Gestion de la variabilité logicielle : Lignes de Produits Logiciels

Tewfik Ziadi

Sorbonne Université/LIP6

tewfik.ziadi@lip6.fr

Two objectives of this presentation

- What are **software variability** and **software product lines**? Where are they? How are they developed? What are the problems, solutions, and challenges?
  - We will presents the **main concepts & tools**
  - We will use **real-world examples**
  - We will be present some **research directions (LIP6)**
  - We will be present some **tools in industry (RedFabRiQ)**

LIVE DEMOS

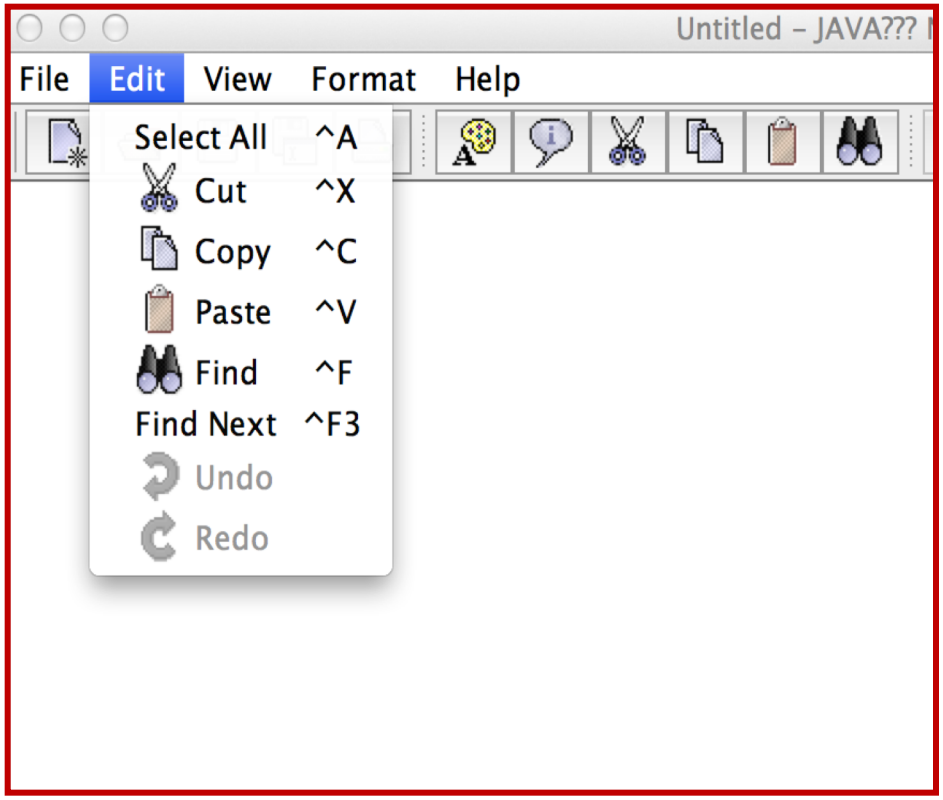
LIP6

SORBONNE UNIVERSITÉ

CNRS dépasser les frontières

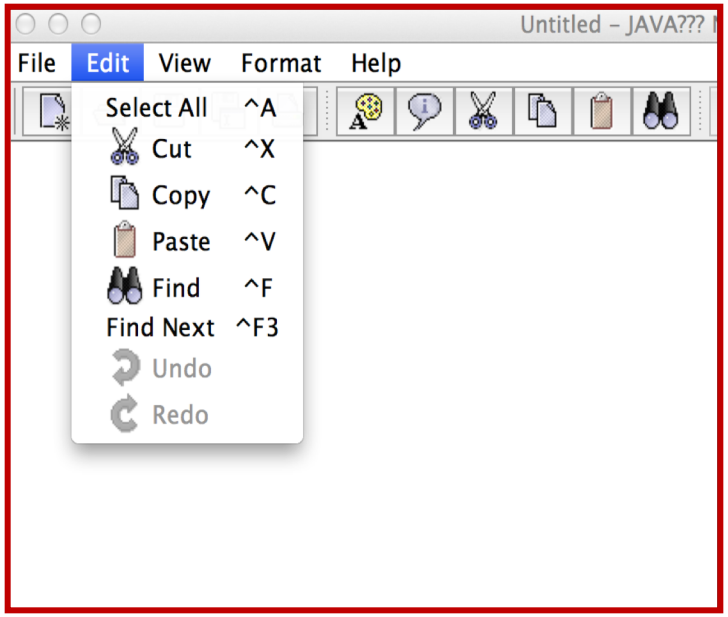
SPL: Motivations

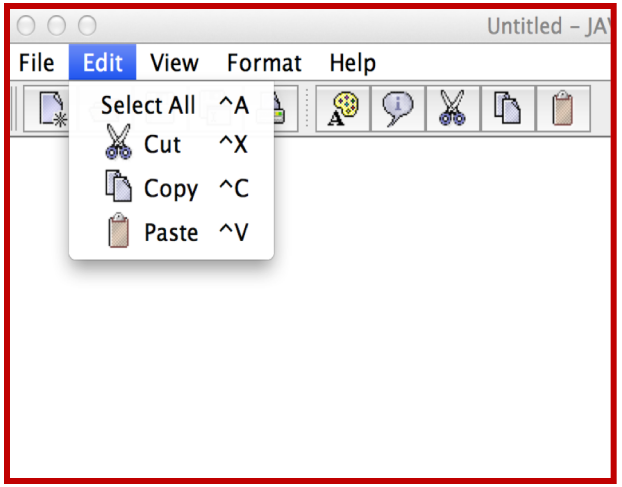
- We have the implementation in Java of a simple Notepad application

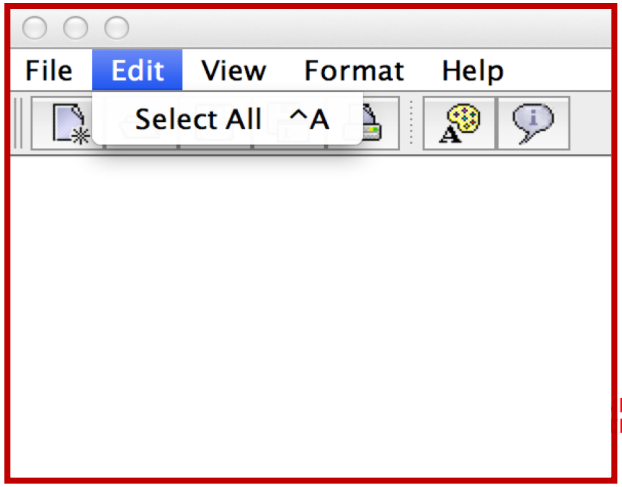


SPL: Motivations

- How can we implement 2 additional variants for the the same application?







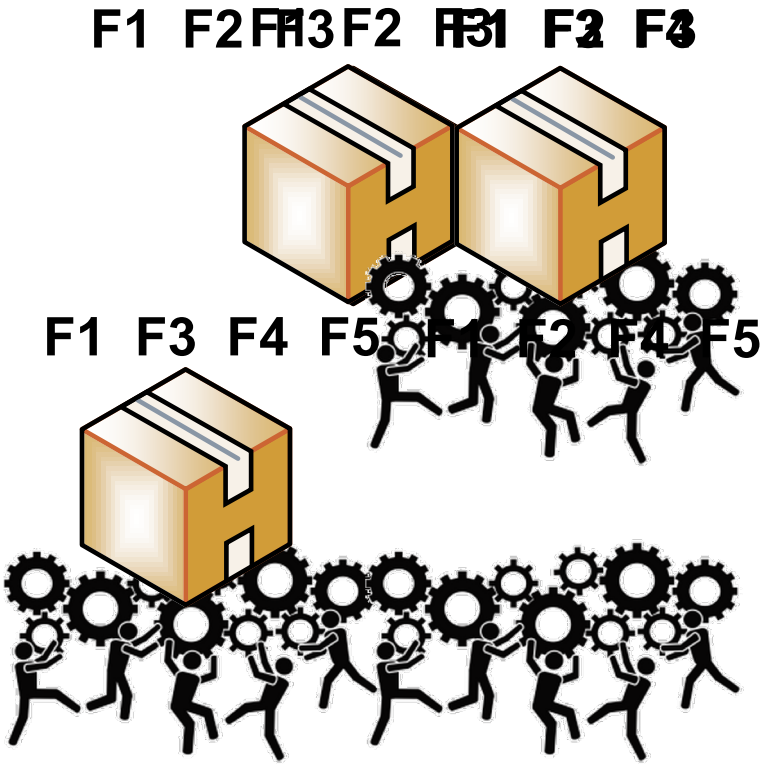
Ad hoc solution

- Copy and past the initial variant and manually delete the unwanted functionalities (copy/cut/finder/undo)

Clone-and-own:

Opportunistic reuse

Clone-and-own



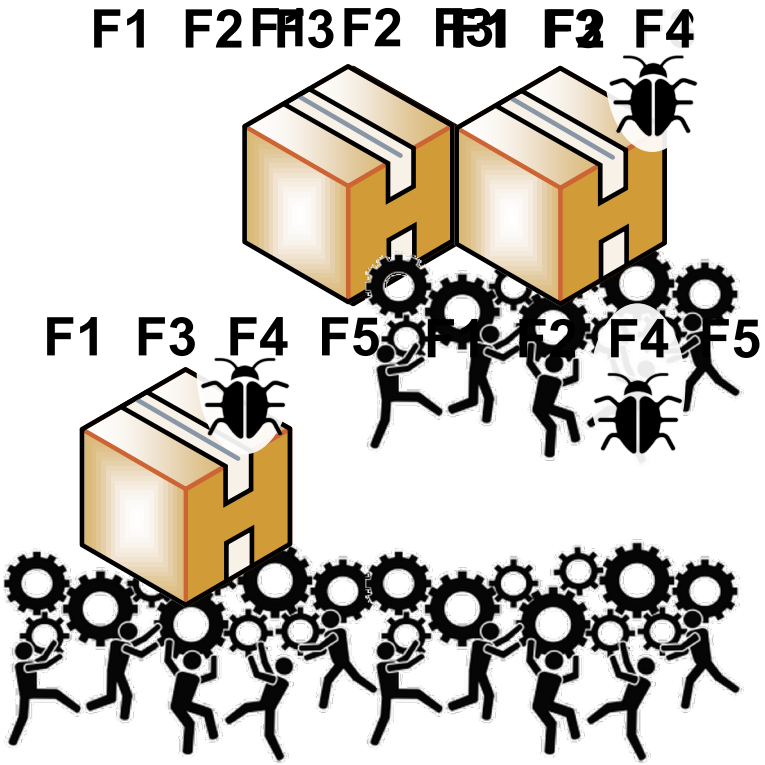
50% of industrial companies use clone-and-own to create variants.

Berger *et al.* A Survey of Variability Modeling in Industrial Practice, VaMoS '13

Clone-and-own

- **Advantages** ?
  - Easy/quick solution to create variants!
- **Problems ?**

Clone-and-own



~~Clone-and-own:~~

~~Opportunistic reuse~~

Systematic Reuse: “**Software Product Lines:**

Systematic reuse

LIVE DEMO

Content

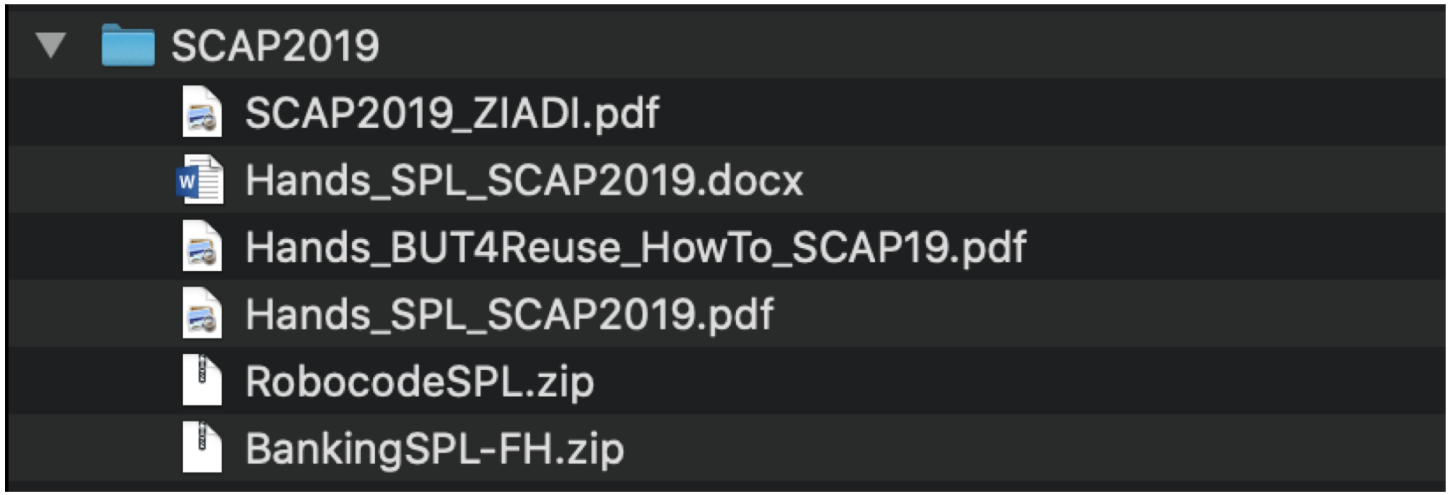
- **Software Product Lines (SPL)**
  - Concepts & existing approaches to implement SPL
  - The Robocode case study

LIVE DEMOS

Slides & Hands-on-Experiements

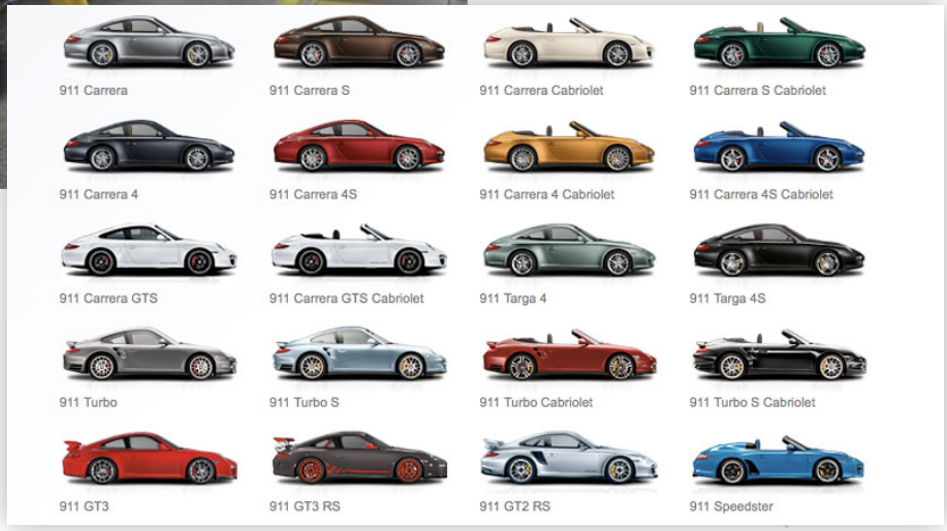
- Using Eclipse/Java

https://pages.lip6.fr/tewfik.ziadi/SCAP2019.zip



Product Lines in Industry

Variants



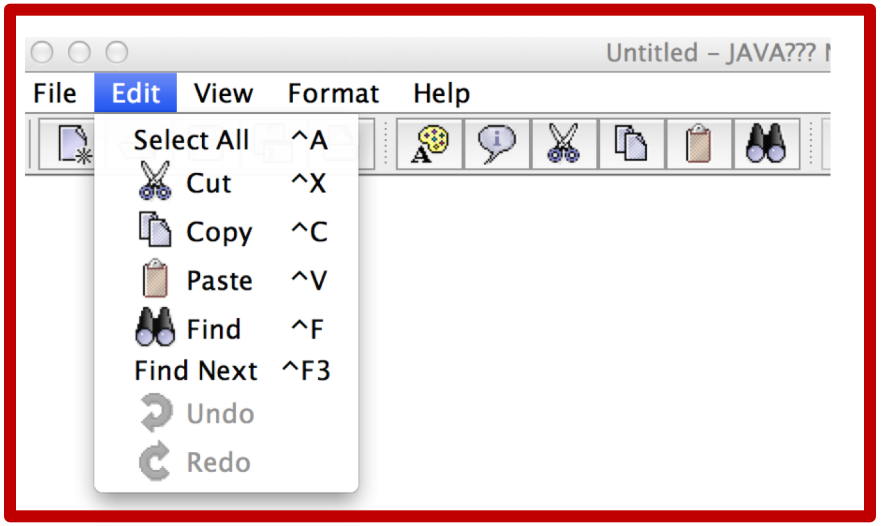
Why can’t build our application as we build our car ?

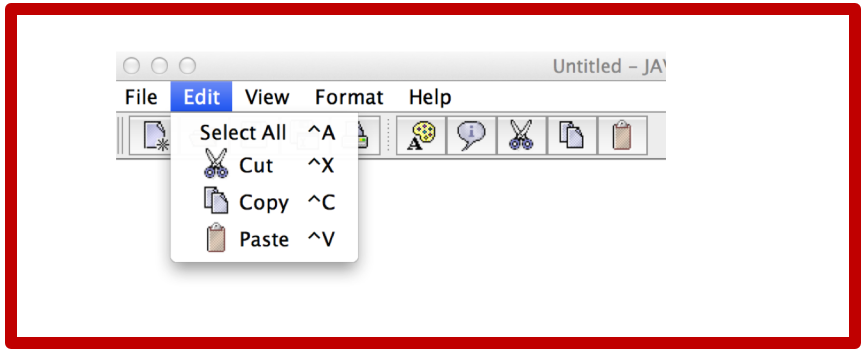
- Sotware systems

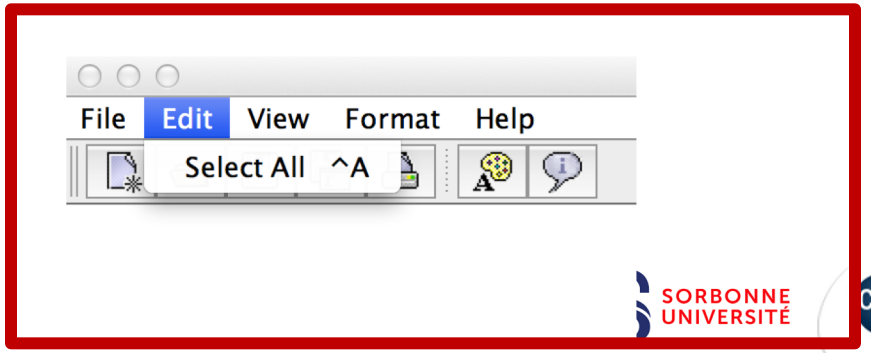
Comme in many **variants**

Printer Firmware

Classical Software



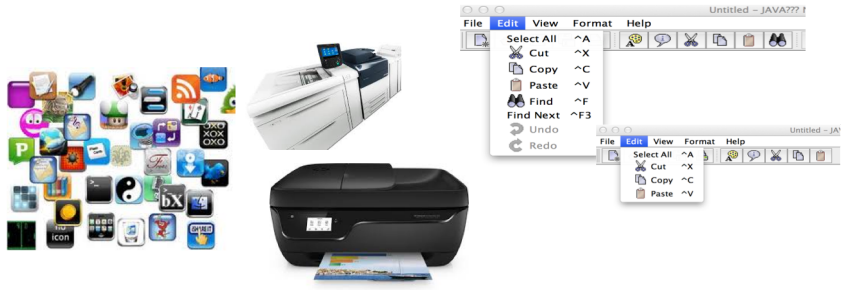




SORBONNE UNIVERSITÉ

dépasser les frontières

Mobile applications



Software Variability

“the ability of a system to be efficiently extended, changed, customized or configured for use in a particular context”

Mikael Svahnberg, Jilles van Gurp, and Jan Bosch (2005)

Software Product Li

Variant
~~**version**~~

A collection (family) of Software **variants**:

➔ The **same domain**
➔ Share a set of **commonalties** but also contain **variability**

- **Time to market**
- **Increase reuse**

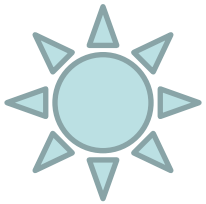
LIP6

SORBONNE UNIVERSITÉ

cnrs dépasser les frontières

Version vs. Variant

- **Version** :
  - Variability in **time**

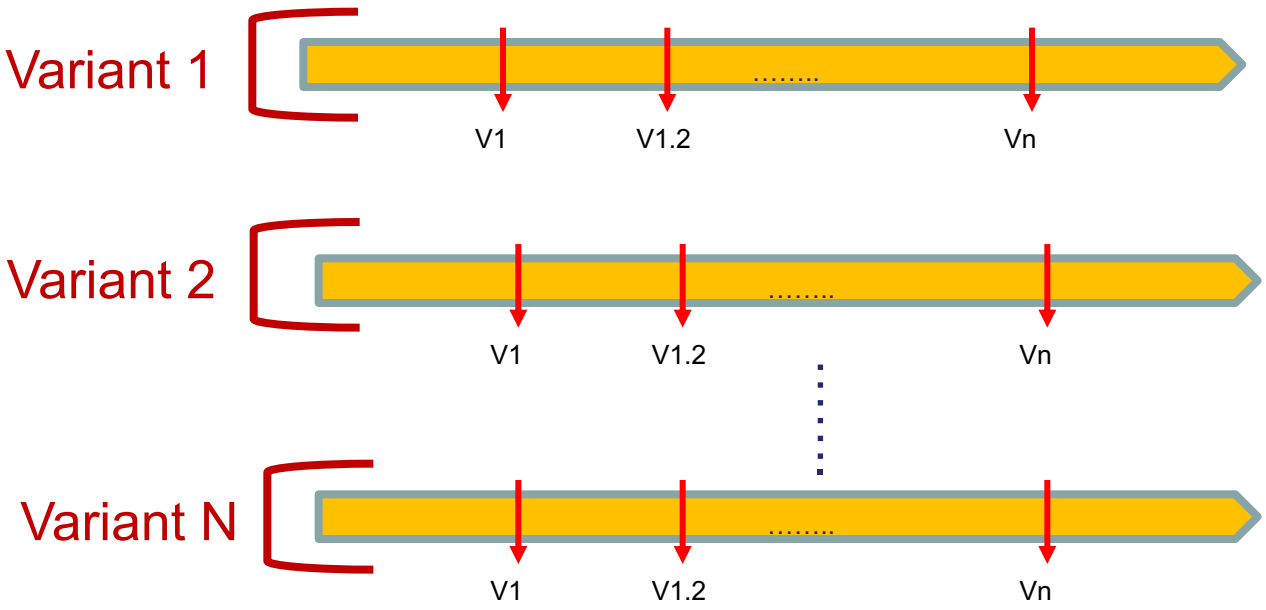


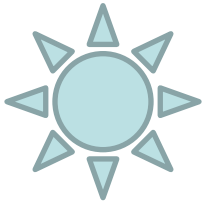
The **same software** that evolves in time

➔ Version Control Systems (git, svn..)

Variante vs. Version

- **Variant** :
  - Variability in **space**

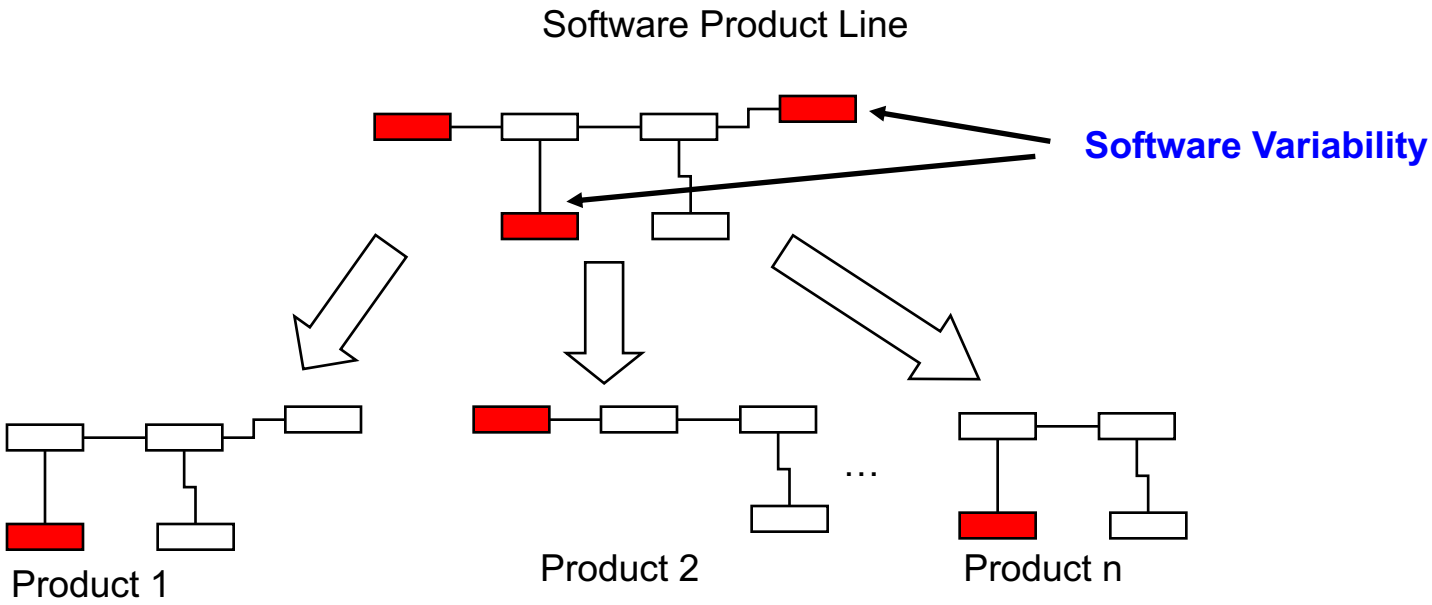




At the same time**, many variants exist**

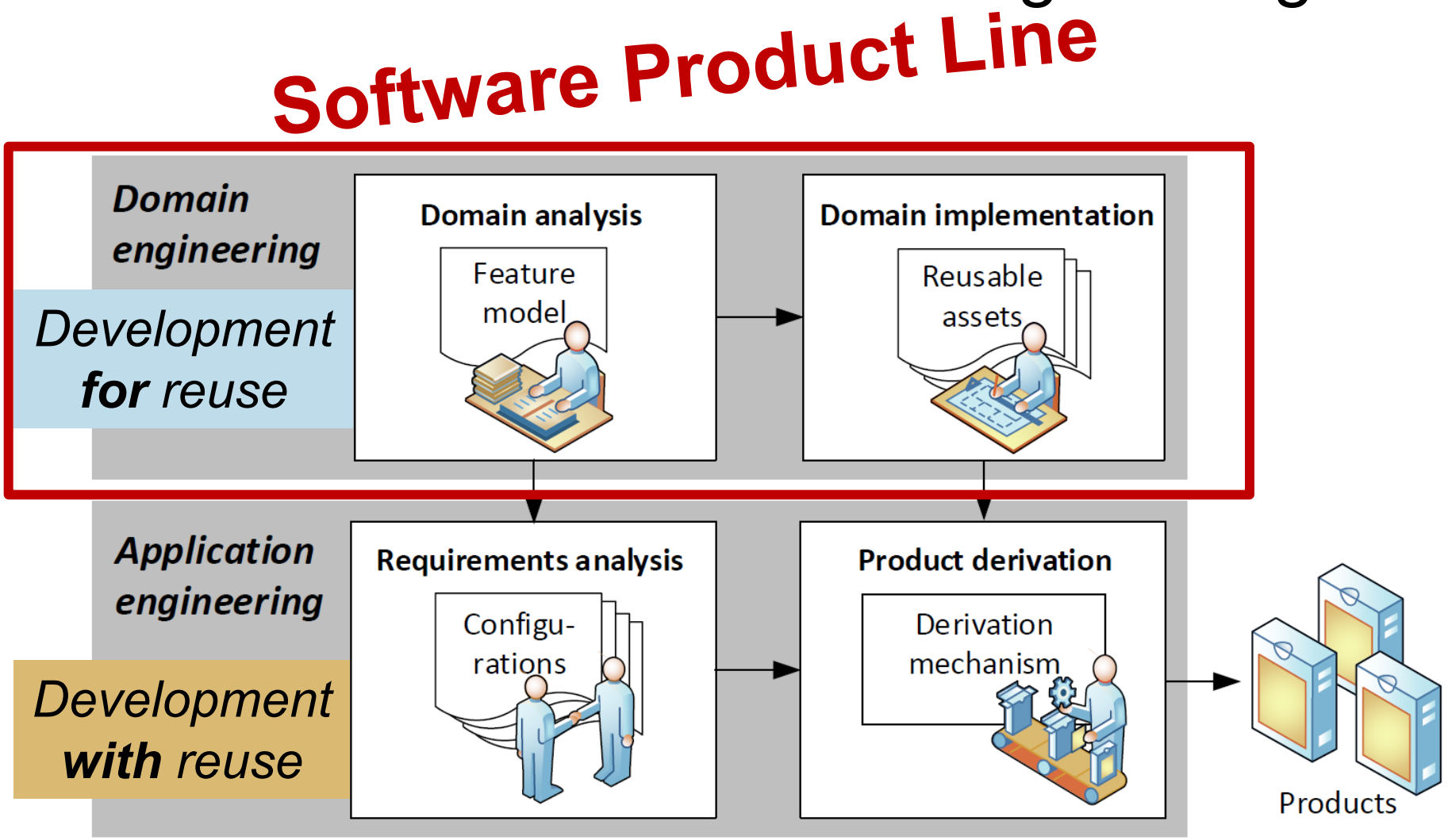
➔ Software Product Lines

Software Product Line

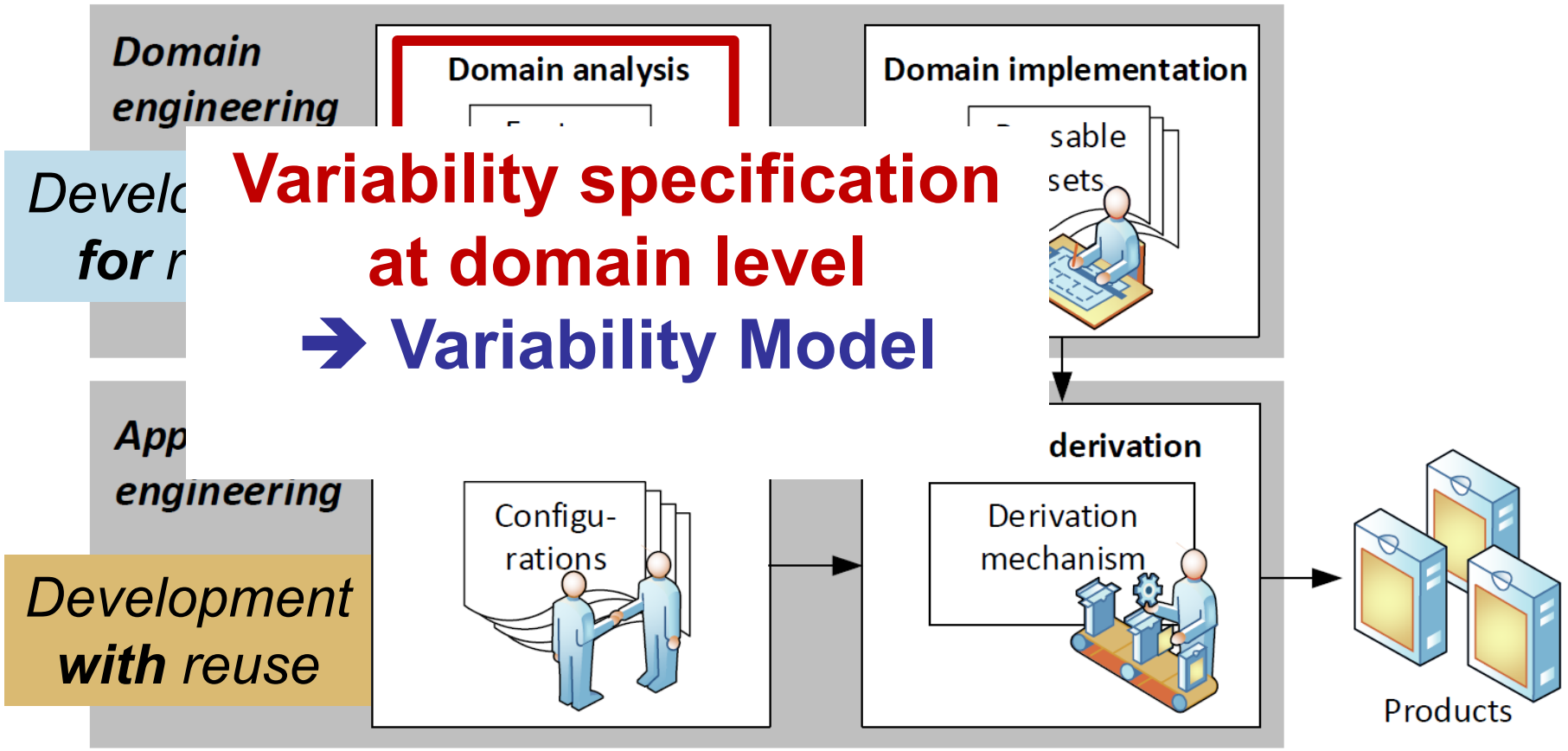


- ✓ **Dimension 1 : Variability Management.**
- ✓ **Dimension 2 : Product Derivation.**

Software Product Line Engineering



Phase1: Domain Analysis



Variability Model

- **Explicit specification of variability**
- **Feature Model: *de facto* standard!**

Feature-Oriented Domain Analysis (FODA) Feasibility Study

Kyo C. Kang
Sholom G. Cohen
James A. Hess
William E. Novak
A. Spencer Peterson

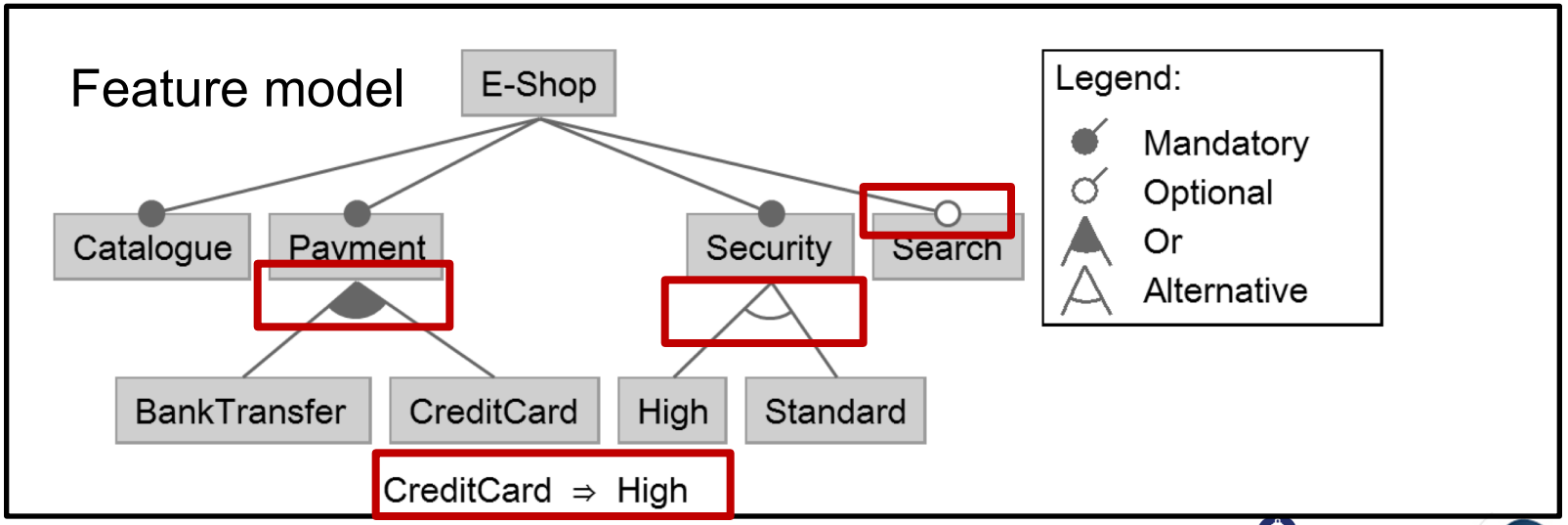
November 1990

- **Research**
  - 5000++ citations!
  - Central for many research work
  - Tools & Languages: GUIDSL/FeatureIDE, SPLOT, FaMa, etc.)
- **Industry**
  - Tools (Gears, pure::variants),

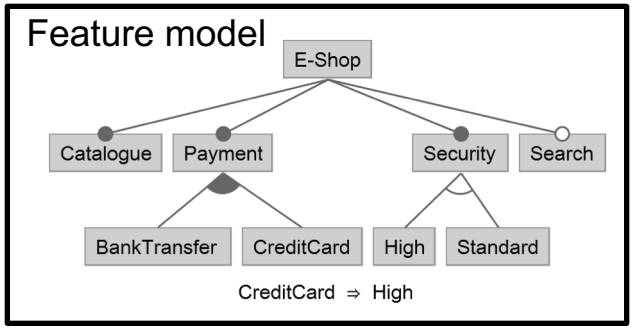
LIP6 SORBONNE UNIVERSITÉ cnrs dépasser les frontières

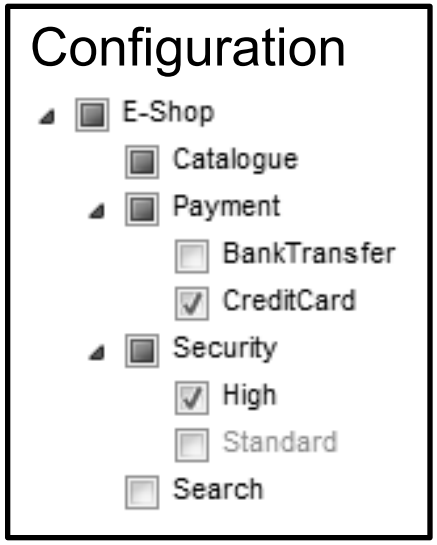
Feature Model

- The concept of **« feature »**
- **Hierarchy**: rooted tree
- Notations for **variability**
  - Optional, alternative, Or, constraints..



Feature Model & Configurations



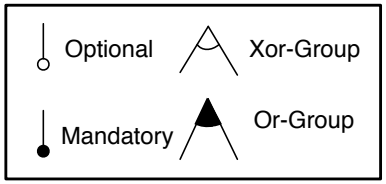


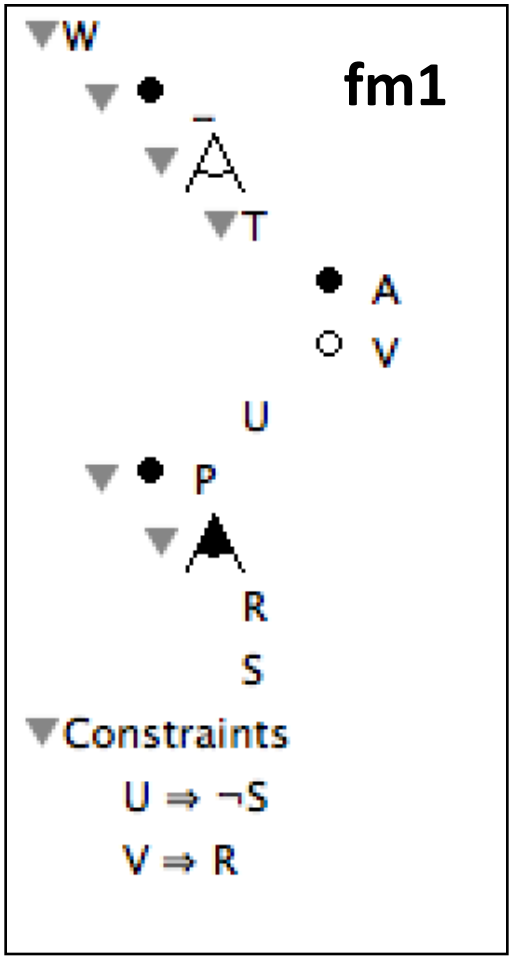
configuration =
set of features selected

Research Direction: **Automated Analysis and Reasoning on Feature models**

(Boolean) Feature Models

Hierarchy + Variability = set of valid configurations



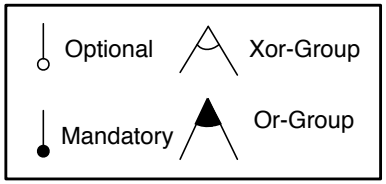


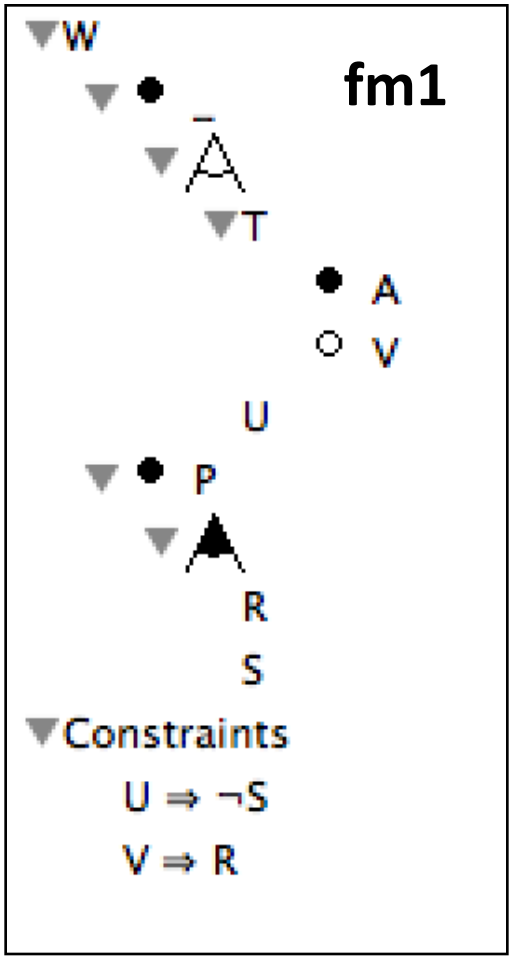
$$
\begin{aligned}
&[\![fm1]\!] = \{ \\
&\{W, P, R, S, T, A, V\}, \\
&\{W, P, S, T, A\}, \\
&\{W, P, R, T, A\}, \\
&\{W, P, R, U\}, \\
&\{W, P, R, T, V, A\}, \\
&\{W, P, R, S, T, A\}, \\
&\}
\end{aligned}
$$

Source: (Acher, 2018)

(Boolean) Feature Models

~ Boolean formula





$$
\begin{aligned}
&\phi_{fm_1} = W \;//\; \text{root} \\
&\wedge\, W \Leftrightarrow P \;//\; \text{mandatory} \\
&//\; \text{Or-group} \\
&\wedge\, P \Rightarrow R \vee S \\
&\wedge\, R \Rightarrow P \wedge\, S \Rightarrow P \\
&\wedge\, V \Rightarrow T \;//\; \text{optional} \\
&\wedge\, A \Leftrightarrow T \;//\; \text{mandatory} \\
&//\; \text{Xor-group} \\
&\wedge\, T \Rightarrow W \\
&\wedge\, U \Rightarrow W \\
&\wedge\, \neg T \vee \neg U \\
&//\; \text{constraints} \\
&\wedge\, V \Rightarrow R \;//\; \text{implies} \\
&\wedge\, \neg U \Rightarrow \neg S \;//\; \text{excludes}
\end{aligned}
$$

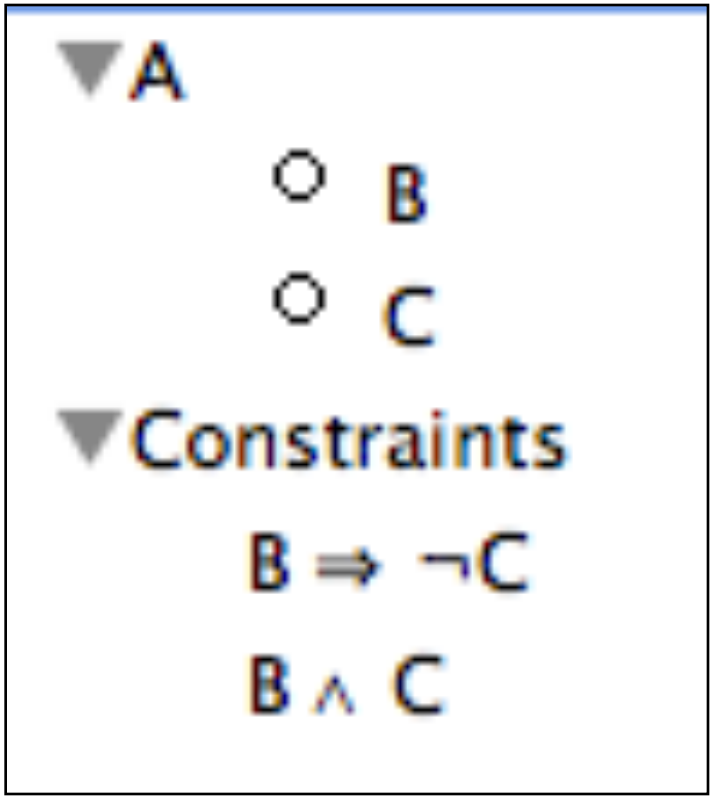
Source: (Acher, 2018)

Feature Models and Automated Reasoning

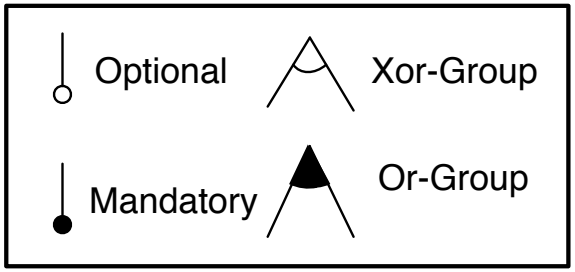
Benavides et al. survey, 2010

	PL													CP						DL			Multi			Others																	No support		
	Batory [5]	Czarnecki *et al.* [30]	Cheyi *et al.* [37]	Mannion *et al.* [51, 52]	Mendonca *et al.* [57]	Mendonca *et al.* [56]	Sun *et al.* [74]	Thüm *et al.* [75]	van der Storm [86, 87]	Zhang *et al.* [102, 101]	Zhang *et al.* [103]	Yan *et al.* [100]	Benavides *et al.* [10, 11, 12]	Benavides *et al.* [15]	Djebbi *et al.* [34]	Trinidad *et al.* [78, 76]	White *et al.* [99]	White *et al.* [97]	Abo Zaid *et al.* [1]	Fan *et al.* [35]	Wang *et al.* [92, 93]	Benavides *et al.* [14]	Benavides *et al.* [16]	Segura [70]	Bachmeyer *et al.* [4]	Cao *et al.* [20]	Fernandez *et al.* [36]	Hemakumar [41]	Cheyi *et al.* [38]	Kang *et al.* [43]	Mendonca *et al.* [55]	Osman *et al.* [59, 60]	Salinesi *et al.* [66]	Van den Broek *et al.* [84]	Van Deursen *et al.* [88]	Von der Massen *et al.* [90]	Von der Massen *et al.* [91]	White *et al.* [98, 96]	Batory *et al.* [7]	Schobbens *et al.* [42, 68, 69]	Trinidad *et al.* [80]	Von der Massen *et al.* [89]
Void feature model	+	+		+		+	+		+	+	+	+	+	+	+	+				+	+	+	+	+		+			+	⊕		+	+	+	+					~	~	
#Products		+		⊕									+	+	+							+	+	+			+					+	+	⊕		+					~	
Dead features		~				+				+	+	+				+			+		+		⊕						⊕		+	+					~	~	~	
Valid product	+	+	+	+			+										+				+				+				⊕					+				~	~	~	
All products	+		+	⊕			+						+										+		+				⊕				+	⊕				~	~	~	
Explanations	+	~					+									+			+		+								⊕		+		+				~	~	~	
Refactoring			+				⊕	+													+							+										~	~	~	
Optimization													⊕		+																+							~	~	~	
Commonality													⊕			+							+				+												~	~	
Filter		+											⊕		+																		+					~	~	
Valid partial configuration	+	+							+																					⊕								~	~	
Atomic sets					+					⊕														+							+										
False optional features										⊕	+					+																							~		⊖	
Corrective explanations																	+														+									⊖	
Dependency analysis																														⊕	+										
ECR					⊕	+																								⊕	+										
Generalization			⊕					+																																		
Core features						+																																			
Variability factor													⊕																											~	⊖	
Arbitrary edit								+																																	
Conditional dead features																												+													
Homogeneity																											+														
LCA					+																																				
Muti-step configuration																		+																								
Roots features					+																																				
Specialization								+																																	
Degree of orthogonality		~																																							
Redundancies																																									~
Variant features																																							~		
Wrong cardinalities																																							~		
Feature model notation	B	C	B	B	B	B	B	B	B	B	C	B	B	C	C	B	B	B	B	B	B	B	C	B	B	B	C	B	C	B	B	C	B	B	B	B	B	C	C	B
Extended feature model													+		+		+	+	+										+		+							+	+		+	
Formalization			+		+	+	+	+		+	+		+			+		+		+					+		+			+					+			+			

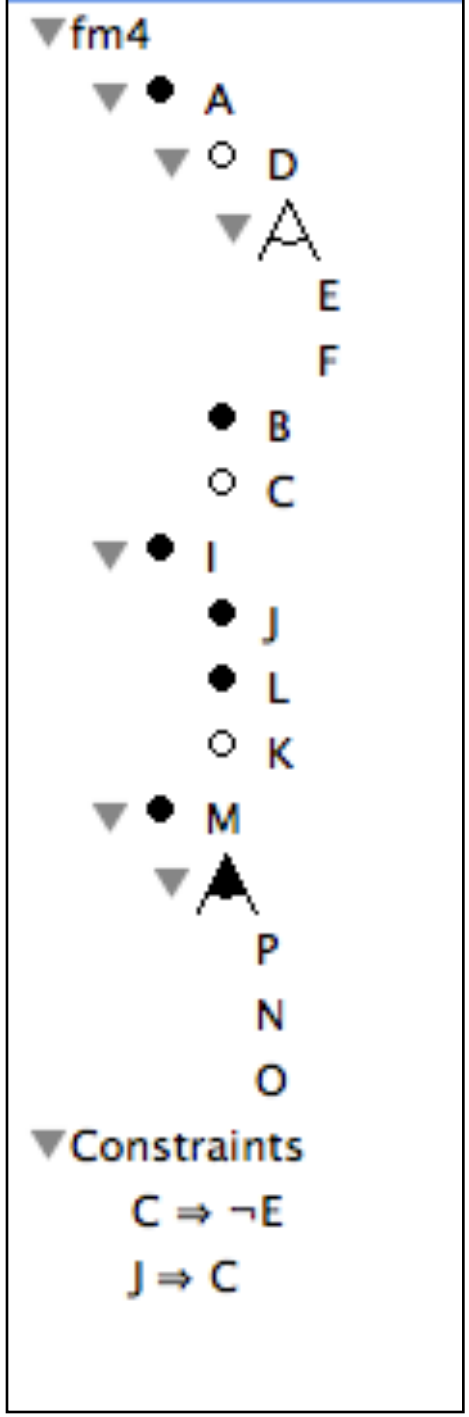
+ Supported ~ No support ⊕ Supported(first reference) ⊖ No support (first reference) B Basic feature model C Cardinality–based feature models



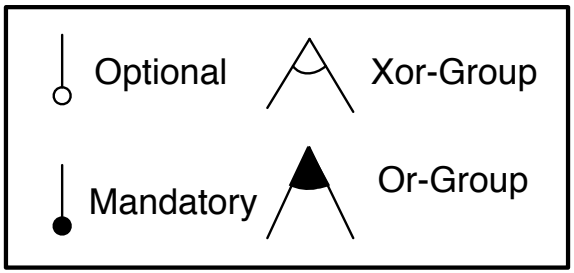
Empty set of configurations



Source: (Acher, 2018)

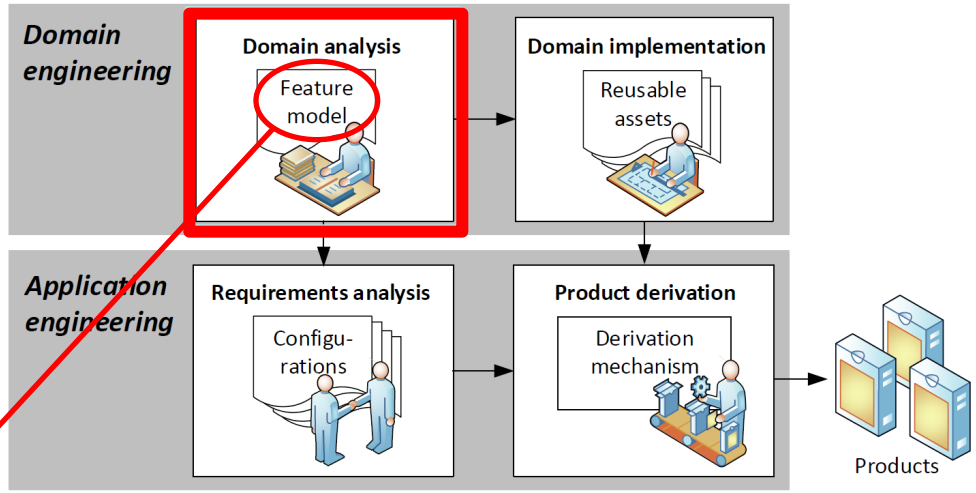


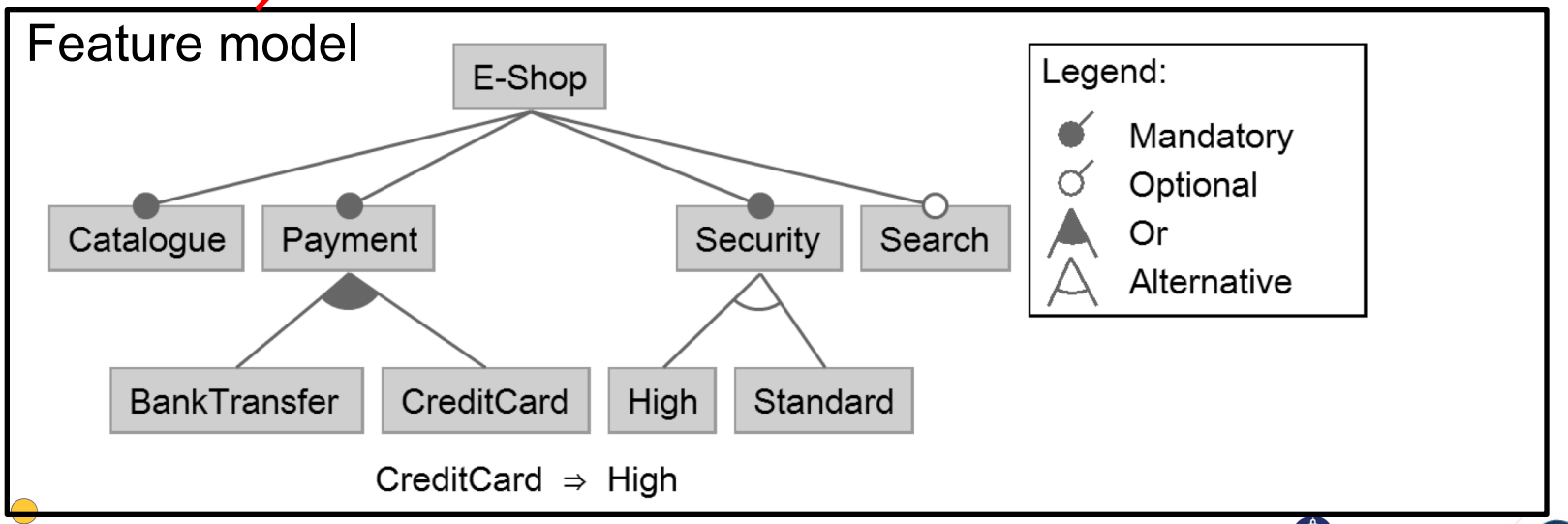
Dead feature



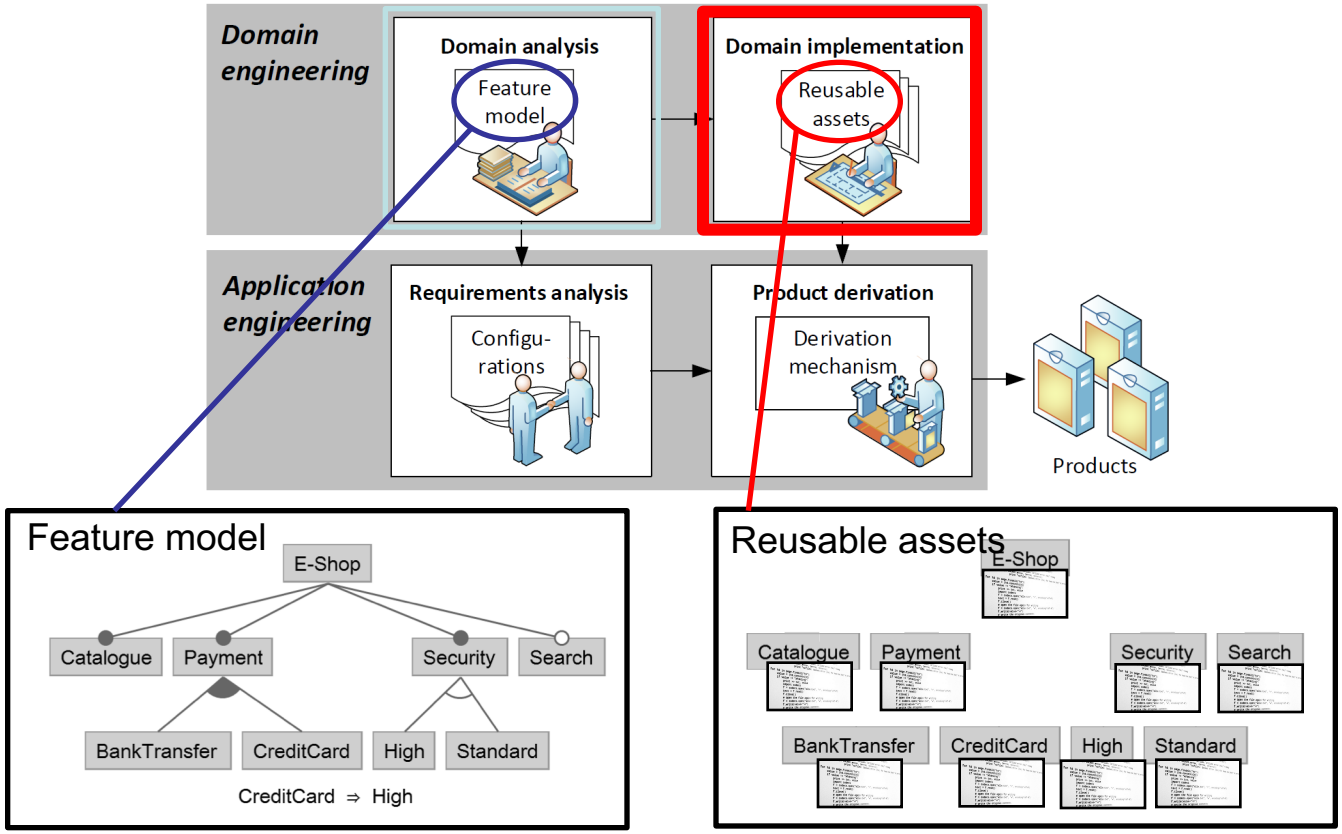
Source: (Acher, 2018)

Software Product Line Engineering

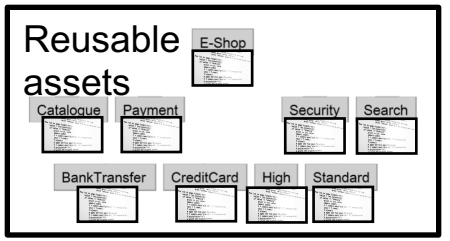




Software Product Line Engineering

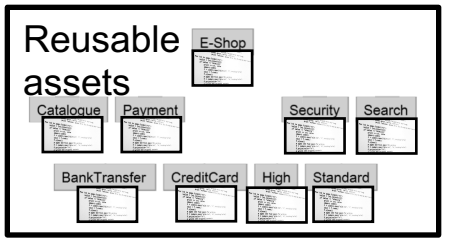


Domain Implementation



- **Objective**: Implement the reusable assets.
  - **Asset**: a **software artefact** needed to implement the product variants of the family.
    - *requirements*, *models, source code files, tests..*
  - **Reusable**: The assets that can be **reused** to implement the different variants:
    - ➔ Includes mechanisms to Implement the **software variability**
    - ➔ Links to the features.

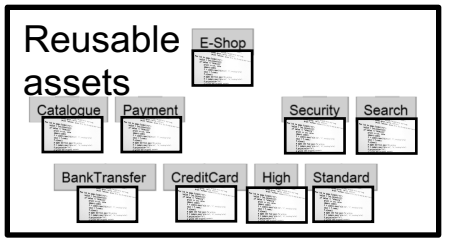
Domain Implementation



- How the reusable assets can be implemented?

Research Direction: **Software Product Lines Implementation**

Domain Implementation



- How the reusable assets can be implemented?

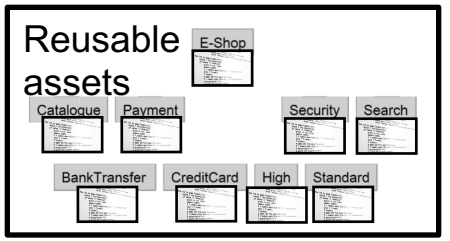
Research Direction: **Software Product Lines Implementation**

State of the art: two families of approaches:

➔ **Annotative** approaches

➔ **Compositional** approaches

Annotative approaches



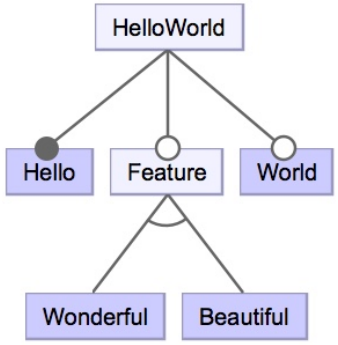
- **Asset Implementation:**
  - A **maximal** product (150% product).
  - **Annotations** to specify the fragments related to each feature.

Annot. approaches: State of the art

- ***Conditional compilation (preprocessors)*:**
  - Implement a maximal **C source code**
  - Add annotations using **compilation directives** (#ifdef)

(150% source code)

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    printf("Hello");
#ifdef Beautiful
    printf(" beautiful");
#endif
#ifdef Wonderful
    printf(" wonderful");
#endif
#ifdef World
    printf(" world");
#endif
    return 0;
}
```



Feature ⇒ World

Annot. approaches: State of the art

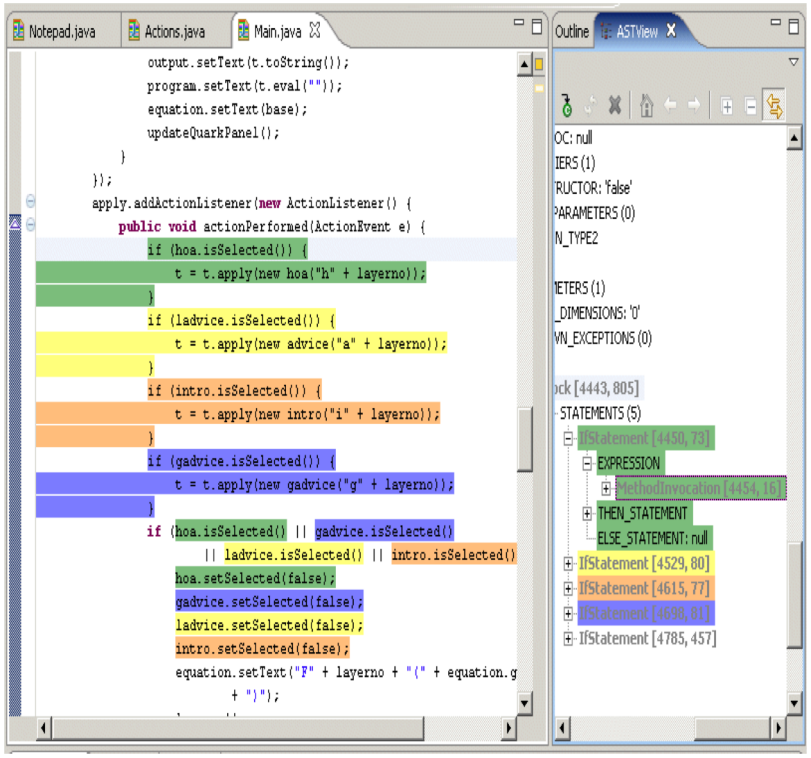
- Preprocessors for Java (integrated in the FeatureIDE tool)
  - **Munge** preprocessor: developed by the developers of Java's Swing library (main author:Thomas Ball)

```
public class HelloWorld {

    public static void main(String[] args){
        System.out.print("Hello");
        /*if[Beautiful]*/
        System.out.print(" beautiful");
        /*end[Beautiful]*/
        /*if[Wonderful]*/
        System.out.print(" wonderful");
        /*end[Wonderful]*/
        /*if[World]*/
        System.out.print(" world");
        /*end[World]*/
    }

}
```

(c) Source: FeatureIDE examples

Annot. approaches: State of the art

- **Colored IDE (CIDE)** [Kästner et al. ICSE]

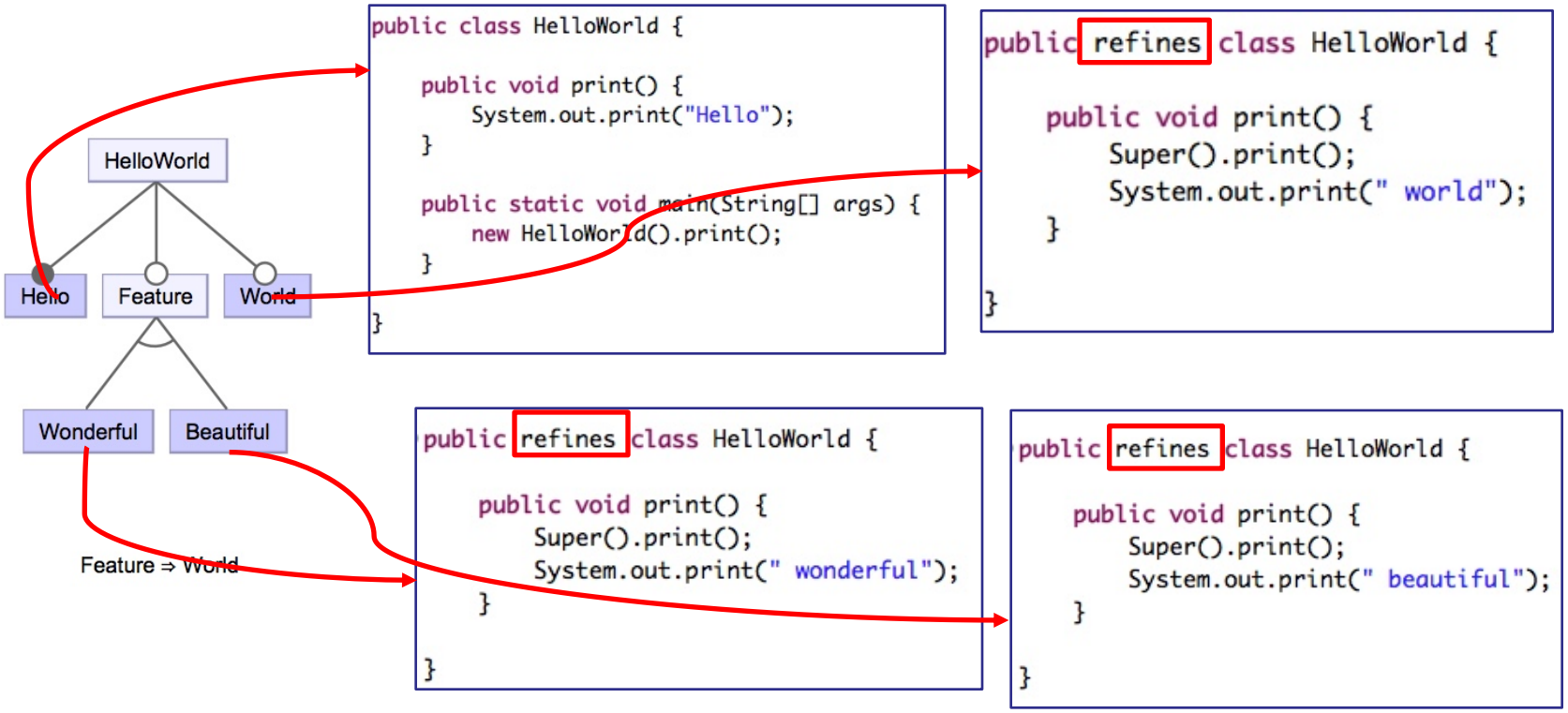


Compositional approaches

- Called also Feature-Oriented Programming/ Modeling approaches.
- **Asset Implementation**
  - the features of the product line are implemented as a set of ***separated fragments (modules)***

Comp. approaches: State of the art

- **AHEAD** framework (Batory et al., 04, IEEE TSE).
  - Based on code refinement

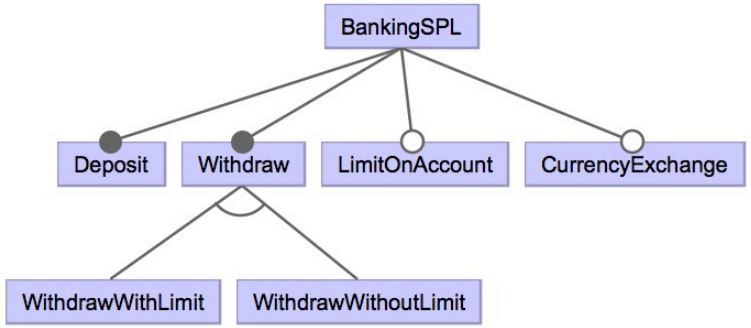


Comp. approaches: State of the art

- **FeatureHouse**, (Apel et al. TSE 10)
  - Based on code refinement
    - A **« base »** code that is a common to all variants.
    - Each feature wil **« refine »** the base code
      - Extend the base code
      - Adding new implementations.
  - A generic compositional approach that supports many languages.

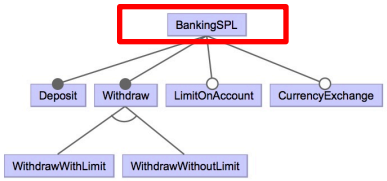
Example: The Banking System SPL

- Simple applications in the banking domain that implement the classical banking functionalities:
  - Deposit money
  - Withdraw
  - Currency conversion.
- **Variability** :
  - Limit on account is optional.
  - The conversion operation is are optional.



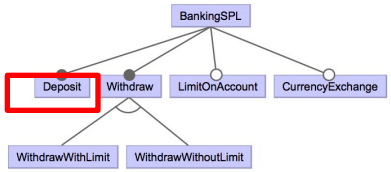
WithdrawWithLimit ⇔ LimitOnAccount

Feature : BankingSPL



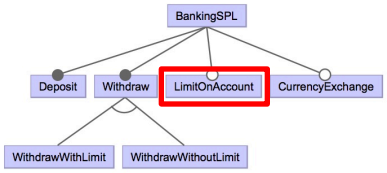
WithdrawWithLimit ⇔ LimitOnAccount

Feature : Deposit



WithdrawWithLimit ⇔ LimitOnAccount

Feature : LimitOnAccount



WithdrawWithLimit ⇔ LimitOnAccount

refine

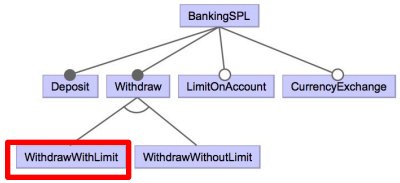
```
public class Account {

    public  String ID;

    private double balance;

    public Account(String id, double initial) {

        this.ID=id;
        this.balance=initial;
    }

    public double getBalance() {

        return this.balance;
    }

    public void display() {
        System.out.println("Account: "+ID+ " Balance ="+balance);
    }

}
```

"Base" code

```
public class Account {

    public void deposit(double amount) {

        balance+=amount;

    }

}
```

```
public class Account {


    final static int DAILY_LIMIT = -1000;


     public void display() {

          original();
          System.out.println("LIMIT ="+DAILY_LIMIT);
       }

}
```

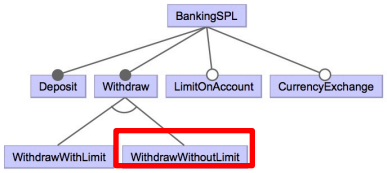
Feature : LimitOnAccount



WithdrawWithLimit ⇔ LimitOnAccount

```
public class Account {

    public boolean withdraw(double amount) {

        if (balance>=amount+DAILY_LIMIT ) {
            balance-=amount;
            return true;
        }
        return false;
    }
}
```

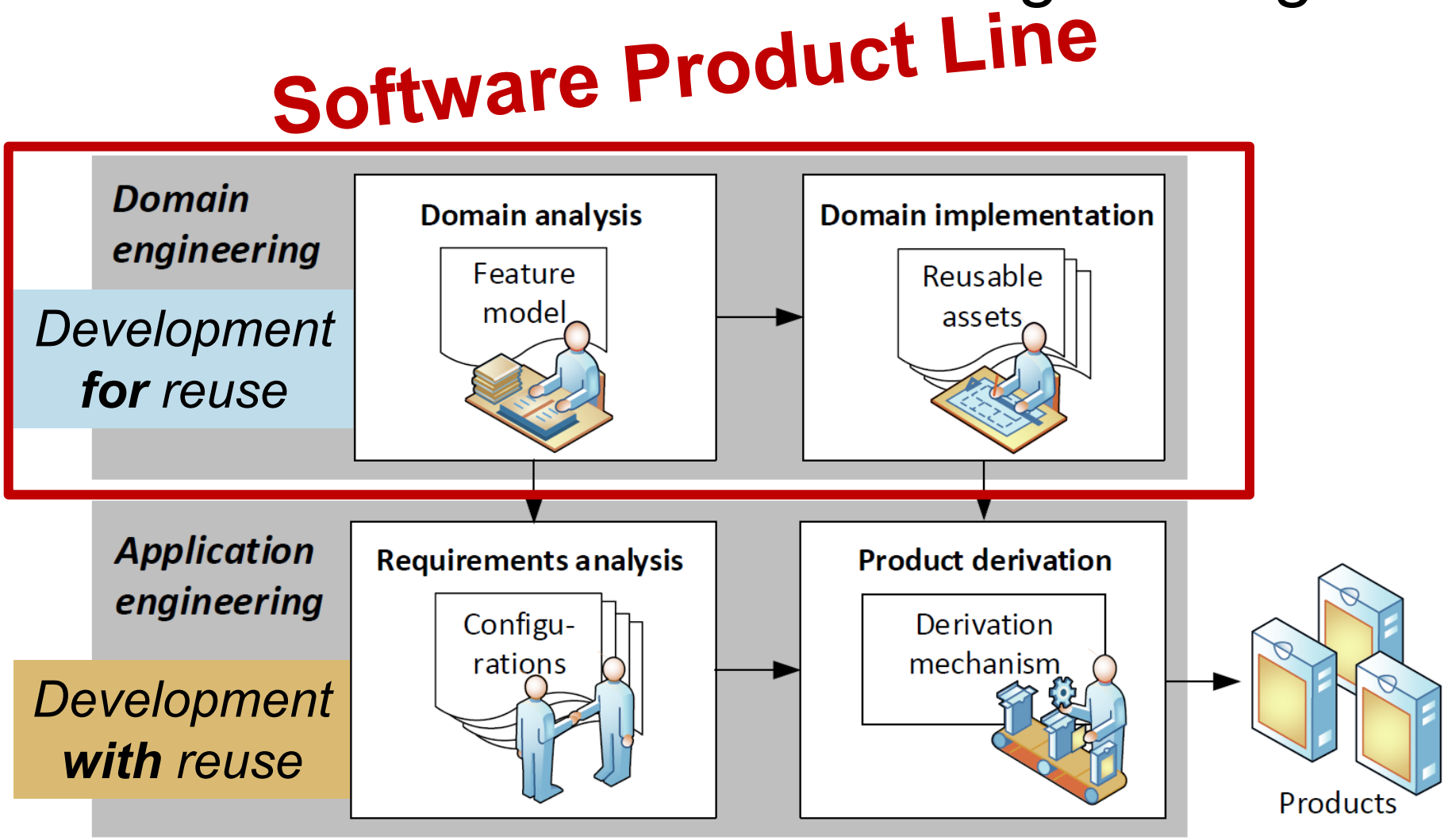
Feature : LimitOnAccount



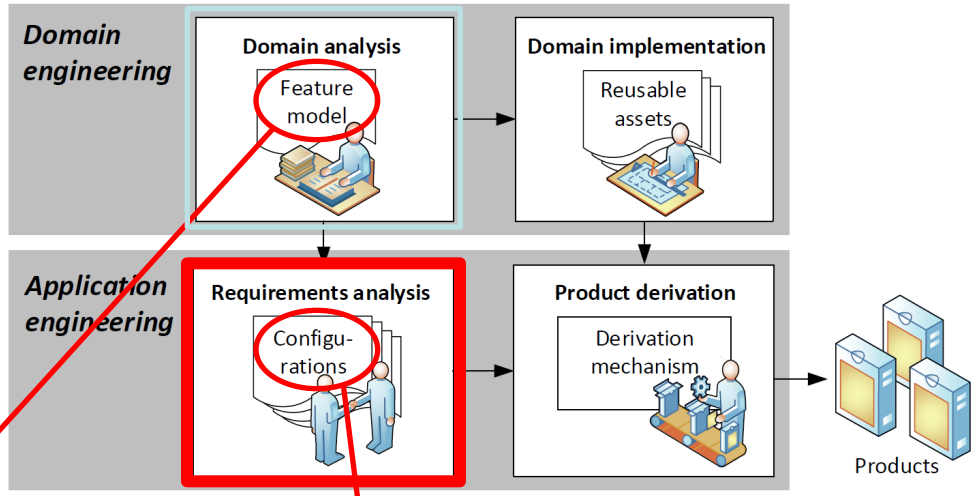
WithdrawWithLimit ⇔ LimitOnAccount

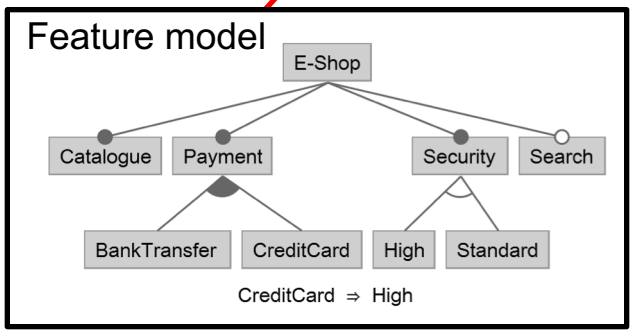
```
public class Account {

    public boolean withdraw(double amount) {

        if (balance>=amount ) {
            balance-=amount;
            return true;
        }
        return false;
    }
}
```

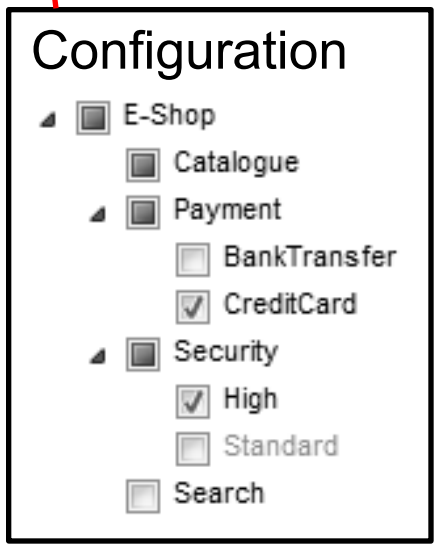
Software Product Line Engineering



Software Product Line Engineering



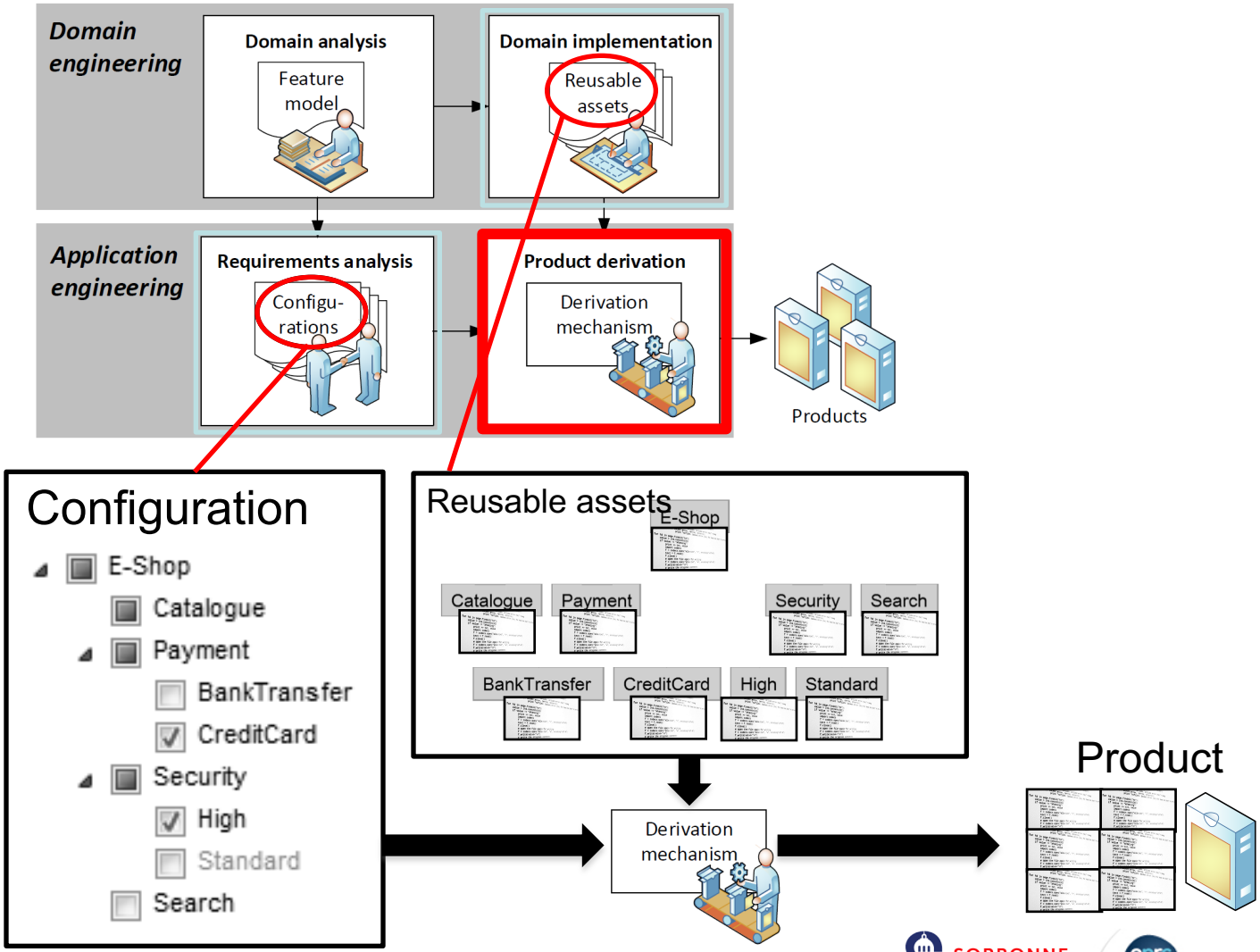




configuration =
set of features selected

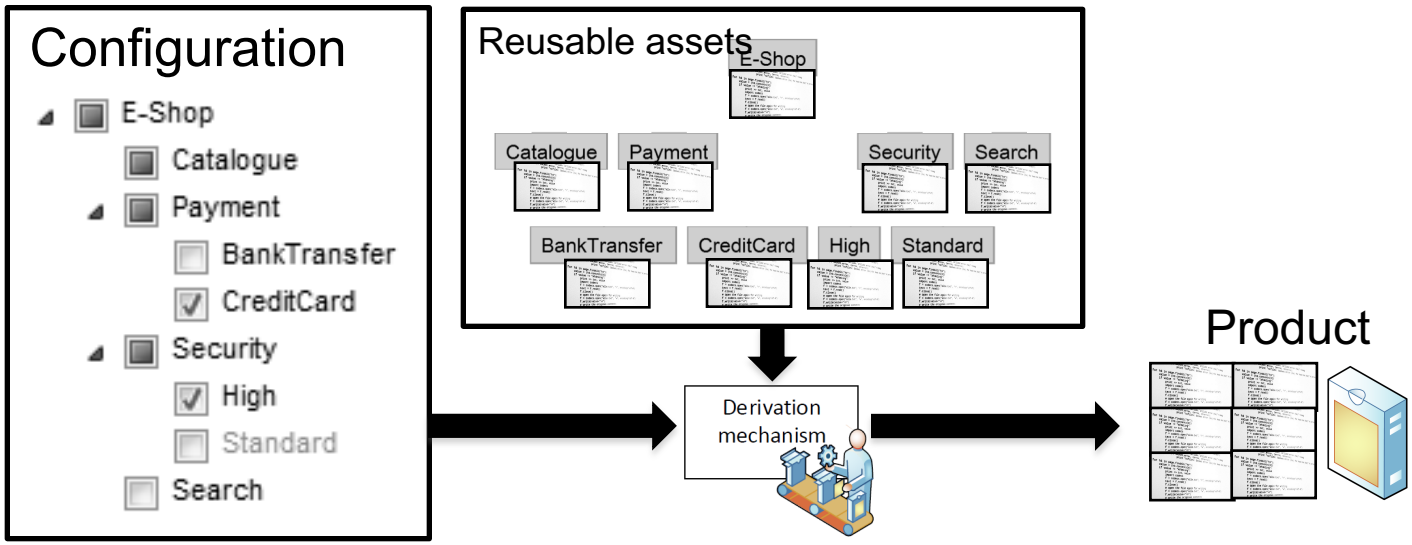
➔ **Only valid configurations**

Software Product Line Engineering



Product Derivation

- **Derive** (generate/synthesize) the product variants, members of the family.
- Generate the assets of the product variant that corresponding to a specific configuration.



Product Derivation

- How product derivation can be achieved ?

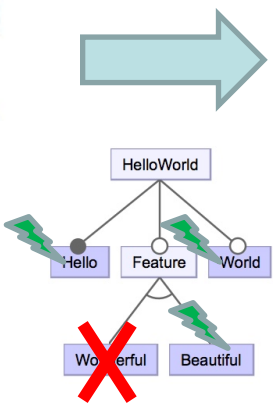
Research Direction: **Product Derivation**

  - Product Derivation depends on the used approach for reusable assets implementation.

Annot. Approaches: Product derivation

- **Product Derivation**
  - A specific product is derived by **removing** fragments related to the disabled features (program transformation).

```
public class HelloWorld {

    public static void main(String[] args){
        System.out.print("Hello");
        /*if[Beautiful]*/
        System.out.print(" beautiful");
        /*end[Beautiful]*/
        /*if[Wonderful]*/
        System.out.print(" wonderful");
        /*end[Wonderful]*/
        /*if[World]*/
        System.out.print(" world");
        /*end[World]*/
    }

}
```



Feature ⇒ World

```
public class HelloWorld {

    public static void main(String[] args){

        System.out.print("Hello");

        System.out.print(" wonderful");

        System.out.print(" world");

    }
```

Comp. Approaches: Product derivation

- **Product derivation** is based on the composition of modules/fragments.
  - We need to implement a **composer** that composes the corresponding fragments for the selected features.

Feature : BankingSPL

```
public class Account {

    public  String ID;

    private double balance;

    public Account(String id, double initial) {

        this.ID=id;
        this.balance=initial;
    }

    public double getBalance() {

        return this.balance;
    }

    public void display() {
        System.out.println("Account: "+ID+ " Balance ="+balance);
    }

}
```

Feature : LimitOnAccount

```
public class Account {


    final static int DAILY_LIMIT = -1000;


     public void display() {

            original();
            System.out.println("LIMIT ="+DAILY_LIMIT);
        }

}
```

Composition(**BankingSPL**, **LimitOnAccount**)?

```
public   class  Account {


    public  String ID;
    private double balance;
    final static int DAILY_LIMIT = -1000;

    public Account(String id, double initial) {
        this.ID=id;
        this.balance=initial;
    }

    public double getBalance() {

        return this.balance;
    }
     private void  display__wrappee__BankingSPL  () {
        System.out.println("Account: "+ID+ " Balance ="+balance);
    }
     public void display() {

            display__wrappee__BankingSPL();
            System.out.println("LIMIT ="+DAILY_LIMIT);
        }
    public boolean withdraw(double amount) {

        if (balance>=amount+DAILY_LIMIT ) {
            balance-=amount;
            return true;
        }
        return false;

    }

}
```

LIVE DEMO

Content

- **Software Product Lines (SPL)**
  - Concepts & existing approaches to implement SPL
  - The Robocode case study

LIVE DEMOS

A real-world case study: Robocode

- Based on a paper presented in SPLC18

Software Product Line Extraction from Variability-Rich Systems: The Robocode Case Study

Jabier Martinez*
Tecnalia
Derio, Spain
jabier.martinez@tecnalia.com

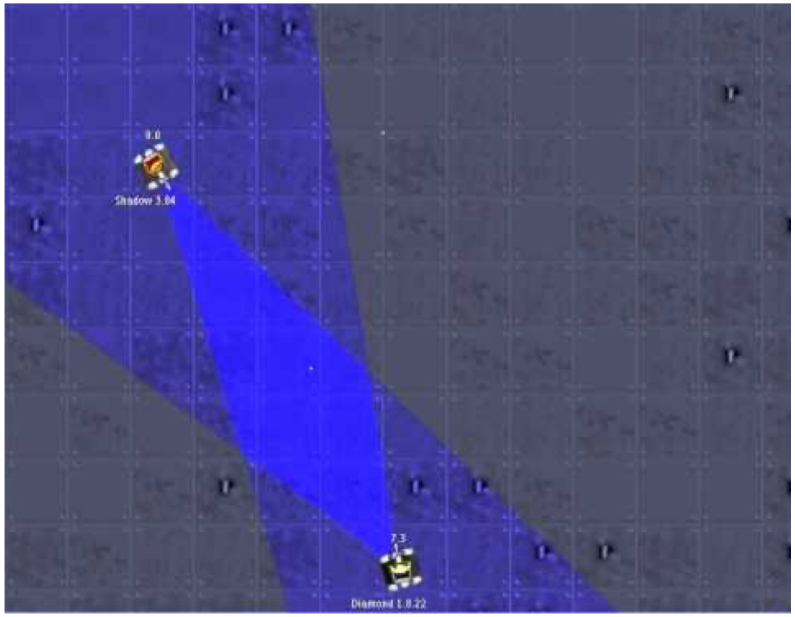
Xhevahire Tërnava
Sorbonne University UPMC
Paris, France
xhevahire.ternava@lip6.fr

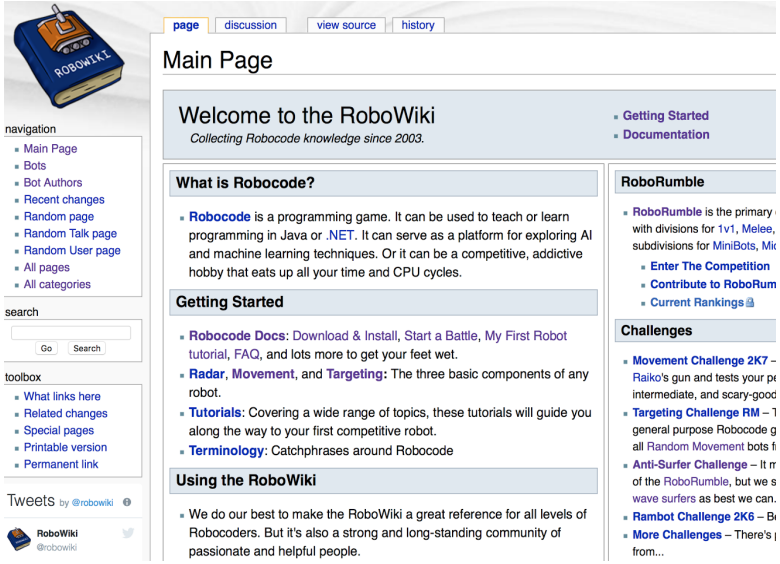
Tewfik Ziadi
Sorbonne University UPMC
Paris, France
tewfik.ziadi@lip6.fr

- Available at https://github.com/but4reuse/RobocodeSPL_teaching

Robocode: the case study

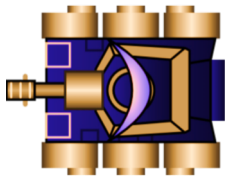
Robocode is a **programming game** where the goal is **to code** a **robot** to compete against other robots in a battle arena.





Robocode: the case study

- API to implement the behavior of the robots
- Each implementation is called **“Bot”**



```java
package sample;
import robocode.HitByBulletEvent;
import robocode.Robot;
import robocode.ScannedRobotEvent;


/**
 * MyFirstRobot - a sample robot by Mathew Nelson.
 * <p/>
 * Moves in a seesaw motion, and spins the gun around at each end.
 *
 * @author Mathew A. Nelson (original)
 */
public class MyFirstRobot extends Robot {

    /**
     * MyFirstRobot's run method - Seesaw
     */
    public void run() {

        while (true) {
            ahead(100); // Move ahead 100
            turnGunRight(360); // Spin gun around
            back(100); // Move back 100
            turnGunRight(360); // Spin gun around
        }
    }

    /**
     * Fire when we see a robot
     */
    public void onScannedRobot(ScannedRobotEvent e) {
        fire(1);
    }

    /**
     * We were hit!  Turn perpendicular to the bullet,
     * so our seesaw might avoid a future shot.
     */
    public void onHitByBullet(HitByBulletEvent e) {
        turnLeft(90 - e.getBearing());
    }
}
```

Robocode: A Variability-Rich System

5 components in each bot:

- **Targeting**
- **Movement**
- **Select Enemy**
- **Radar**
- **Gun & Energy Management**

➔ **Many variations for each component**

More than **3000** variants of bots are available from RobotWiki

Robot Name	Wiki Page	Movement	Targeting	
Acero	http://robowiki.net/wiki/Acero	Anti-Pattern at distance	Head-On Targeting	m
Acraepheus	http://robowiki.net/wiki/Acraepheus	Adapting Stop and Go	Averaged Velocity, Heading Circular Targeting, En	O
AFlatNatural	http://robowiki.net/wiki/AFlatNatural	Oscillator Movement	Virtual Guns Array (26 simple targeters)	/
AgentSmith	http://robowiki.net/wiki/AgentSmith	Danger Prediction	Virtual Guns Array (Linear, Circular, Head On Targe	/
AIR	http://robowiki.net/wiki/AIR	Gilgalad's Movement strategy	Gilgalad's Targeting strategy	/
Aleph	http://robowiki.net/wiki/Aleph	Circle Movement, Minimum Risk Moveme	Dynamic Clustering, Play It Forward	O
Ali	http://robowiki.net/wiki/Ali	Wave Surfing	Dynamic Clustering	/
AlphaAurora	http://robowiki.net/wiki/AlphaAurora	/	Circular Targeting	/
Anarchy	http://robowiki.net/wiki/Anarchy	Pattern Movement	Head-On Targeting	/

	Robot Name	Wiki Page	Movement	Targeting	Fighting	Code License	Code Java	Code Size
1								
2	**Acero**	http://robowiki.net/wiki/Acero	Anti-Pattern at distance	Head-On Targeting	melee and One-on-one	None	Yes	
3	**Acraepheus**	http://robowiki.net/wiki/Acraepheus	Adapting Stop and Go	Averaged Velocity, Heading Circular Targeting,	One-on-one	None	Yes	
4	**AFlatNatural**	http://robowiki.net/wiki/AFlatNatural	Oscillator Movement	Virtual Guns Array (26 simple targeters)	/	public domain	Yes	
5	**BlitzBat**	http://robowiki.net/wiki/BlitzBat	Minimum Risk Movement	Head-On Targeting	One-on-One Melee	RWPCL	Yes ?	
6	**BrokenSword**	http://robowiki.net/wiki/BrokenSword	Minimum Risk Movement	Shadow/Melee Gun	One-on-One Melee	zlib license	Yes ?	
7	**BulletCatcher**	http://robowiki.net/wiki/BulletCatcher	Pusedo Not Moving, Random Movement	GuessFactor Targeting	One-on-one	/	Yes ?	
8	**BulletSimBot**	http://robowiki.net/wiki/BulletSimBot	Fires Prediction	/	/	/	No	
9	**Caligula**	http://robowiki.net/wiki/Caligula	Linear Movement (special version)	Linear Targeting	One-on-one	/	No	
0	**Canon**	http://robowiki.net/wiki/Cannon	Musashi Trick, SandboxFlattener	Dynamic Clustering	One-on-one	/	Yes ?	
1	**Capulet**	http://robowiki.net/wiki/Capulet	Minimum Risk Movement	Circular Targeting	One-on-One Melee	/	No	
2	**CassiusClay**	http://robowiki.net/wiki/CassiusClay	Wave Surfing	GuessFactor Targeting	/	RWPCL	Yes ?	
3	**Centaur**	http://robowiki.net/wiki/Centaur	/	Dynamic Clustering-GuessFactor Targeting	One-on-One Melee		Yes ?	
4	**Chalk**	http://robowiki.net/wiki/Chalk	/	Dynamic Clustering	/	Public domain	No	
5	**Charo**	http://robowiki.net/wiki/Charo	Random Movement	GuessFactor Targeting	/	Public domain	Yes	
6	**CHCI3**	[illegible]	/	[illegible]	[illegible]	[illegible]	[illegible]	
7	**ChristmasCard**	[illegible]	[illegible]	/	[illegible]	[illegible]	[illegible]	
8	**Chupacabra**	http://robowiki.net/wiki/Chupacabra	Pathfinding	Pattern matching	One-on-one	/	No	
9	**Cigaret**	[illegible]	Random Movement	Wave (based Statistical Targeting)	/	/	No	
0	**CirclingBot**	[illegible]	[illegible]		/	Public domain	Yes	
1	**Claude**	[illegible]	[illegible]		/	RWPCL	Yes	
2	**CloudBot**	http://robowiki.net/wiki/Cloudbot	Wave Surfing	GuessFactor Targeting	One-on-one	RWPCL	No	
3	**ColdBreath**	http://robowiki.net/wiki/ColdBreath	Neural Movement (wave surfing, GuessFa	Neural Targeting (GuessFactor)	One-on-One Melee	/	Yes	
4	**Combat**	http://robowiki.net/wiki/Combat	Wave Surfing (Dynamic Clustering)/ Fla	GuessFactor Targeting (with Dynamic	One-on-One Melee	/	No	
5	**ConceptA**	http://robowiki.net/wiki/ConceptA	kNN-True Wave Surfing	kNN-GuessFactor	/	CC BY-ND 3.0	No	
6	**Connavar**	http://robowiki.net/wiki/Connavar	Stop & Go, and Random Movement	GuessFactor Targeting	/	/	Yes	
7	**CopyKat**	http://robowiki.net/wiki/CopyKat	Mirror Movement	Symbolic Pattern Matching	One-on-one	RWPCL	Yes	

➔ 1500 bots made source code publicly available

➔ WikiRobot snippets

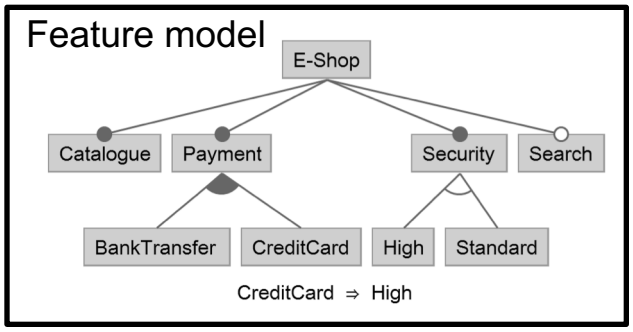
➔ **All follow the same architecture**

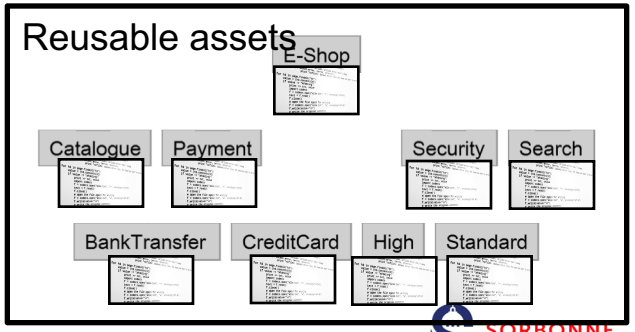
SORBONNE UNIVERSITÉ

dépasser les frontières

Robocode SPL Implementation

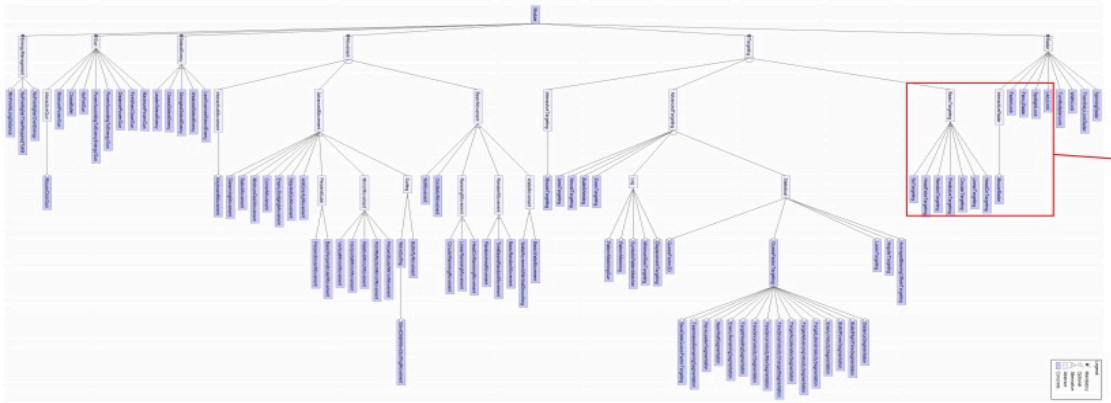
- **Inputs**:
  - Source code of bots
  - RoboWiki Snippet
  - Documentation
- **Output**: Robocode SPL

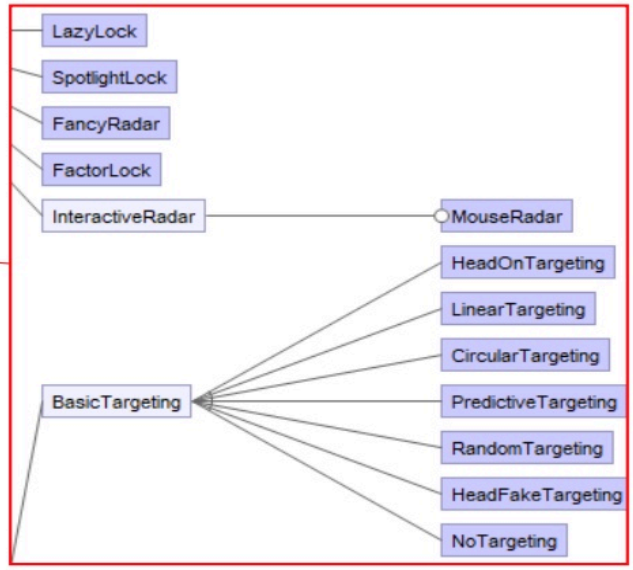




Domain Analysis

- **Objective**: Identify features and construct the feature model

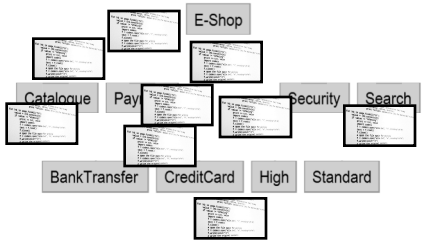




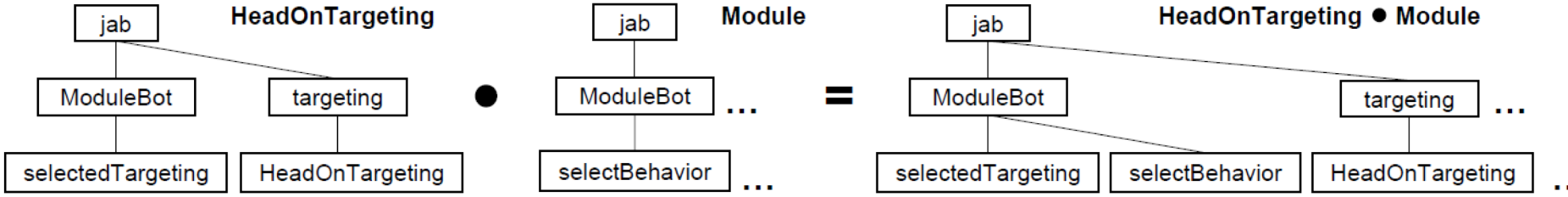
115 features:

- **7** mandatory features.
- **22** abstract features
- **93** are behavioral features

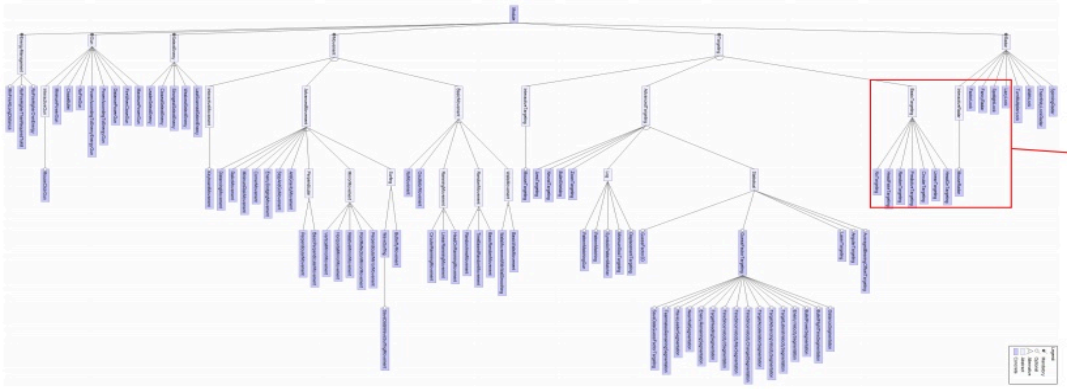
Domain Implementation

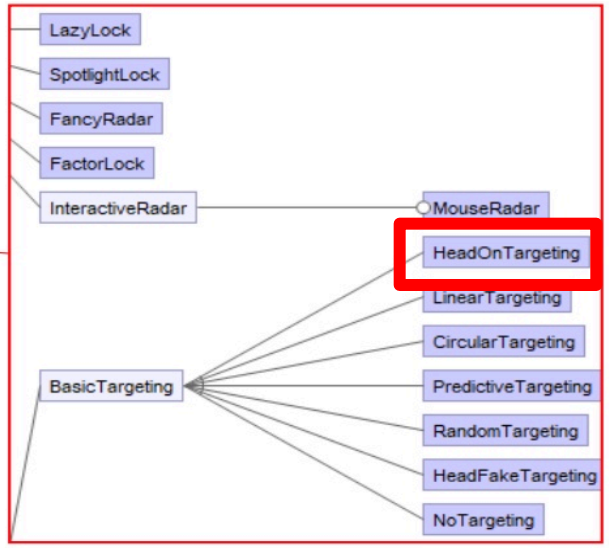


- **Objective**: Implement the reusable assets.
  - ➔ The use of the FeatureHouse framework [FH,13]
    - ➔A compositional approach at the source code level



[FH,99] Apel et al., Language-Independent and Automated Software Composition: The FeatureHouse Experience. IEEE Trans. Software Eng. 39(1): 63-79 (2013)






```java
package jab.targeting;

public class HeadOnTargeting extends Targeting {

  public HeadOnTargeting(Module bot) {
    super(bot);
  }

  public void target() {
    if (bot.enemy != null) {
      double absoluteBearing = bot.getHeadingRadians() +
     bot.enemy.bearingRadians;      bot.
     setTurnGunRightRadians(robocode.util.Utils.
     normalRelativeAngle(absoluteBearing - bot.
     getGunHeadingRadians()));
    }
  }
}
```

```java
package jab;

public class ModuleBot extends Module {
  Targeting selectedTargeting = new HeadOnTargeting(this)
    ;
}
```

“Base” refinement

```java
package jab;

public class ModuleBot extends Module {
  ...
  Targeting selectedTargeting = new HeadOnTargeting(this)
    ;
  ...
  protected void selectBehavior() {
    radar = selectedRadar;
    movement = selectedMovement;
    targeting = selectedTargeting;
    selectEnemy = selectedSelectEnemy;
    gun = selectedGun;
  }
    ...
}
```

SORBONNE UNIVERSITÉ

cnrs dépasser les frontières

Robocode SPL

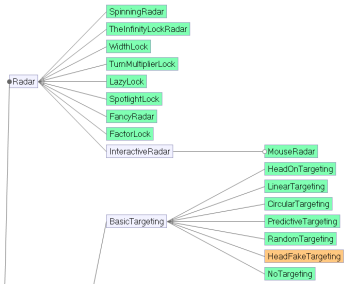
Domain Analysis

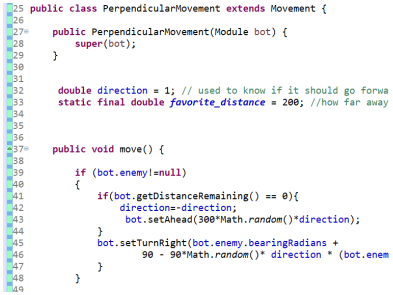
Domain Implementation

FeatureHouse

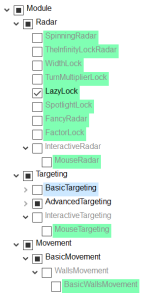
Configuration & Product Derivation

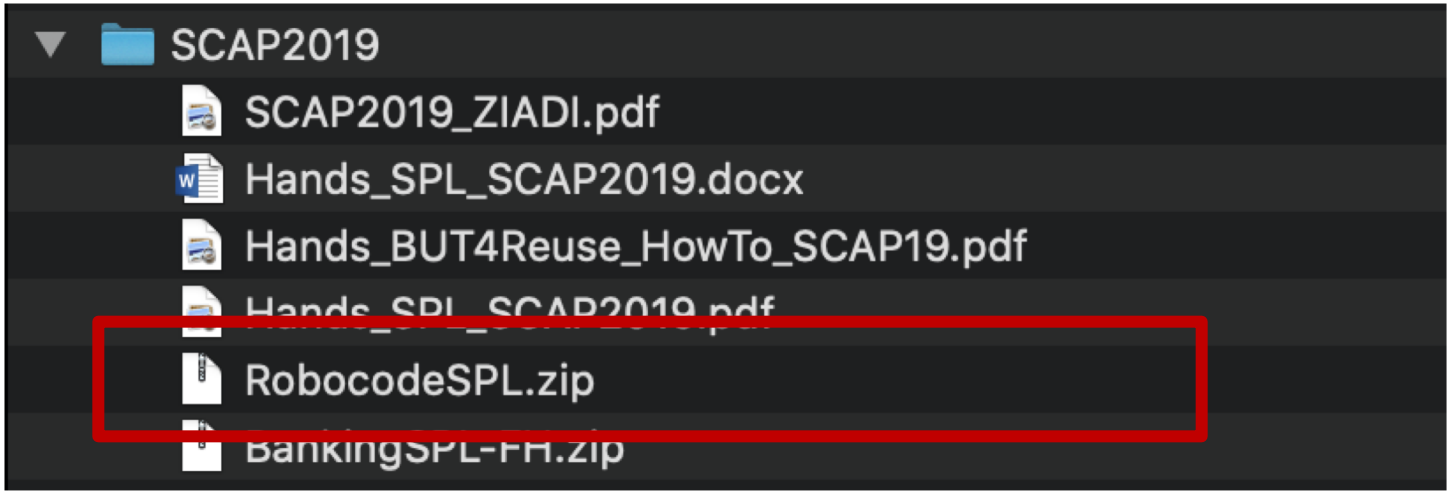
Bots





```
public class PerpendicularMovement extends Movement {

    public PerpendicularMovement(Module bot) {
        super(bot);
    }


     double direction = 1; // used to know if it should go forwa
     static final double favorite_distance = 200; //how far away


    public void move() {

        if (bot.enemy!=null)
        {
            if(bot.getDistanceRemaining() == 0){
                direction=-direction;
                 bot.setAhead(300*Math.random()*direction);
            }
            bot.setTurnRight(bot.enemy.bearingRadians +
                    90 - 90*Math.random()* direction * (bot.enem
            }
        }
```

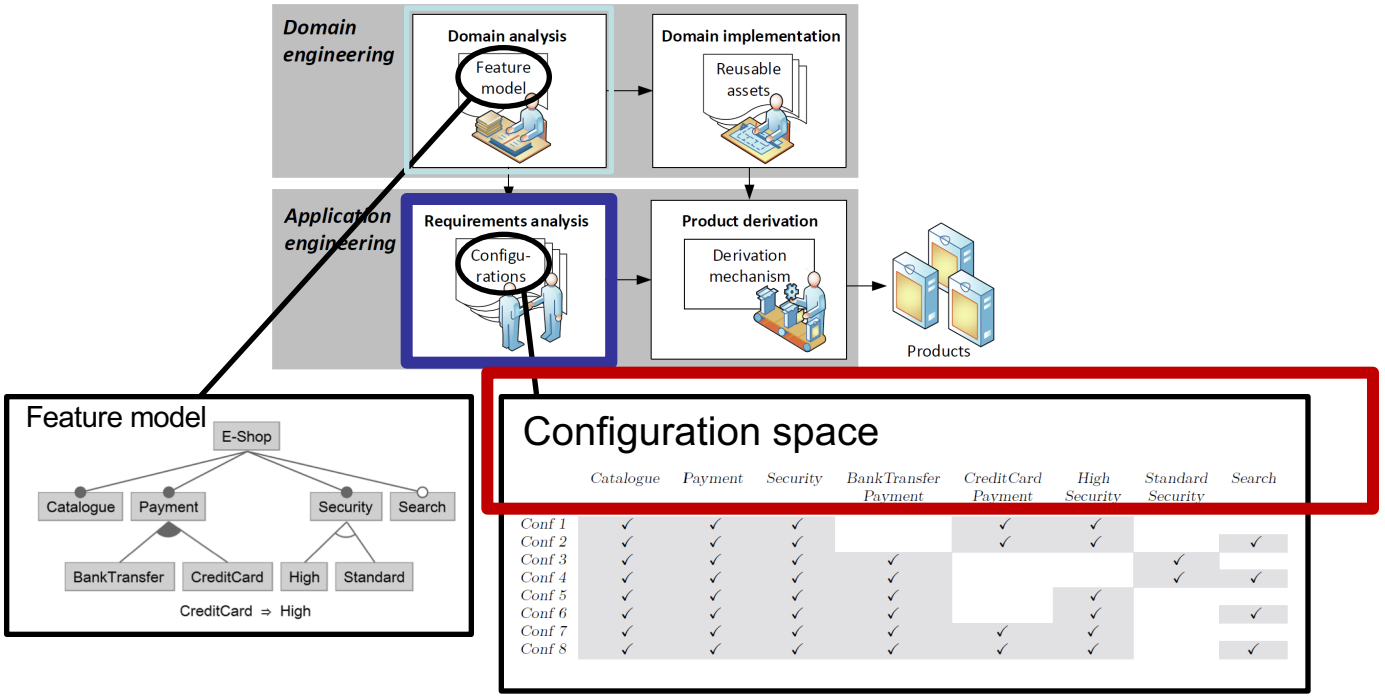




https://pages.lip6.fr/tewfik.ziadi/SCAP2019.zip

LIVE DEMO

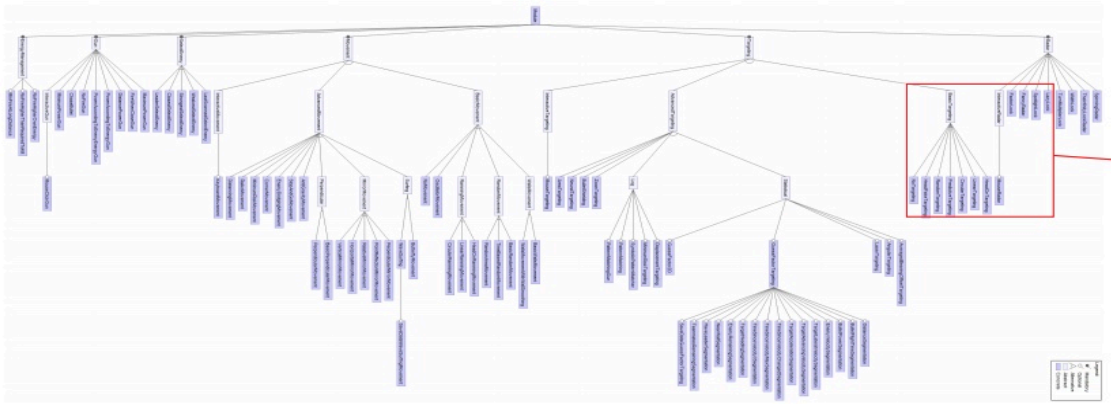
Software product line engineering

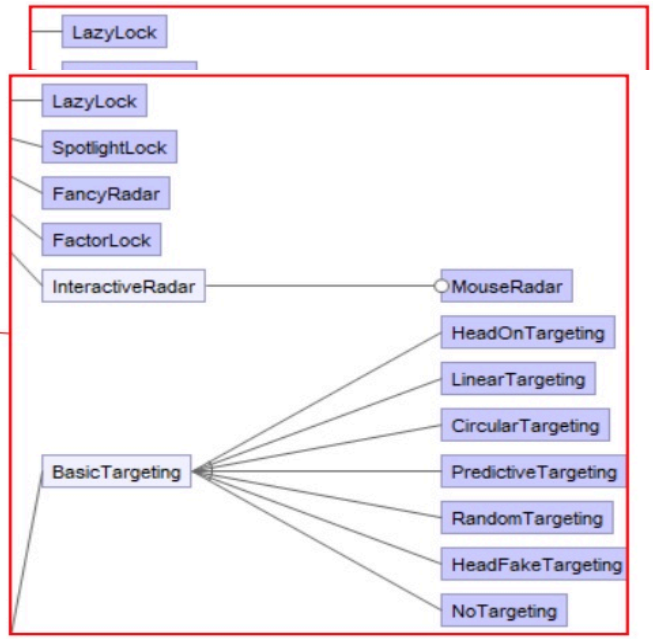


Research Direction: Configuration Space Analysis

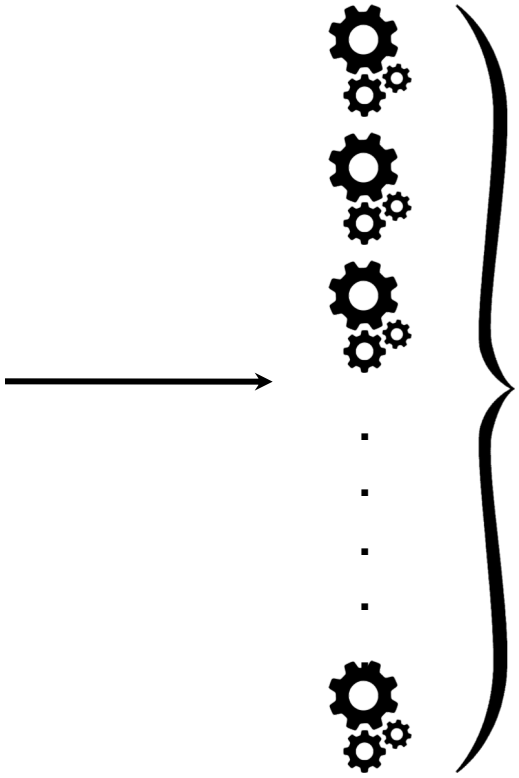
What are the best Robocode configurations?

- Battle competition!



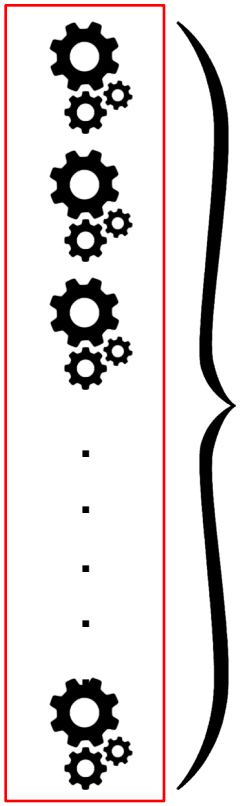


Challenge: What is the **best** configuration?



Configurations

Challenge: What is the **best** configuration?



Configurations

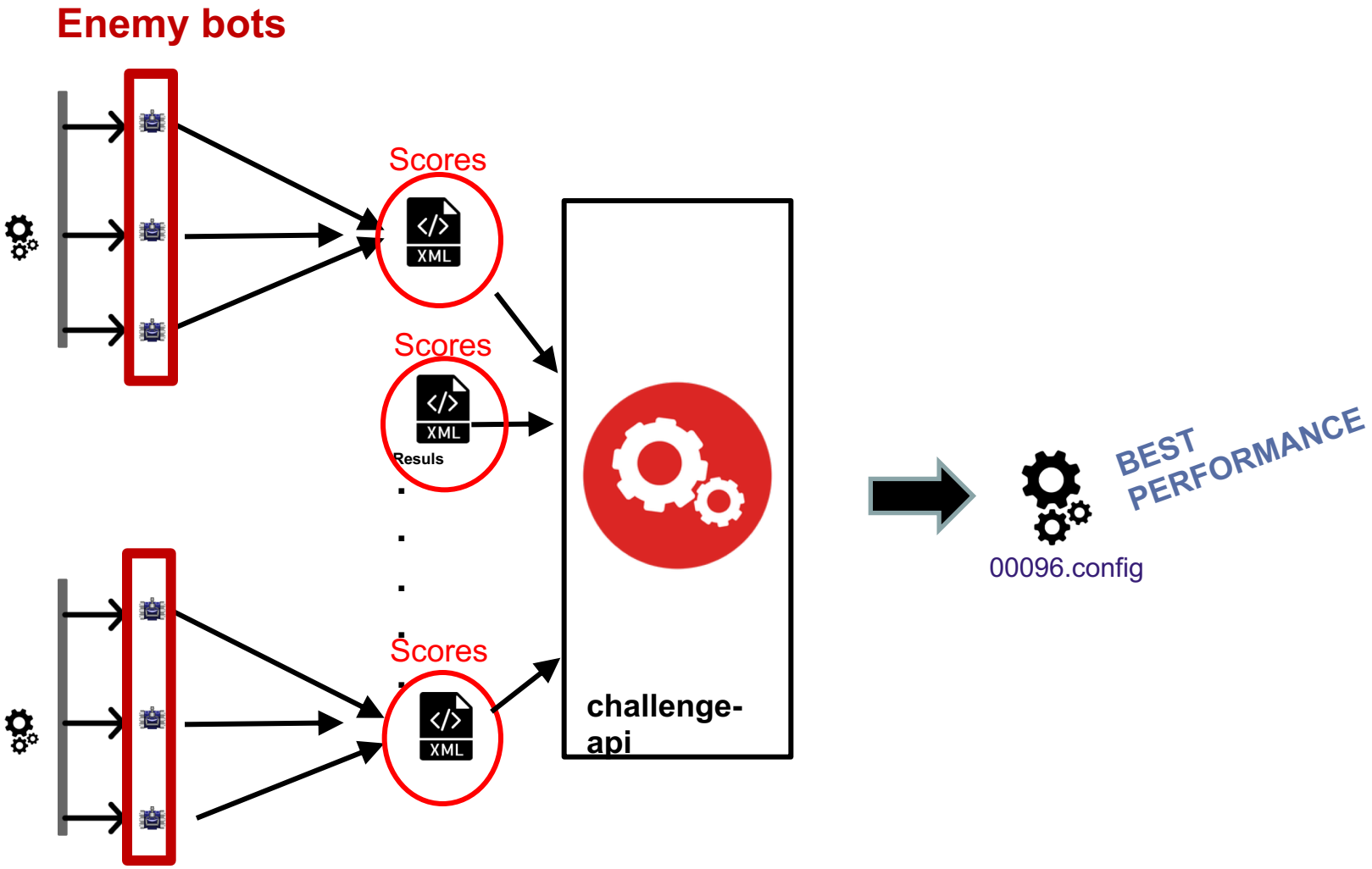
Robocode SPL Challenge

Enemy bots

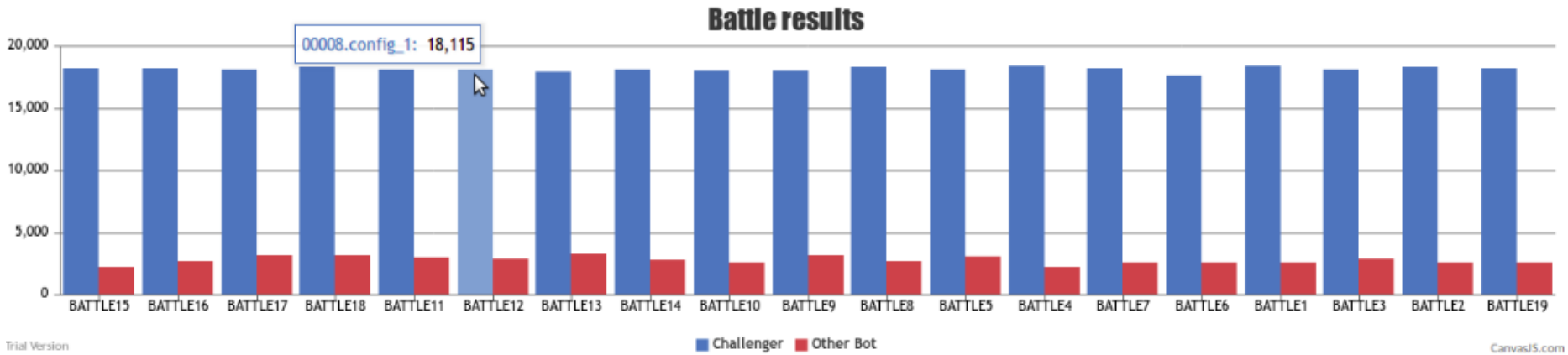
BEST PERFORMANCE

00096.config

Challenge: What is the **best** configuration?



Challenge: What is the **best** configuration?

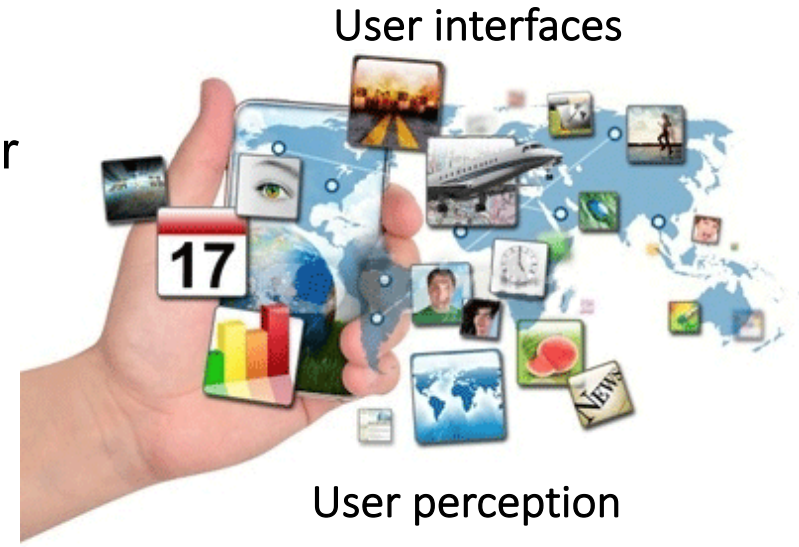


BEST PERFORMANCE : 00095.config

Human-Computer Interaction Systems

- What about systems without quantity attributes but more based on **human assessments** ?
  - Human assessments are subjective by nature in HCI systems.

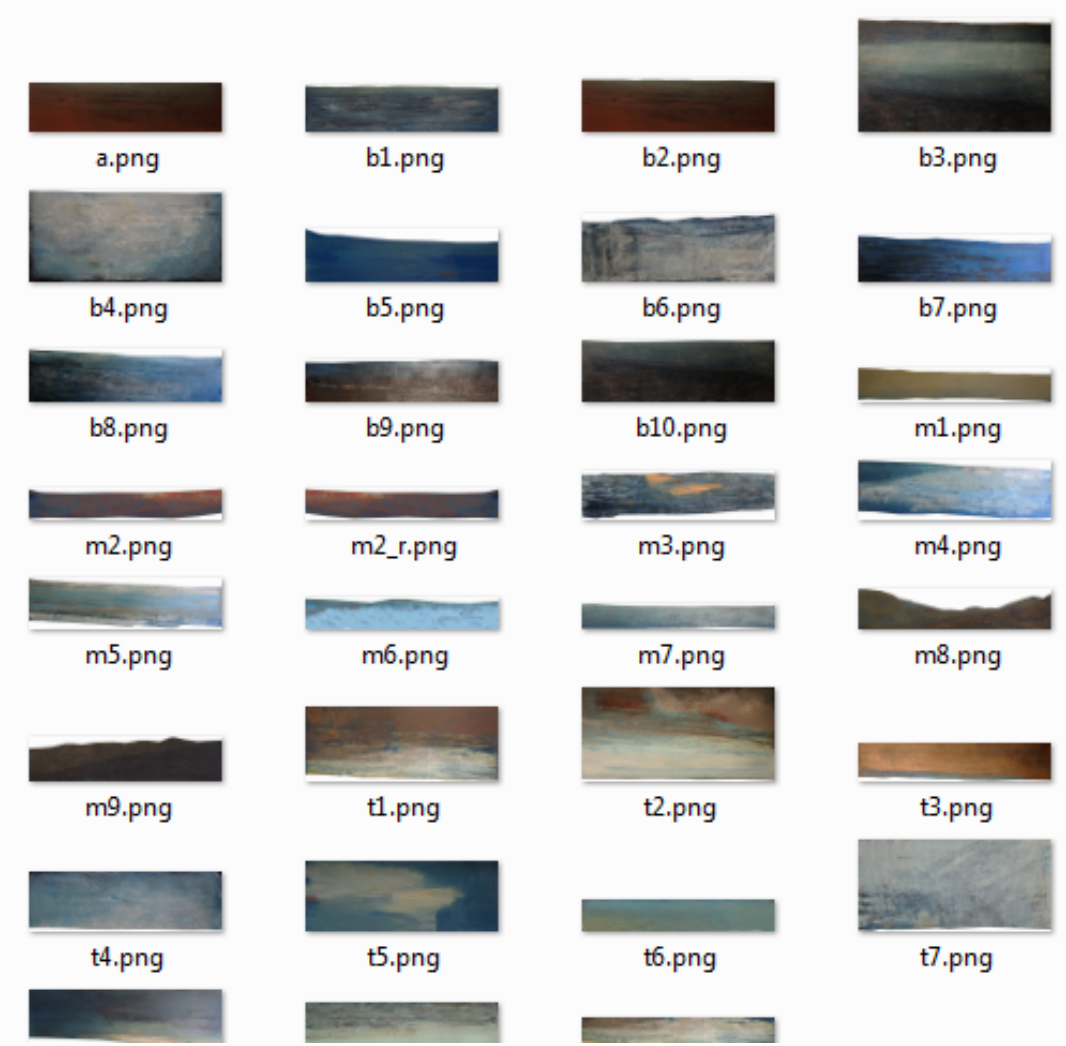
- Our **SPLRank** approach:
  - Martinez et al, 2017 (book chapter
  - Martinez et al, 2018 (ICSR2018)



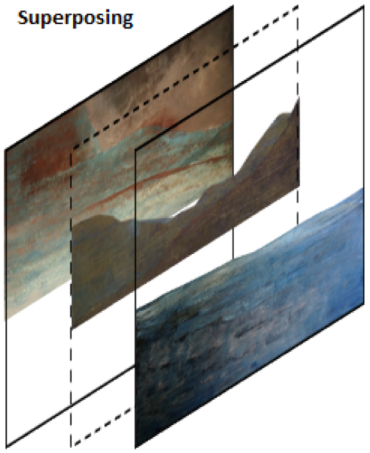
Gabriele Rossi

Professional Art Painter based on Paris


sky
middle
ground

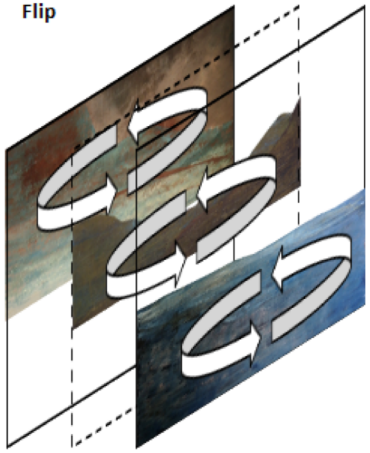

a.png
b1.png
b2.png
b3.png
b4.png
b5.png
b6.png
b7.png
b8.png
b9.png
b10.png
m1.png
m2.png
m2_r.png
m3.png
m4.png
m5.png
m6.png
m7.png
m8.png
m9.png
t1.png
t2.png
t3.png
t4.png
t5.png
t6.png
t7.png

Superposing



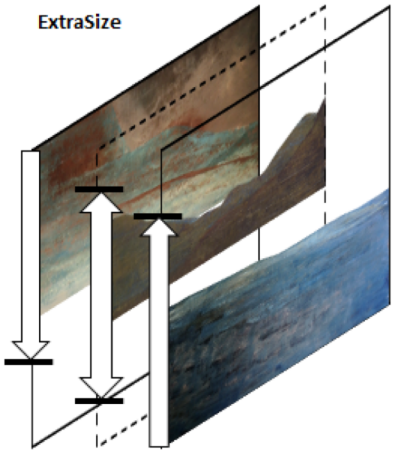
Config: S1, M1, G1

Flip



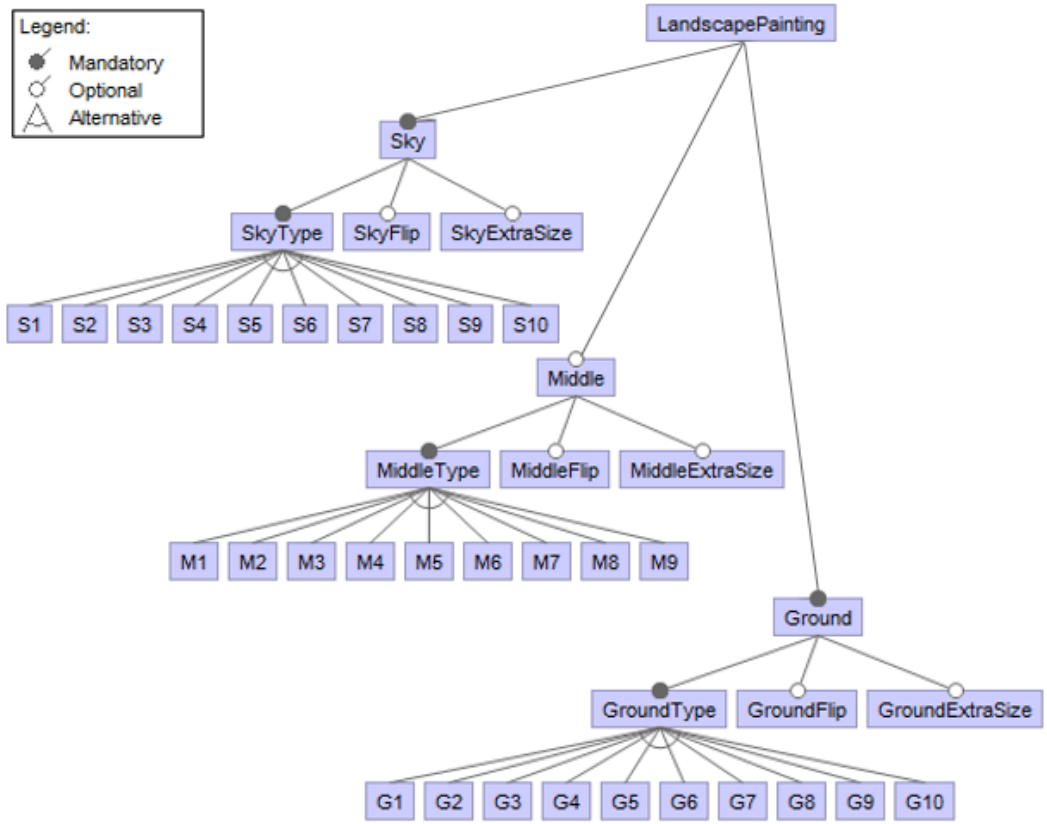
Config: S1, M1, G1, GroundFlip

ExtraSize

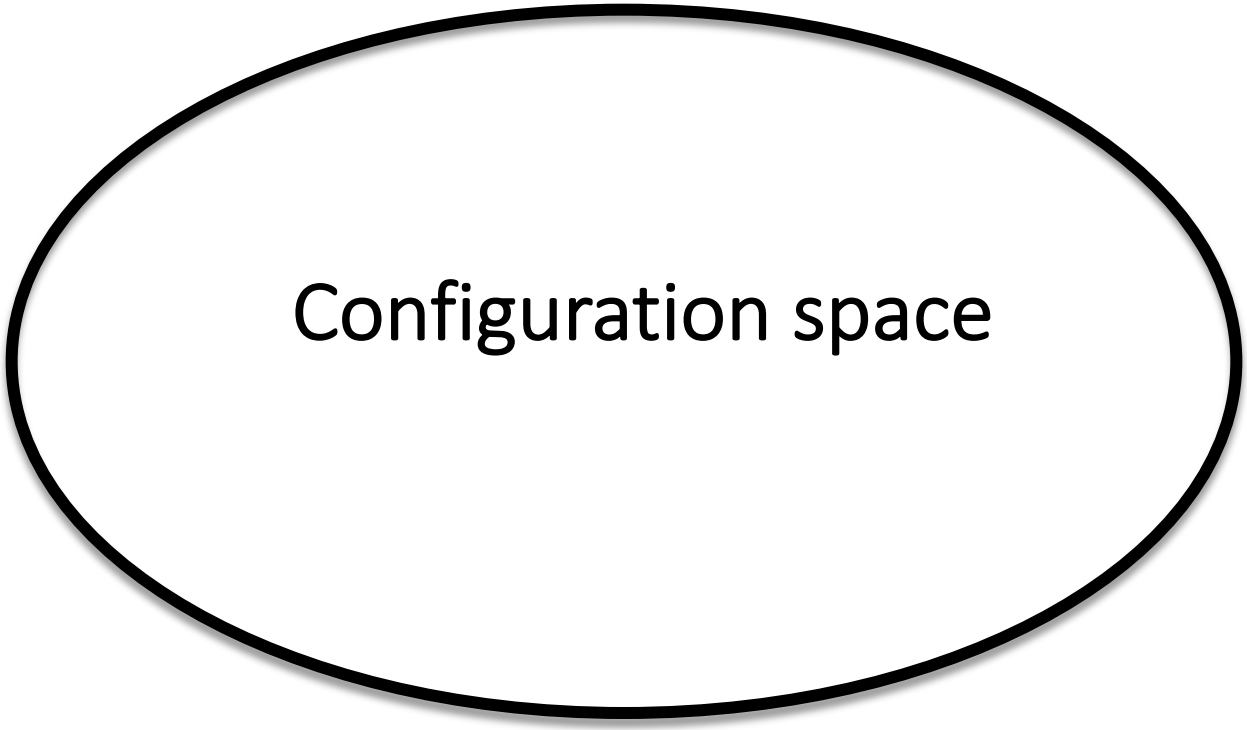


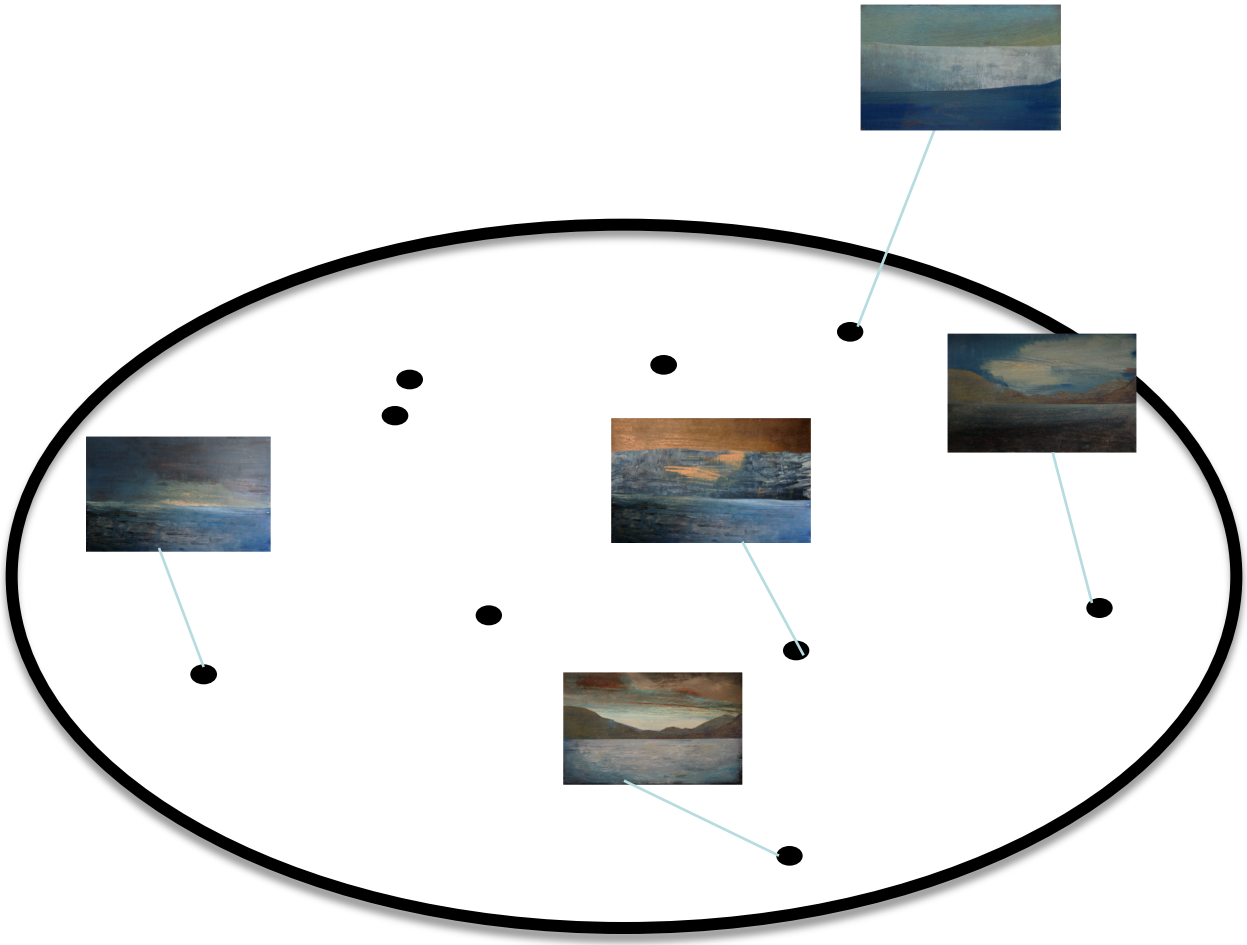
Config: S1, M1, G1, SkyExtraSize, MiddleExtraSize, GroundExtraSize

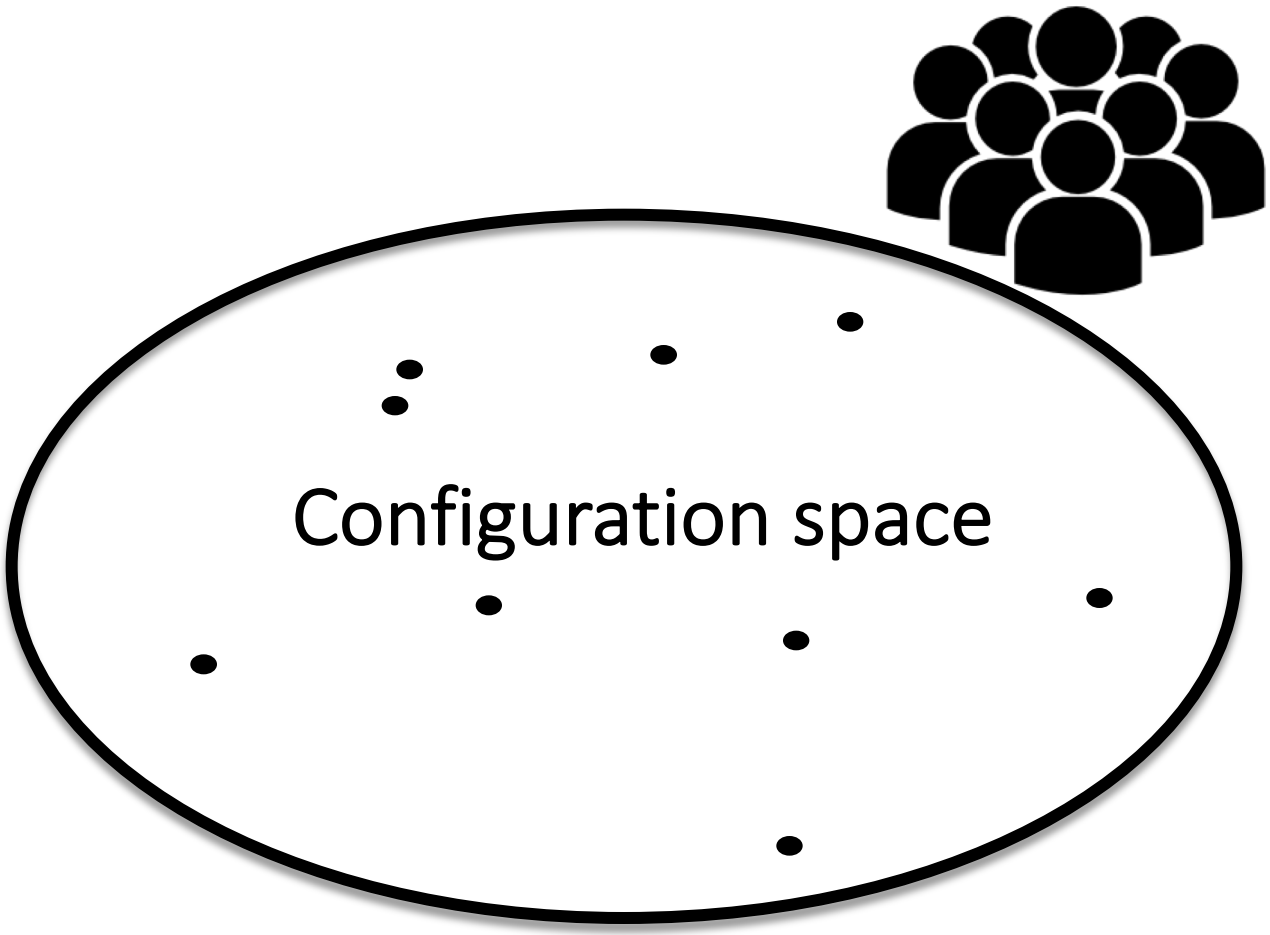
Generative-art SPL

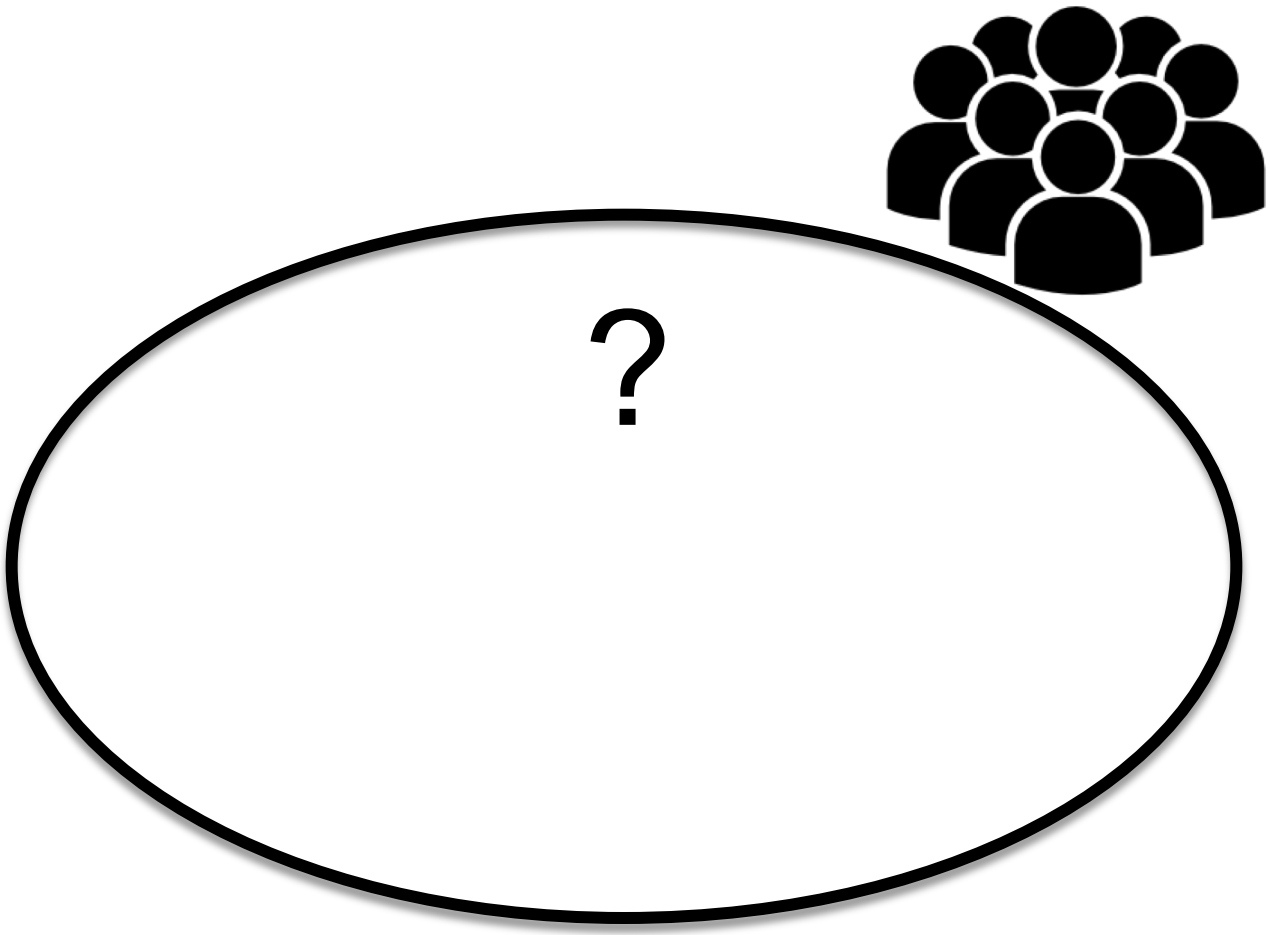


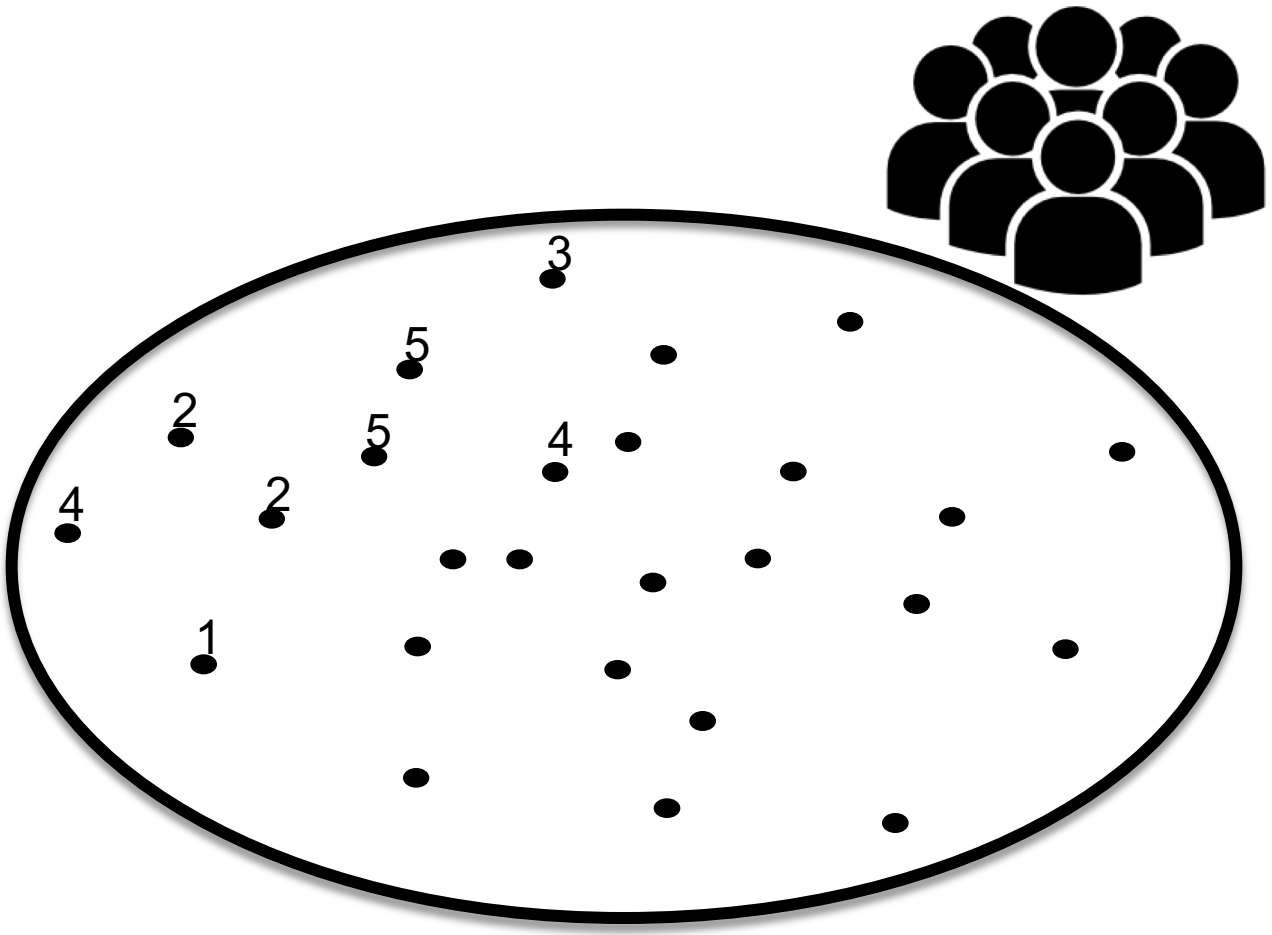
59200 possible artworks


Configuration space




Configuration space




3
5
2
5
4
2
4
1

Generative-art: Challenges

- **RQ1**: Given the combinatorial explosion of configurations and the limited resources, how can one identify and select the optimal subset of products that are relevant for user assessment?

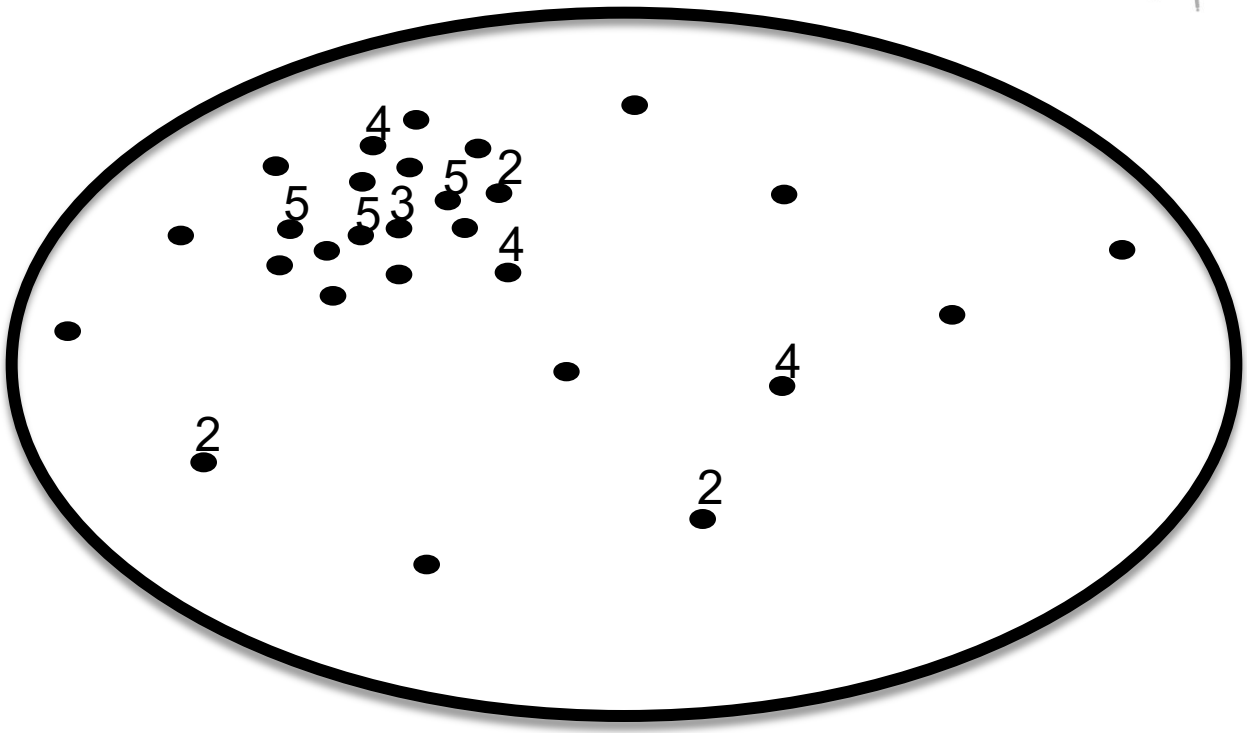
- **RQ2**: Given a subset of assessed product variants, how can one infer the user feedback of the non assessed products? How can one aggregate user feedback to calculate new predictions even for the already assessed products?

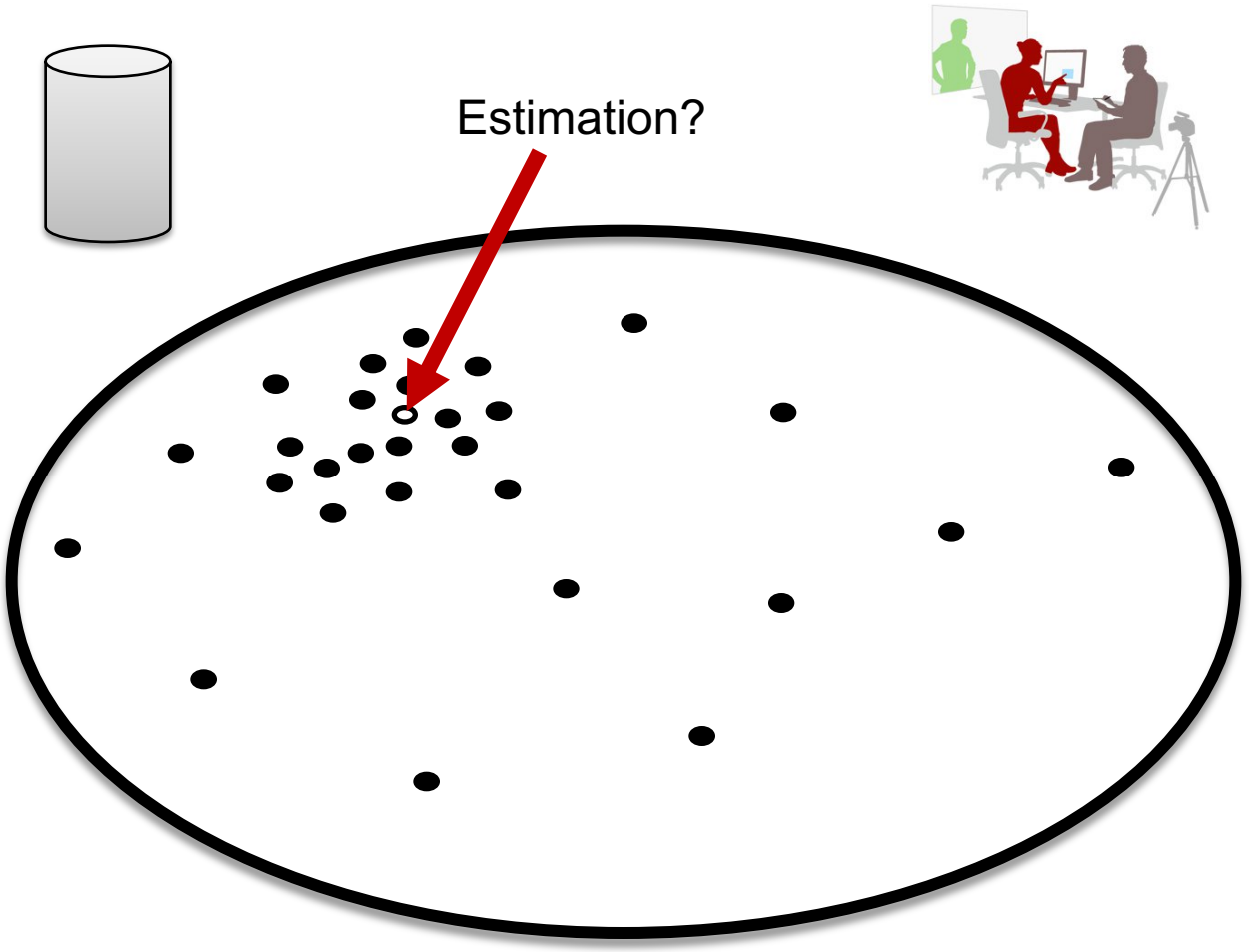
Generative-art: The approach

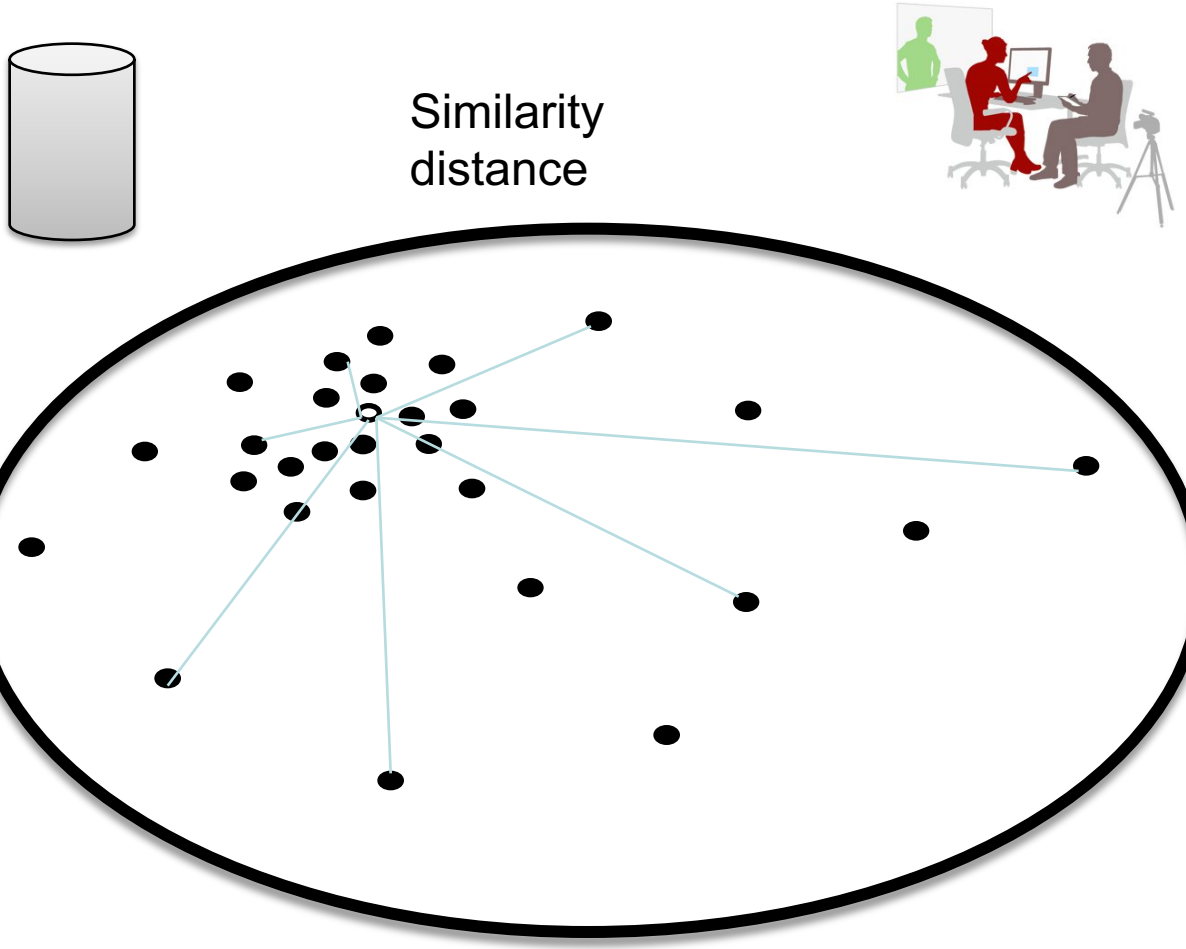
- **Phase 1:** Data set creation through evolution.

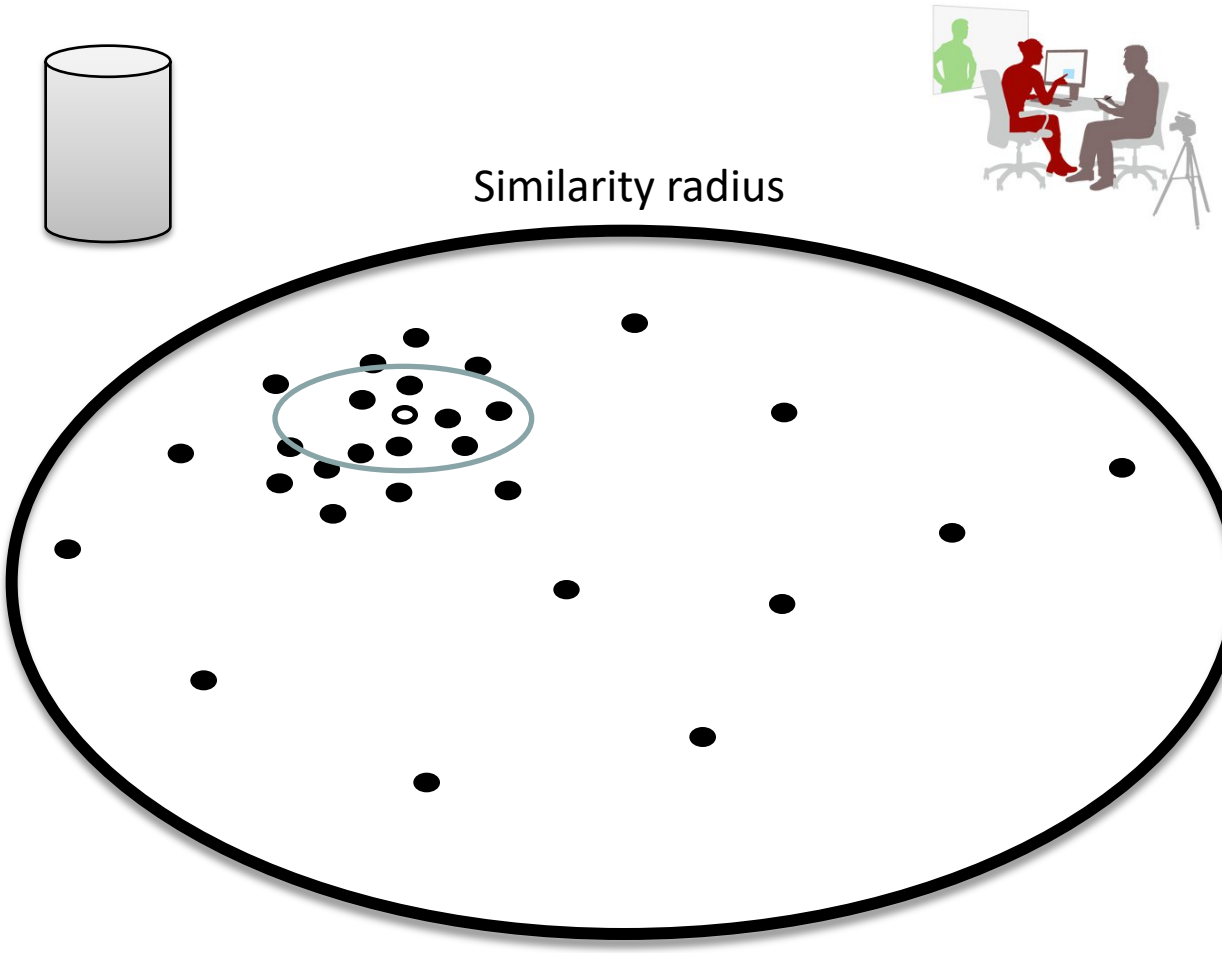
- **Phase 2**: Ranking computation
  - Similarity distance.
  - Weighted mean score computation.
  - Empirical selection of the approach settings.

Interactive Genetic Algorithm

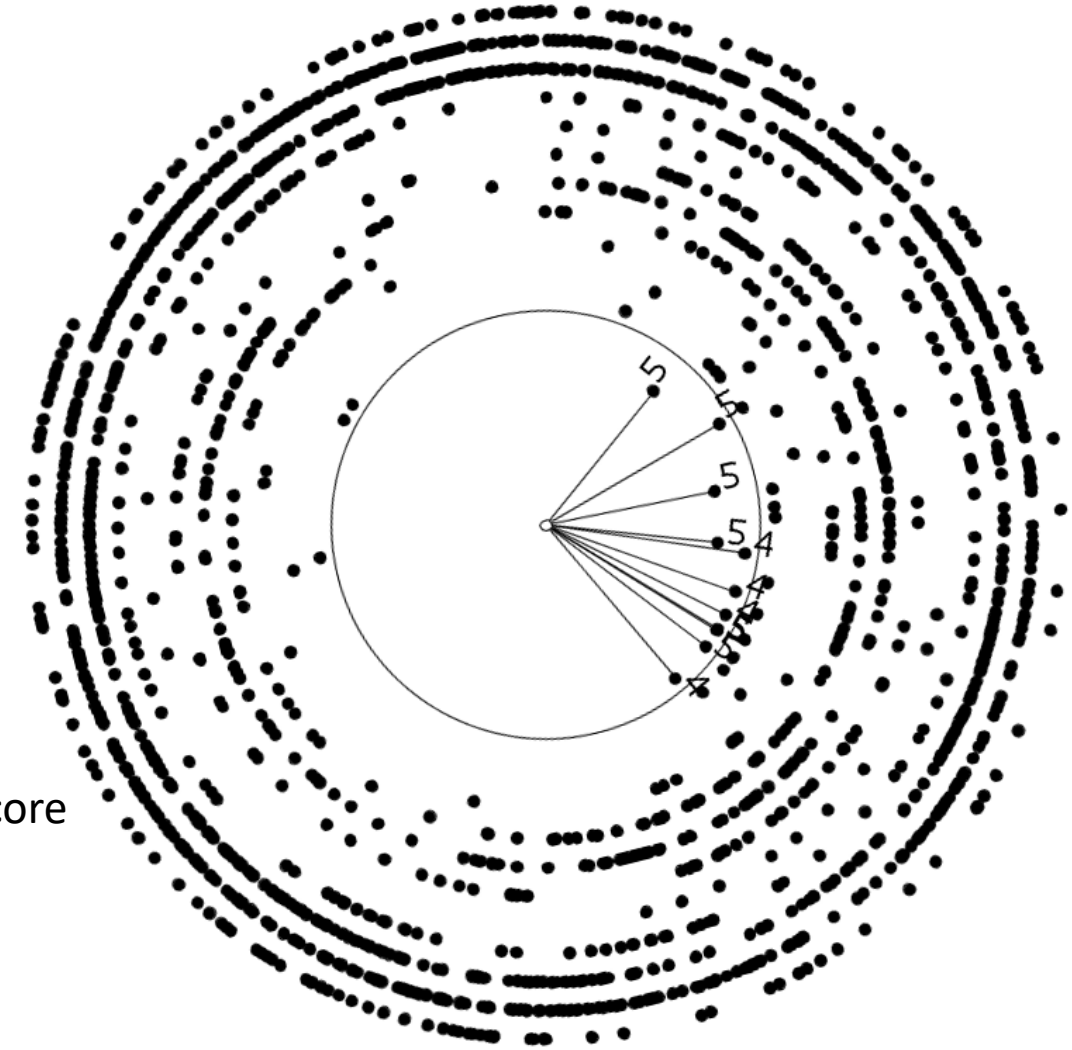

4
5
5
3
5
2
4
4
2
2


Estimation?


Similarity
distance


Similarity radius

Weighted mean score

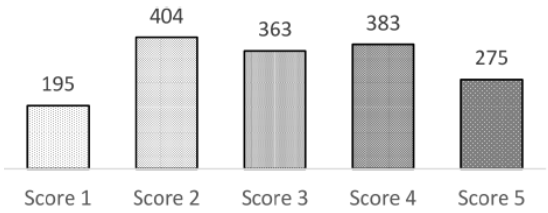



MANIFIESTA 7
Au Theatre de Verre
LE 8 MARS 2014
Installations
Performances
Expositions
Concerts
A PARTIR DE 17H
17 RUE DE LA CHAPELLE
www.theatredeverre.com
3€

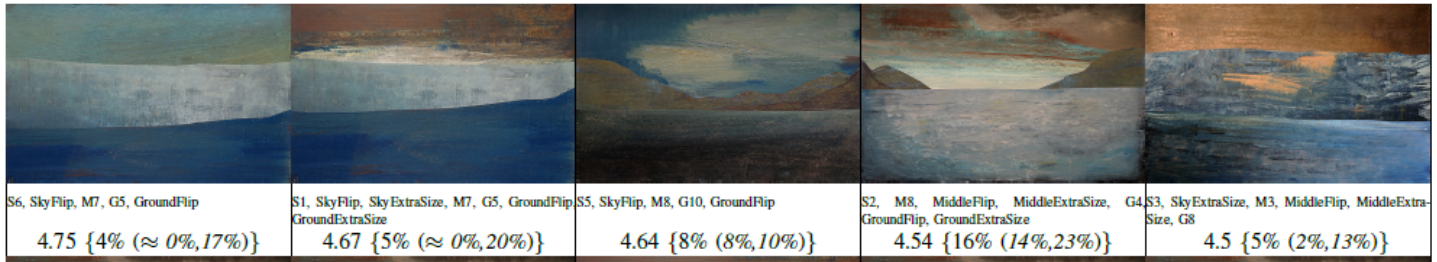
4 hours and 42 minutes

around 150 persons

1,620 votes
81 generations



TOP 10. EACH CONFIGURATION IN THE RANKING IS DEFINED BY: WMEAN { GCONF% (*ndenc%, nsimc%*)}



S6, SkyFlip, M7, G5, GroundFlip 4.75 {4% (≈ *0%,17%*)}	S1, SkyFlip, SkyExtraSize, M7, G5, GroundFlip, GroundExtraSize 4.67 {5% (≈ *0%,20%*)}	S5, SkyFlip, M8, G10, GroundFlip 4.64 {8% (*8%,10%*)}	S2, M8, MiddleFlip, MiddleExtraSize, G4, GroundFlip, GroundExtraSize 4.54 {16% (*14%,23%*)}	S3, SkyExtraSize, M3, MiddleFlip, MiddleExtraSize, G8 4.5 {5% (*2%,13%*)}
S2, M8, G4, GroundFlip 4.42 {33% (*36%,32%*)}	S2, M8, G10, GroundFlip 4.41 {43% (*48%,27%*)}	S1, SkyFlip, SkyExtraSize, M5, MiddleFlip, MiddleExtraSize, G4, GroundFlip 4.4 {6% (≈ *0%,23%*)}	S2, M8, G4 4.36 {48% (*56%,23%*)}	S2, M8, MiddleFlip, G10, GroundFlip 4.35 {37% (*40%,30%*)}

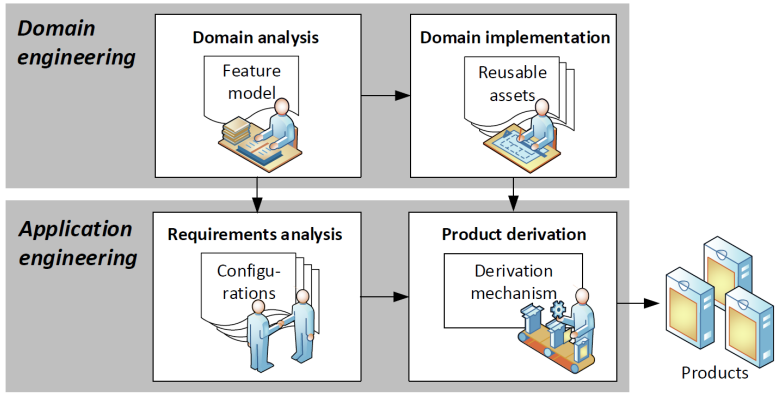
. . .

. . .

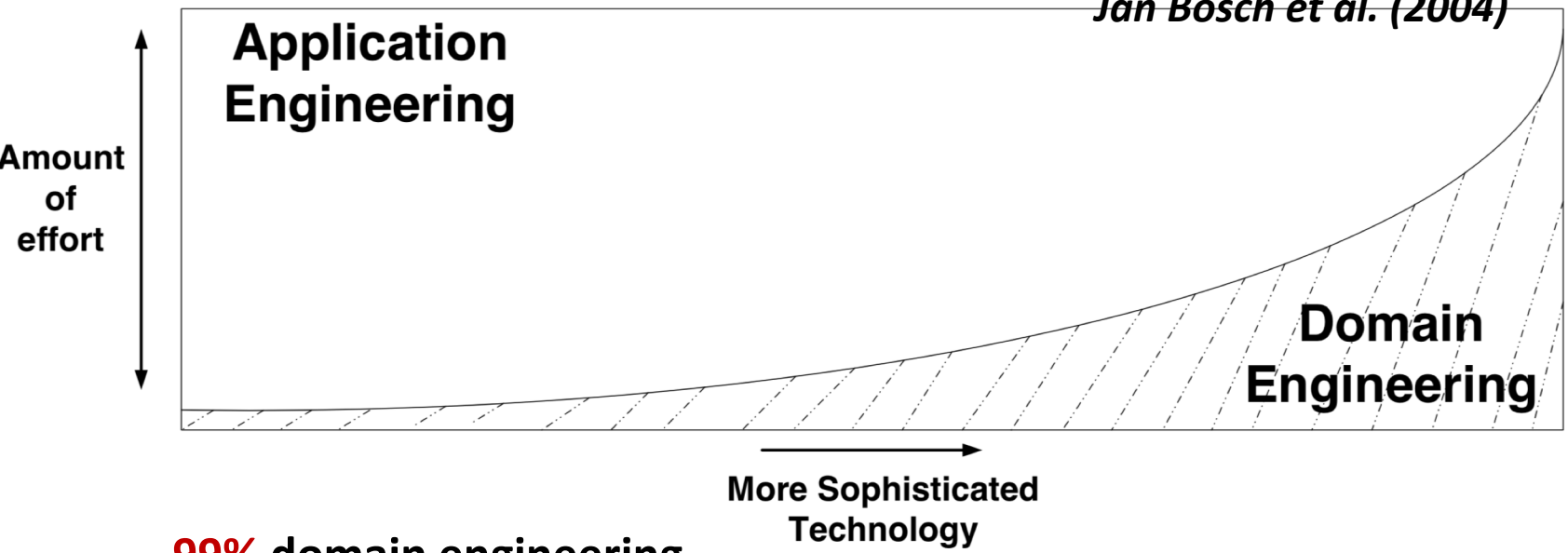
. . .

BOTTOM 5. EACH CONFIGURATION IN THE RANKING IS DEFINED BY: WMEAN { GCONF% (*ndenc%, nsimc%*)}

S5, SkyExtraSize, M6, MiddleExtraSize, G5, GroundFlip 1.5 {4% (*2%,8%*)}	S4, SkyFlip, M4, MiddleFlip, G6, GroundFlip 1.5 {3% (*1%,11%*)}	S1, SkyFlip, M1, MiddleExtraSize, G5, GroundFlip, GroundExtraSize 1.33 {5% (≈ *0%,20%*)}	S1, SkyFlip, M1, G5, GroundFlip, GroundExtraSize 1.33 {3% (≈ *0%,11%*)}	S5, M5, G10, GroundFlip, GroundExtraSize 1 {3% (≈ *0%,13%*)}



Jan Bosch et al. (2004)



99% domain engineering,
1% application engineering?

Conclusion

- Software Product Lines to manage Software Variability in a Systematic way.

- A very active research area with many research directions:
  - Software Product Line Implementation
  - Product Derivation
  - Configuration Space Exploration
  - Re-engineering Software Variability into SPL