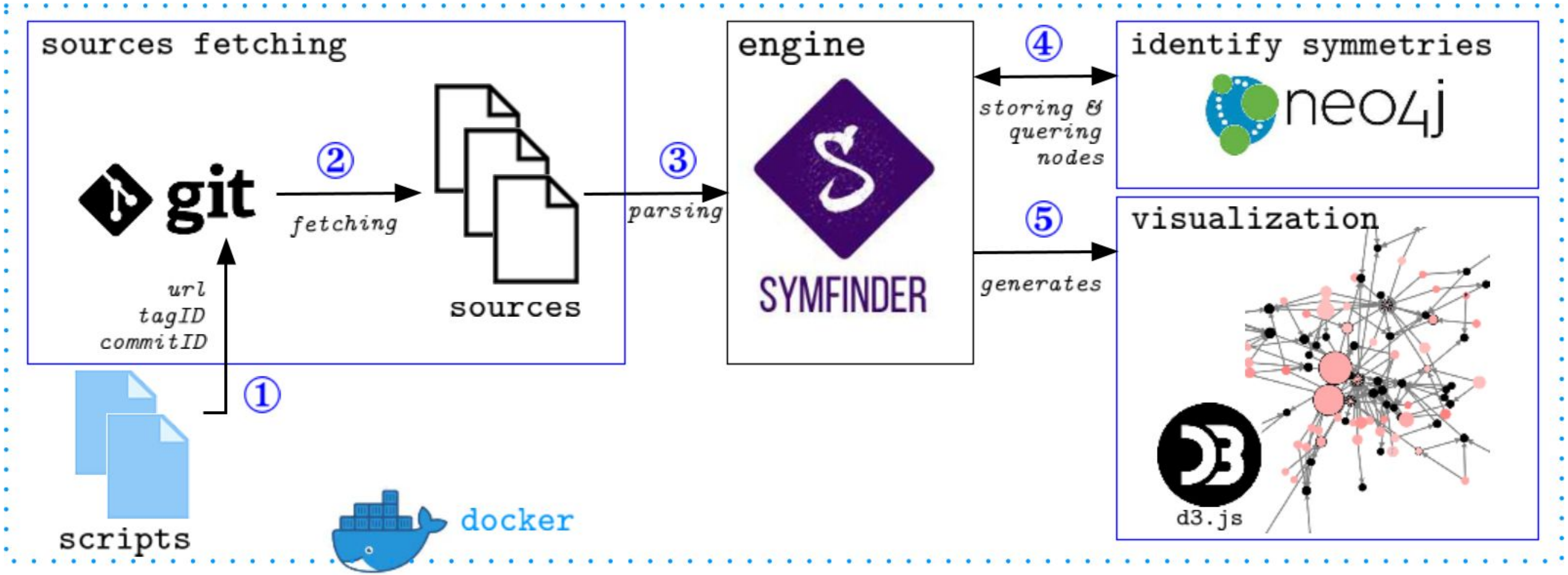
Language: Java

Expected variability implementations:

Different types of charts: cartesian charts, pie charts, line charts...

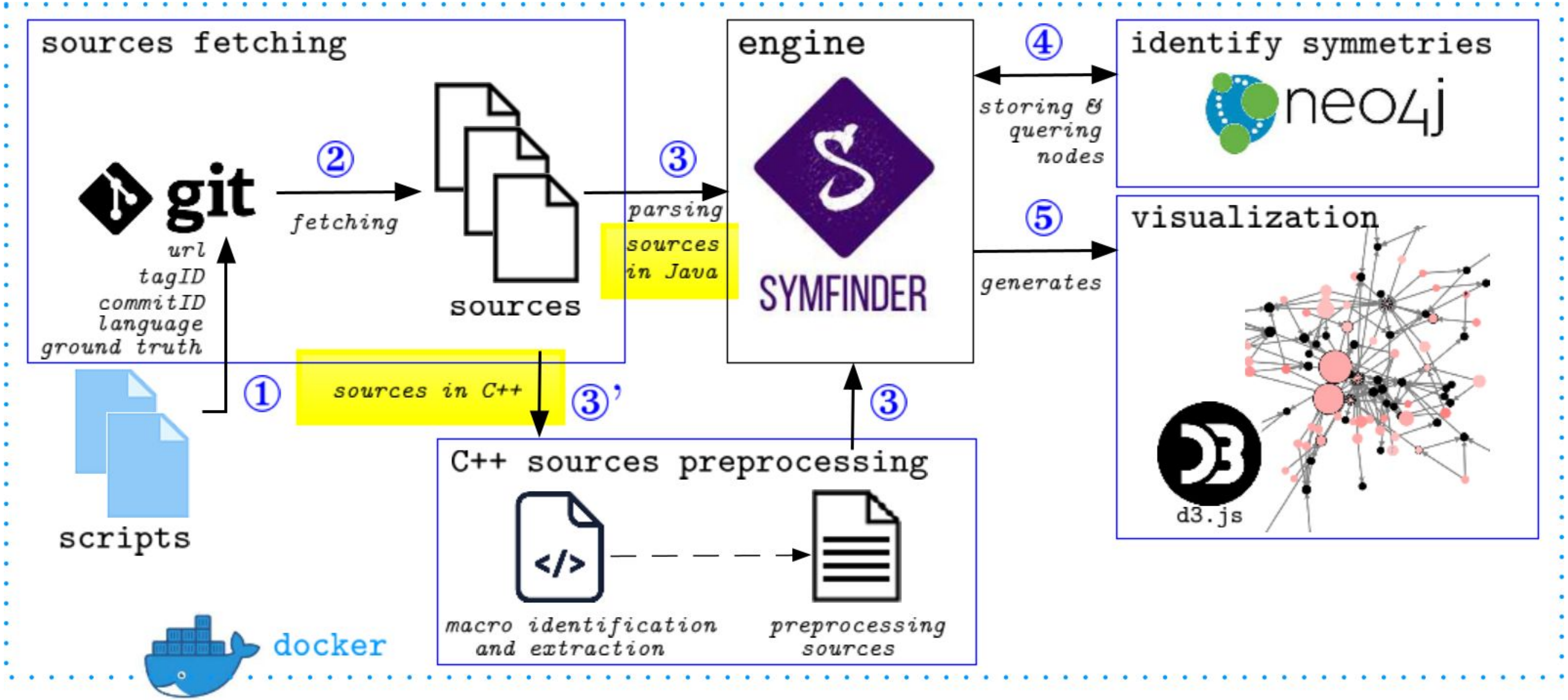
Time units: millisecond, day, month...

symfinder – 2019 version

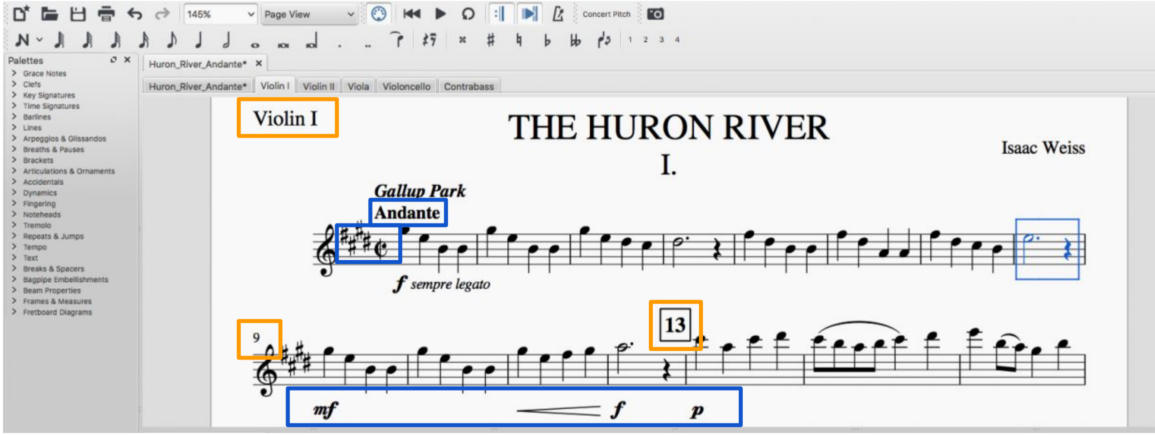


Johann Mortara, Xhevahire Tërnava, and Philippe Collet. 2019. symfinder: A Toolchain for the Identification and Visualization of Object-Oriented Variability Implementations. In 23rd International Systems and Software Product Line Conference - Volume B (SPLC '19), September 9–13, 2019, Paris, France.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

symfinder – 2020 version



Project ID card - MuseScore



©MuseScore

Description: Free music notation and composition software

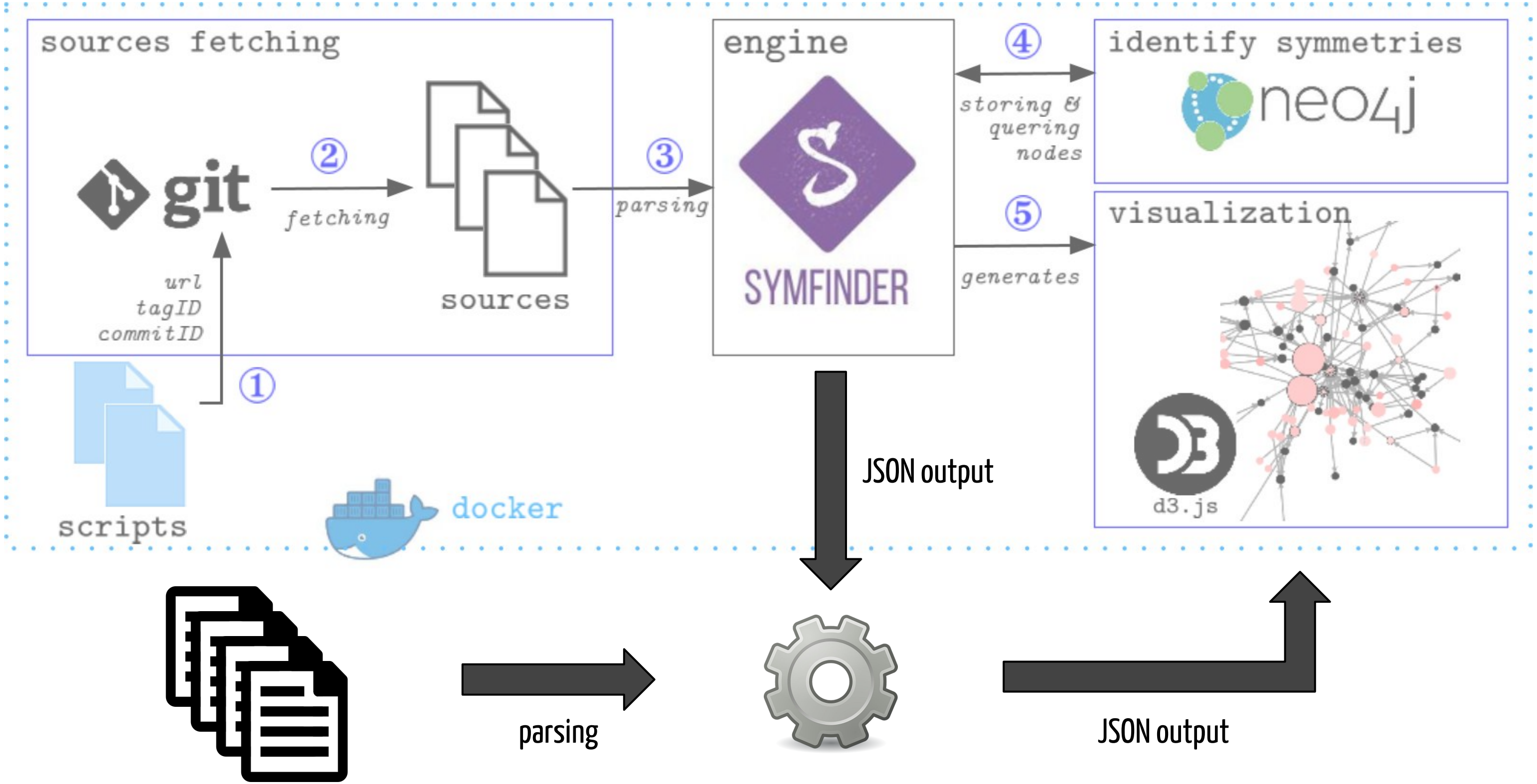
Language: C++

Expected variability implementations: elements composing a music sheet

Musical elements: key, time signature, tempo, wedges...

Textual elements: instrument name, measure number, harmony, fingering...

Automated mapping with features traces



JSON output

parsing

JSON output

Files containing features traces

Automated mapping

Project ID card - ArgoUML-SPL

Description: SPL extracted from ArgoUML, a UML diagramming application.

Language: Java

Features traces are available

Main features from the extracted domain:

- Draw UML diagrams
  - activity
  - collaboration
  - deployment
  - sequence
  - state
  - use case

```
ACTIVITYDIAGRAM.txt:

[…]
org.argouml.uml.diagram.state.ui.FigTransition
org.argouml.uml.ui.behavior.state_machines.ActionNewCallEvent createEvent(Object)
org.argouml.uml.diagram.ui.ActionAddConcurrentRegion Refinement
org.argouml.uml.diagram.ui.ActionAddConcurrentRegion isEnabled() Refinement
[…]
```

Identifying and Mapping Implemented Variabilities in Java and C++ Systems using symfinder

Get the Paper

Johann Mortara – Philippe Collet – Xhevahire Tërnava

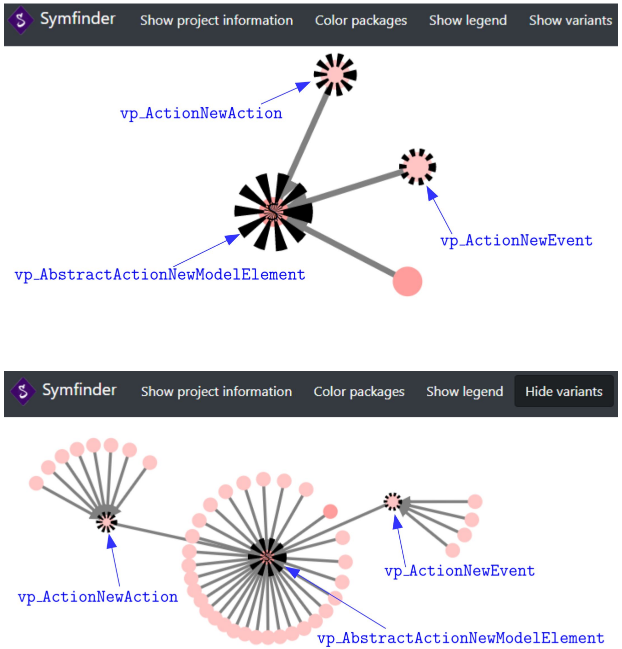
Update QR Code

symfinder website:
https://deathstar3.github.io/symfinder-demo/

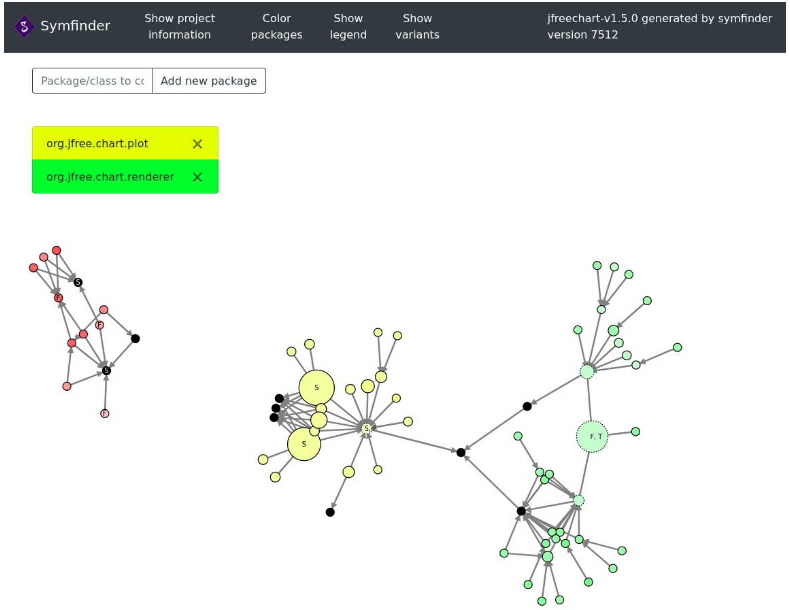
Live demo page:
https://deathstar3.github.io/symfinder-demo/splc2020.html

GitHub repository to get symfinder:
https://github.com/DeathStar3/symfinder

Visualization improvements



Allowing display of all variants on the visualization



jfreechart-v1.5.0 generated by symfinder version 7512

org.jfree.chart.plot ✕

org.jfree.chart.renderer ✕

Coloring nodes belonging to a given package (Java) or namespace (C++)

Automated mapping with features traces

image symfinder

○ *vp* / variant

○ *vp* / variant mapped to at least one trace

Variability-rich OO (Java or C++) system in a <u>single</u> code base

SYMFINDER

Identification of symmetries

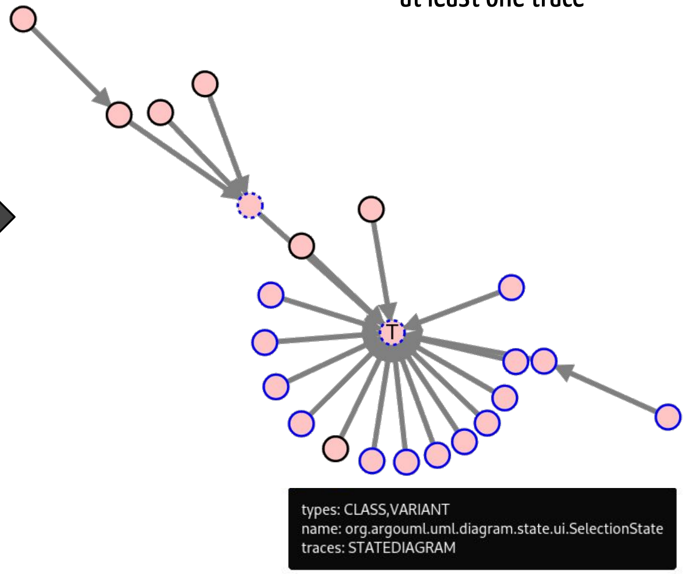
JSON output

JSON output

Files containing features traces

Automated mapping

JSON output



types: CLASS,VARIANT
name: org.argouml.uml.diagram.state.ui.SelectionState
traces: STATEDIAGRAM

Visualization of identified *vp*-s and variants mapped to traces with a blue border