Software Product Line Engineering:

From Research Ideas to industrial Tools

Tewfik Ziadi
Sorbonne Université/LIP6
&
RedFabriQ
Tewfik.Ziadi@lip6.fr

About me

- **My Current Positions**

  - **Since 2005, Associate Professor, HDR** at Sorbonne University, Paris France.

  - **Leader** of the MoVe research team in the LIP6 lab

  - **Leader of the R&D** division at RedFabriQ / Mobioos AI (Part Time)

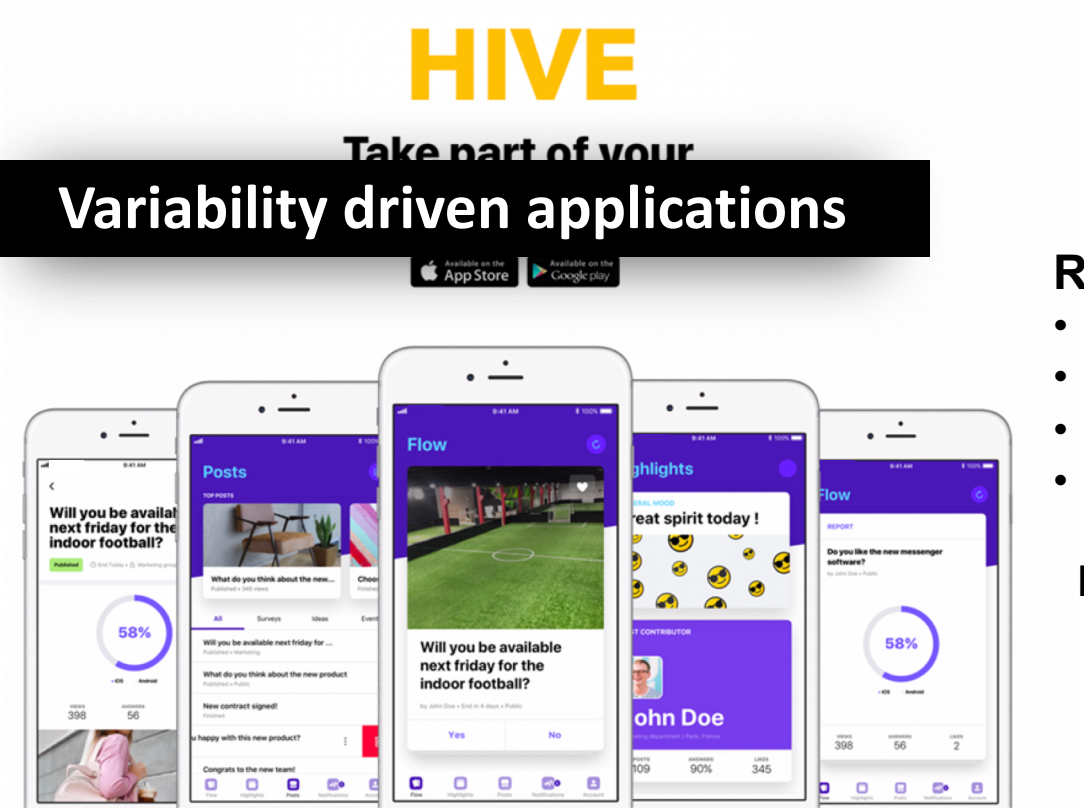
    

SORBONNE UNIVERSITÉ

CRÉATEURS DE FUTURS DEPUIS 1257

- **55 300** students
- **4 400** PhD students.
- **6 700** faculty members

redfabriq



mobioos

R&D Lab:

- Software Variability
- Software Product Lines
- Domain Specific Languages
- Low code platform

PhD Thesis (CIFRE)

LIP6

SORBONNE UNIVERSITÉ

cnrs dépasser les frontières

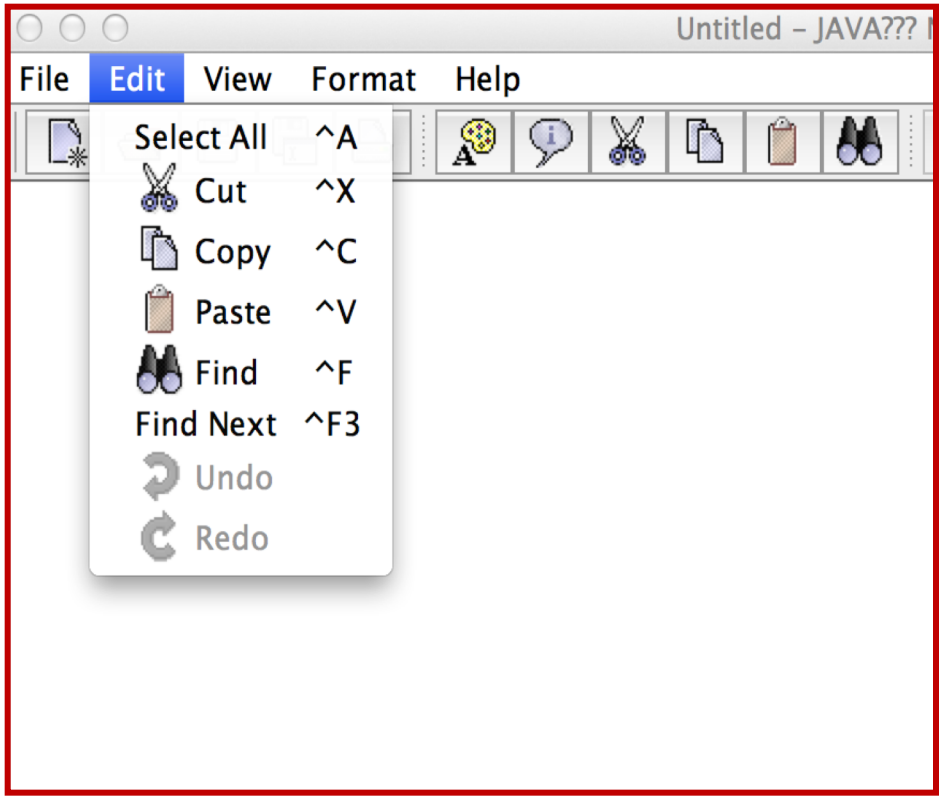
Content

- **Software Product Lines fr (SPL)**
  - Concepts & existing approaches to implement SPL
    - **FeatureIDE**: an open source tool
    - The Robocode case study
  - **SPL in industry:** the Mobioos Forge

LIVE DEMOS

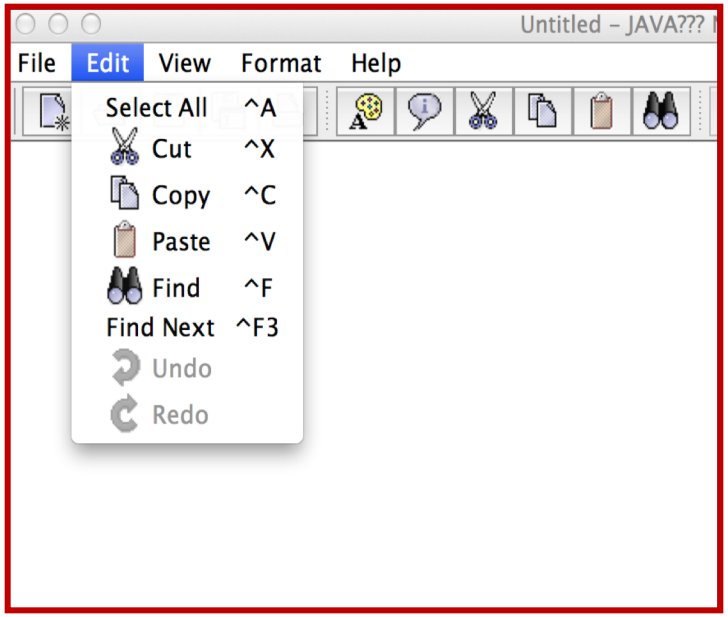
SPL: Motivations

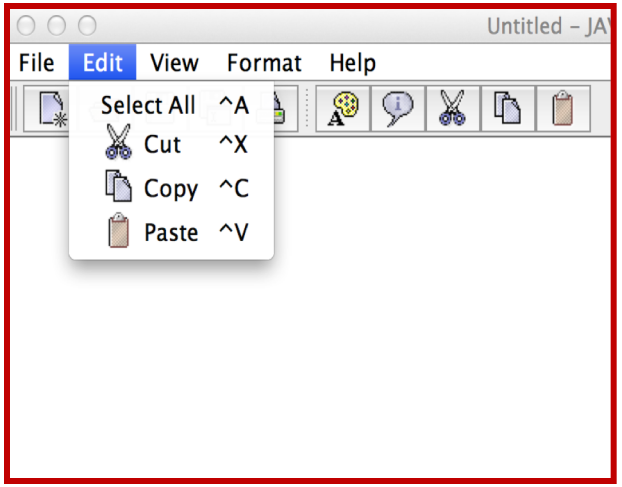
- We have the implementation in Java of a simple Notepad application

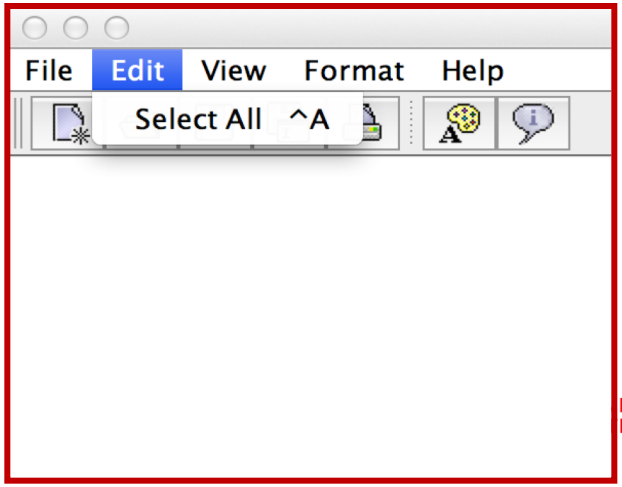


SPL: Motivations

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







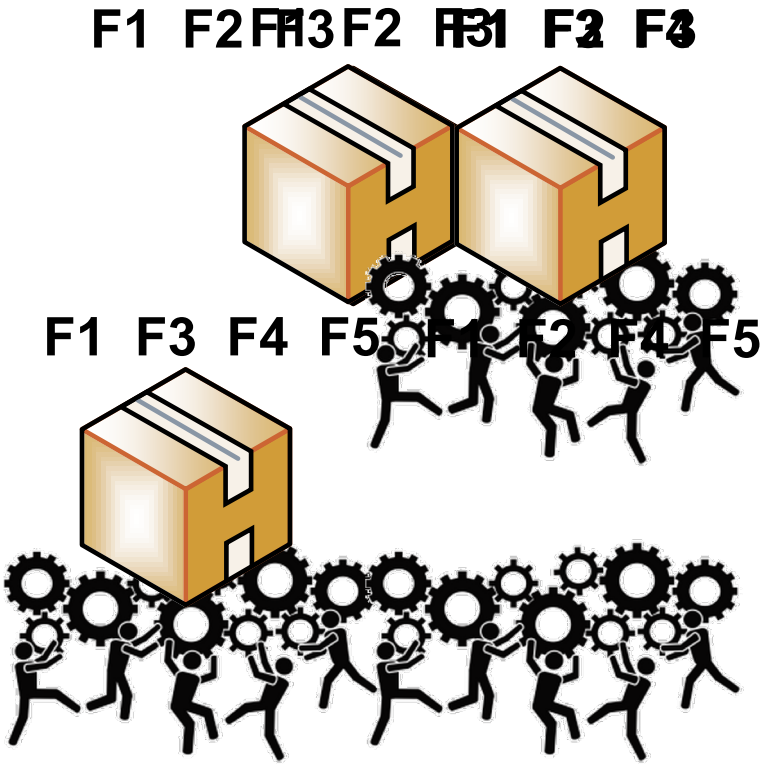
Ad hoc solution

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

Clone-and-own:

Opportunistic reuse

Clone-and-own



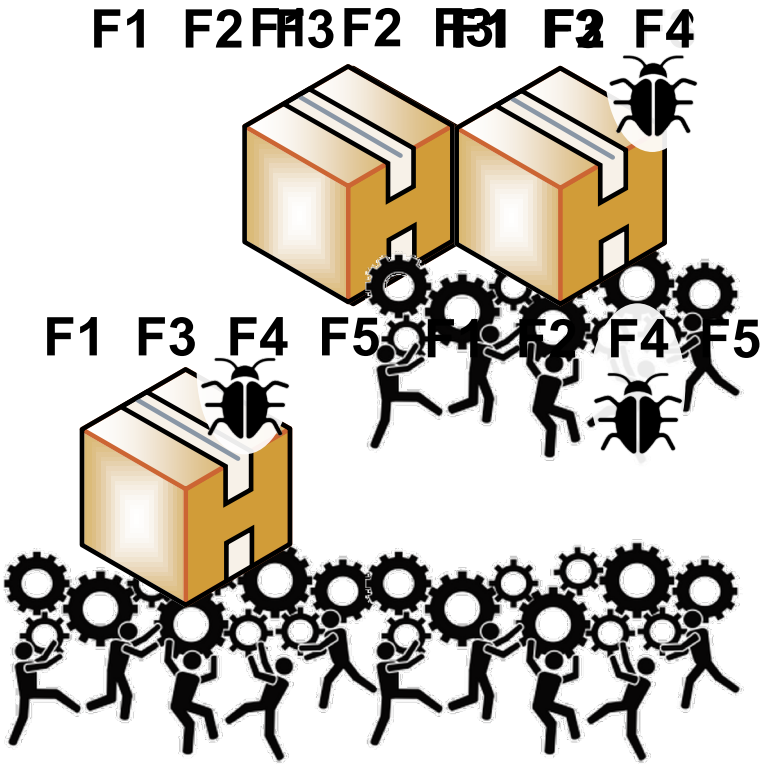
50% of industrial companies use clone-and-own to create variants.

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

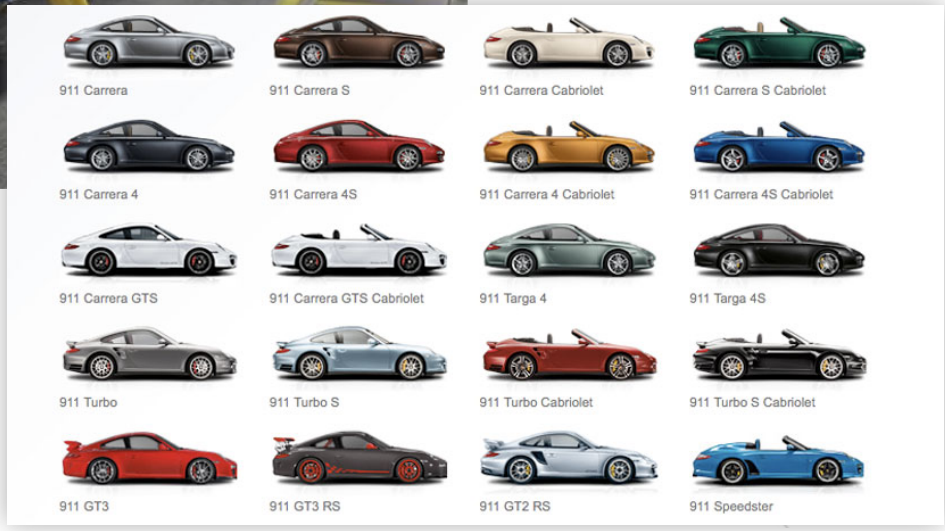
Content

- **Software Product Lines fr (SPL)**
  - Concepts & existing approaches to implement SPL
    - **FeatureIDE**: an open source tool
    - The Robocode case study
  - **SPL in industry:** the Mobioos Forge

LIVE DEMOS

Product Lines in Industry

Variants



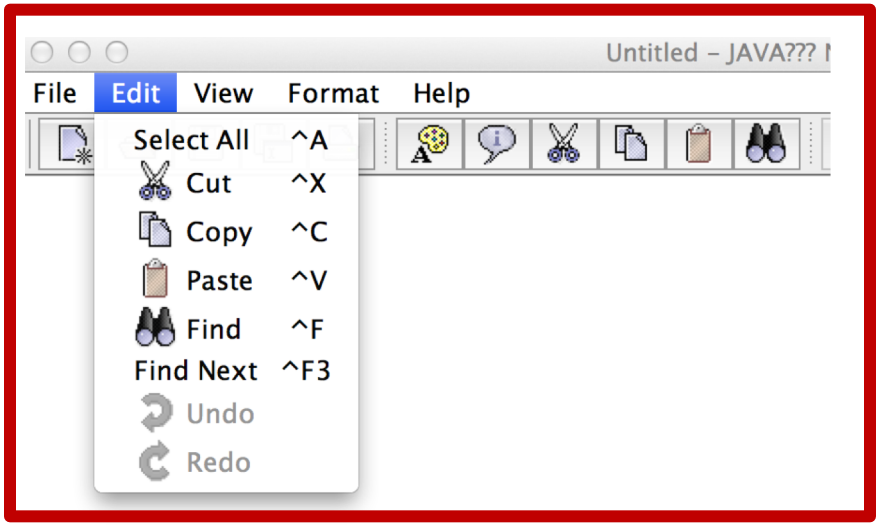
Why can’t build our application as we build our car ?

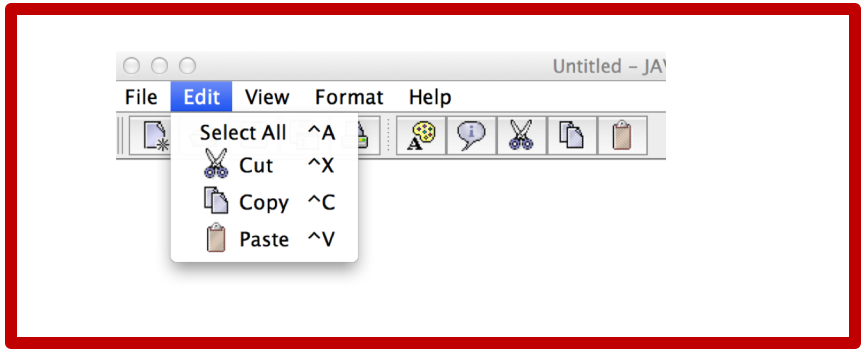
- Sotware systems

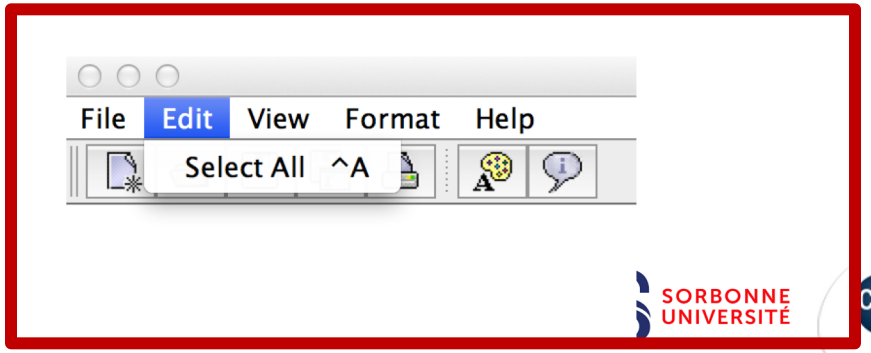
Comme in many **variants**

Printer Firmware

Classical Software

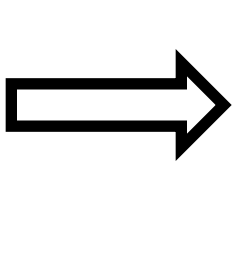


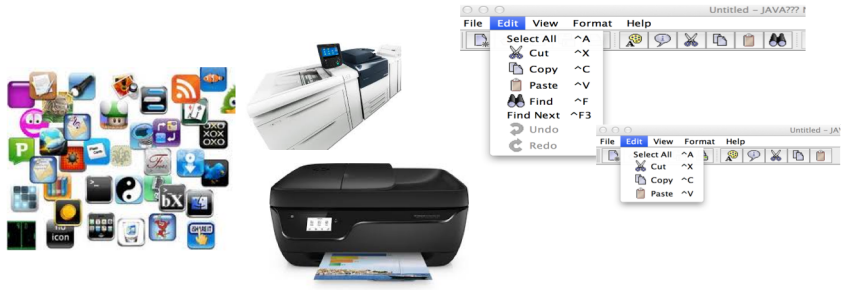




SORBONNE UNIVERSITÉ

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**

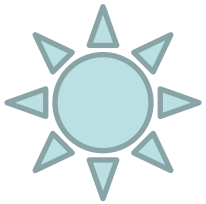
LIP6

SORBONNE UNIVERSITÉ

CNRS dépasser les frontières

Version vs. Variant

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

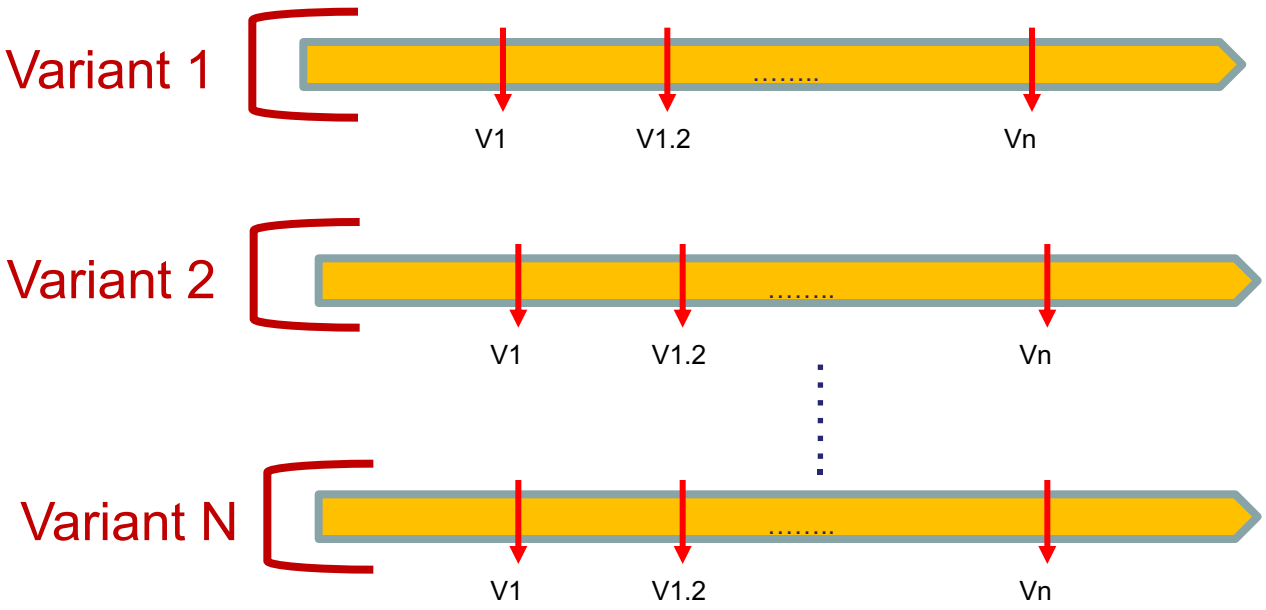


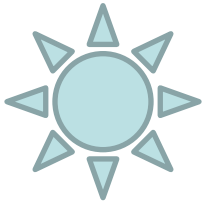
The **same software** that evolves in time

➔ Version Control Systems (git, svn..)

Variante vs. Version

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

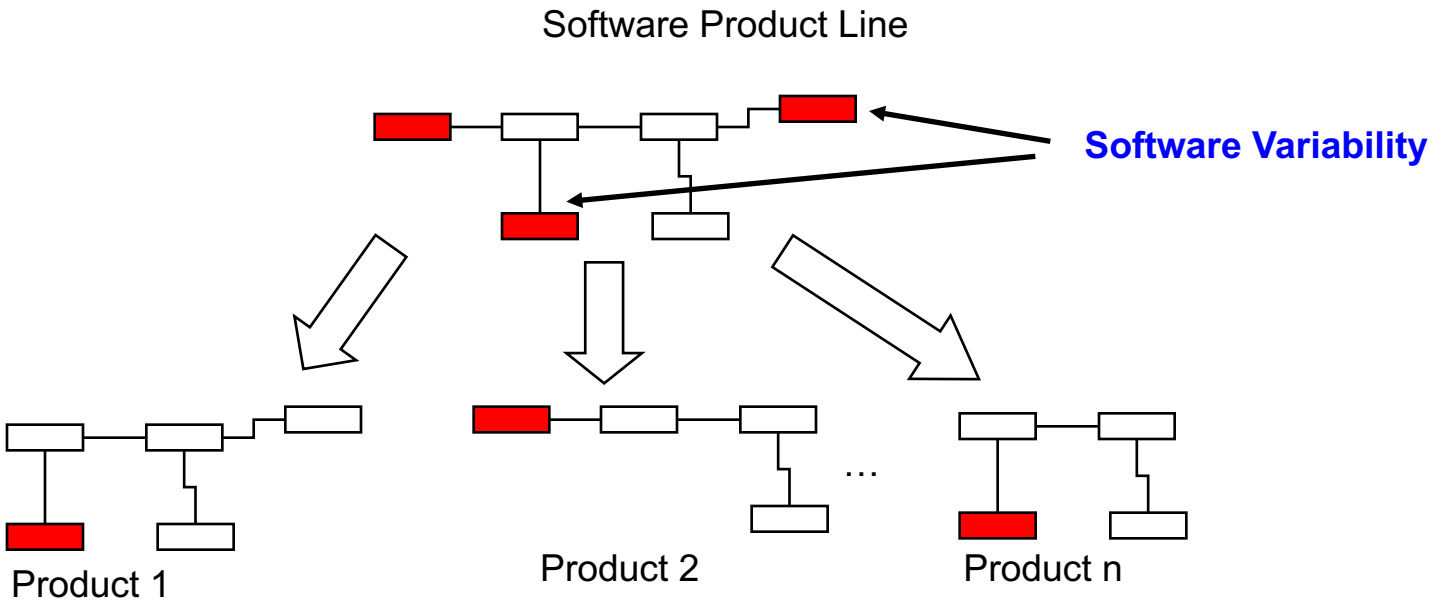




At the same time**, many variants exist**

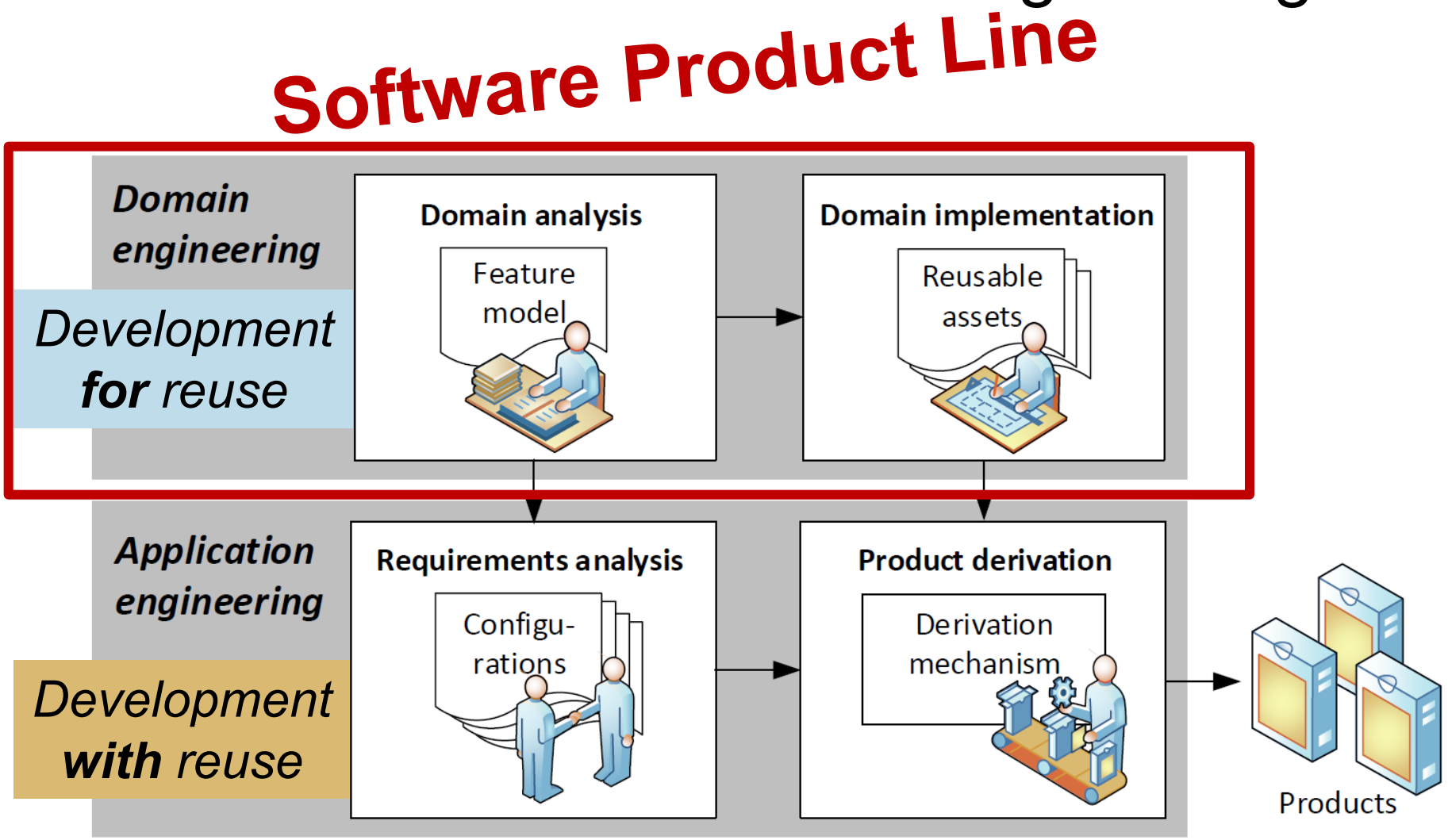
➔ Software Product Lines

Software Product Line

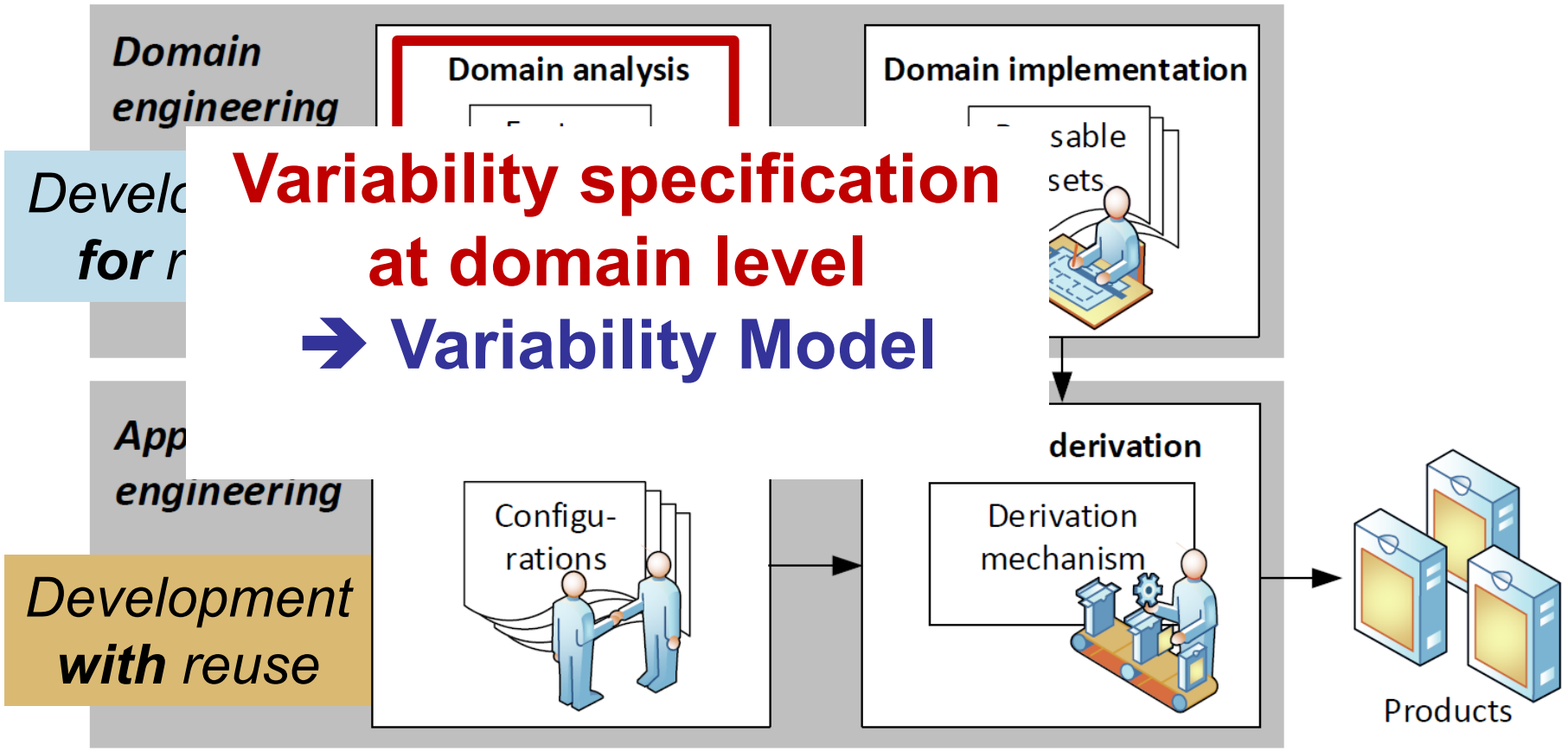


- ✓ **Dimension 1 : Variability Management.**
- ✓ **Dimension 2 : Product Derivation.**

Software Product Line Engineering

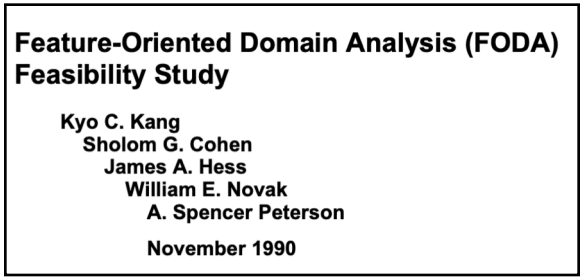


Phase1: Domain Analysis



Variability Model

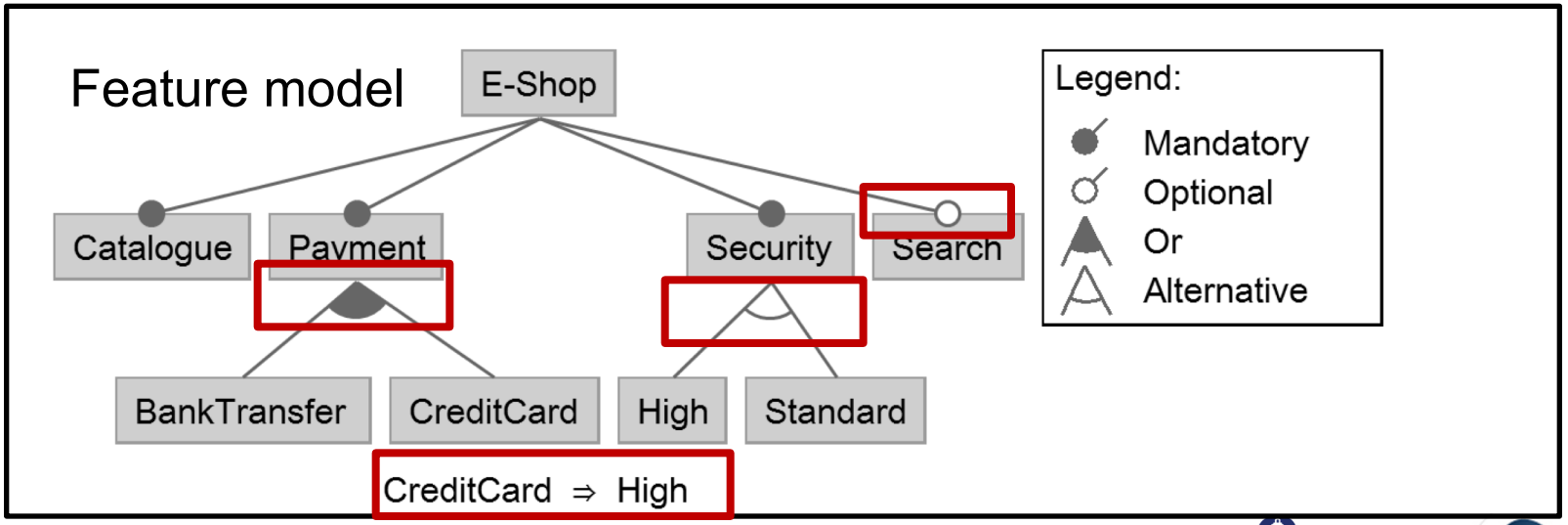
- **Explicit specification of variability**
- **Feature Model: *de facto* standard!**



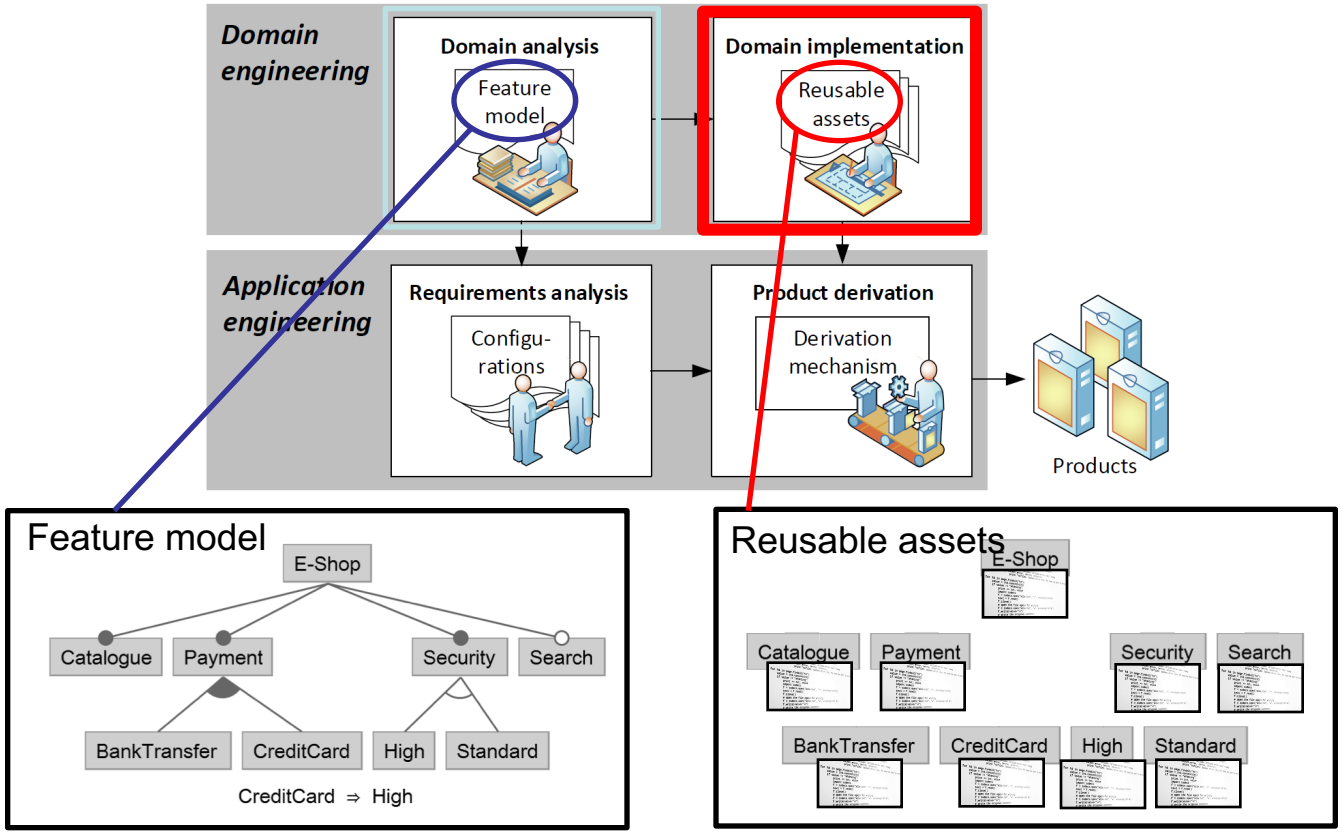
- **Research**
  - 5000++ citations!
  - Central for many research work
  - Tools & Languages: GUIDSL/FeatureIDE, SPLOT, FaMa, etc.)
- **Industry**
  - Tools (Gears, pure::variants),

Feature Model

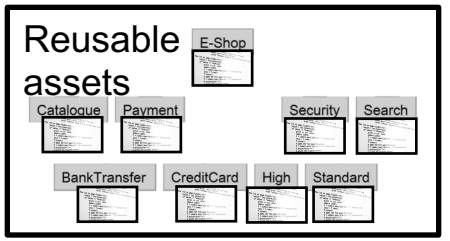
- The concept of **« feature »**
- **Hierarchy**: rooted tree
- Notations for **variability**
  - Optional, alternative, Or, constraints..



Software Product Line Engineering

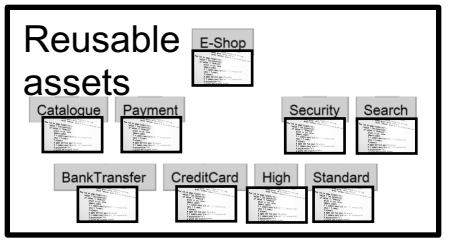


Domain Implementation



- **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.

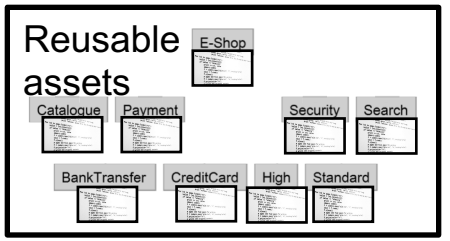
Domain Implementation



- How the reusable assets can be implemented?

Research Direction: **Software Product Lines Implementation**

Domain Implementation



- How the reusable assets can be implemented?

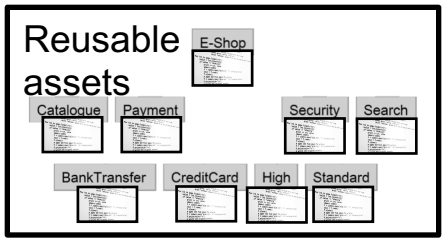
Research Direction: **Software Product Lines Implementation**

State of the art: two families of approaches:

➔ **Annotative** approaches

➔ **Compositional** approaches

Annotative approaches



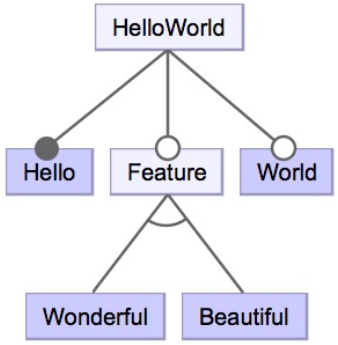
- **Asset Implementation:**
  - A **maximal** product (150% product).
  - **Annotations** to specify the fragments related to each feature.

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

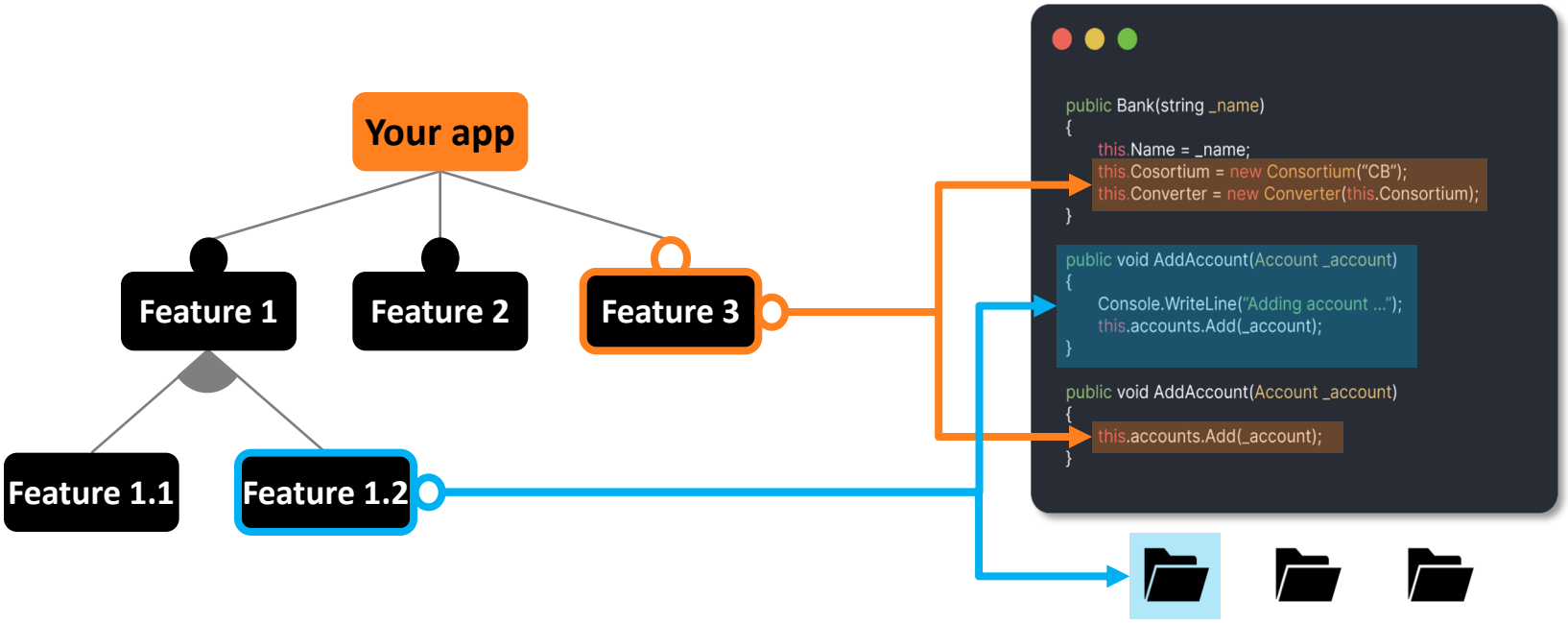
- Preprocessors for Java (integrated in the FeatureIDE tool)
  - **Munge** preprocessor: developed by the developers of Java's Swing library (main author:Thomas Ball)

```
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

- **Mobioos Forge**



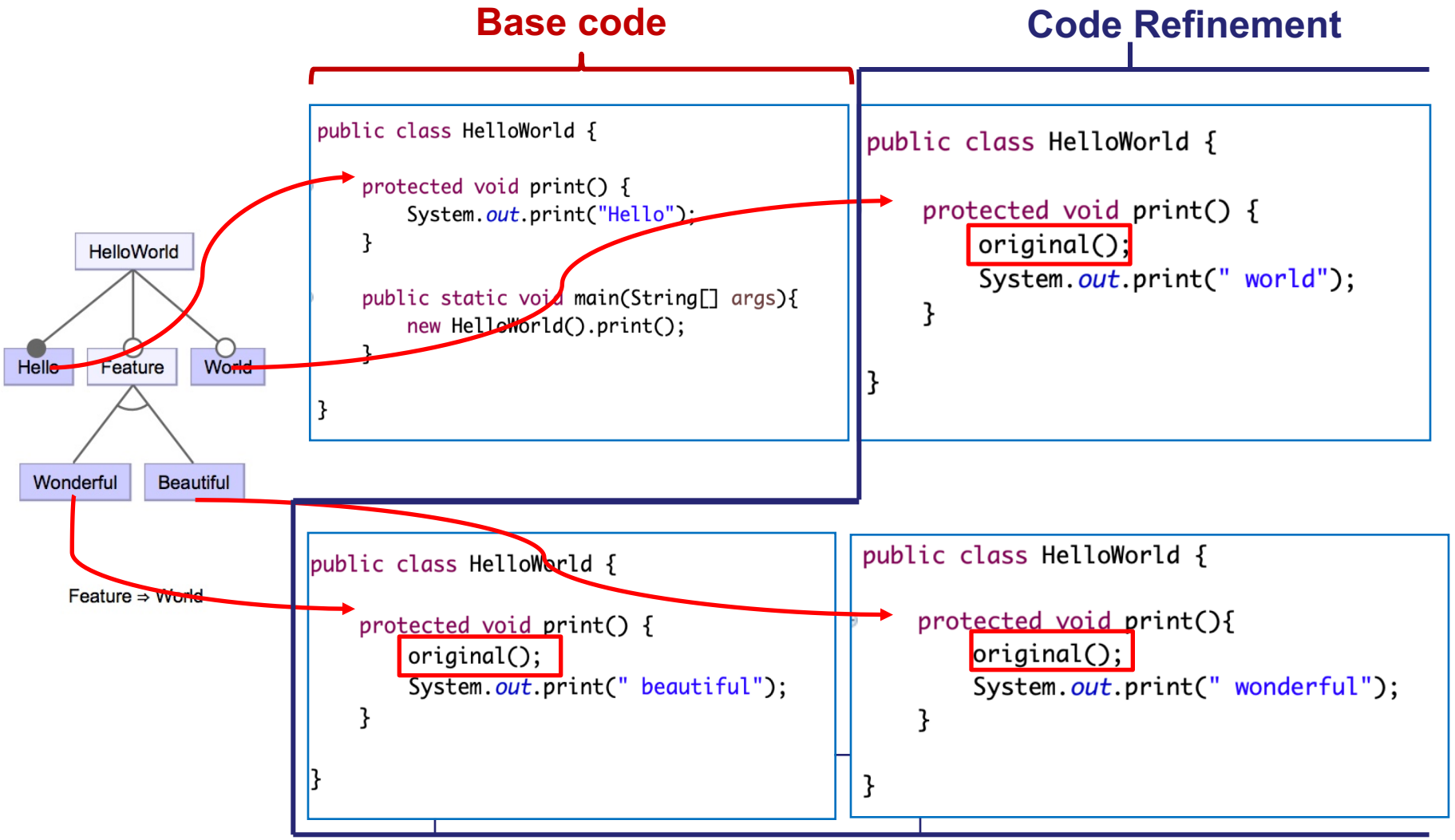
Compositional approaches

- 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

- **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: Java, C/C++, C#. etc

FeatureHouse: Illustration

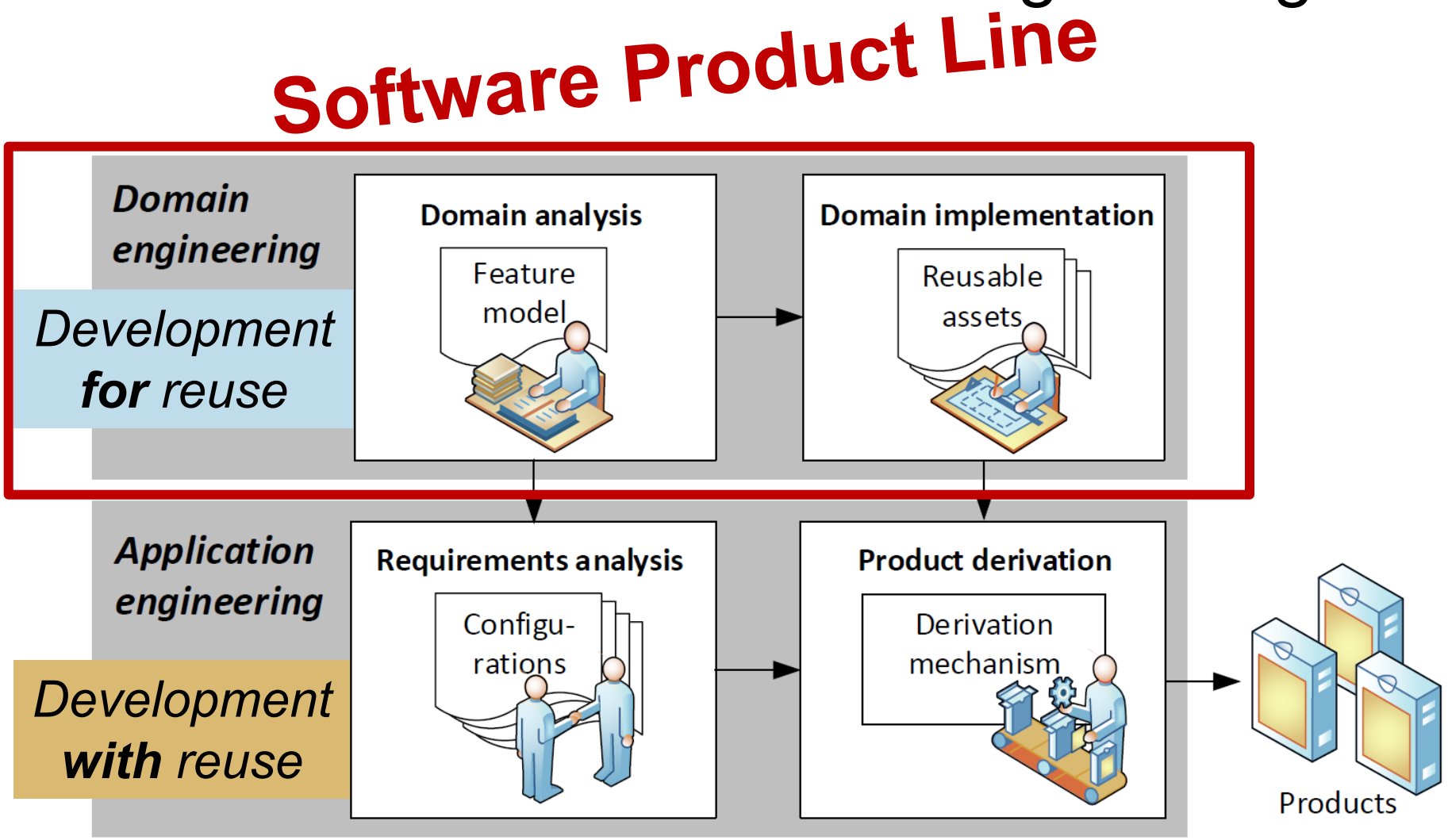


Comp. approaches: State of the art

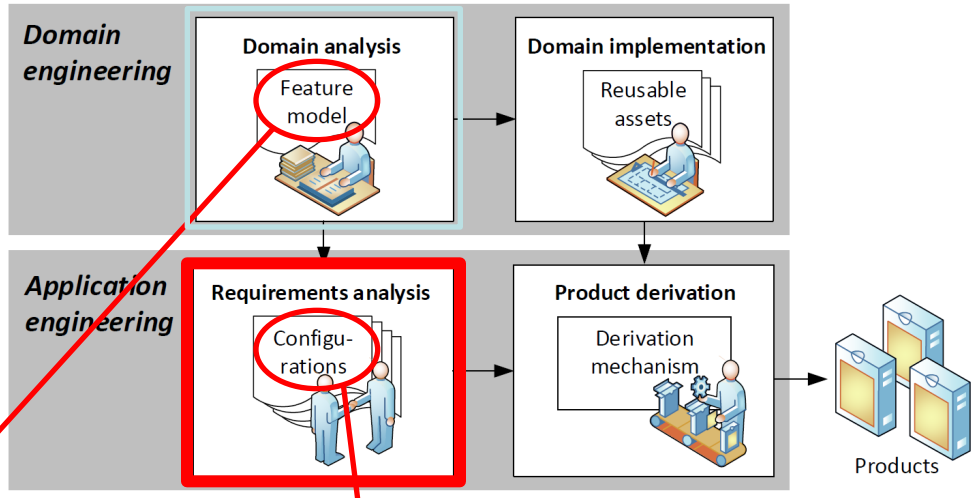
- **Delta-oriented Programming.** Schaefer, et al. SPLC 2010

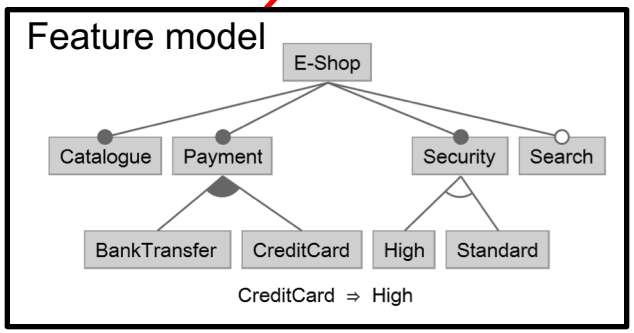
- **Aspect Oriented Programming** (ex. AspectJ, Kiczales et al. 07] .

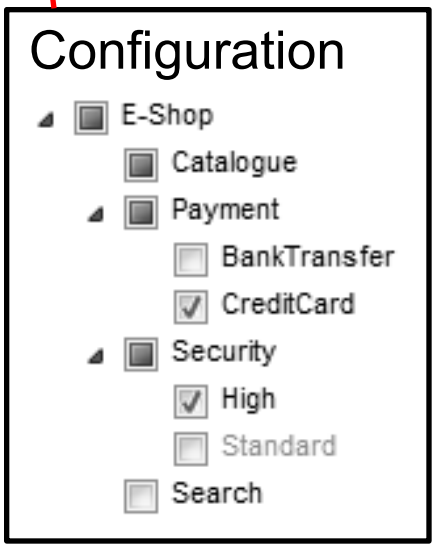
Software Product Line Engineering



Software Product Line Engineering



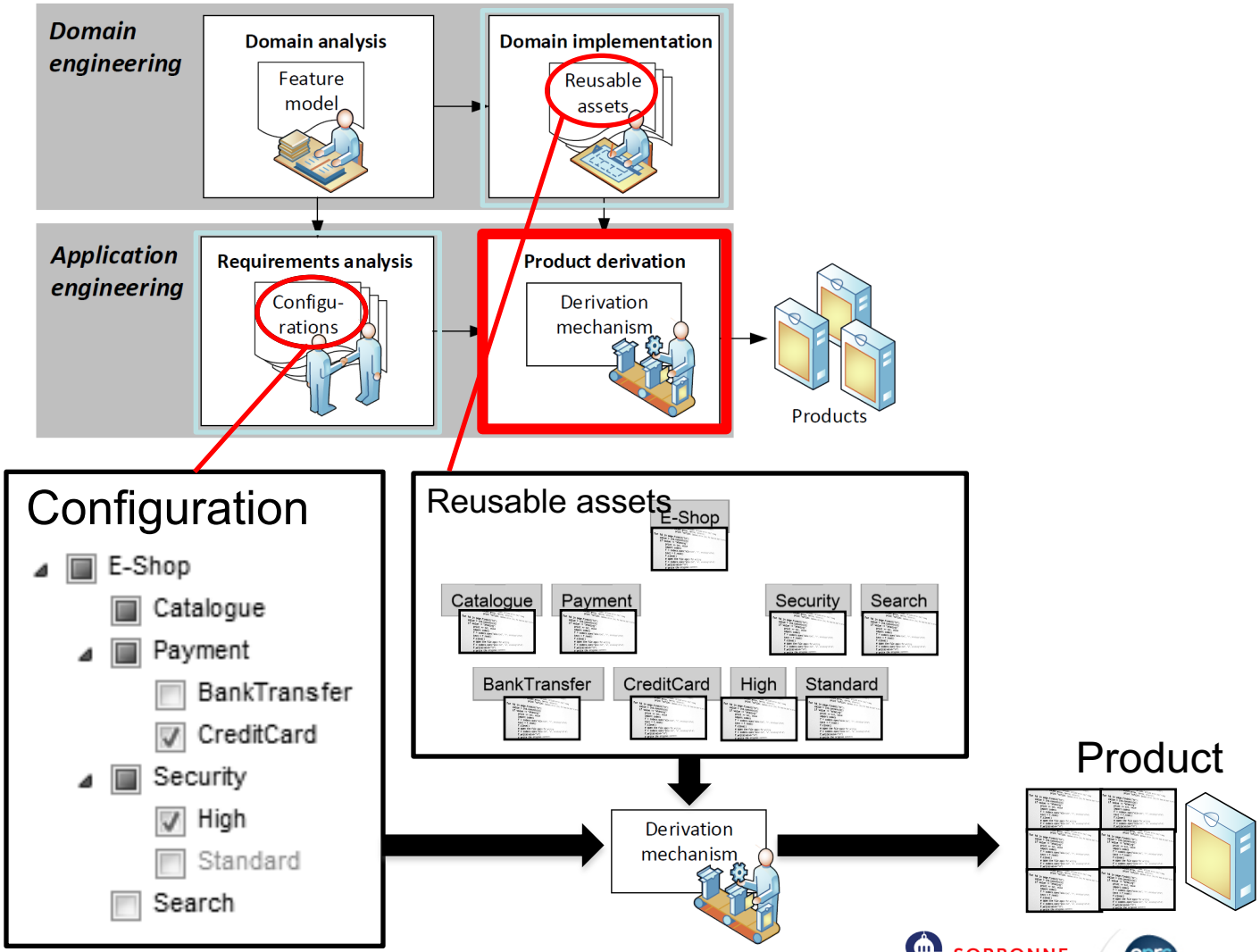




configuration =
set of features selected

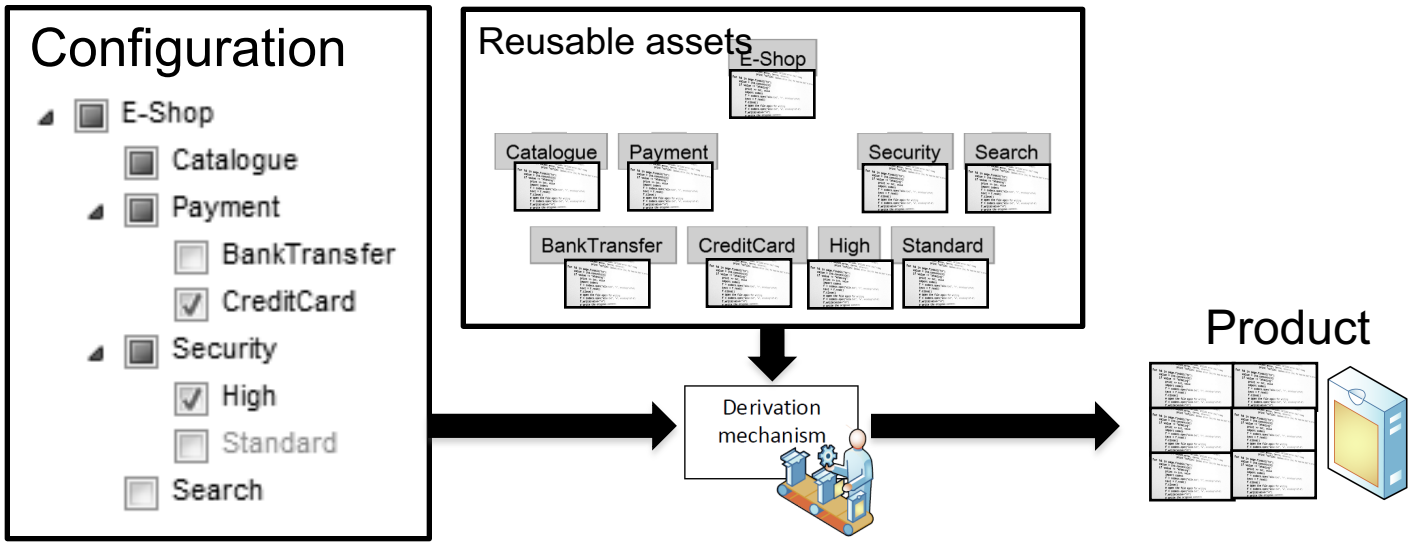
➔ **Only valid configurations**

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 ?

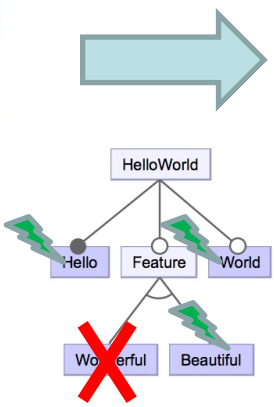
Research Direction: **Product Derivation**

  - 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**)?

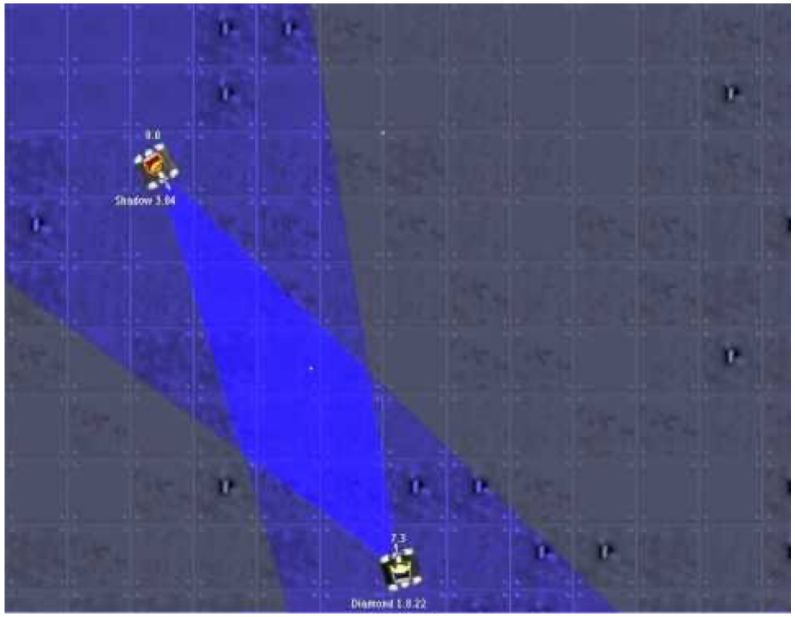
```
public   class  Account {


    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;

    }

}
```

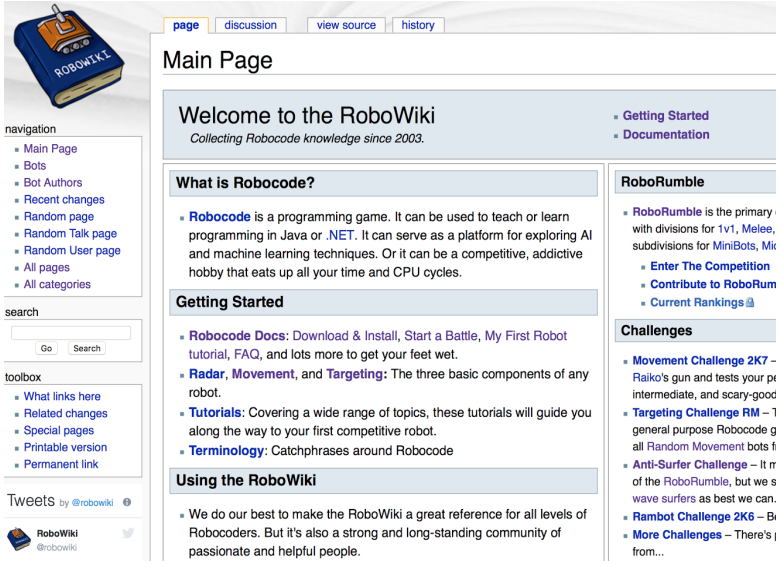
A real-world case study: Robocode

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

Robocode: the case study

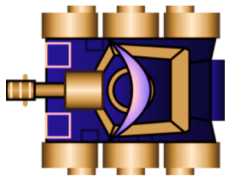
Robocode is a **programming game** where the goal is **to code** a **robot** to compete against other robots in a battle arena.





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	Fighting	Code License	Code Java	Code Size
Acero	http://robowiki.net/wiki/Acero	Anti-Pattern at distance	Head-On Targeting	melee and One-on-one	None	Yes	
Acraepheus	http://robowiki.net/wiki/Acraepheus	Adapting Stop and Go	Averaged Velocity, Heading Circular Targeting,	One-on-one	None	Yes	
AFlatNatural	http://robowiki.net/wiki/AFlatNatural	Oscillator Movement	Virtual Guns Array (26 simple targeters)	/	public domain	Yes	

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
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]
7	ChristmasCard	[illegible]	[illegible]	/		[illegible]	[illegible]
8	Chupacabra	http://robowiki.net/wiki/Chupacabra	Pathfinding	Pattern matching	One-on-one	/	No
9	Cigaret	http://robowiki.net/wiki/Cigaret	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

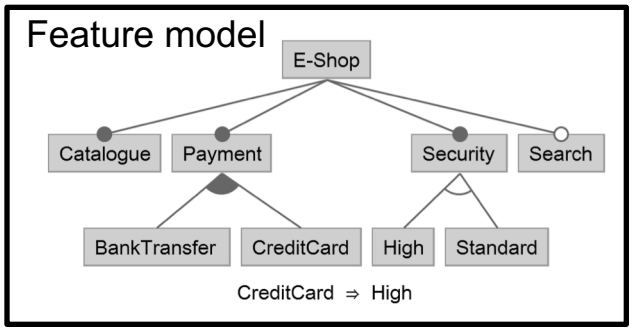
➔ **1500 bots made source code publicly available**

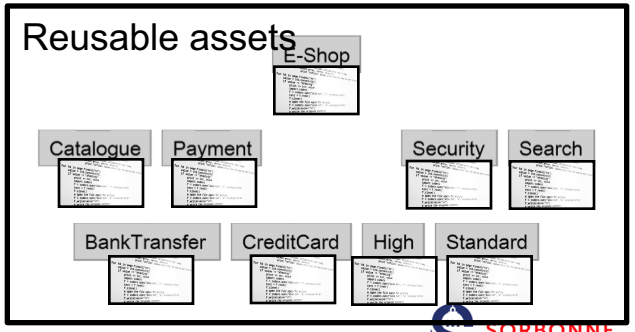
➔ **WikiRobot snippets**

➔ **All follow the same architecture**

Robocode SPL Implementation

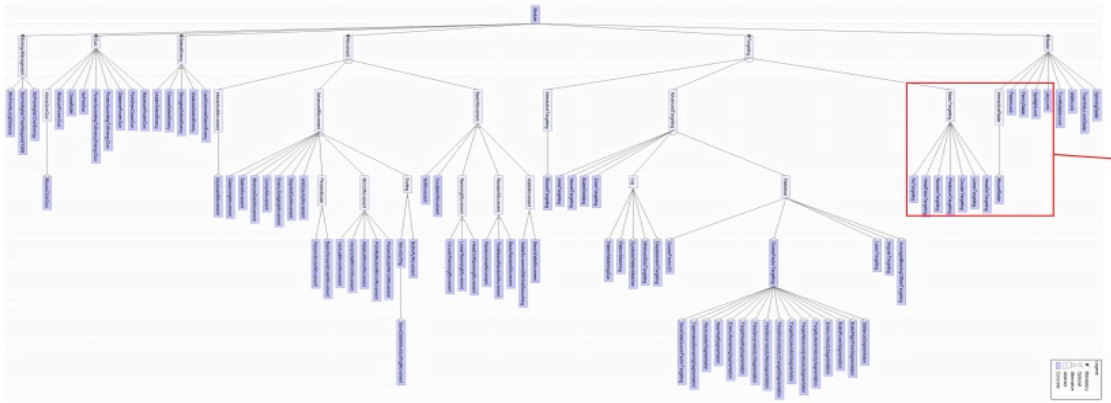
- **Inputs**:
  - Source code of bots
  - RoboWiki Snippet
  - Documentation
- **Output**: Robocode SPL

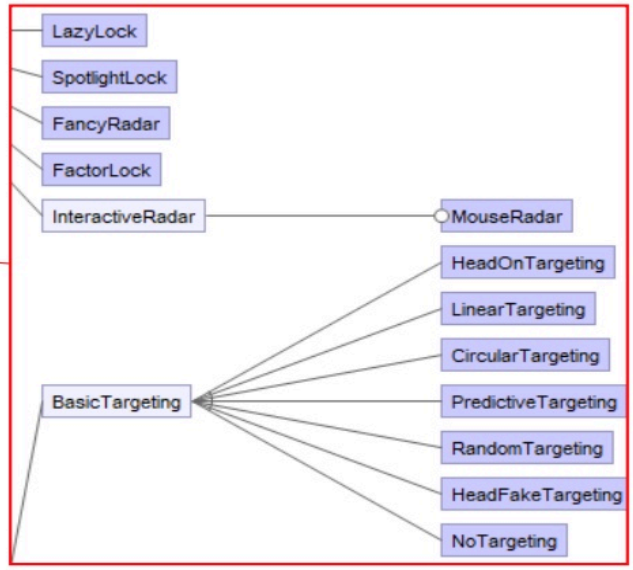




Domain Analysis

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

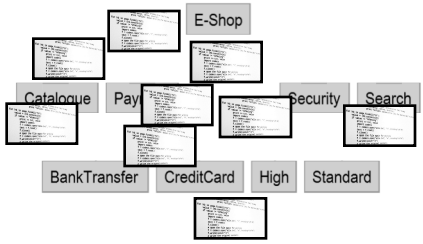




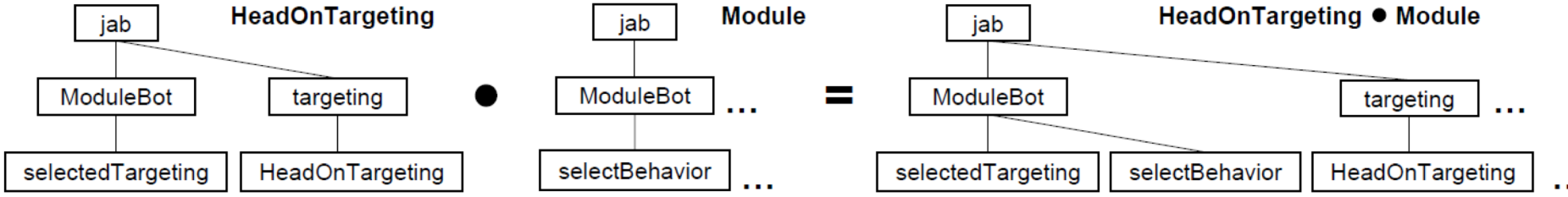
115 features:

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

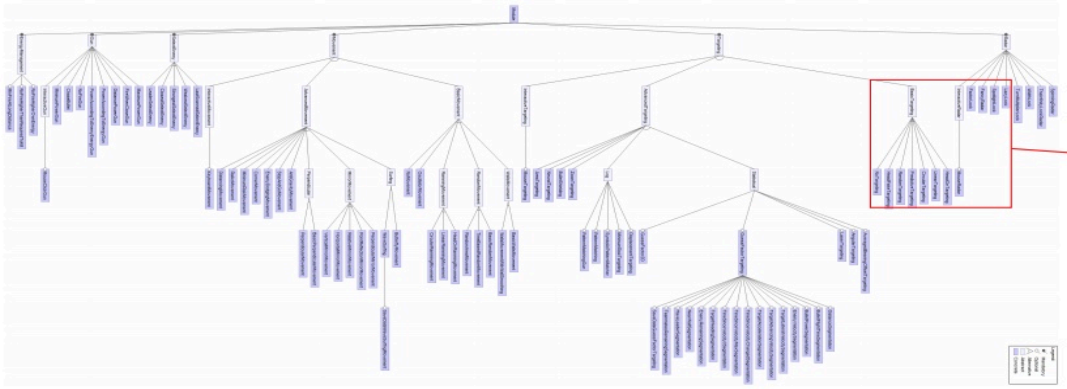
Domain Implementation

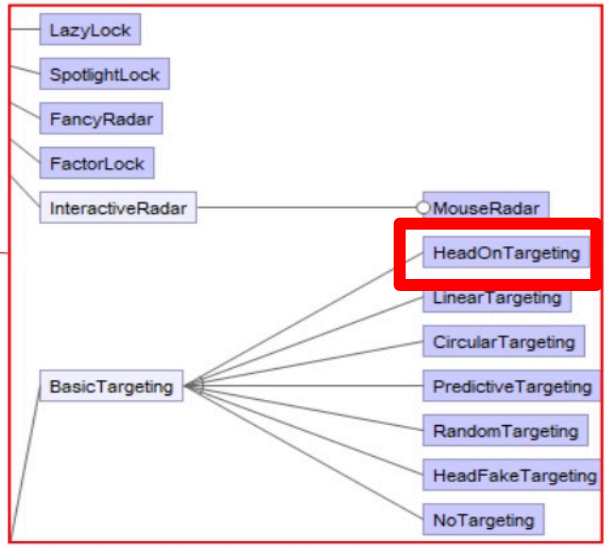


- **Objective**: Implement the reusable assets.
  - ➔ The use of the FeatureHouse framework [FH,13]
    - ➔A compositional approach at the source code level



[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)
    ;
}
```

“Refinement”

“Base”

```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

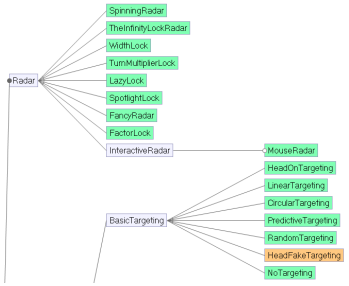
Domain Analysis

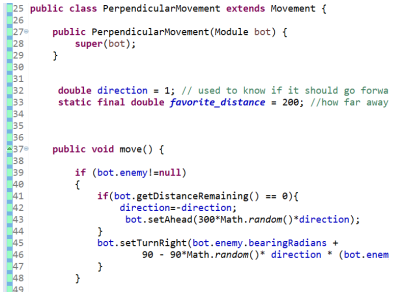
Domain Implementation

FeatureHouse

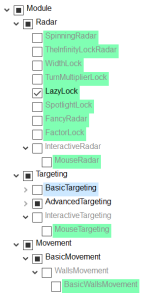
Configuration & Product Derivation

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
            }
        }
```



FeatureIDE: LIVE DEMO

redfabriq

- RedFabriQ is now a Start-up-Studio
  - From project creation to Start-up creation

- Mobioos Forge: A tool implemented at the R&D division at RedFabriQ

LIP6

SORBONNE UNIVERSITÉ

cnrs
dépasser les frontières

Mobioos Forge

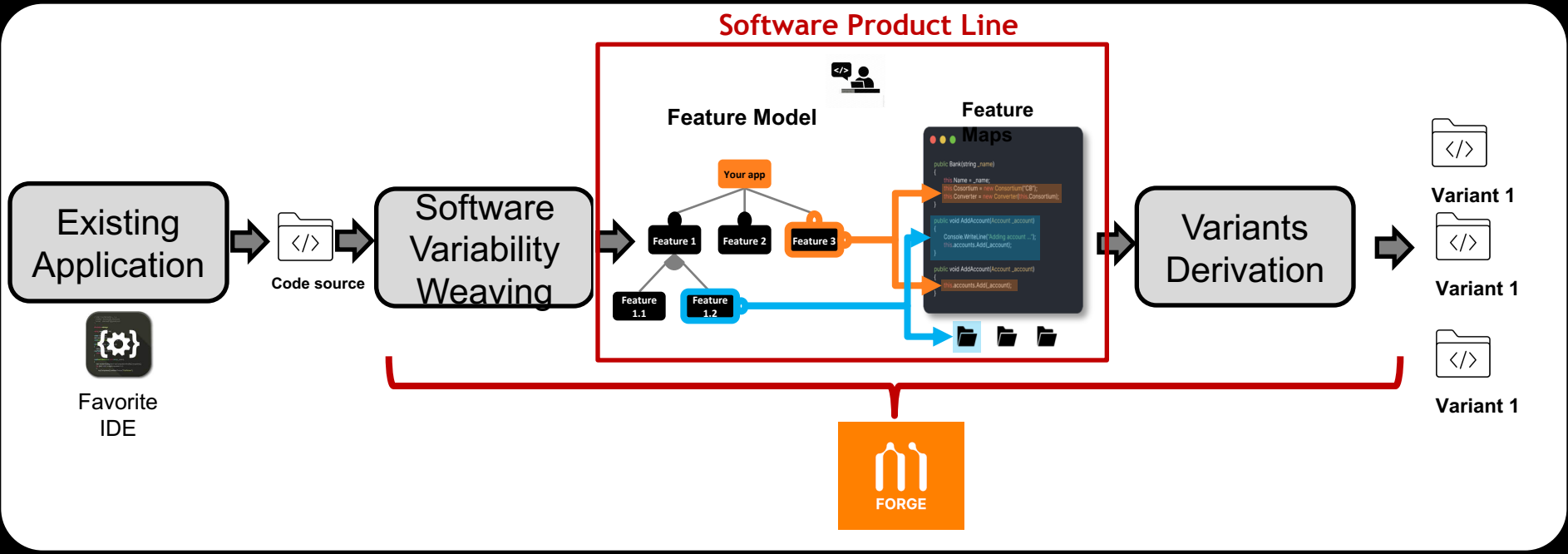
- From research ideas to a real industrial tool!
- Requirements & Objectives
  - Managing software variability in entreprise applications (web, mobile, etc.)
    - Multi-languages, multi-plateforms!
  - Do not radically change the ways that developers work.
  - Do not radically modify the source code of applications.

Mobioos Forge

..Coding first, and weaving variability after…

Implementing SPL with Mobioos Forge

https://forge.mobioos.ai/



Mobioos Forge