An Automated Approach to Manage MAS-Product Line Methods

EMAS at AAMAS 2017
São Paulo

Sara Casare
Tewfik Ziadi
Anarosa A. F. Brandão
Zahia Guessoum

Universidade de São Paulo
Escola Politécnica

UPMC PARIS UNIVERSITAS
LIP6

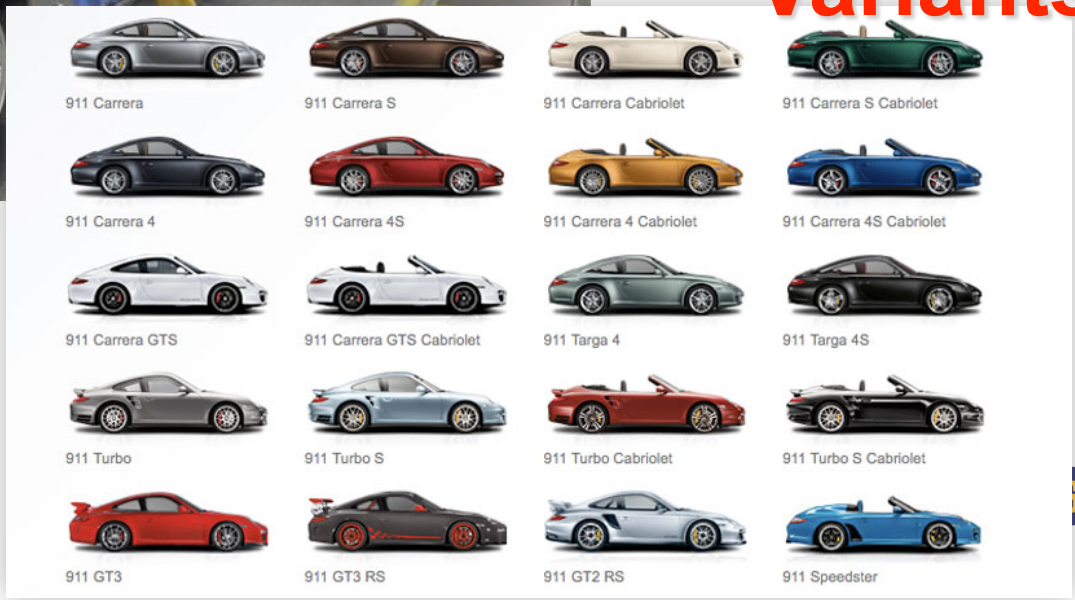
Software Variability in MAS

Intrinsic properties of MAS, such as modularity, make them variability-rich systems

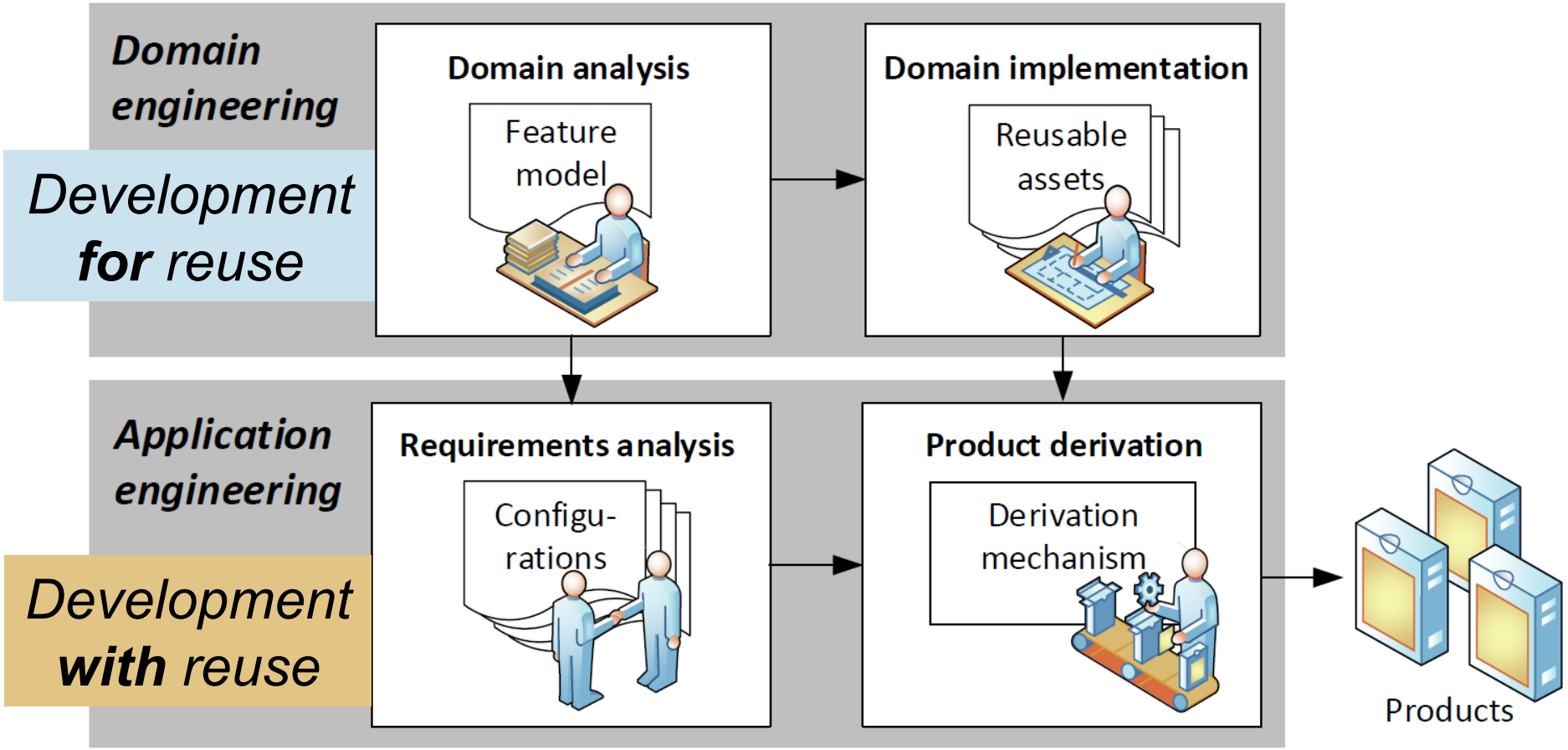
- Software variability
  - the ability of a software system to be changed, customized or configured for use in a particular context
- **Software Product Line Engineering** (SPLE)
  - a systematic approach for variability management proposed by the Software Engineering community

Software Product Line Engineening is inspired in Product Lines

Variants



Software Product Line Engineering

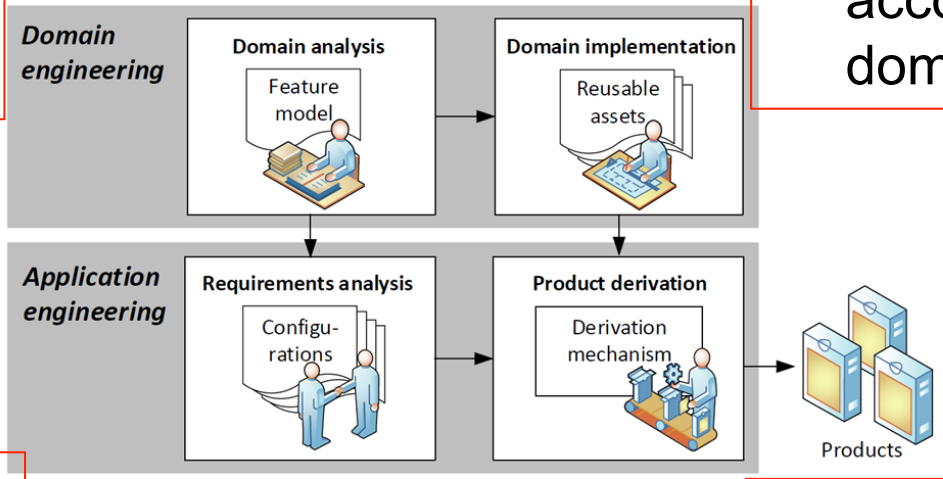


Software Product Line Engineering
Main Activities

- Define domain scope
- Specify variabilities among members of a same family

- Implement reusable artifacts
- Organize artifacts according to the domain variabilities

Development ***for*** *reuse*



Development ***with*** *reuse*

- Select the required variabilities for a particular product

- Generate (or derive) the final product

Software Product Lines for MAS Development

MAS- Product Lines

- ✓ Define a family of multiagent systems according to their **variabilities**
  - similarities and differences among MAS belonging to the same family
- ✓ Derive single MAS applying these variabilities according to project needs

Software Product Lines for MAS Development

Only few AOSE methods that support the development of MAS-PL

- How can we integrate the different SPLE activities with any AOSE method, concerning both MAS-PL domain and application engineering?
- How can we automatically generate a new MAS-PL based on any AOSE method and SPLE best practices according to project needs?

Our proposition

Meduse for MAS-Product Lines

- ✓ Automated approach to manage methods for developing MAS-Product Lines
- ✓ Provide steady methodological support for MAS-Product Line development following SPLE best practices
  - Capitalize knowledge on SPLE activities
- ✓ Extend AOSE methods to explicitly support MAS-PL development by integrating SPLE activities with them according to project needs

Meduse for MAS-PL

Meduse for MAS-PL uses SPLE principles in **two levels**

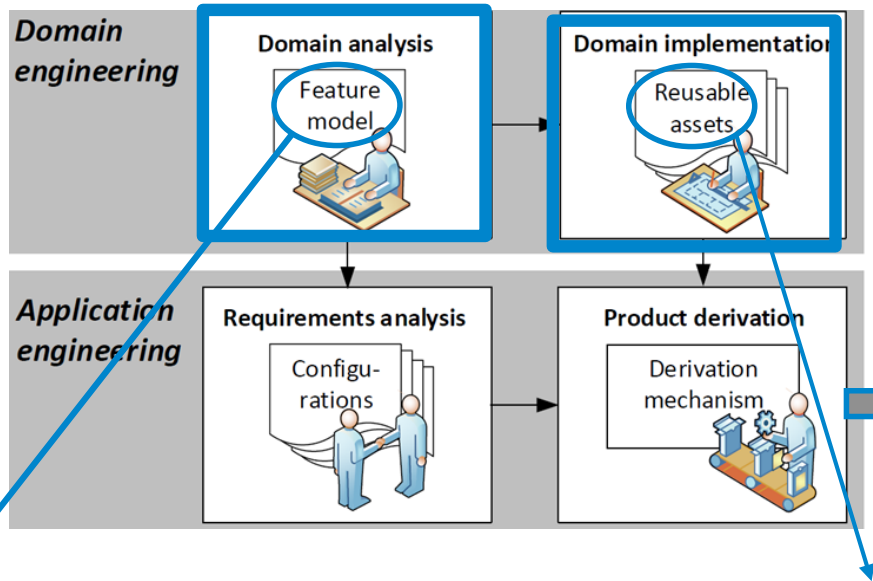
- ✓ SPLE principles are used in a meta-level fashion to **generate method variants to develop MAS-PL**

- ✓ These method variants are used to develop MAS-PL and then to **generate MAS variants**, i.e. product variants

Applying Software Product Line to Build Families of Methods for MAS-PL

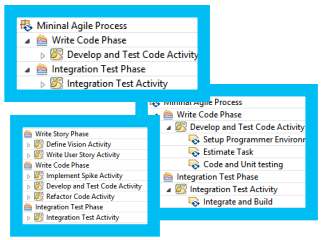
MAS-PL Method Domain Knowledge

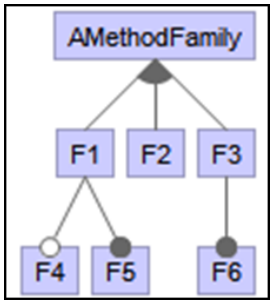
MAS-PL Project Needs



*Development **for** reuse*

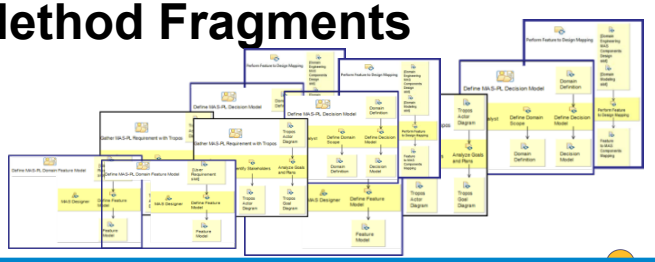
MAS-PL Methods





1. Specify MAS-PL methods variability using a Feature model

2. Build Artifacts: Reusable Method Fragments



Universidade de São Paulo

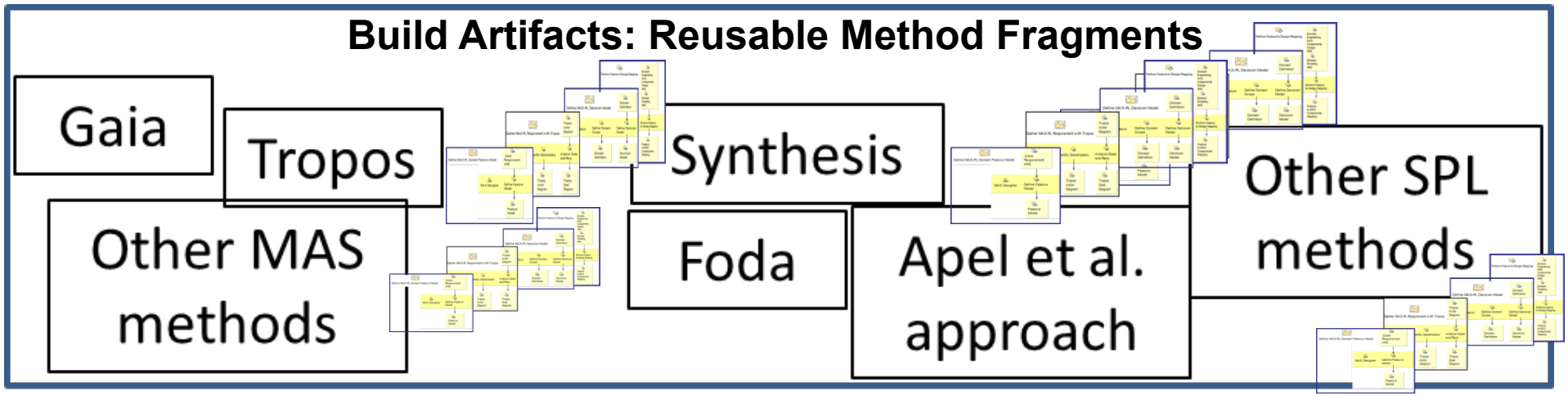
Escola Politécnica

PARIS UNIVERSITAS

Meduse for MAS-PL: Applying Software Product Line to Build Families of Methods

- Set of **method fragments for MAS-PL** that we can reuse

- Sourced from AOSE methods and SPLE approaches

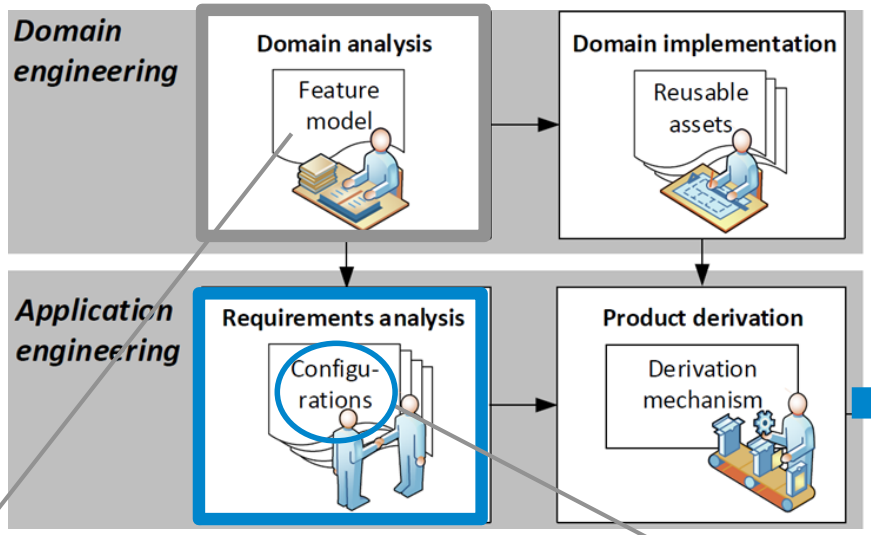


Meduse for MAS-PL: Applying Software Product Line to Build Families of Methods

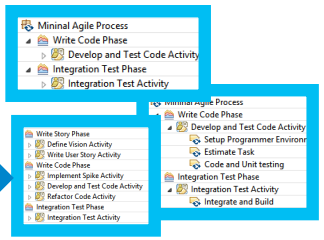
*Development **with** reuse*

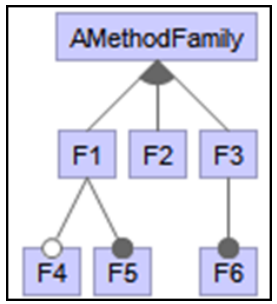
MAS-PL Method Domain Knowledge

MAS-PL Project Needs



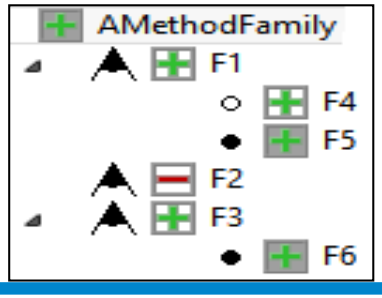
MAS-PL Methods





2. Specify MAS-PL methods variability using a Feature model

3. Select required MAS-PL method variabilities using a Method Configuration



Universidade de São Paulo

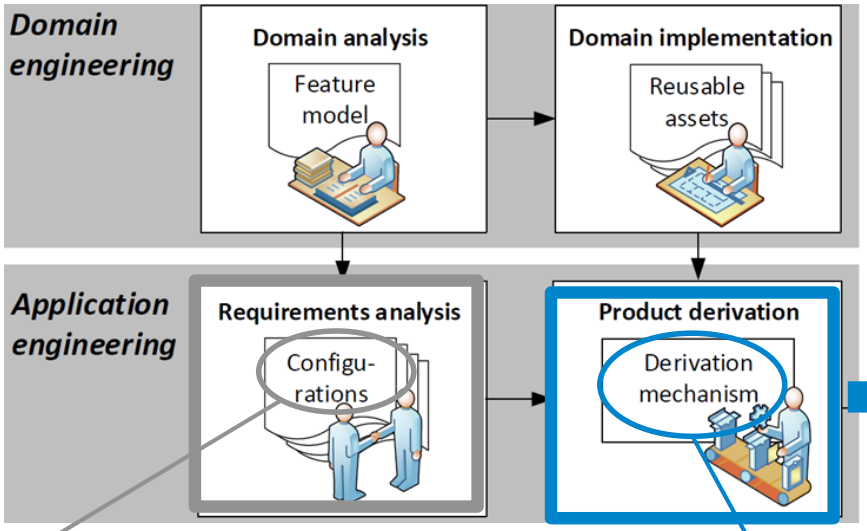
Escola Politécnica

Meduse: Applying Software Product Line to Build Families of Methods

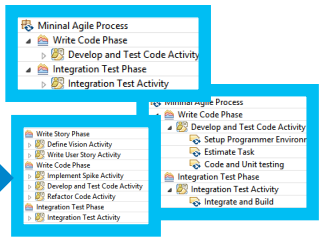
*Development **with** reuse*

MAS-PL Method Domain Knowledge

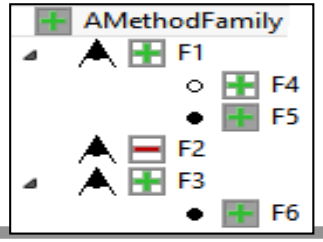
MAS-PL Project Needs



MAS-PL Methods



3.Select required method variabilities using a Method Configuration



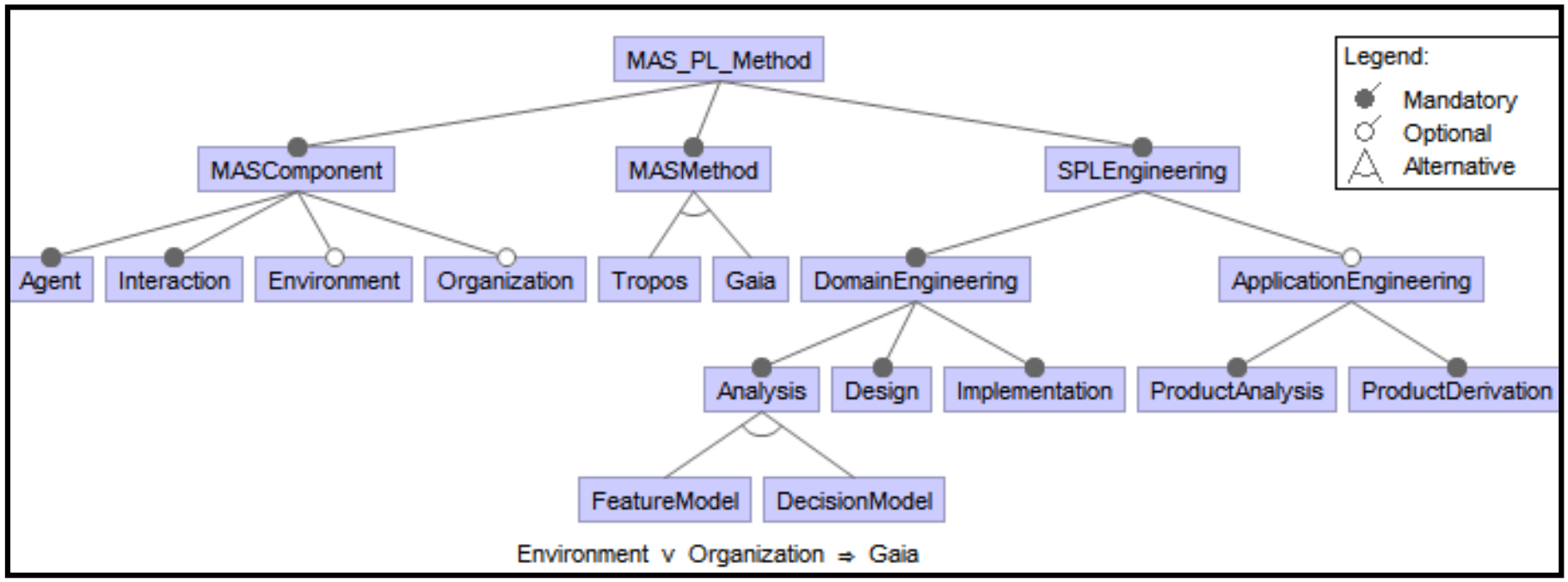
4. Generate a particular MAS-PL method using a Method Derivation tool

Meduse Composer

Case study: A family of MAS-PL methods using Gaia and Tropos

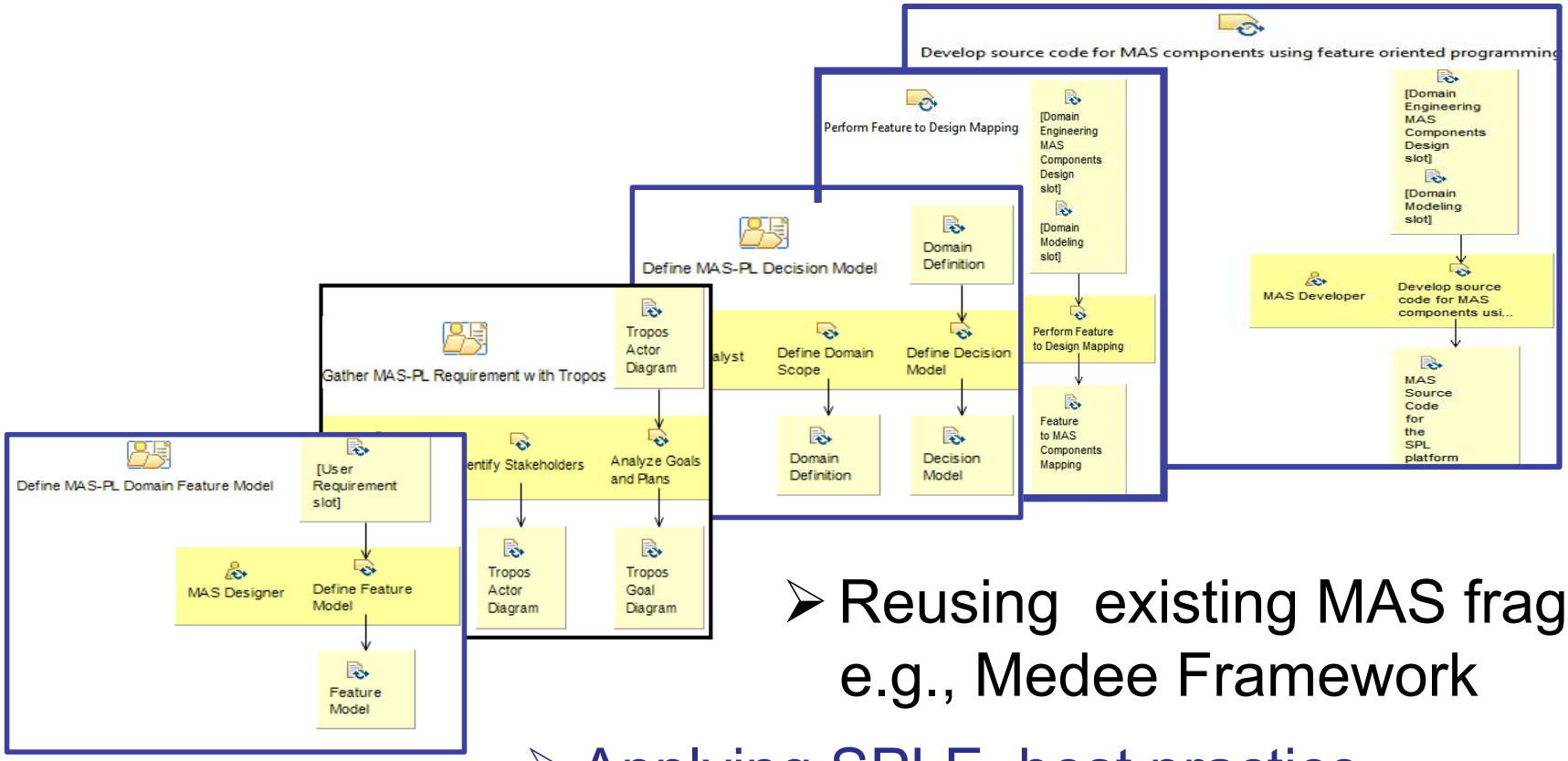
1) Specifying method variability using a Feature Model

18 features



Case study: A family of MAS-PL methods using Gaia and Tropos

2) Building Artifacts: Reusable Method Fragments for MAS-PL



- Reusing existing MAS fragments e.g., Medee Framework
- Applying SPLE best practice

Universidade de São Paulo

Escola Politécnica

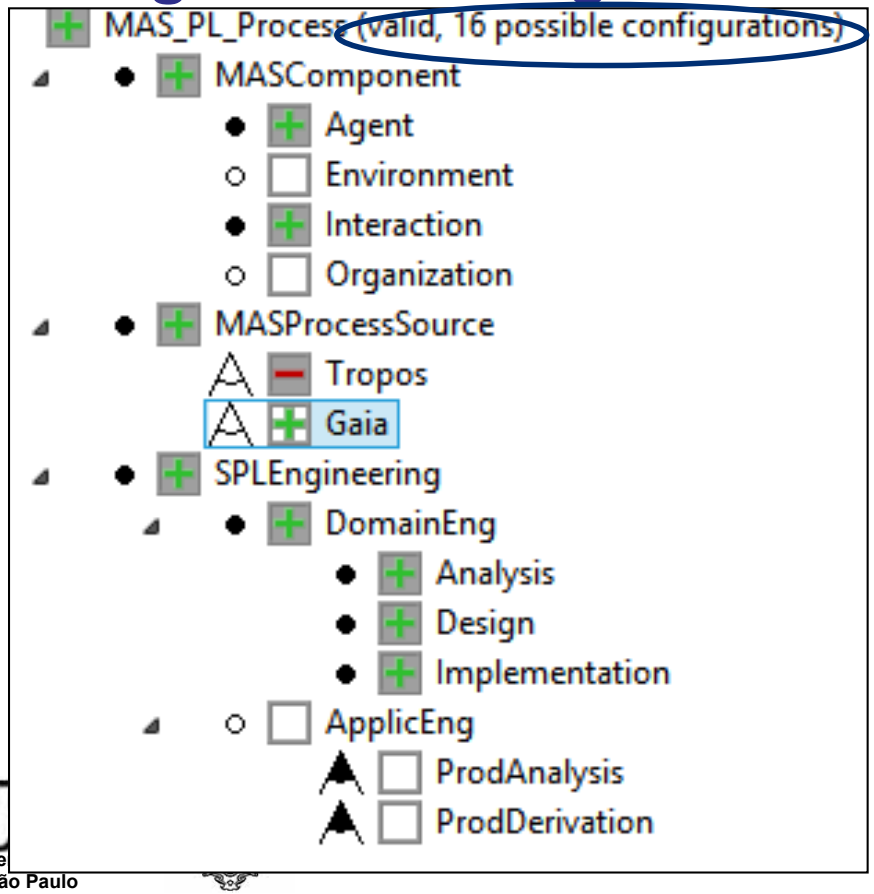
UPMC PARISUNIVERSITAS

LIP6

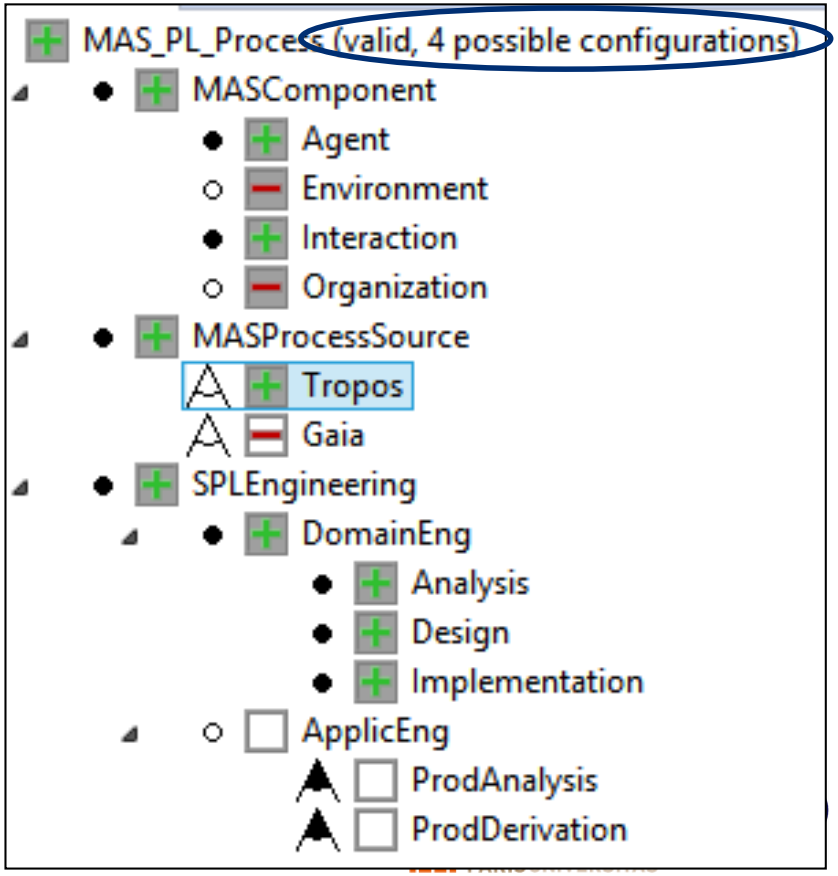
Case study: A family of MAS-PL methods

3) Selecting required variabilities using Method Configuration

16 possible method configuration using Gaia

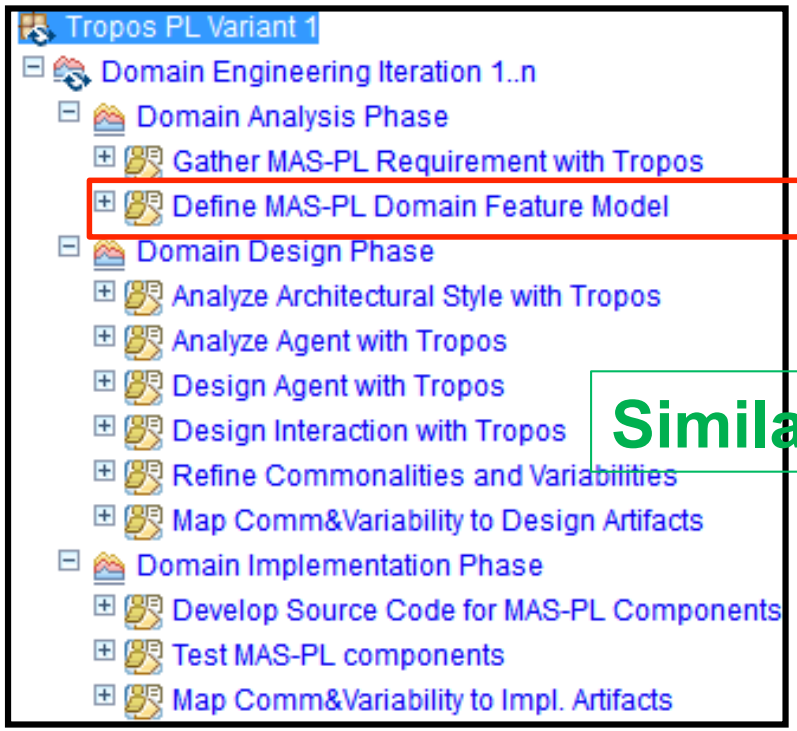


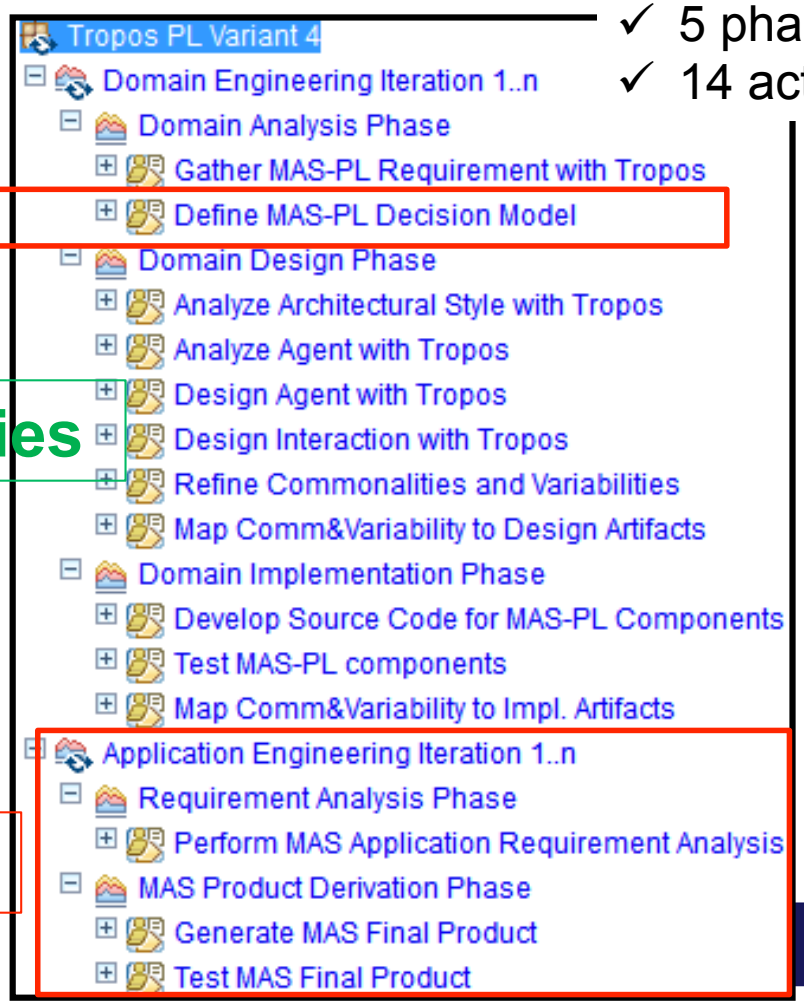
4 possible method configuration using Tropos



Case study: A family of MAS-PL methods

4) Generating variants of MAS-PL method based on Tropos





- ✓ 2 iterations
- ✓ 5 phases
- ✓ 14 activities

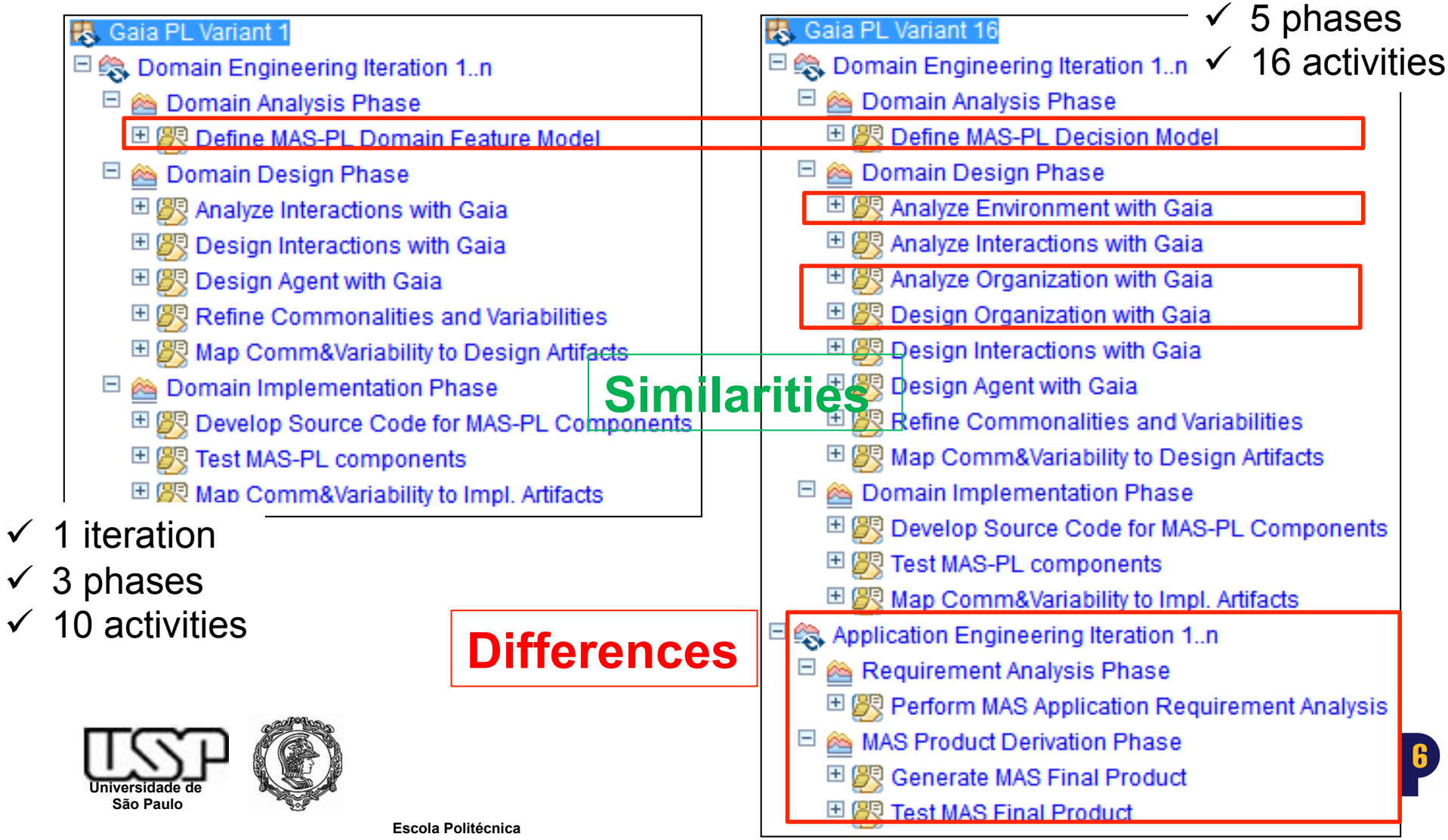
Similarities

- ✓ 1 iteration
- ✓ 3 phases
- ✓ 11 activities

Differences

Case study: A family of MAS-PL methods

4) Generating variants of MAS-PL method based on Gaia



Similarities

Differences

- ✓ 1 iteration
- ✓ 3 phases
- ✓ 10 activities

- ✓ 2 iterations
- ✓ 5 phases
- ✓ 16 activities

Conclusion and perspectives

- Meduse for MAS-PL provides steady methodological support for MAS-PL development by integrating AOSE methods & state-of-art SPLE development approaches

- Case Study
  - 20 MAS-PL method variants based on Tropos & Gaia
  - Some methods were used by Master students in the University Pierre et Marie Curie (Paris 6) to generate a family of multiagent teams to the Agent Contest.

- We plan to propose another MAS-PL case study and some criteria to measure the quality of generated method variants

Thanks for you attention

Meduse for MAS-PL is available at:

https://pages.lip6.fr/Tewfik.Ziadi/emas2017/

Acknowledgement

CNPq

National Council of Technological and Scientific Development

Universidade de São Paulo

Escola Politécnica

UPMC PARIS UNIVERSITAS

LIP6

BACK UP

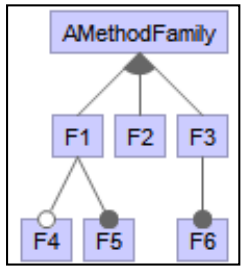
Building Family of Methods for MAS-Product Line with Meduse

1.Domain Analysis

2. Domain Implementation

MAS-PL Method Domain Knowledge

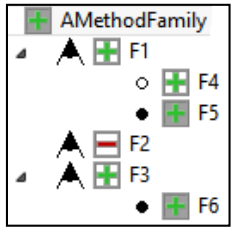
Feature Model



Features & Delta Modules Connection

List of Delta Modules, Conditions & MAS & SPL Fragments

MAS-PL Project Needs

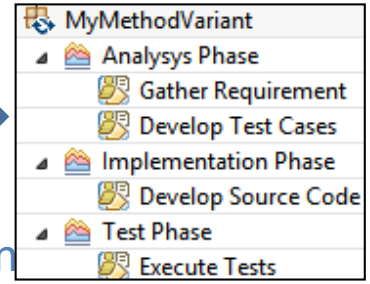


MAS-PL Method Configuration

Feature Selection

Meduse Composer

MAS-PL Method Variant

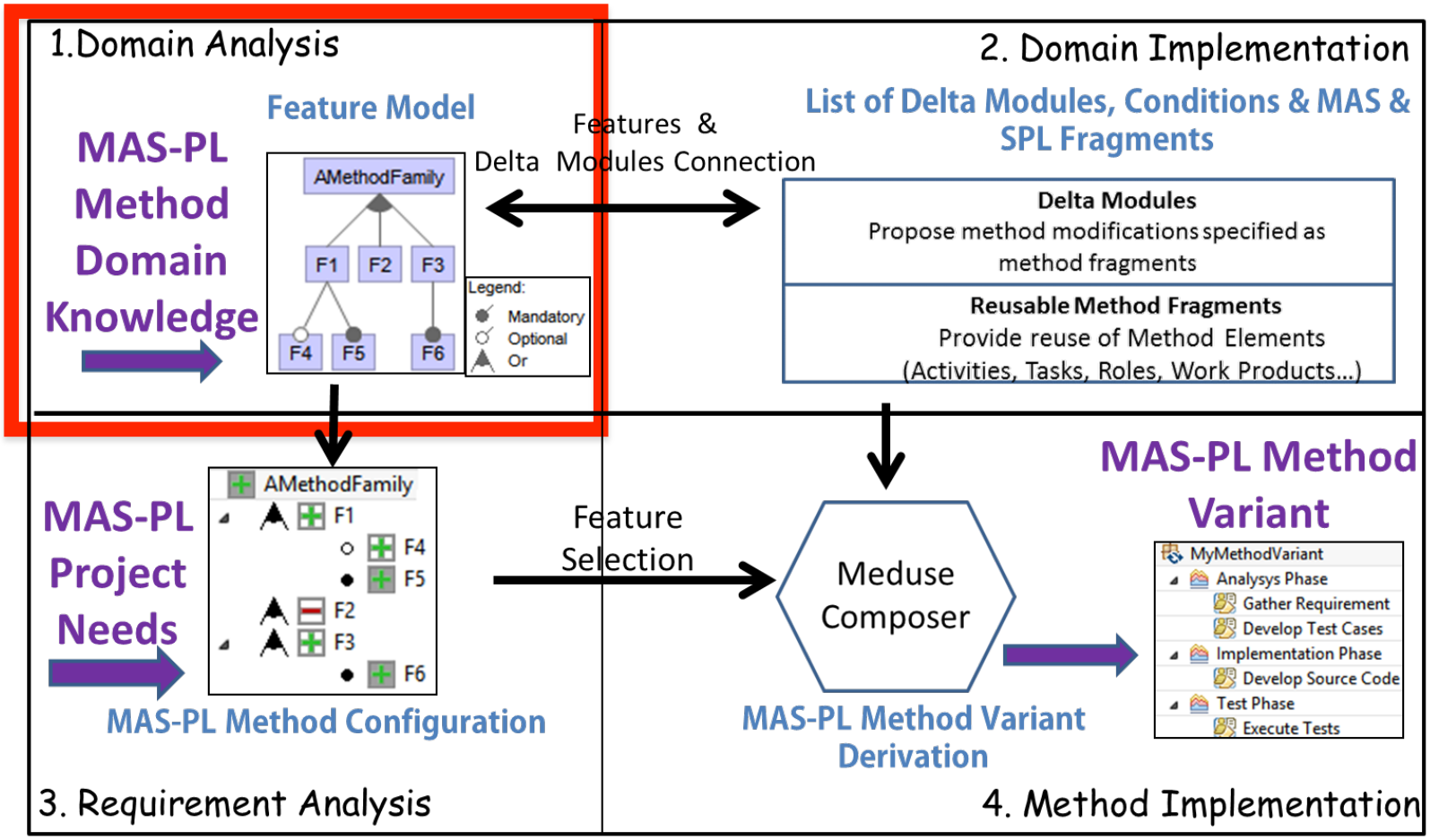


MAS-PL Method Variant Derivation

3. Requirement Analysis

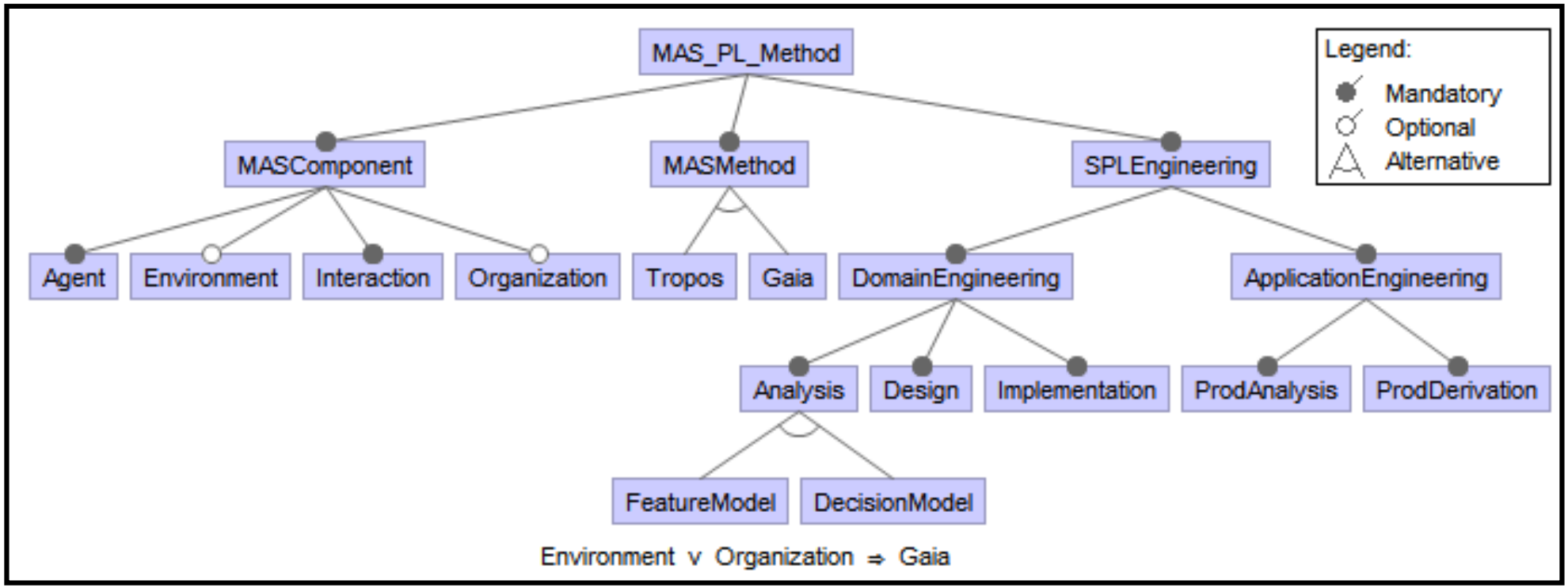
4. Method Implementation

Domain Analysis
Family of Methods for MAS-Product Line

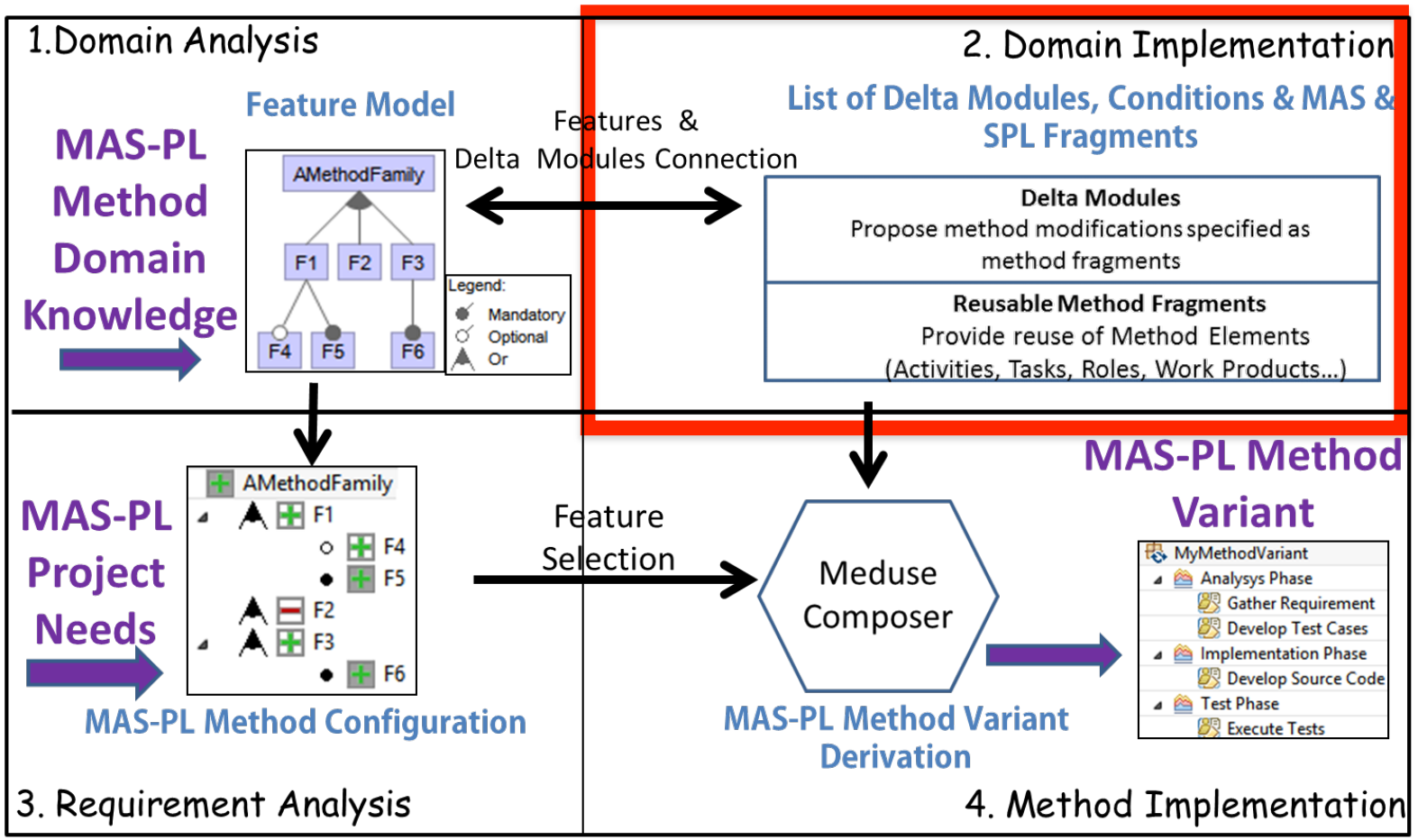


Domain Analysis
Family of Methods for MAS-Product Line

Commonalties and **variabilities** in a method family through the use of **feature model**

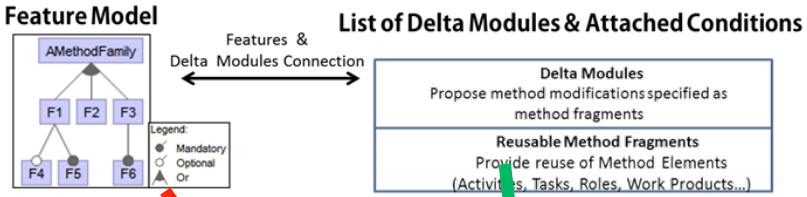


Domain Implementation
Family of Methods for MAS-Product Line



Domain Implementation for MAS-PL Methods

Associate Features to Method fragments through Delta Modules and Conditions



List of Delta modules & Attached conditions

[D_TroposFMAnalysis *when* Tropos and FeatureModel]

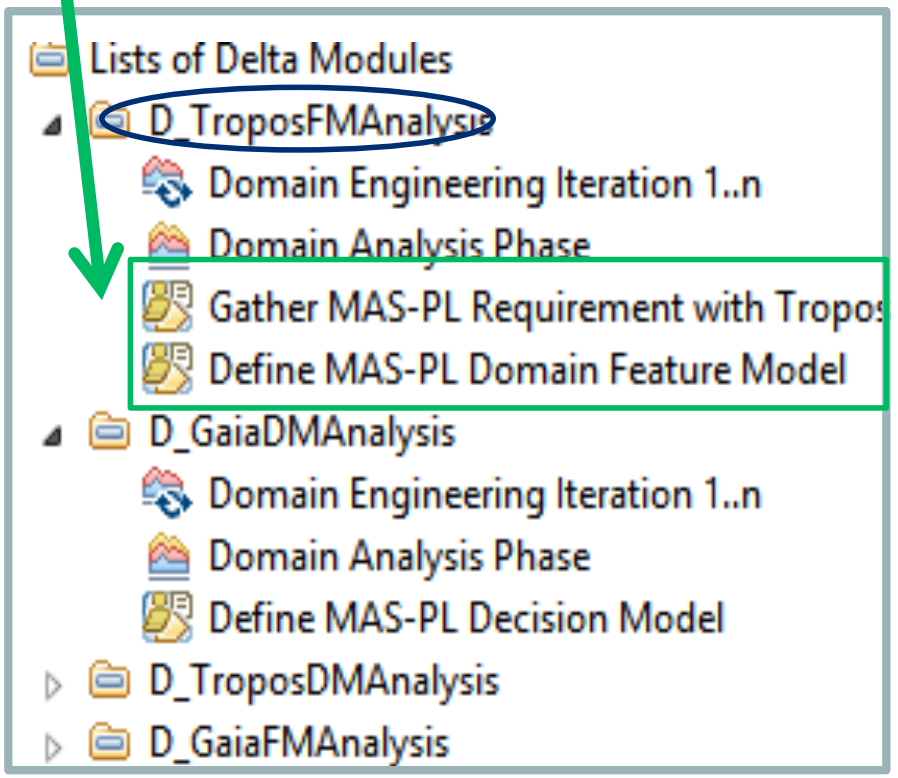
[D_GaiaDMAnalysis *when* Gaia and DecisionModel]

[D_TroposDMAnalysis *when* Tropos and DecisionModel]

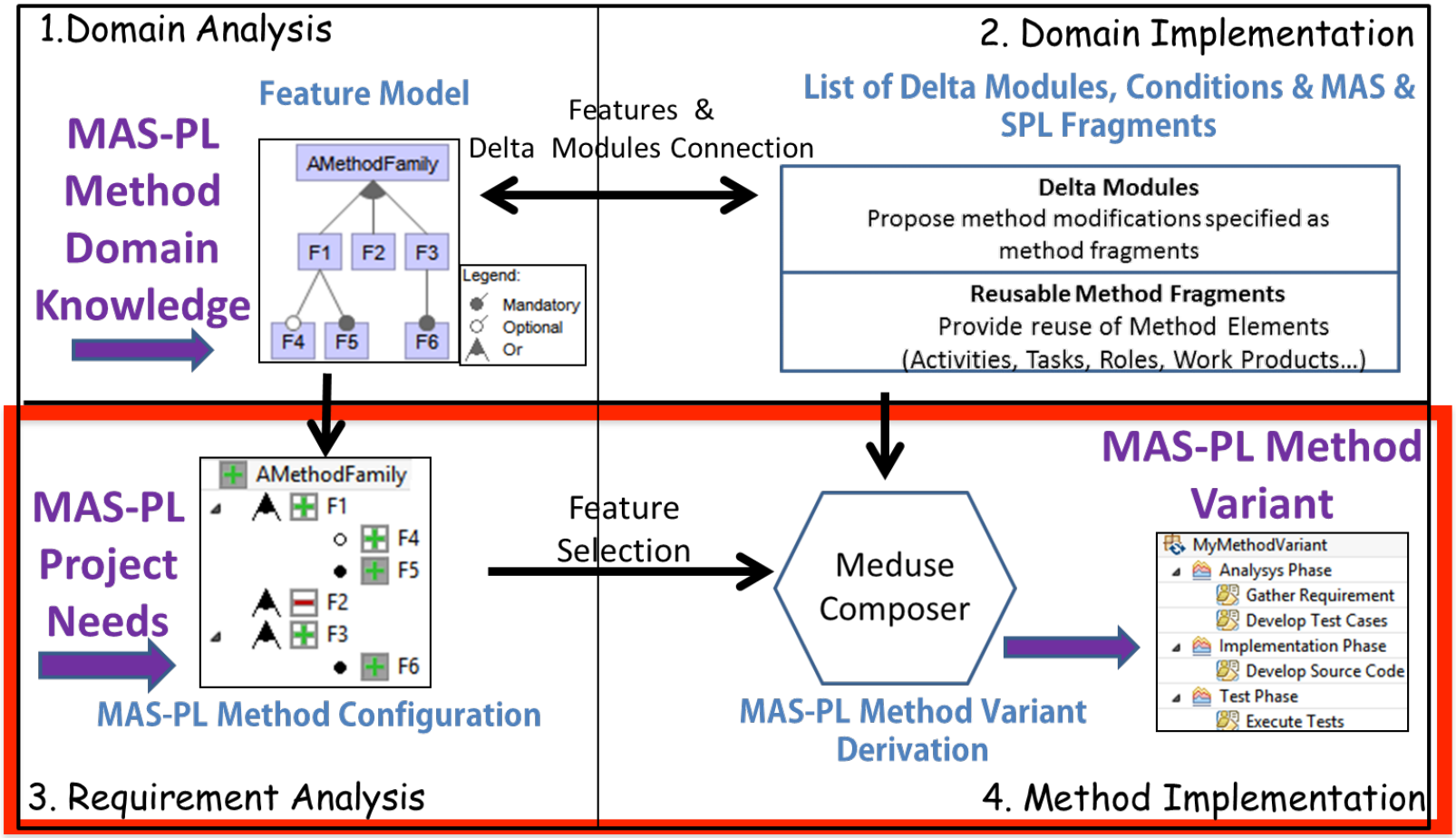
[D_GaiaFMAnalysis *when* Gaia and FeatureModel]

.....

[D_ApplicEngIteration when ApplicationEngineering]



Requirement Analysis & MAS-PL Method Variant Derivation



Case study: MAS-PL methods configuration and variants

MAS-PL Method Configuration																			MAS-PL Method Variants	
Configuration #	MAS-PL Features																		num. of Selected Features	
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18		
	MAS Component	Agent	Environment	Interaction	Organization	MAS Method	Gaia	Tropos	SPL Emgineering	Domain Engineer.	Analysis	Feature Model	Decision Model	Design	Implementation	Applic. Engineer.	Product Analysis	Product Derivation		
1	*	*	_	*	_	*	_	x	*	*	*	x	_	*	*				11	Tropos-PL variants: V1: Domain Eng. w/ FM & A.I.
2	*	*	_	*	_	*	_	x	*	*	*	_	x	*	*				11	V2: Domain Eng. w/ DM & A.I.
3	*	*	_	*	_	*	_	x	*	*	*	x	_	*	*	x	x	x	14	V3: Domain&Application Eng. w/ FM & A.I.
4	*	*	_	*	_	*	_	x	*	*	*	_	x	*	*	x	x	x	14	V4: Domain&Application Eng. w/ DM & A.I.
5	*	*		*		*	x	_	*	*	*	x	_	*	*				11	Gaia-PL Variants: V1: Domain Eng. w/ FM & A.I.
6	*	*		*		*	x	_	*	*	*	_	x	*	*				11	V2: Domain Eng. w/ DM & A.I.
7	*	*	x	*		*	x	_	*	*	*	x	_	*	*				12	V3: Domain Eng. w/ FM & A.E.I.
.....																				
17	*	*		*	x	*	x	_	*	*	*	x	_	*	*	x	x	x	15	V13: Domain&Application Eng. w/ FM & A.I.O.
18	*	*		*	x	*	x	_	*	*	*	_	x	*	*	x	x	x	15	V14: Domain&Application Eng. w/ DM & A.I.O.
19	*	*	x	*	x	*	x	_	*	*	*	x	_	*	*	x	x	x	16	V15: Domain&Application Eng. w/ FM & A.I.E.O.
20	*	*	x	*	x	*	x	_	*	*	*	_	x	*	*	x	x	x	16	V16: Domain&Application Eng. w/ DM & A.I.E.O.

FM = Feature Model DM = Decision Model A = Agent I = Interaction E = Environment O = Organization

(*) mandatory feature (x) optional selected featured (-) optional disable feature ( ) optional unselected feature

Universidade de São Paulo

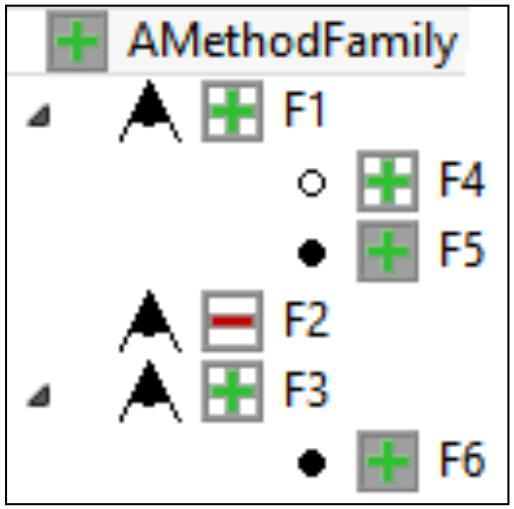
Escola Politécnica

UPMC PARIS UNIVERSITAS

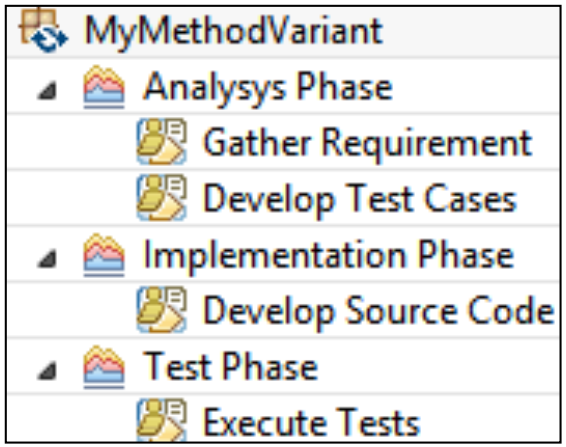
LIP6

Requirement Analysis & MAS-PL Method Variant Derivation

Step1: Create MAS-PL Method configuration



Step2: Compose MAS & SPL method fragments to derive specific variant



1. Find all delta modules that shall be applied to the method variant.
2. Method fragment composition using model transformations

Variability in Software Development Methods

- A software development method:

A structured set of steps required to develop a software system.

Variability in Software Processes

- A software process (SP):

A structured set of steps required to develop a software system.

  - **fine-grained elements**:
    - *tasks*, *work products*, and *roles*,
  - **coarse-grained:**
    - *activities*, *phases*, *iterations,*

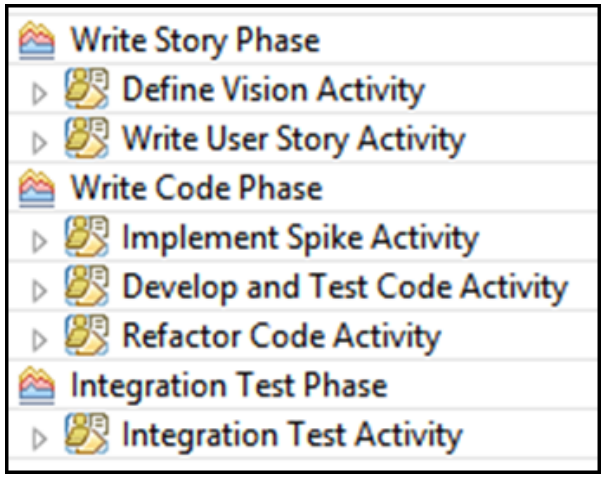
    process patterns, and *delivery process*

Variability in Software Processes

- A software process (SP):

A structured set of steps required to develop a software system.

  - **fine-grained elements**:
    - *tasks*, *work products*, and *roles*,
  - **coarse-grained:**
    - *activities*, *phases*, *iterations, process patterns,* and *delivery process*

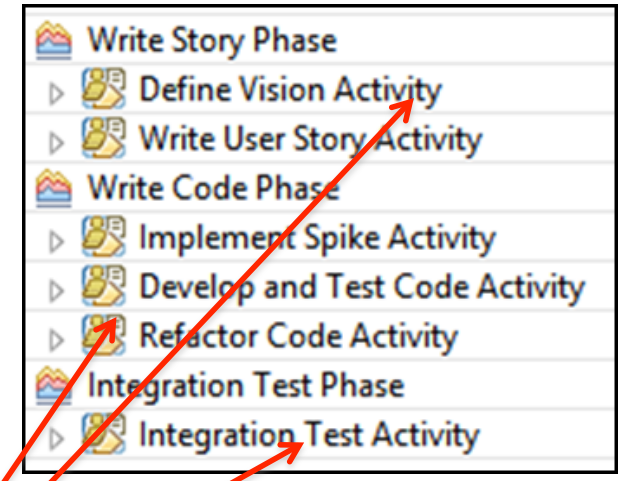


SPEM notations

Variability in Software Processes

- A software process (SP):
  - **fine-grained elements**:
    - *tasks*, *work products*, and *roles*,
  - **coarse-grained:**
    - *activities*, *phases*, *iterations,*

    process patterns, and *delivery process*



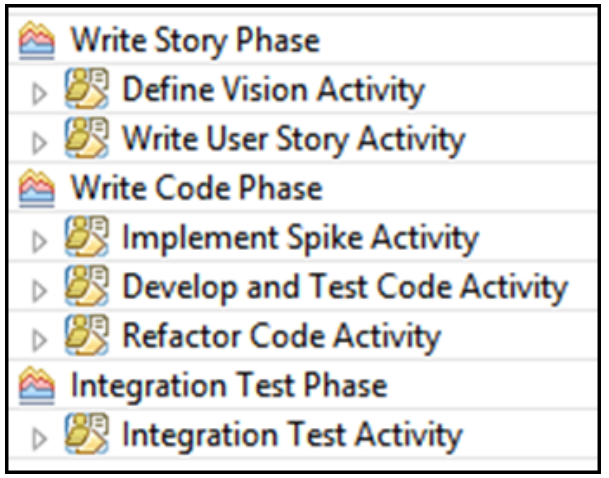
SPEM notations

Optional activities

Variability in Software Processes

- A software process (SPs):
  - **fine-grained elements**:
    - like *tasks*, *work products*, and *roles*,
  - **coarse-grained:**
    - *activities*, *phases*, *iterations,*

    process patterns, and *delivery process*



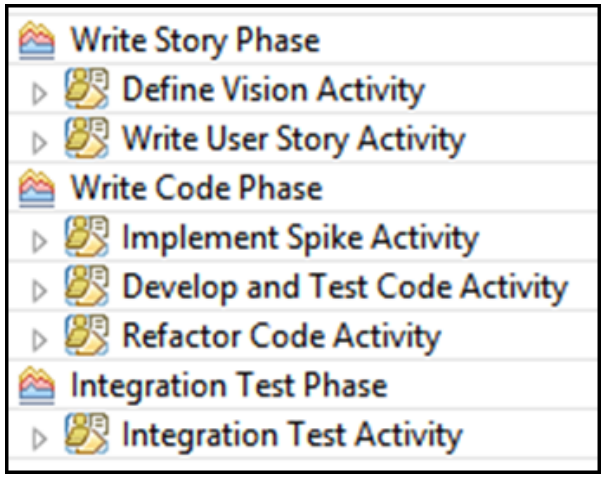
SPEM notations

- The need of **tailoring** software processes

Variability in Software Processes

- A software process (SPs):
  - **fine-grained elements**:
    - like *tasks*, *work products*, and *roles*,
  - **coarse-grained:**
    - *activities*, *phases*, *iterations,*

    process patterns, and *delivery process*



SPEM notations

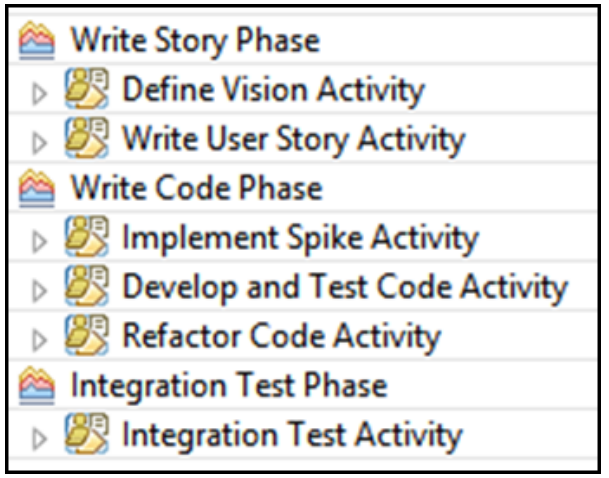
- The need of **tailoring** software processes
  - The need to manage **variability** in SPs.

RQ: **How can we tailor Software Processes**?

Variability in Software Processes

- A software process (SPs):
  - **fine-grained elements**:
    - like *tasks*, *work products*, and *roles*,
  - **coarse-grained:**
    - *activities*, *phases*, *iterations,*

    process patterns, and *delivery process*



- The need of **tailoring** software processes
  - The need to manage **variability** in SPs.

How can we tailor Software Processes?

Idea: The use of **Software Process Lines**