Identifying and Visualizing Variability in Object-Oriented Variability-Rich Systems

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

[1] Sorbonne Université, Paris, France

[2] Université Côte d'Azur, Sophia Antipolis, France

SPLC'19, Paris - September 12, 2019

Problem

Identifying variability places in reusable code assets

Context: The implementation of a variability-rich system in a single code base

Problem

Identifying variability places in reusable code assets

Context: The implementation of a variability-rich system in a single code base

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

```
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)
  }
}
```

vp_draw

- Diverse techniques
  - → Lack of approaches on identifying **variation points** with **variants**

Problem

Identifying variability places in reusable code assets

Context: The implementation of a variability-rich system in a <u>single</u> code base

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);
  }
```

vp_draw

```java
  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)
  }
}
```

- Diverse techniques

  → Lack of approaches on identifying **variation points** with **variants**

But, do they have a common property so they can be uniformly identified?

Theory of centers

Assumption: Variation points are **centers** of *attention and activity* in design

Theory of centers

Assumption: Variation points are **centers** of *attention and activity* in design

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

Christopher Alexander

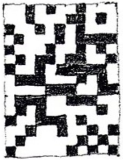
Books 1 - 4: **"The Nature of Order"**; $>$ 30 years; $\approx$ 2.500 pages

Theory of centers

Assumption: Variation points are **centers** of *attention and activity* in design

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

Christopher Alexander



random
hard to describe

ordered
easy to describe

There are **15 ways** for making a center, and...

A **center** is commonly formed by a **local symmetry**

Books 1 - 4: **"The Nature of Order"**; > 30 years; ≈ 2.500 pages

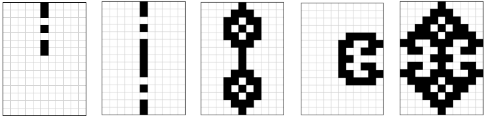
Symmetry in nature and human-made artifacts

Symmetry represents immunity to a possible change

Symmetry in nature and human-made artifacts

Symmetry represents immunity to a possible change

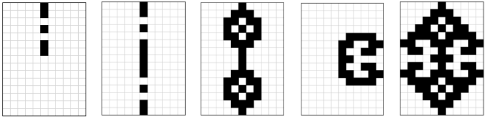
Local symmetries:
$\rightarrow$ it's all about their density!



Symmetry in nature and human-made artifacts

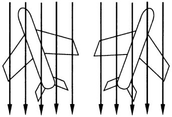
Symmetry represents immunity to a possible change

Local symmetries:
→ it's all about their density!



It's "everywhere"... and also in code






Symmetry in software constructs

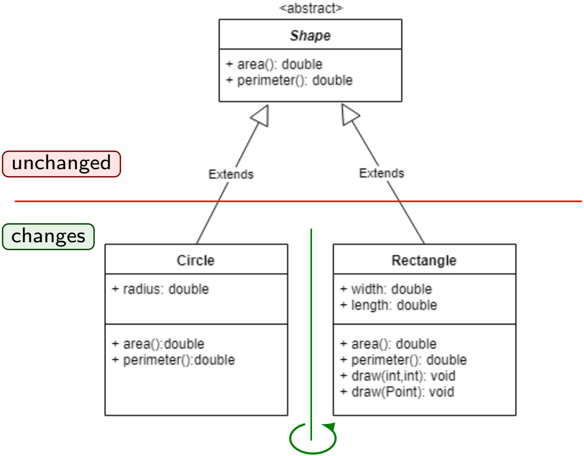
We rely on previous work on symmetry in software

- **Symmetry breaking in software patterns.** James Coplien and Liping Zhao. 2000. In International Symposium on Generative and Component-Based SE.
- **Symmetry in class and type hierarchy.** Liping Zhao and James Coplien. 2002. In Proceedings of the Fortieth International Conference on Tools Pacific.
- **Understanding symmetry in object-oriented languages.** Liping Zhao and James Coplien. 2003. Journal of Object Technology.
- **Patterns, symmetry, and symmetry breaking.** Liping Zhao. 2008. ACM.
- **Toward a general formal foundation of design. Symmetry and broken symmetry.** James Coplien and Liping Zhao. 2019. Monograph (Working draft).

...and extend it in SPL engineering

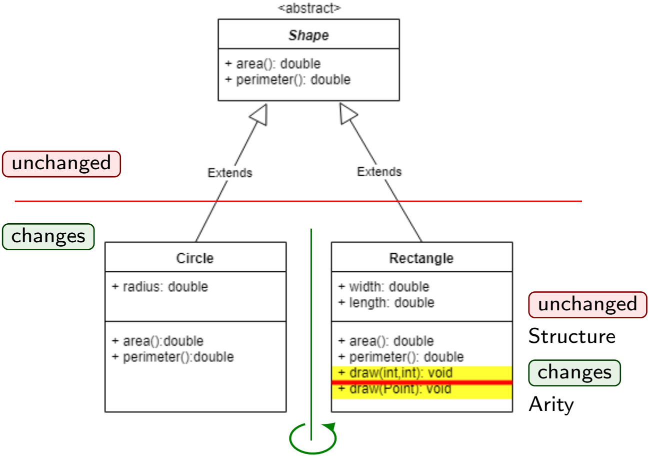
Symmetry in software constructs

Symmetry in subtyping (inheritance)



Symmetry in software constructs

Symmetry in overloading



Identifying variation points with variants

Variability implementation technique	$\leftrightarrow$	$\boxed{\text{local symmetry}}$
■ variation point (commonality)	$\leftrightarrow$	$\boxed{\text{unchanged}}$
■ variants (variability)	$\leftrightarrow$	$\boxed{\text{changed}}$

Identifying variation points with variants

Variability implementation technique $\leftrightarrow$ local symmetry

- variation point (commonality) $\leftrightarrow$ unchanged
- variants (variability) $\leftrightarrow$ changed

$\rightarrow$ variation points with variants can be uniformly identified by simply identifying local symmetries in core assets

Identifying variation points with variants

Variability implementation technique $\leftrightarrow$ local symmetry

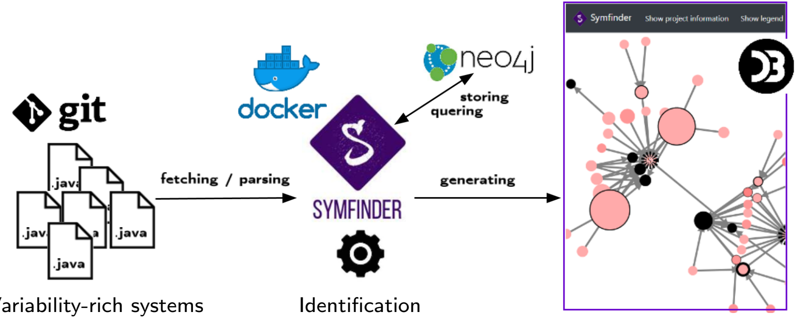
- variation point (commonality) $\leftrightarrow$ unchanged
- variants (variability) $\leftrightarrow$ changed

$\rightarrow$ variation points with variants can be uniformly identified by simply identifying local symmetries in core assets

Symmetry in 9 techniques:

- Class as type
- Class subtyping
- Method overriding
- Method overloading
- Strategy pattern
- Factory pattern
- Decorator pattern
- Template pattern
- Observer pattern

Automatic identification of vp-s with variants



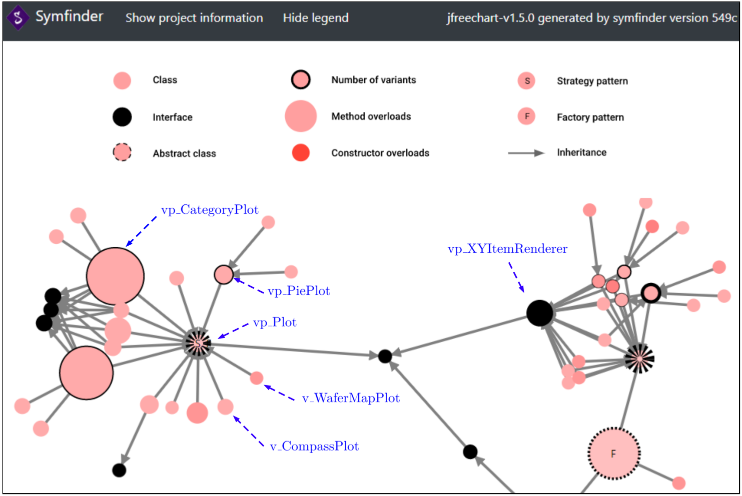
Variability-rich systems in a single code base

Identification of symmetries

Visualization of vp-s

Automatic visualization of variation points

Example: JFreeChart (tag v1.5.0)



Validation

8 case studies: Java, open-source, git, variability-rich

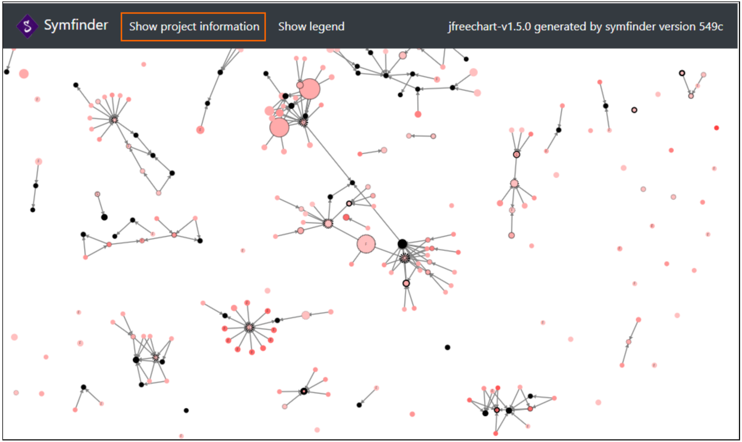
	Case study	**Anaysed LoC**
tag	Java AWT	69,974
tag	Apache CXF 3.2.7	48,655
tag	JUnit 4.12	9,317
tag	Apache Maven 3.6.0	105,342
tag	JHipster 2.0.28	2,535
tag	JFreeChart 1.5.0	94,384
commit	JavaGeom	32,755
commit	ArgoUML	178,906

Validation

Metric 1: **#places** with a higher density of **vp-s**

Metric 2: **#vp-s** and **#variants**, at method and class level, for each

Filtering out vp-s
#vp-s with #variants

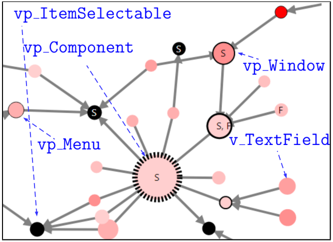


Validation

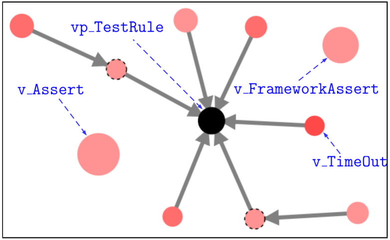
Three discerned patterns of variability

1. Places with a higher density of variability at method level have a higher density of variability at class level

Java AWT



JUnit 4.12

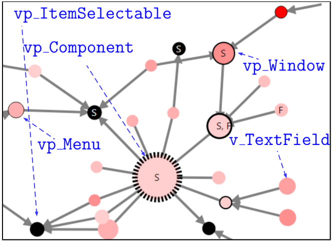


Validation

Three discerned patterns of variability

1. Places with a higher density of variability at method level have a higher density of variability at class level
2. **#vp-s** seems highly correlated with **#LoC**

Java AWT

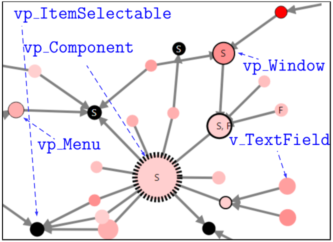


Validation

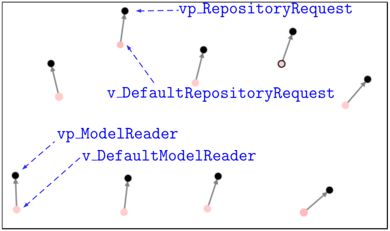
Three discerned patterns of variability

1. Places with a higher density of variability at method level have a higher density of variability at class level
2. **#vp-s** seems highly correlated with **#LoC**
3. Smaller [larger] number of trees but a higher [lower] density of vp-s
   → Code bases more [less] variability-rich

Java AWT



Maven 3.6.0



Future Work

- Identification of symmetry in other language features
- Building **symfinder** as a GitHub App
- Exploit other software metrics to discern other patterns of variability
- Extrapolate the other 14 centers' properties, *e.g.*, good shape

Summary

Identifying and Visualizing Variability in Object-Oriented Variability-Rich Systems

An automatic **identification** and **visualization** of different kinds of vp-s with variants, through *local symmetry*, in a uniform way

Validated in **8 Java-based systems**
Developed **2 metrics**: density and #vp-s
Discerned **3 first patterns of variability**

Availability

- Public release: tag **splc2019-artifact**
  https://github.com/DeathStar3/symfinder
- **symfinder** demonstration
  https://deathstar3.github.io/symfinder-demo/

Get the Paper