POLYTECH
NICE SOPHIA

Une aiguille dans une botte de code

Rétro-Ingénierie, Maintenance et Évolution du Logiciel

Johann Mortara

19/01/2021

Mon parcours

2014-2016 : PeiP à Polytech Nice Sophia

2016-2019 : Sciences Informatiques à Polytech Nice Sophia (Parcours AL en 5A)

2019 : Thèse au laboratoire I3S

Mon parcours

2014-2016 : PeiP à Polytech Nice Sophia

2016-2019 : Sciences Informatiques à Polytech Nice Sophia (Parcours AL en 5A)

2019 : Thèse au laboratoire I3S

Identification, visualisation et gestion de variabilité au sein de grands systèmes orientés objets hautement variables

Mon parcours

2014-2016 : PeiP à Polytech Nice Sophia

2016-2019 : Sciences Informatiques à Polytech Nice Sophia (Parcours AL en 5A)

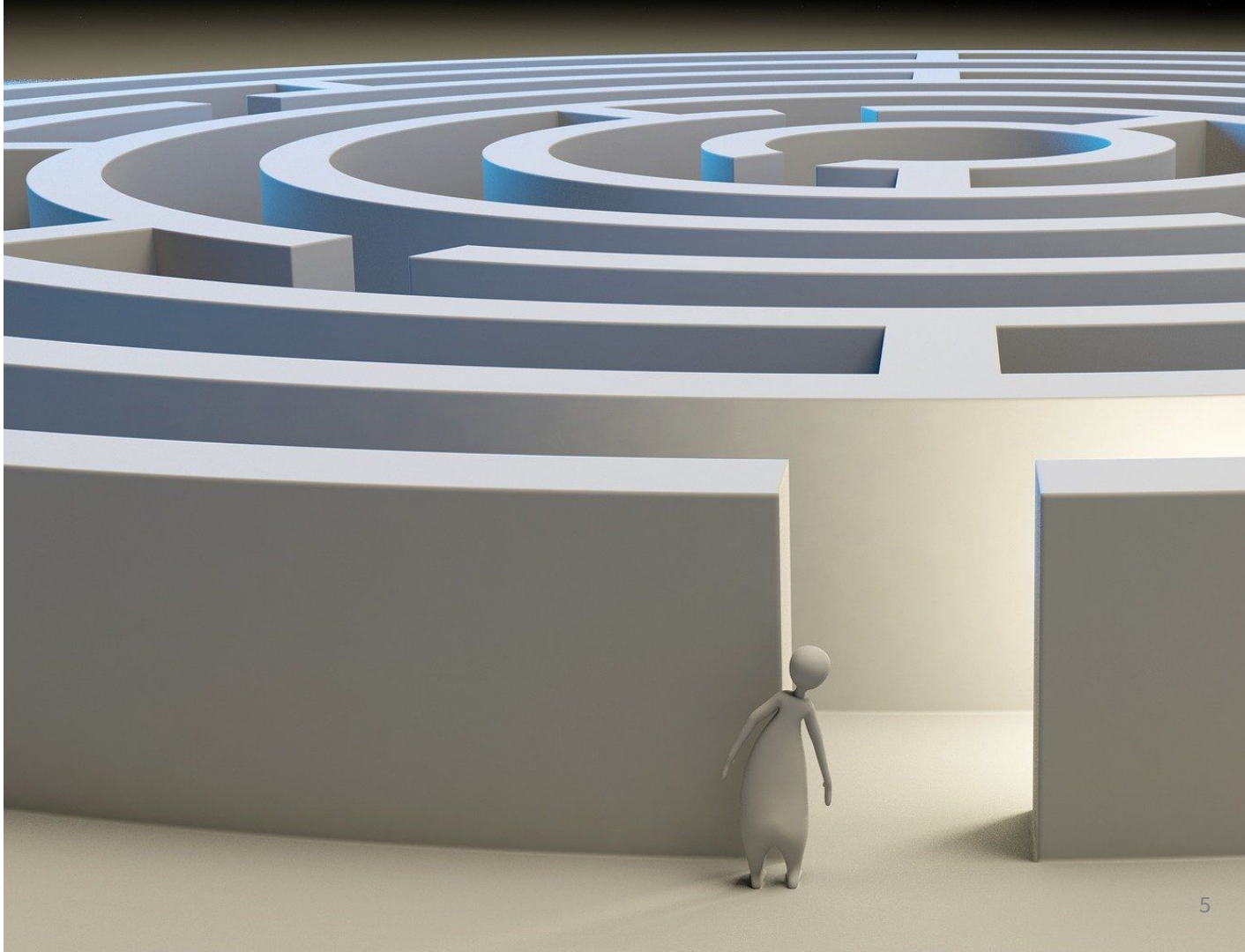
2019 : Thèse au laboratoire I3S

Identification, visualisation et gestion de variabilité au sein de grands systèmes orientés objets hautement variables

Pardon ?

1.

Quel est le problème ?



Source : Image by Arek Socha from Pixabay

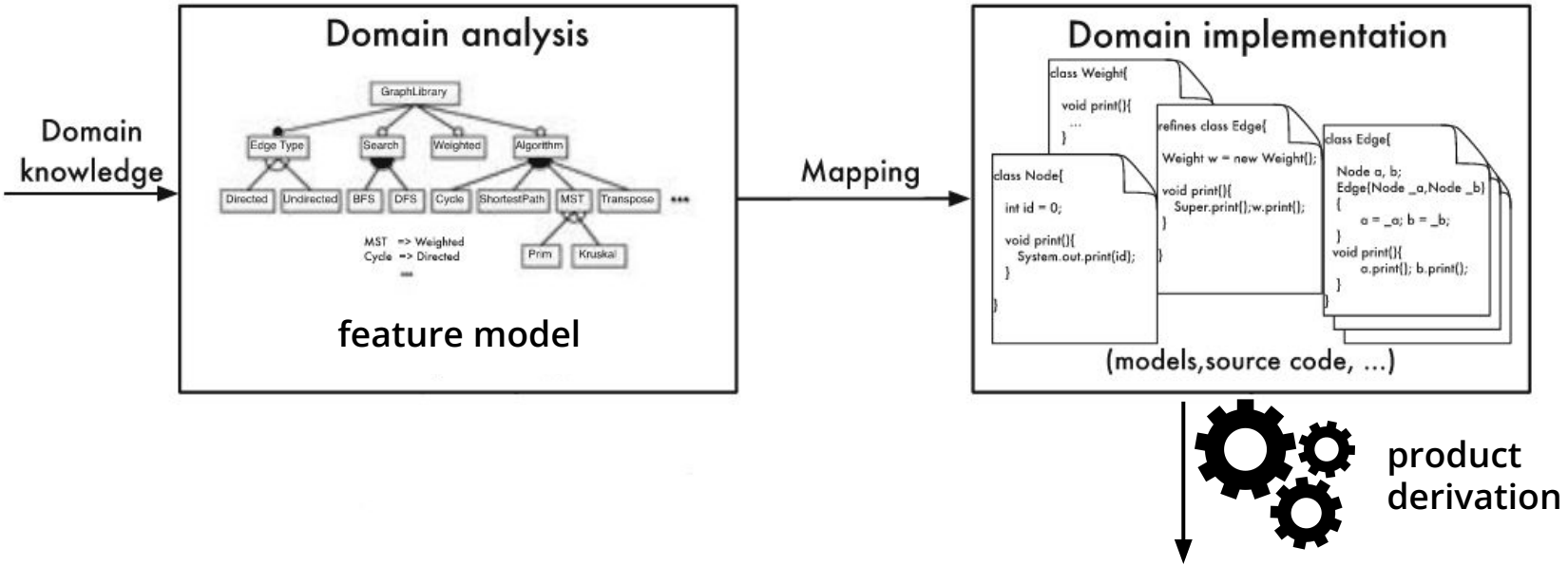
Vari-quoi ?

Variabilité : Capacité d'un logiciel à être étendu ou configuré pour un contexte précis [Svahnberg2005]

Exemples :

- support de plusieurs langues
- différentes connexions aux serveurs de mails d'un client mail (POP3, SMTP...)
- extensions des fonctionnalités avec des plugins
- ...

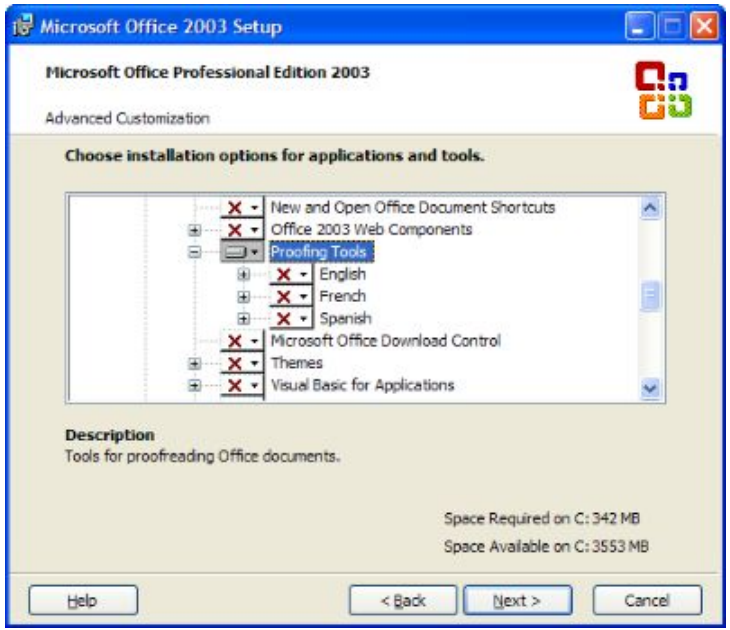
Les Lignes de Produits Logiciels (SPL)



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

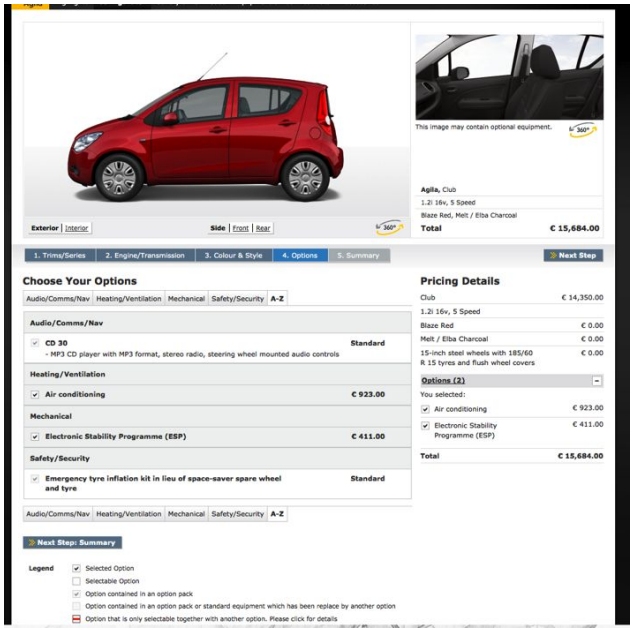
Exemples de lignes de produits

Des produits logiciels…



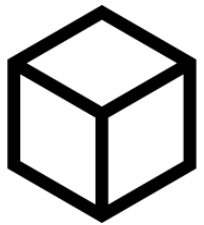
© Microsoft

… mais pas seulement !



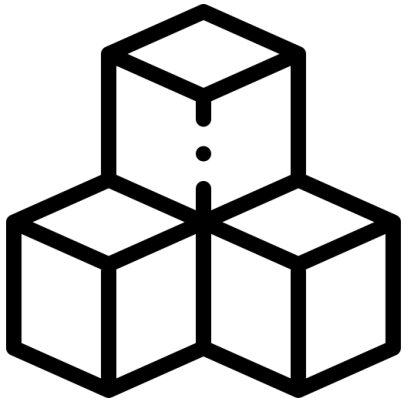
© Opel

Il était une fois...



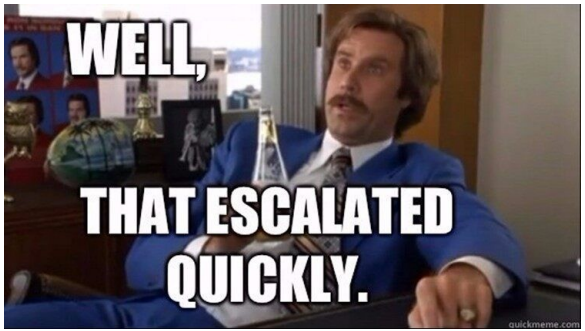
Votre petit projet

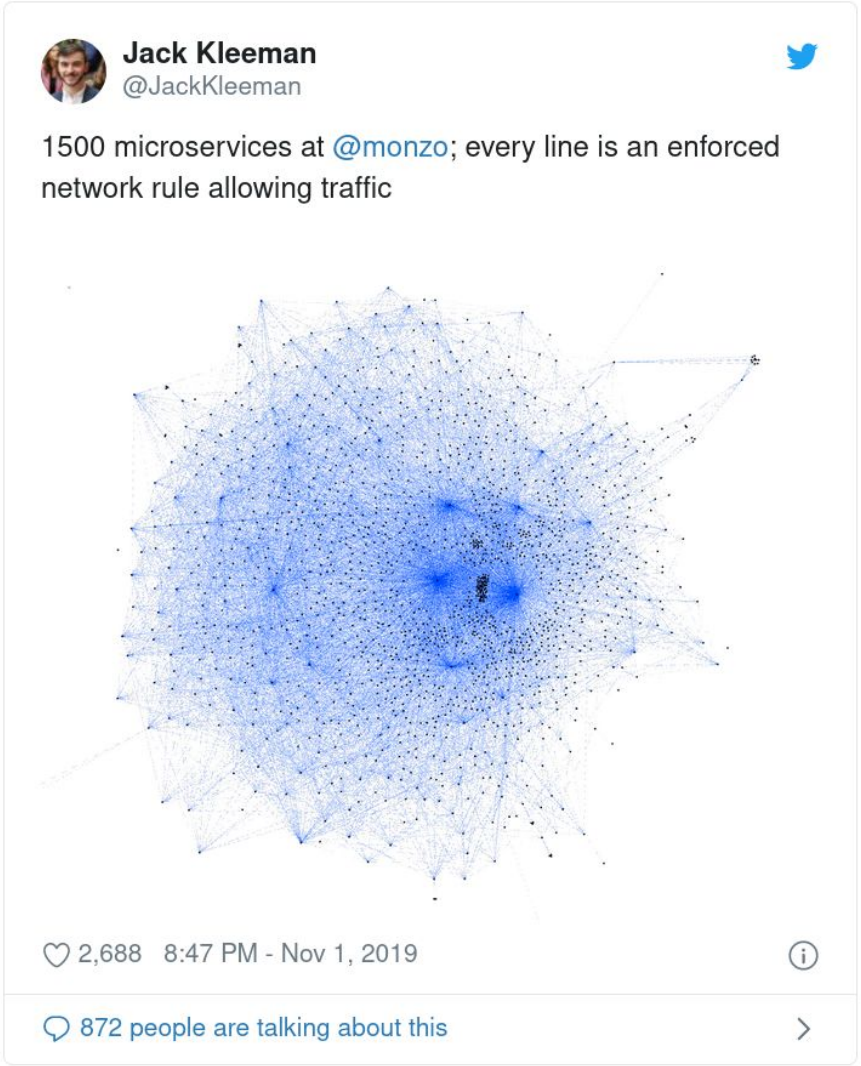
“Ah tiens et si on faisait ça aussi ?”



Votre *“petit”* projet

Et là, c'est le drame !





Source : https://twitter.com/JackKleeman/status/1190354757308862468?s=09

Quelques systèmes hautement variables

16.000 options gérées dans 25M LoC [Acher2018]

24.000 différentes plateformes en 2015 [Open2015]

2.000+ options générant des variantes pour différentes plateformes, niveaux de sécurité... [Acher2018]

Quelques systèmes hautement variables

16.000 options gérées dans 25M LoC [Acher2018]

24.000 différentes plateformes en 2015 [Open2015]

2.000+ options générant des variantes pour différentes plateformes, niveaux de sécurité… [Acher2018]

⇒ pas de modèle formel ⇒ pas de SPL !

Que faire pour revenir à une SPL ?

On reprend tout à zéro :

- analyse du domaine
- écriture du code de chaque fonctionnalité
- ...

Feature model ⟶ Implémentation

Que faire pour revenir à une SPL ?

On reprend tout à zéro :

- analyse du domaine
- écriture du code de chaque fonctionnalité
- ...

Feature model ⟶ Implémentation

On identifie les implémentations de variabilité présentes pour reconstruire un *feature model*.

Feature model ⟵ Implémentation

Que faire pour revenir à une SPL ?

On reprend tout à zéro :

- analyse du domaine
- écriture du code de chaque fonctionnalité
- ...

Feature model ⟶ Implémentation

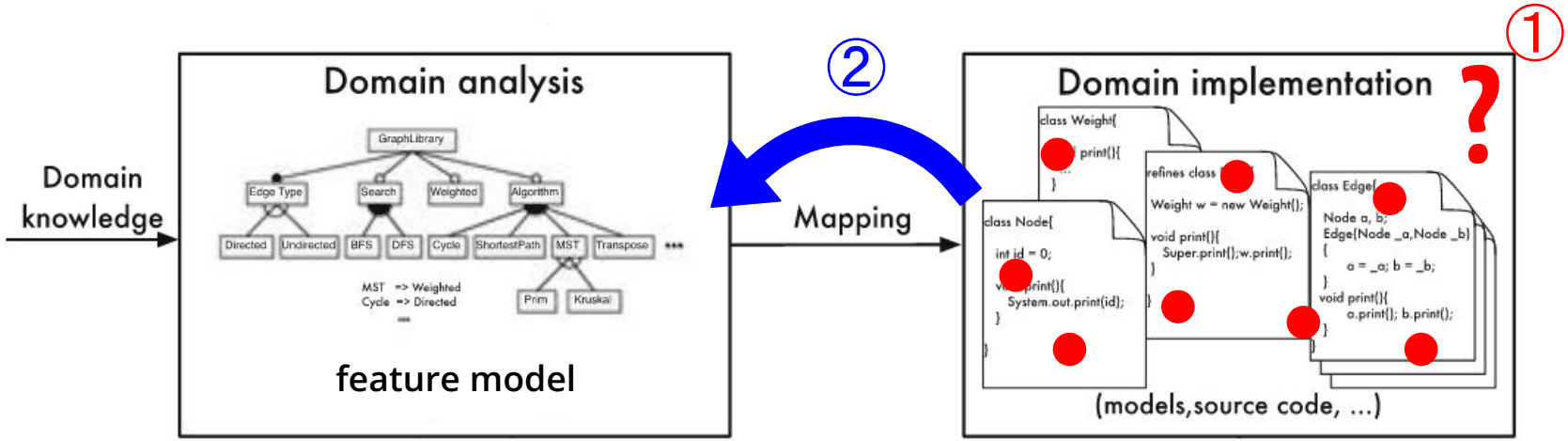
On identifie les implémentations de variabilité présentes pour reconstruire un *feature model*.

Feature model ⟵ Implémentation

⇒ **rétro-ingénierie**

On identifie les implémentations de variabilité présentes

pour les relier à un feature model



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

2.

Comment s'y prend-on ?

Découpons le problème !

Source : Image by cpenzin from Pixabay

Identifier les implémentations de variabilité présentes pour les relier à un feature model

Les implémentations de variabilité sont enfouies dans le code, il nous faut donc les identifier.

Données à notre disposition : code source du système étudié

⇒ 1ère sous-question : Comment identifier des implémentations de variabilité d'un système en ayant pour seules données son code source ?

Identifier les implémentations de variabilité présentes **pour les relier à un feature model**

Les implémentations de variabilité implémentent la variabilité (c'est fou non ?)

i.e. ces implémentations peuvent être reliées à des fonctionnalités du domaine (liées au métier), donc à des fonctionnalités d'un feature model

⇒ Besoin de s'assurer que les implémentations identifiées sont correctes

⇒ 2e sous-question : Est-ce que les implémentations de variabilité identifiées correspondent vraiment à de la variabilité ?

Identifier les implémentations de variabilité présentes pour les relier à un feature model
Pourquoi ?

Zones fortement variables représentent du code complexe

⟶ points d'intérêt du code qui doivent être connus des architectes / développeurs pour assurer leur qualité

⇒ 3^{e} sous-question : Comment indiquer à un utilisateur les zones de forte densité d'un projet ?

Nos sous-questions

1. Comment identifier des implémentations de variabilité d'un système en ayant pour seules données son code source ?
2. Est-ce que les implémentations de variabilité identifiées correspondent vraiment à de la variabilité ?
3. Comment indiquer à un utilisateur les zones de forte densité d'un projet ?

3.1

À la recherche de la variabilité perdue

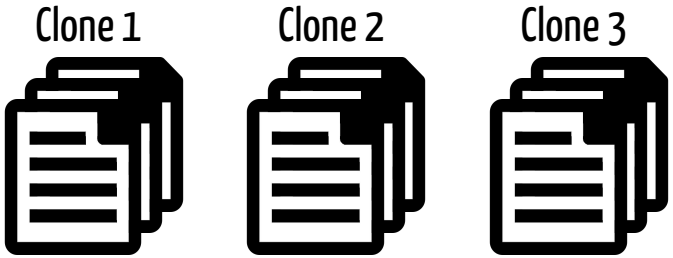
Source : Image by Mila Kusmenko from Pixabay

Étape 1 : que sait-on faire ?

Rechercher dans l'état de l'art (articles scientifiques...) si des techniques existent déjà

Technique 1 : comparaison de clones

Comparaison entre les clones et lien avec les fonctionnalités [Assunção2017]



Peut-on s'en servir ?

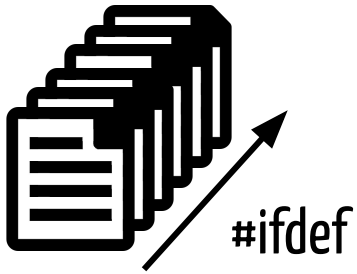
Non, car la variabilité d'un système OO est gérée la plupart du temps en une unique base de code.

Étape 1 : que sait-on faire ?

Rechercher dans l'état de l'art (articles scientifiques...) si des techniques existent déjà

Technique 2 : directives de préprocesseurs / annotations dans une unique base de code

Identifier les directives ou annotations et les relier à des features [Liebig2010]



Peut-on s'en servir ?

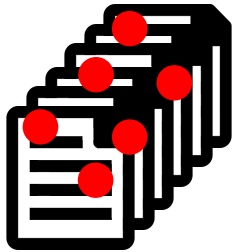
Pas toujours, car la majorité des systèmes OO n'ont pas ces informations.

Étape 1 : que sait-on faire ?

Rechercher dans l'état de l'art (articles scientifiques...) si des techniques existent déjà

Notre contexte :

- Une unique base de code
- Pas d'annotations / directives



Comment faire ?

Besoin de trouver une technique se reposant uniquement sur le code du système.

Étape 2 : que cherche-t-on exactement ?

Question : Comment les systèmes OO implémentent leur variabilité ?

Exemples :

- paramètres
- propriétés
- annotations (ex : #ifdefs)

Exemples spécifiques aux systèmes orientés objet :

- **héritage / implémentation d'interfaces**
- **surcharge de méthodes**
- **surcharge de constructeurs**
- **patrons de conception**

Étape 2 : que cherche-t-on exactement ?

Question : Comment les systèmes OO implémentent leur variabilité ?

On observe manuellement !

Points de variation et variantes

```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)
  }
}
```

Points de variation et variantes

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)
  }
}
```

Points de variation et variantes

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)
  }
}
```

Étape 2 : que cherche-t-on exactement ?

Question : Comment les systèmes OO implémentent leur variabilité ?

Intuition d'après les observations : utilisation des mécanismes OO

- héritage / implémentation d'interfaces
- surcharge de méthodes
- surcharge de constructeurs
- patrons de conception

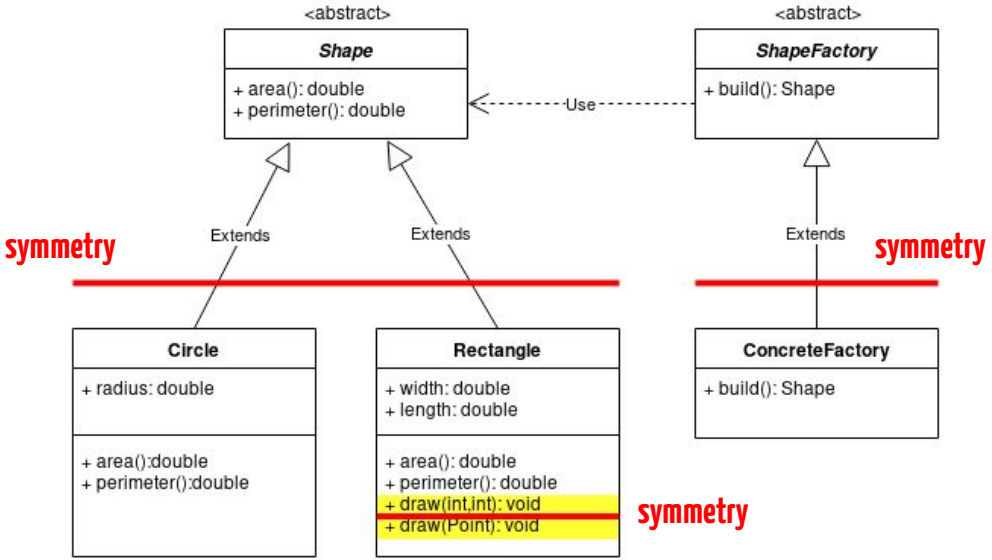
⇒ Nouvelle question : Comment identifier ces implémentations de variabilité ?

Intuition

- Présence de **symétries dans des bases de code orientées objets** [Coplien2019] inspiré de la théorie des centres de Christopher Alexander [Alexander2002].

- Ces symétries sont présentes dans les **mécanismes d'implémentation de la variabilité**.

⇒ Utilisation des symétries pour détecter les implémentations de variabilité ?



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.

Intuition

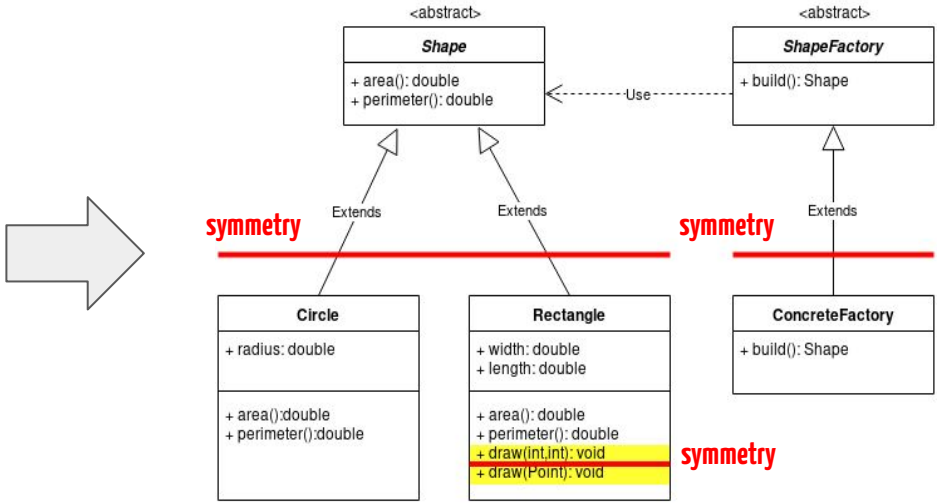
Code source

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

```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) { // Point defined
  // rectangle at (p.x, p.y, width, length)
  }
}
```

```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;
  }
}
```

Identification des symétries



Étape 3 : vérification de l'intuition

On commence petit !

1. Prendre un petit projet qui a les propriétés recherchés,
2. Vérifier qu'on retrouve bien ces propriétés.

Choix d'un projet

Caractéristiques recherchées :

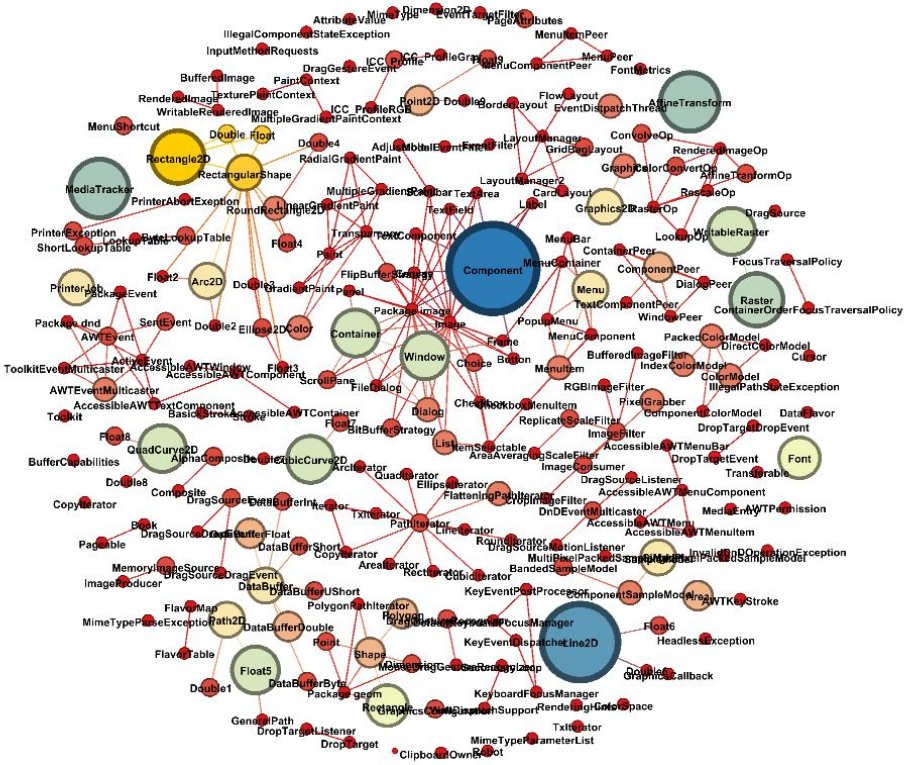
- implémenté dans un langage orienté objet (ex : Java)
- une seule base de code
- intuition de la présence de variabilité

Candidat choisi : **Java AWT**

- Partie graphique du JRE
- Permet de créer des environnements graphiques avec différents types de composants ⟵ **variabilité**

Premières expérimentations

- Identification manuelle des points de variation
- Première visualisation de la variabilité sous la forme d'un graphe montre des zones de haute densité de variabilité



Nos sous-questions

1. Comment identifier des implémentations de variabilité d'un système en ayant pour seules données son code source ? Sur un projet, la densité de symétries semble être un moyen viable pour identifier des points de variation et leurs variantes.

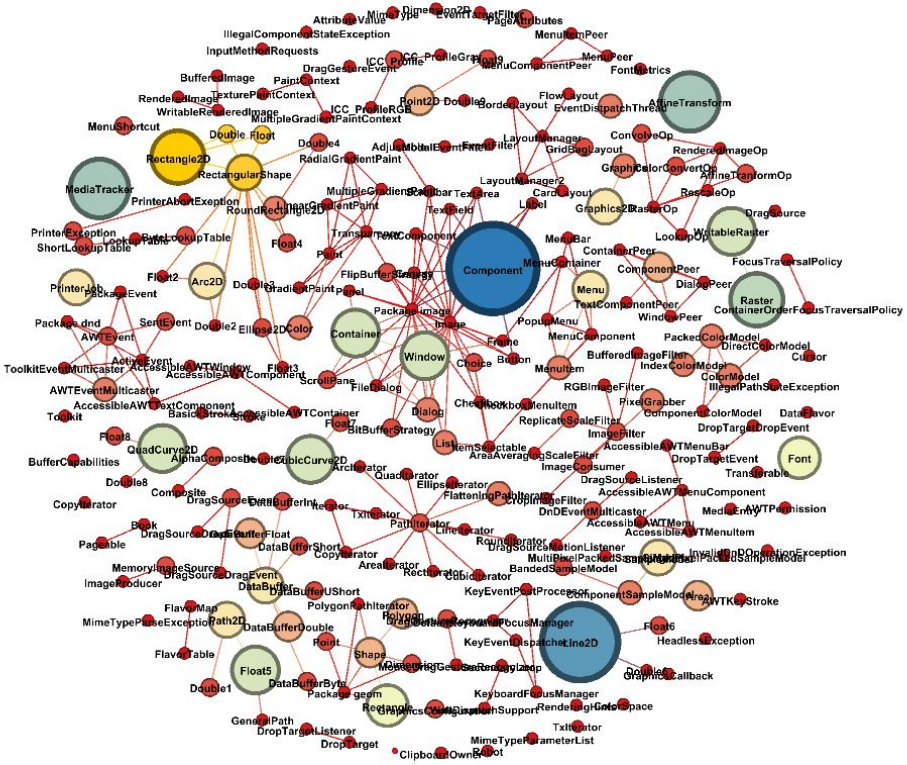
2. Est-ce que les implémentations de variabilité identifiées correspondent vraiment à de la variabilité ? **Bonne question…**

3. Comment indiquer à un utilisateur les zones de forte densité d'un projet ? Sur un projet, un graphe semble approprié.

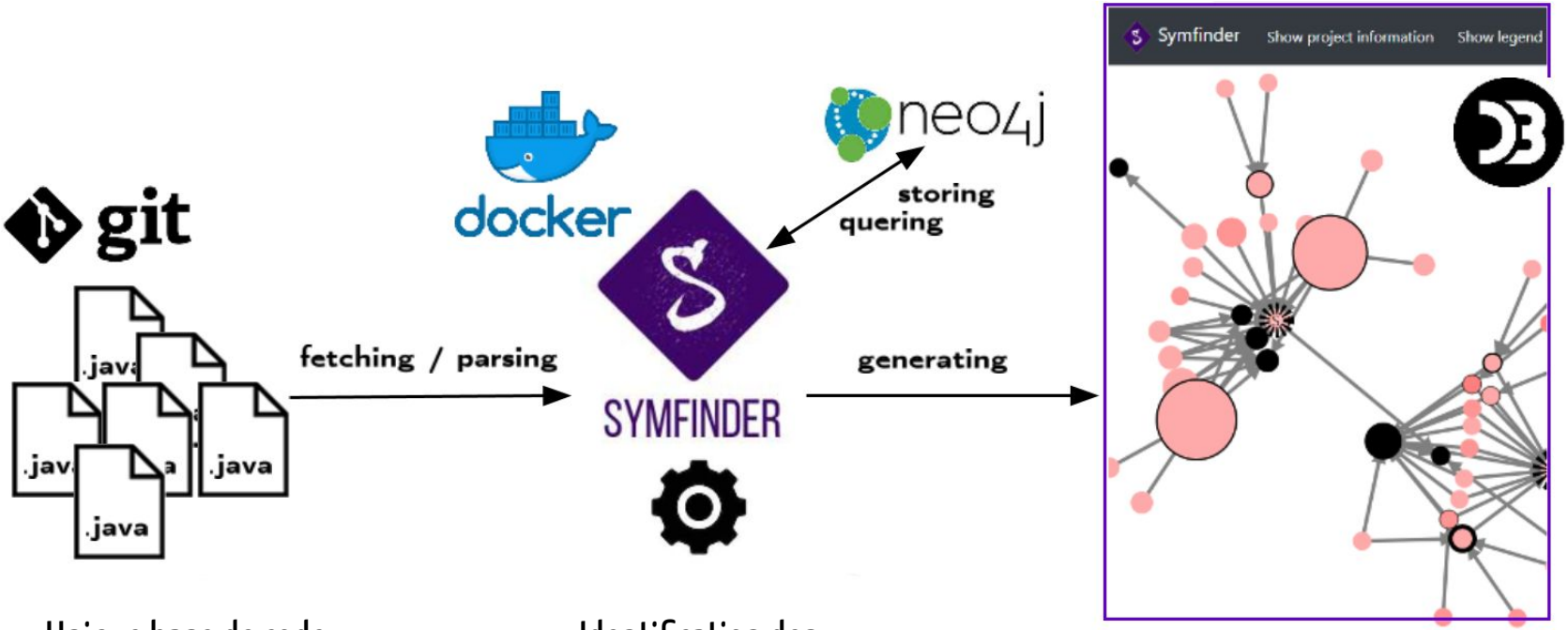
Premières expérimentations

- Identification **manuelle** des points de variation
- Première visualisation de la variabilité

⇒ besoin d'une solution automatisée pour valider l'intuition sur d'autres projets



symfinder



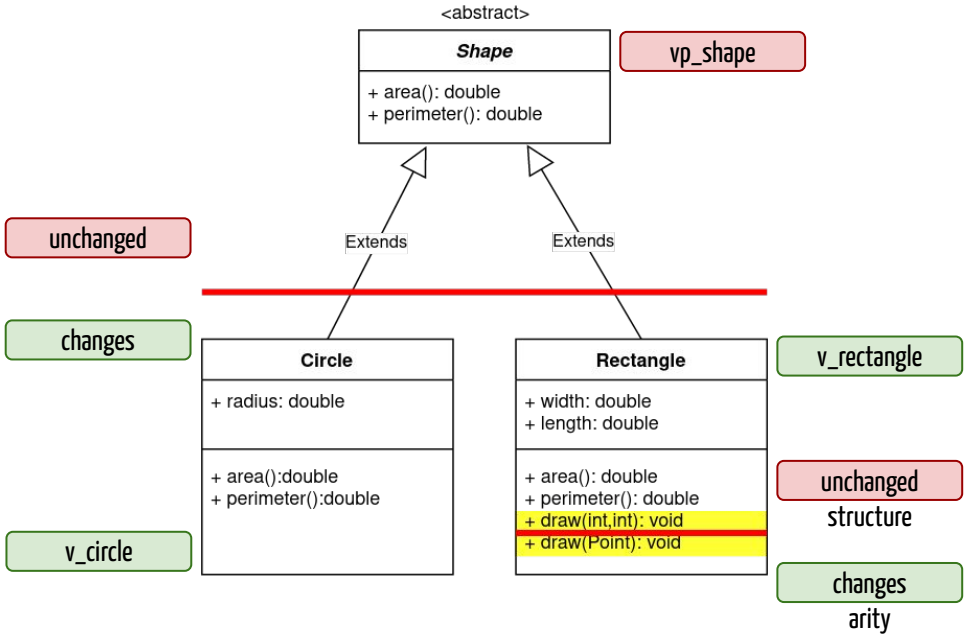
Unique base de code d'un système variable

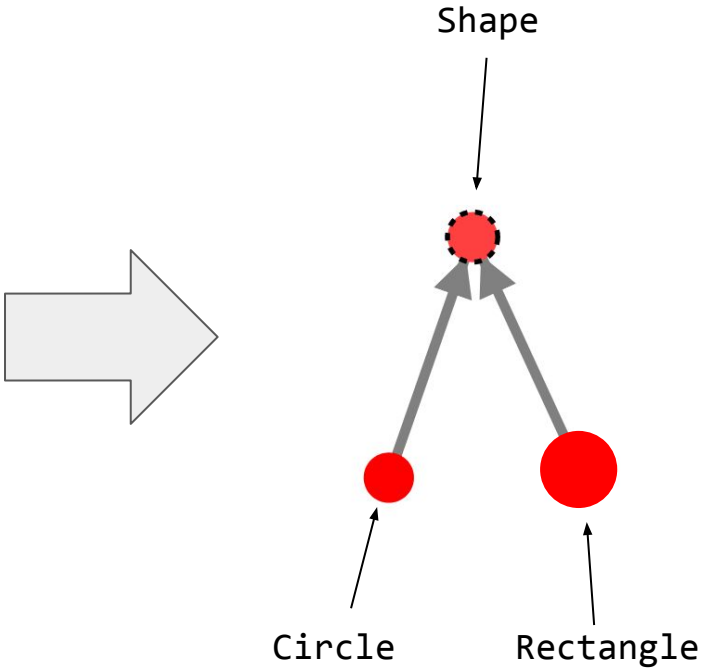
Identification des symétries

Visualisation des *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.

Visualisation d'un petit exemple





Vérification de l’approche symfinder

On exécute symfinder sur des projets qui ont (potentiellement) les caractéristiques recherchées !

JFreeChart

Bibliothèque permettant de tracer différents types de graphiques

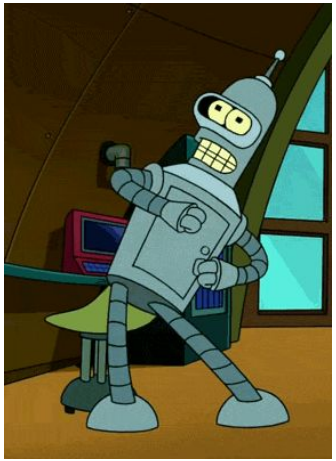
JFreeChart

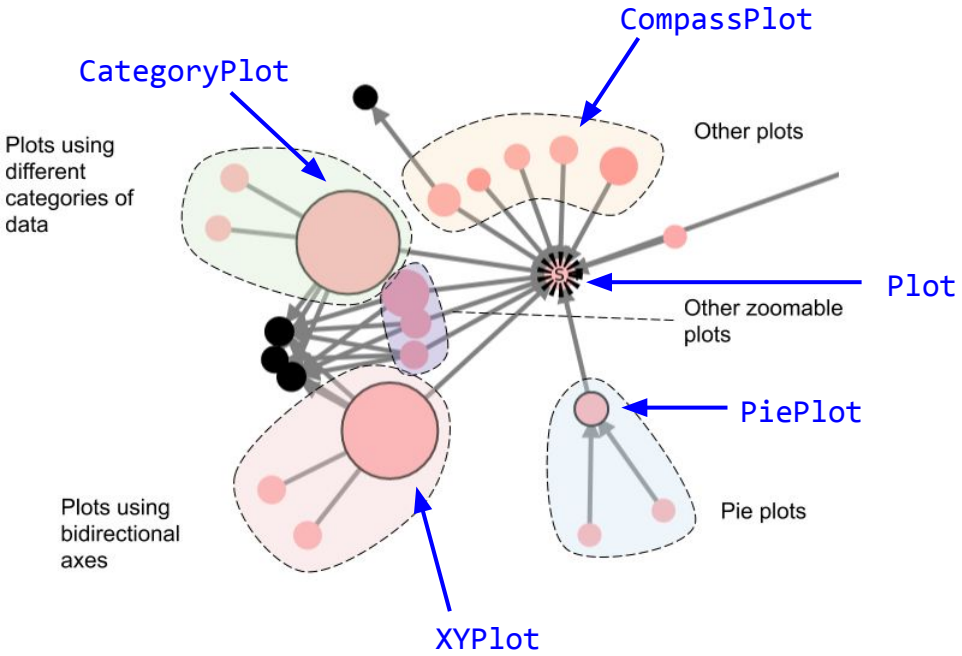
Bibliothèque permettant de tracer **différents types de graphiques**

variabilité ?

JFreeChart

Bibliothèque permettant de tracer **différents types de graphiques** ← **variabilité !**





JHipster

Outil de configuration de projets à partir d'un choix de pile technologique

Construit comme une ligne de produit

JHipster

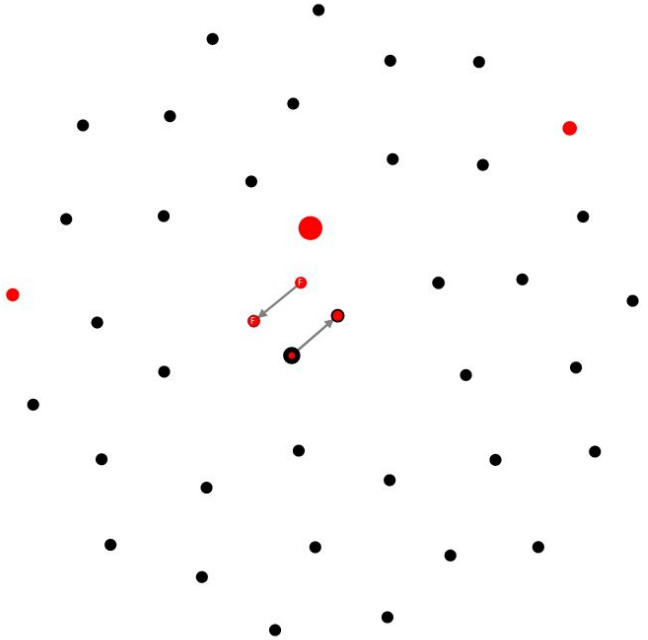
Outil de configuration de projets à partir d'un choix de pile technologique

Construit comme une ligne de produit ⟵ **variabilité ?**

JHipster

Outil de configuration de projets à partir d'un choix de pile technologique

Construit comme une ligne de produit ← ~~variabilité~~



Que s'est-il passé ?

JHipster projet multi-langages et sur plusieurs dépôts ⟶ variabilité éparpillée

Bilan néanmoins positif !

Prouve qu'on ne voit rien quand il n'y a rien à voir.

⟶ Pas de faux-positifs

Des projets similaires mais variés

Système	# LoC	# *vp*-s	# variantes
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

Nos sous-questions

1. Comment identifier des implémentations de variabilité d'un système en ayant pour seules données son code source ? La densité de symétries semble être un moyen viable pour identifier des points de variation et leurs variantes.

2. Est-ce que les implémentations de variabilité identifiées correspondent vraiment à de la variabilité ? Bonne question…

3. Comment indiquer à un utilisateur les zones de forte densité d'un projet ? Un graphe semble approprié.

3.2

Comment vérifier la pertinence de notre détection ?

Source : Image by StockSnap from Pixabay

Question : Est-ce qu'on peut relier nos vp-s et variantes à des fonctionnalités ?

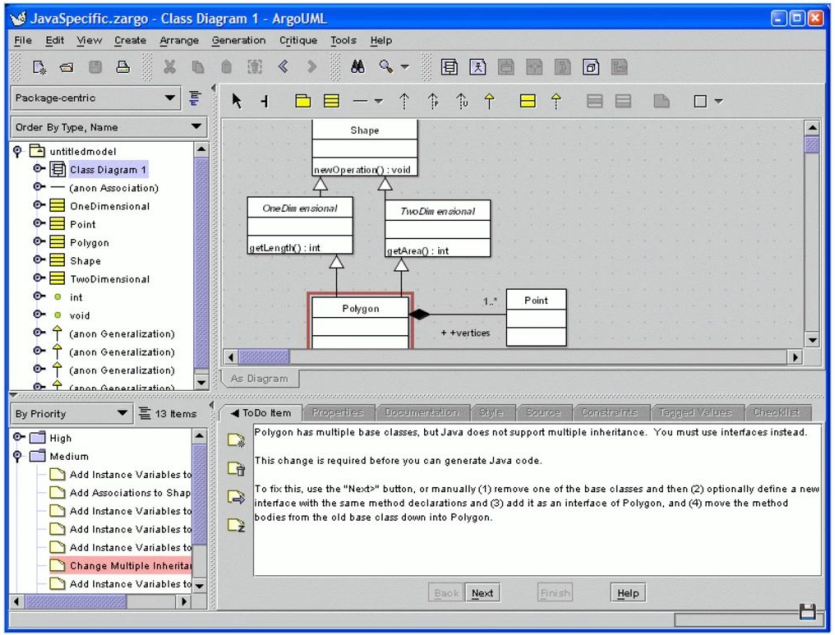
Besoin d'un projet pour lequel on a les informations que l'on cherche (i.e. un mapping implémentation ⟷ fonctionnalité existant, voire un feature model)

Mais quel projet ?

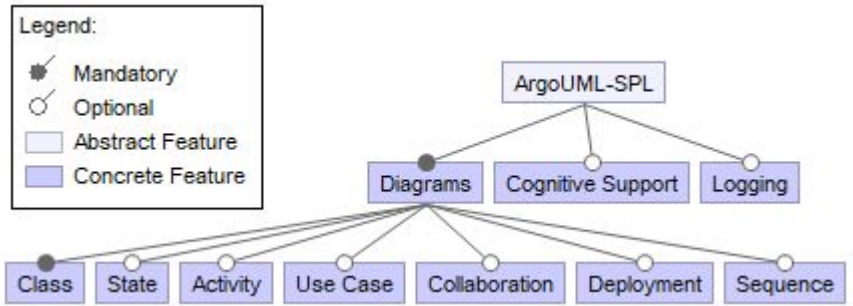
Rechercher dans l'état de l'art !

- indice pour un projet de qualité
- possibilité de comparaison avec d'autres techniques similaires

ArgoUML-SPL [Couto2011]

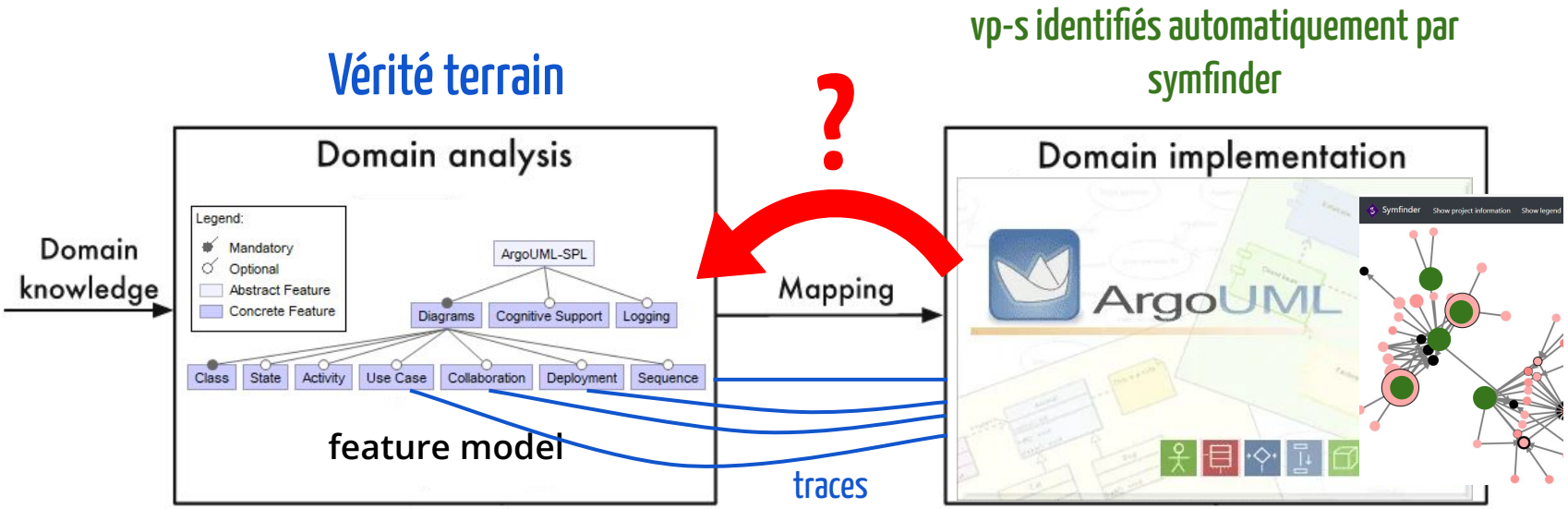


Éditeur d'ArgoUML



Feature model d'ArgoUML-SPL

Question: Est-ce que les *vp-s* identifiés dans ArgoUML correspondent à des fonctionnalités du feature model ?



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

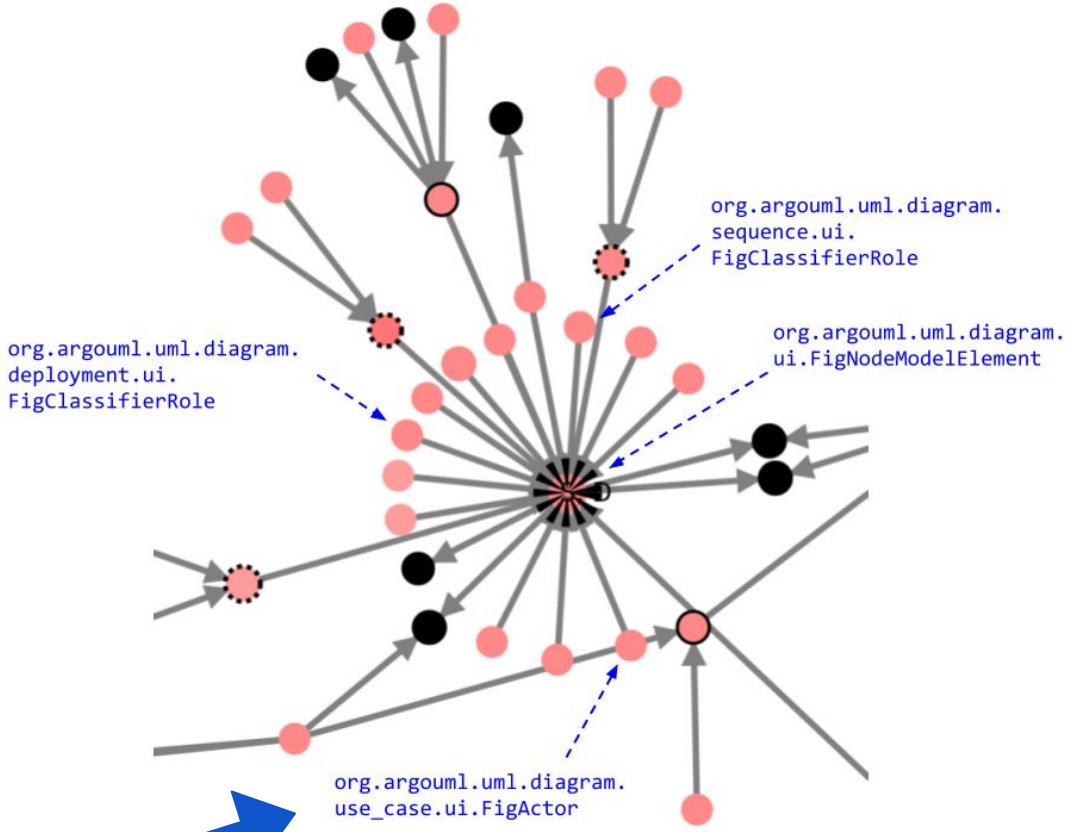
Que voit-on ?

Fonctionnalité : *Sequence*

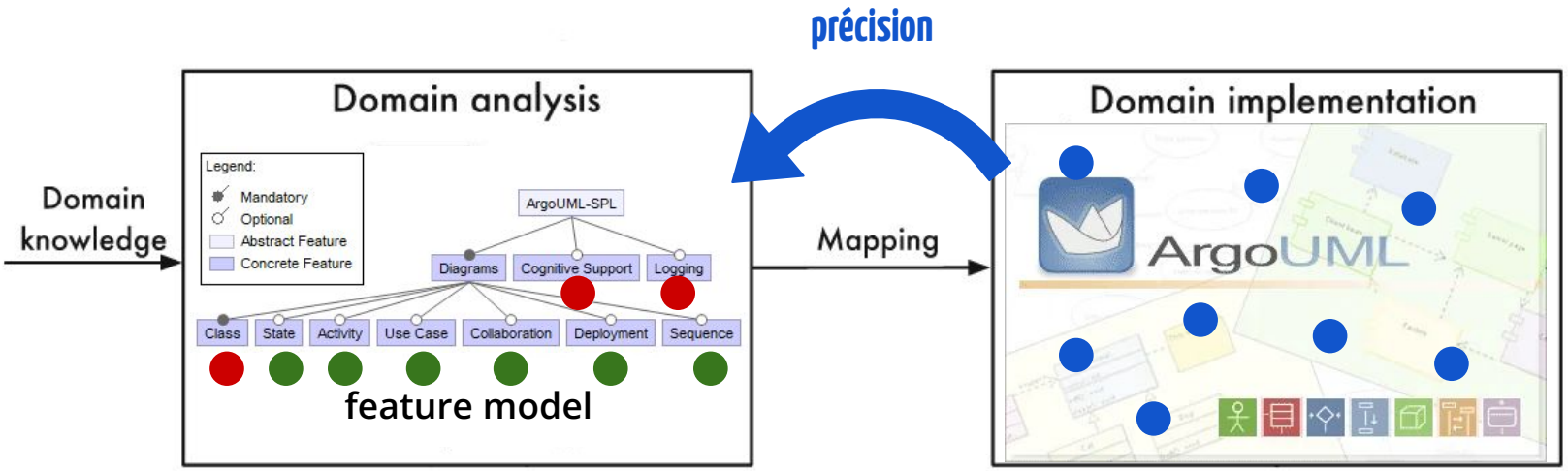
```
//#if defined(SEQUENCEDIAGRAM)
//@#$LPS-SEQUENCEDIAGRAM:GranularityType:Package
public class FigClassifierRole extends FigNodeModelElement
```

Fonctionnalité : *Use Case*

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



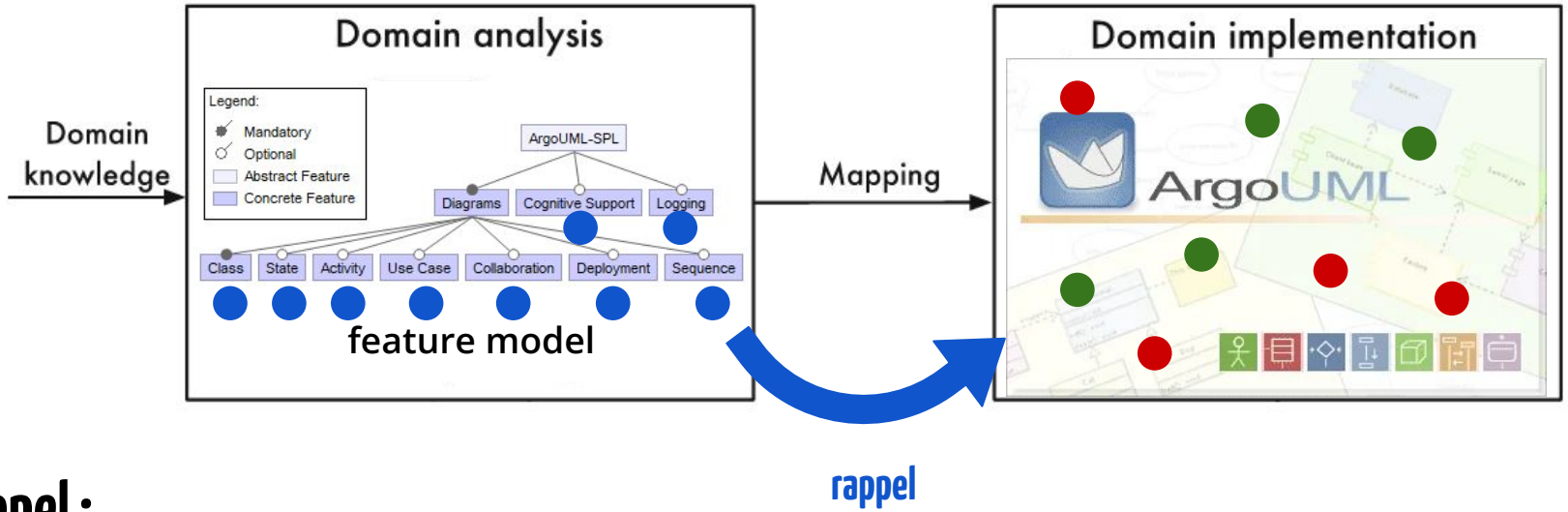
Pertinence des *vp-s*



Précision :

Pourcentage des vp-s et variantes identifiés qui peuvent être reliés à des fonctionnalités

Pertinence des *vp-s*



Rappel :

Pourcentage des traces de fonctionnalités qui peuvent être reliées à des vp-s et variantes identifiés

Résultats obtenus

Précision :

38 %

Résultat faible, mais attendu :

- fonctionnalités à gros grain
- symétries pas uniquement liées à de la variabilité

Rappel :

83 %

Bon résultat !

Analyse manuelle des 17% de traces restantes

⟶ ne sont pas liées à de la variabilité

Johann Mortara, Xhevahire Tërnava, and Philippe Collet. 2020. Mapping Features to Automatically Identified Object-Oriented Variability Implementations: The case of ArgoUML-SPL. In Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS '20), February 5–7, 2020, Magdeburg, Germany. ACM, New York, NY, USA, 9 pages.

Nos sous-questions

1. Comment identifier des implémentations de variabilité d'un système en ayant pour seules données son code source ? La densité de symétries semble être un moyen viable pour identifier des points de variation et leurs variantes.

2. Est-ce que les implémentations de variabilité identifiées correspondent vraiment à de la variabilité ? La majorité des implémentations de variabilité est identifiée, mais beaucoup de faux-positifs le sont également.

3. Comment indiquer à un utilisateur les zones de forte densité d'un projet ? Un graphe semble approprié.

Prenons un peu de recul…

Rappel du sujet initial :

Identification, visualisation et gestion de variabilité au sein de grands systèmes orientés objets hautement variables

Prenons un peu de recul…

Rappel du sujet initial :

Identification, visualisation et gestion de variabilité au sein de

grands systèmes orientés objets hautement variables

Prenons un peu de recul…

Rappel du sujet initial :

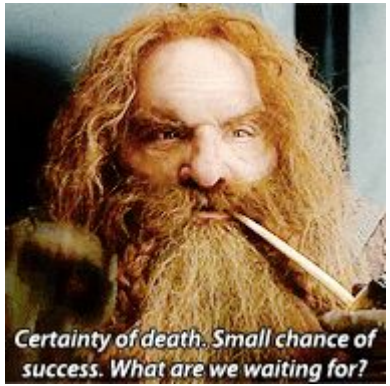
Identification, visualisation et gestion de variabilité au sein de

grands systèmes orientés objets hautement variables

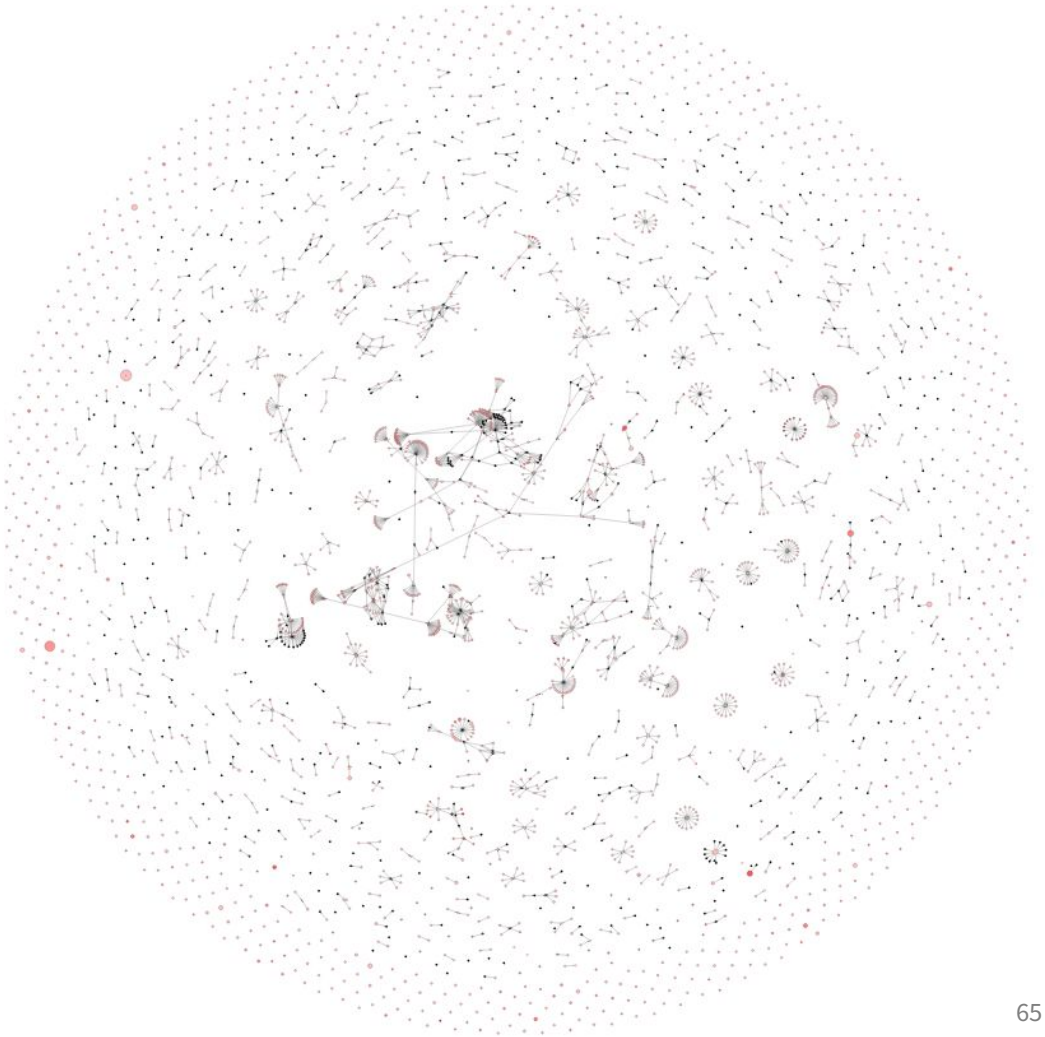
Plus grand système étudié : ArgoUML (179k LoC) ⟶ il nous faut plus grand !

Apache NetBeans

- 4.5 M LoC
- Plus grand projet Java de la fondation Apache
- Présence de variabilité ? *Who knows ?*



Passage à l'échelle de l'outil ✔



Passage à l'échelle de la visualisation ✘

Nos sous-questions

1. Comment identifier des implémentations de variabilité d'un système en ayant pour seules données son code source ? La densité de symétries semble être un moyen viable pour identifier des points de variation et leurs variantes.

2. Est-ce que les implémentations de variabilité identifiées correspondent vraiment à de la variabilité ? La majorité des implémentations de variabilité est identifiée, mais beaucoup de faux-positifs le sont également.

3. Comment indiquer à un utilisateur les zones de forte densité d'un projet ? Un graphe semble approprié sur de petits systèmes mais devient illisible sur de grands systèmes.

3.3

Vers une visualisation plus intuitive

Source : Image by FunkyFocus from Pixabay

Question : Quelle nouvelle visualisation adopter ?

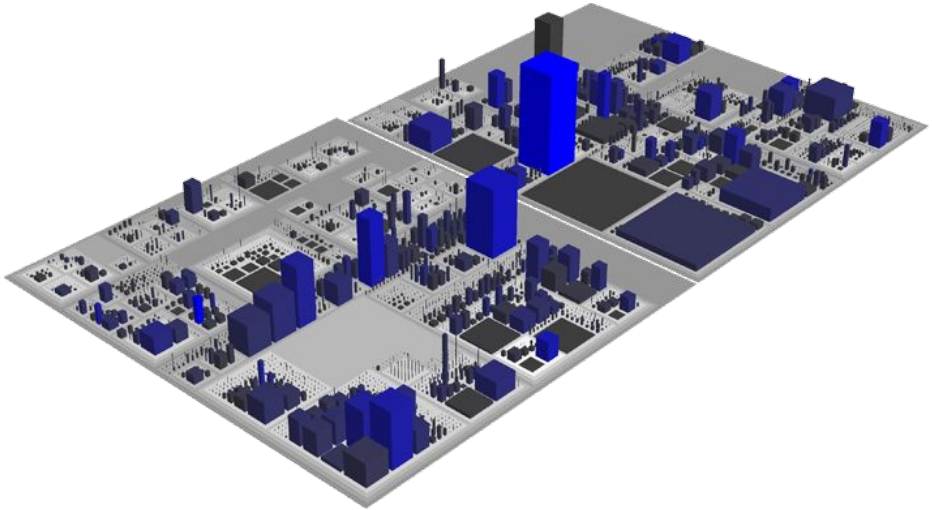
Rechercher dans l'état de l'art !

- Quelles visualisations ont été développées pour aider la compréhension de grands systèmes ?
- Quelles visualisations sont utilisées dans les analyses de systèmes variables ?

CodeCity [Wettel2007] et Evostreet

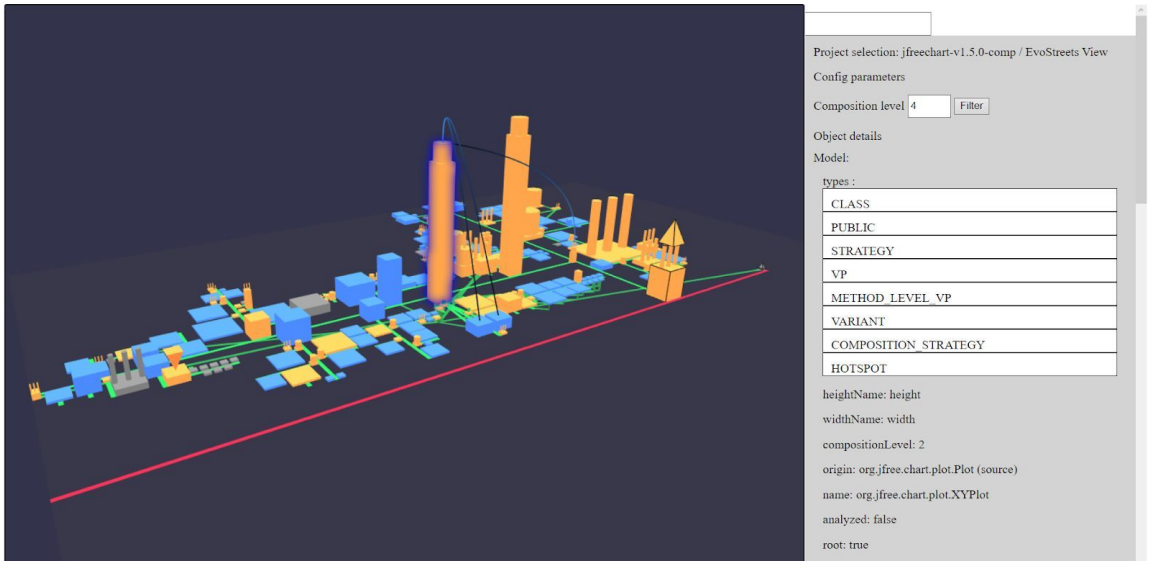
Utilisation de la métaphore de la ville pour représenter un système orienté objet, en se reposant sur des métriques de qualité du code, pour détecter des *code smells*.

Visualisations intégrées à des systèmes de suivi de la qualité du code tels que SonarQube.



From **CodeCity** to **VariCity**

Utilisation de la **métaphore de la ville** pour représenter un système orienté objet, en se reposant sur des **métriques de variabilité**, pour **identifier des concentrations de vp-s et variantes**.



Dédicace à Paul-Marie DJEKINNOU, Florian FOCAS et François RIGAUT !

Nos sous-questions

1. Comment identifier des implémentations de variabilité d'un système en ayant pour seules données son code source ? La densité de symétries semble être un moyen viable pour identifier des points de variation et leurs variantes.

2. Est-ce que les implémentations de variabilité identifiées correspondent vraiment à de la variabilité ? La majorité des implémentations de variabilité est identifiée, mais beaucoup de faux-positifs le sont également.

3. Comment indiquer à un utilisateur les zones de forte densité d'un projet ? Une visualisation sous forme de ville semble donner de bons résultats.

En résumé

Image par kropekk_pl de Pixabay

Quelques conseils

1. Suivre les pistes des intuitions ; on peut être (agréablement) surpris du résultat.
2. Ne pas trop en faire dès le départ ; une hypothèse, bien que simple, peut être suffisante.
3. Un échec n'en est jamais vraiment un.

   ⟶ Rétro-ingénierie à la main pour comprendre pourquoi ça n'a pas marché dans ce cas.
4. Il y a de (très) grandes chances pour que vous ne répondiez pas totalement à vos questions, et c'est normal !

Que faire en cas de problème ?

Réflexe : ~~pleurer~~ comprendre ce qui est arrivé.

⟶ on plonge dans le code pour chercher “à la main” ce que l’outil n’a pas trouvé.

Ce n’est jamais une perte de temps !

On trouve quelque chose ?
⟶ On corrige l’outil.

On ne trouve rien ?
⟶ C’est qu’il n’y a rien à voir (pas de faux-positif) ⟶ tout va bien !

Merci !