Context variability approaches in the today's systems

José Miguel Horcas

José A. Galindo

Introduction

Variability Intensive Systems (VIS)

Mobile systems

Cloud & Edge Computing

Operating systems

Artificial Intelligence

Automotive

QA & Cyber-security

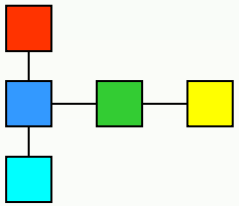
Smart homes

Smart city

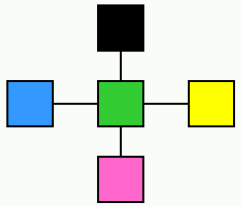
How to create a VIS?

Software Product Lines

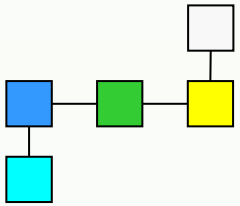
Traditional approach (mass production)



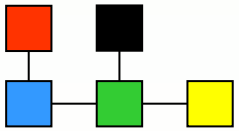
Product 1



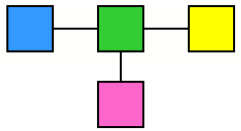
Product 2



Product 3



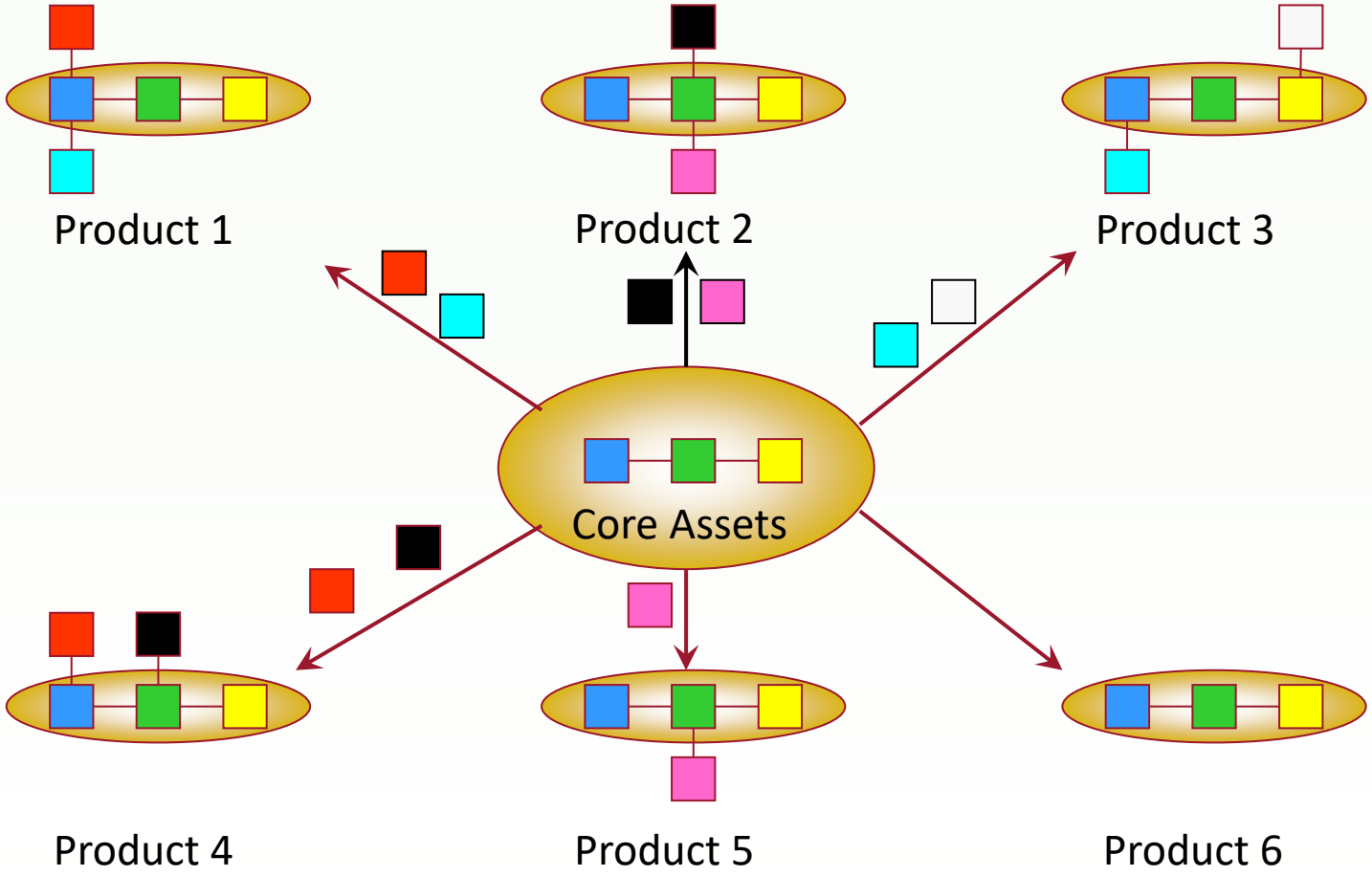
Product 4



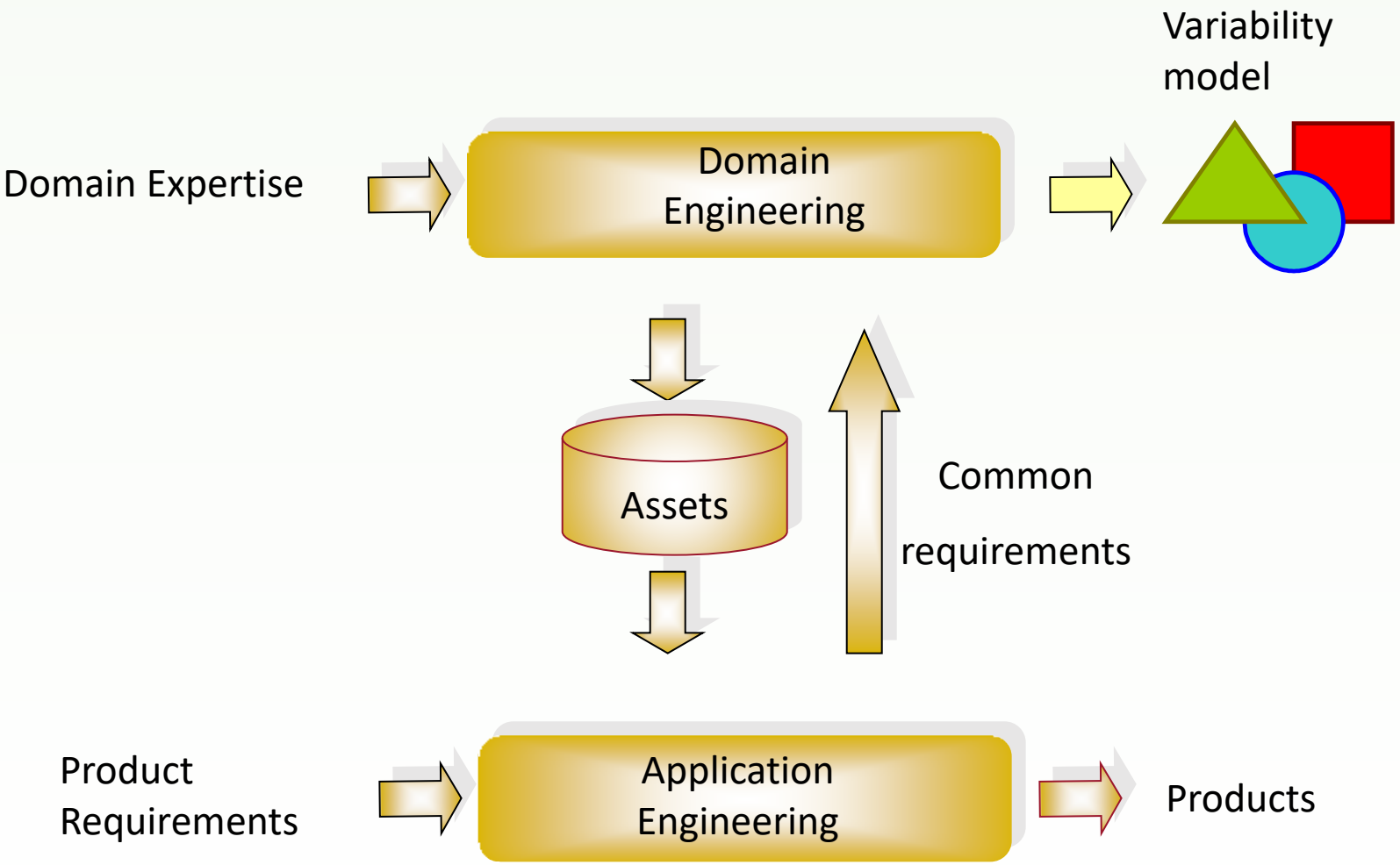
Product 5

Product 6

SPL approach (mass customization)

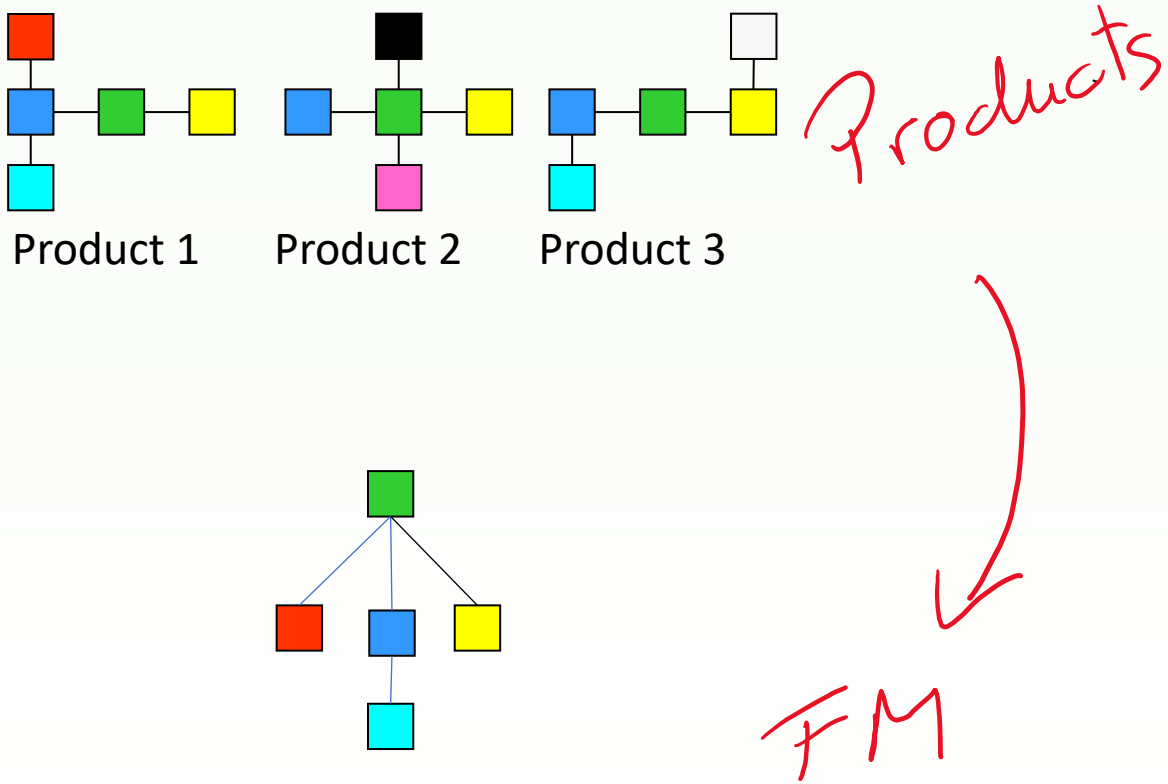


SPL approach

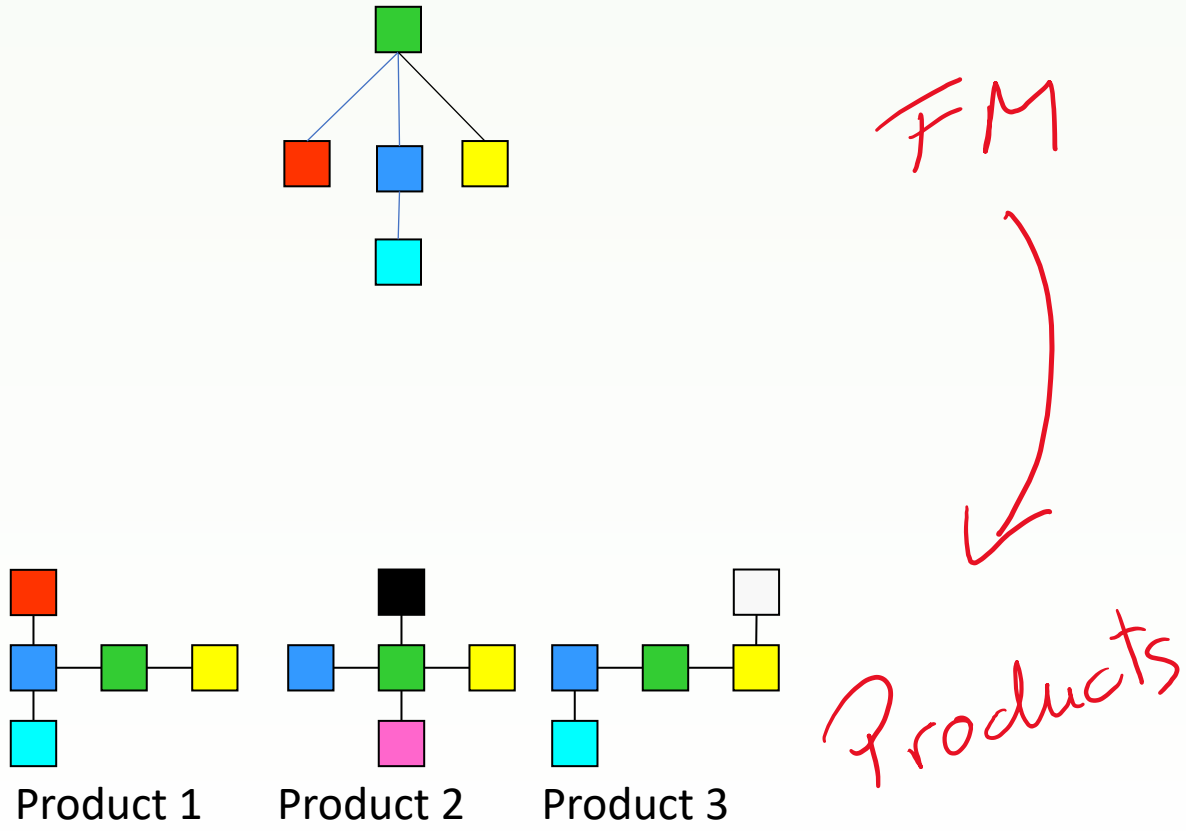


SPL transition approaches

Reactive approach

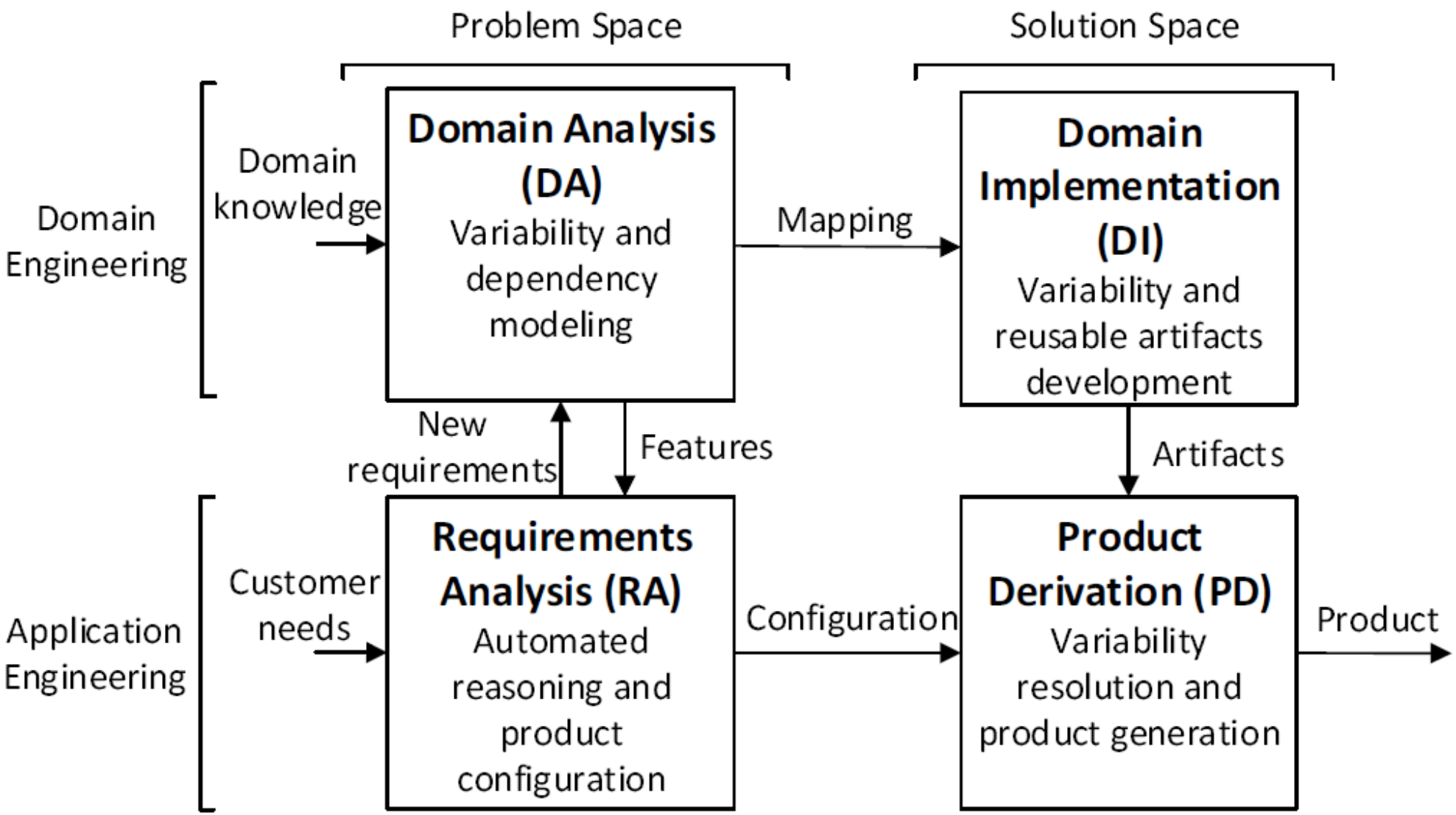


Proactive approach



Let’s learn proactive

SPL activities for proactive approach



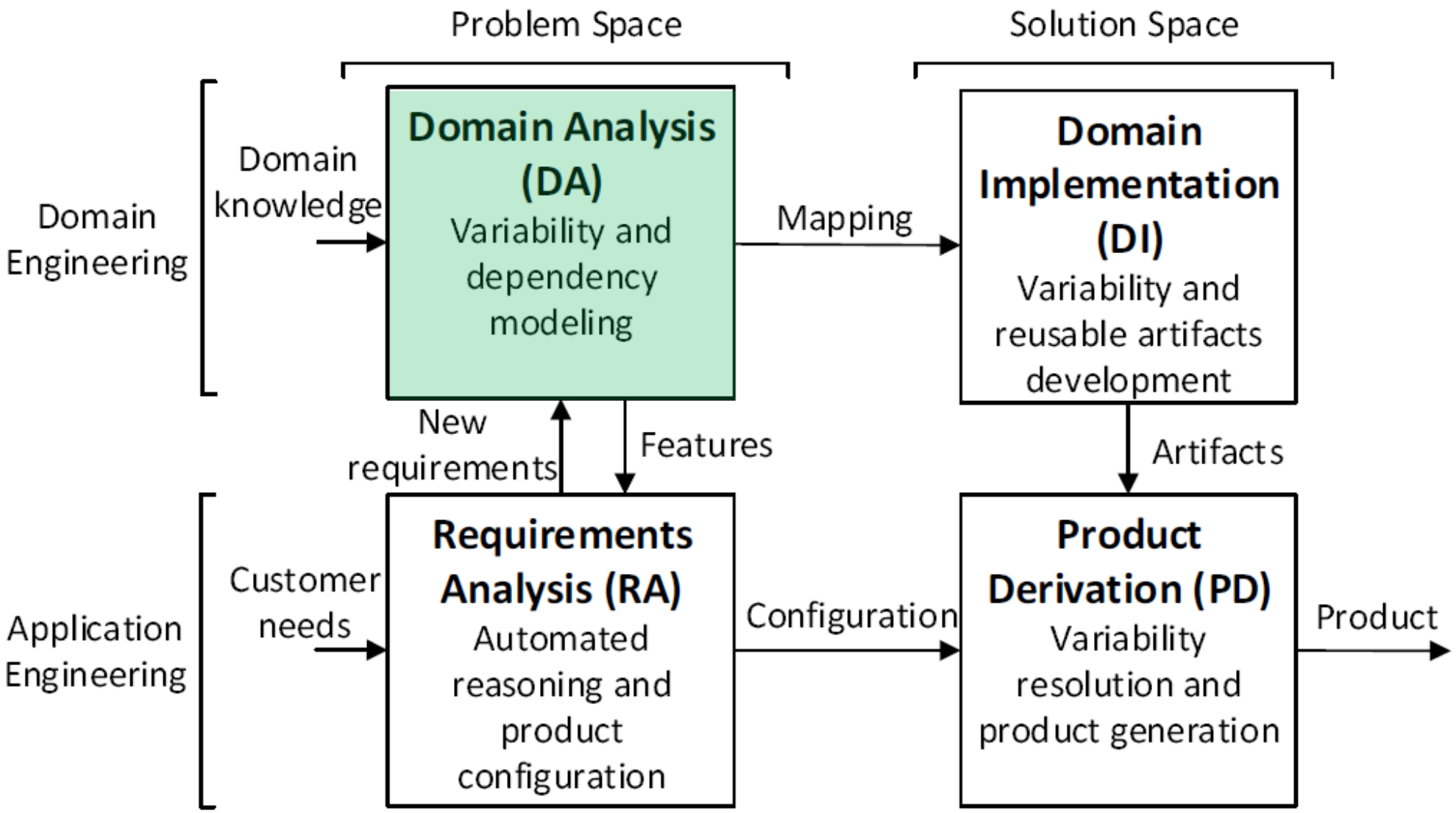
SPL tools

- FeatureIDE and UVL: http://www.featureide.com/
- Clafer: http://t3-necsis.cs.uwaterloo.ca:8094/
- Glencoe: https://glencoe.hochschule-trier.de/
- SPLOT: http://www.splot-research.org/
- FaMa: https://www.isa.us.es/fama/?FaMa_Current_Projects
- pure::variants: https://www.pure-systems.com/

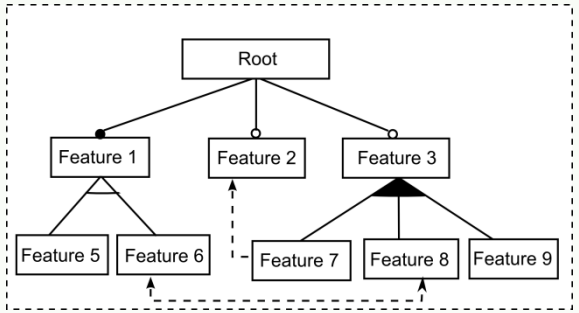
…

1. Variability modeling

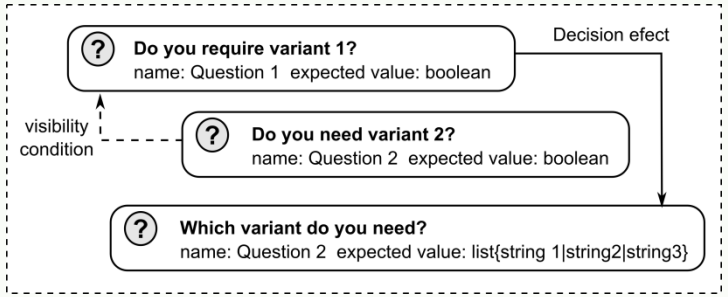
SPL activities for proactive approach



Modeling techniques

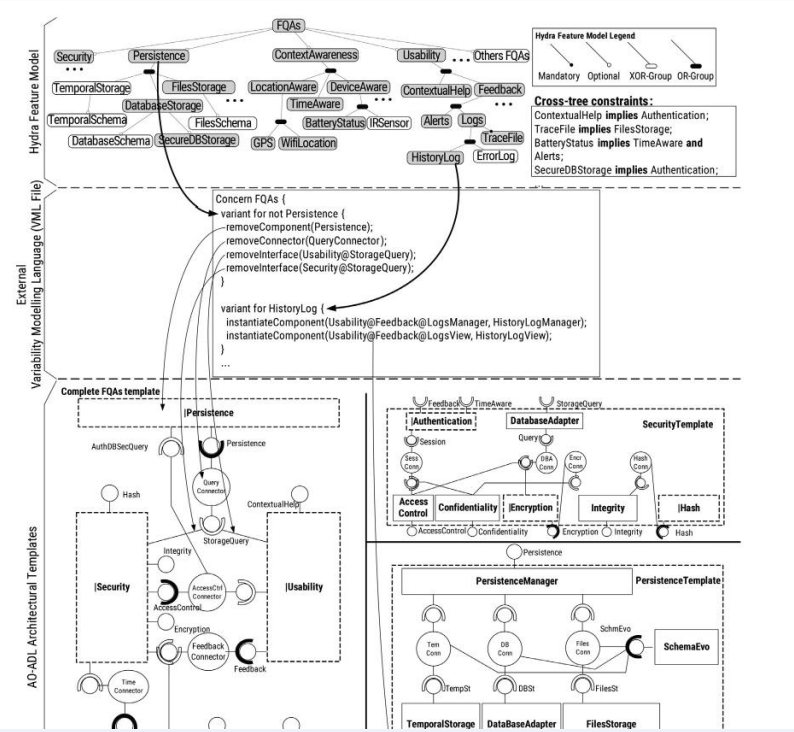


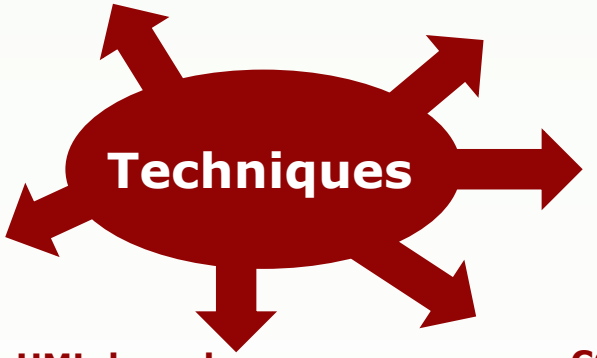
Feature modelling



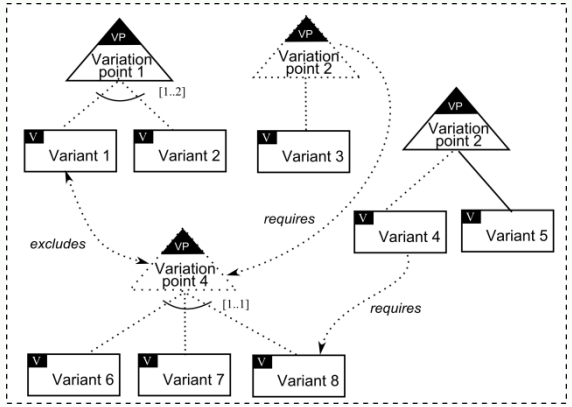
Decision modelling

Ad-hoc solutions: tables, textual docs, …

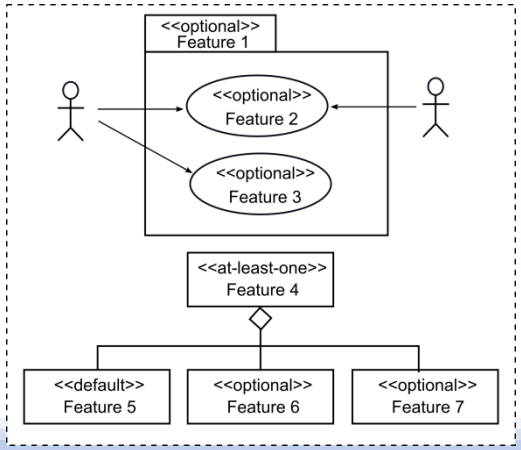




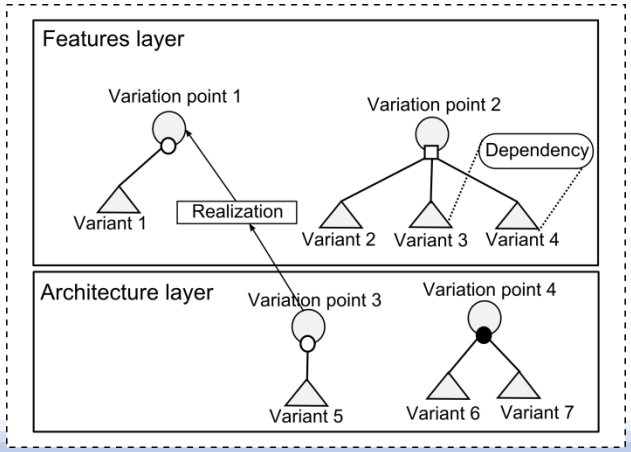
Orthogonal variability modelling



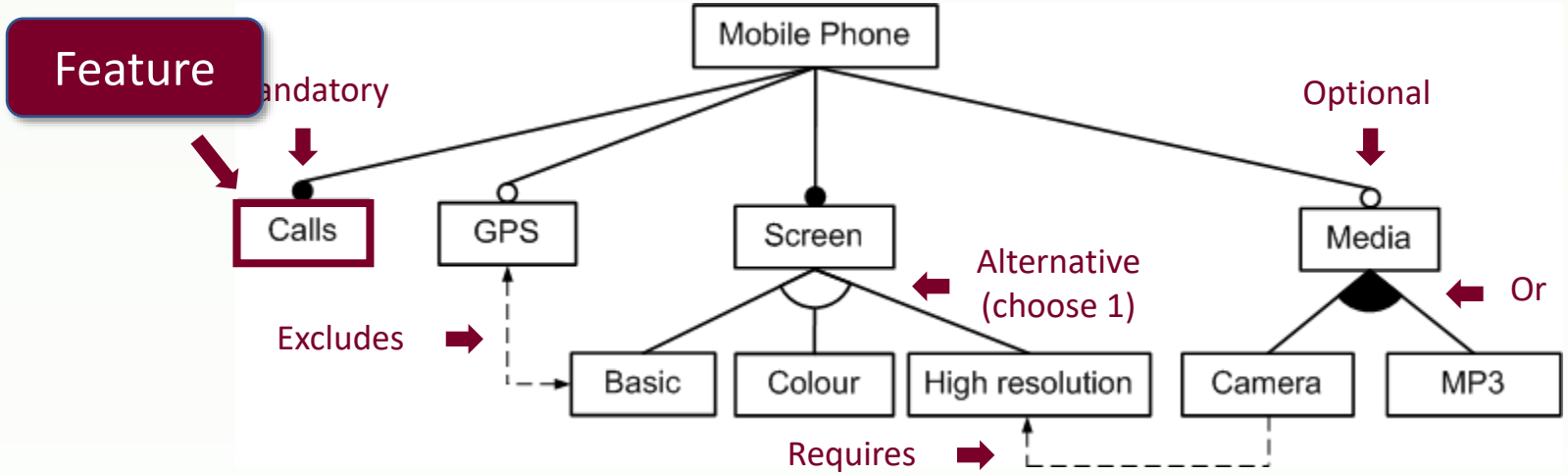
UML-based



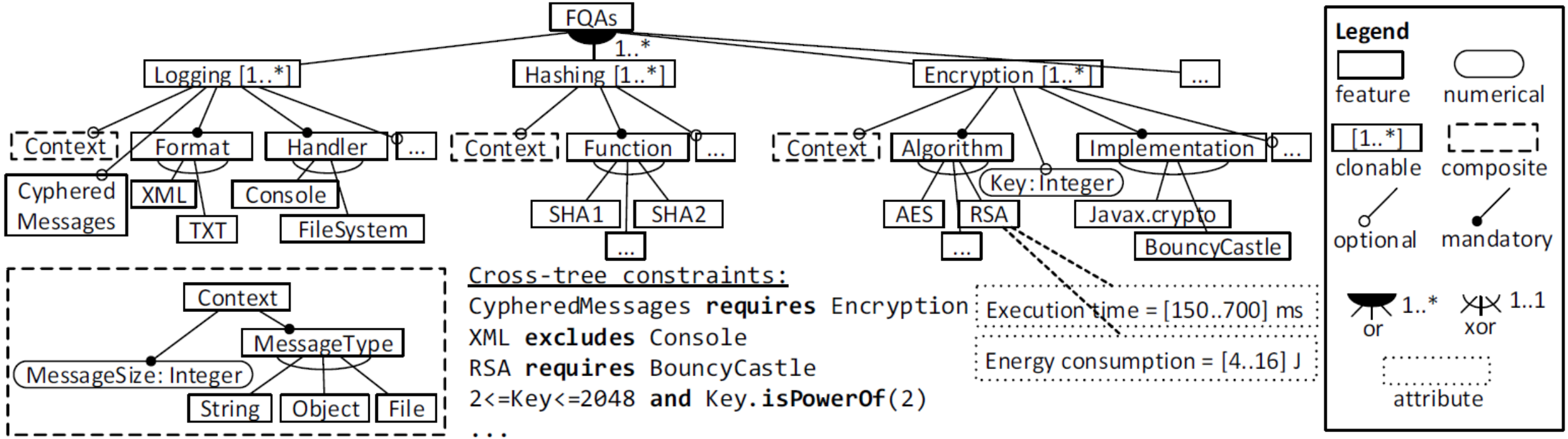
COVAMOF



Feature models



Feature models



180 features → 5719775658520318301491199 configurations (5*10^24)

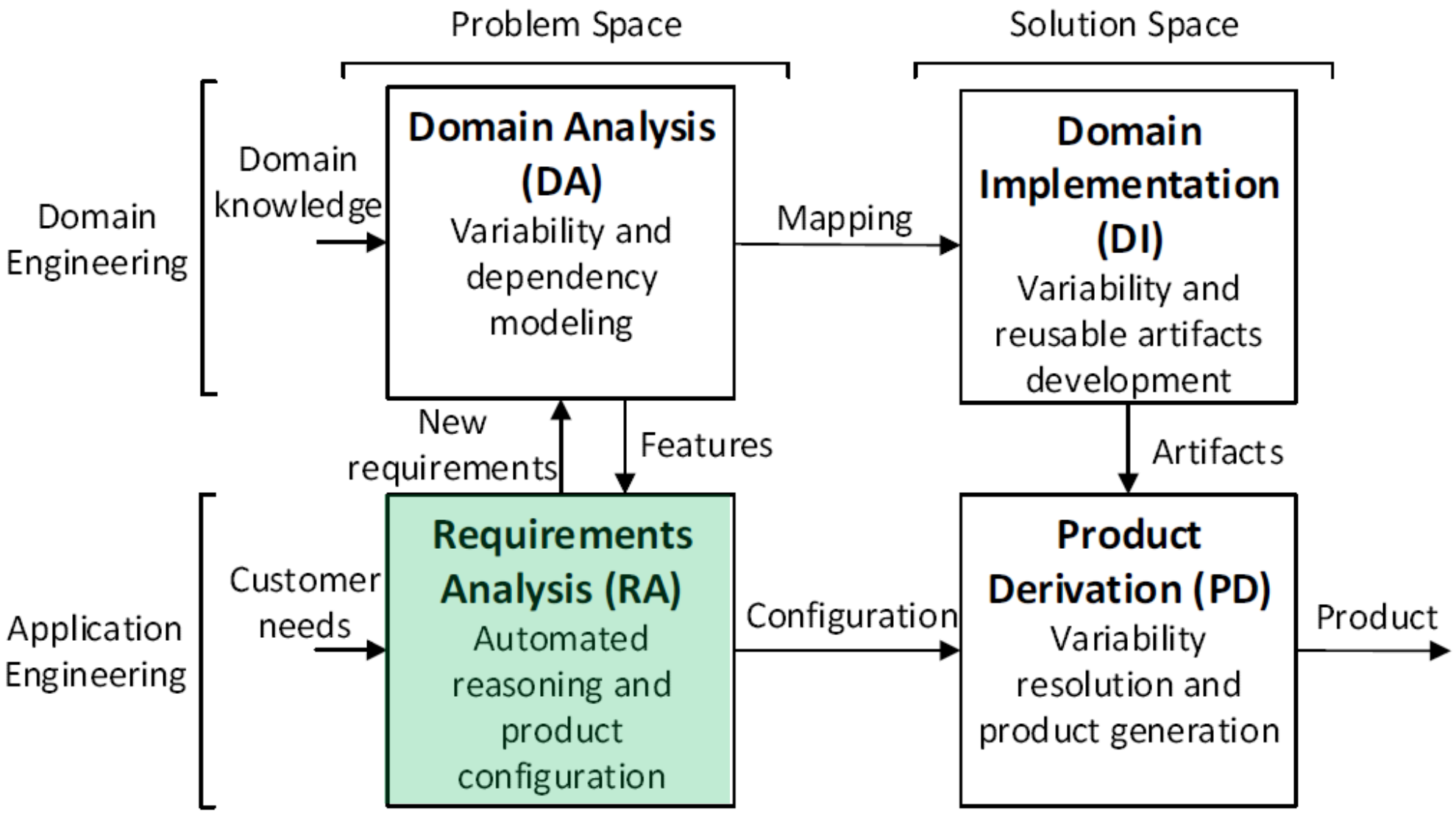
Feature models

Tools	Optional feat.	Xor group	Or group	Abstract feat.	Mutex-Group	Cardinalities	Multi decomp.	Multi-feature	Typed feature	Numerical feat.	Feat. attribute	Binding time	Default value	Delta value	Range	Simple const.	Prop. log. const.	First-order const.	Relational expr.	Arithmetic expr.	Type const.	Default const.	Compositions	Conf. reference	Containers	Model version	Multi-views	Configuration	Partial conf.
Glencoe	●	●	●	○	○	●	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	○	○	○	○	◐	○	●	●
SPLOT	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	●	◐	○	○	○	○	○	○	○	○	○	○	●	●
FaMa	●	●	●	○	○	○	○	○	○	○	●	◐	○	○	○	●	◐	○	○	○	○	○	○	○	○	○	○	●	●
Clafer	●	●	●	●	●	●	○	●	●	●	◐	◐	◐	○	◐	●	●	●	●	●	◐	●	◐	◐	◐	○	○	●	●
FeatureIDE	●	●	●	●	○	○	○	○	○	○	◐	○	○	○	○	●	●	○	○	○	○	○	○	○	○	○	○	●	●
pure::variants	●	●	●	●	○	○	○	○	○	◐	◐	○	○	○	○	●	●	○	◐	◐	○	○	●	◐	◐	○	○	●	●

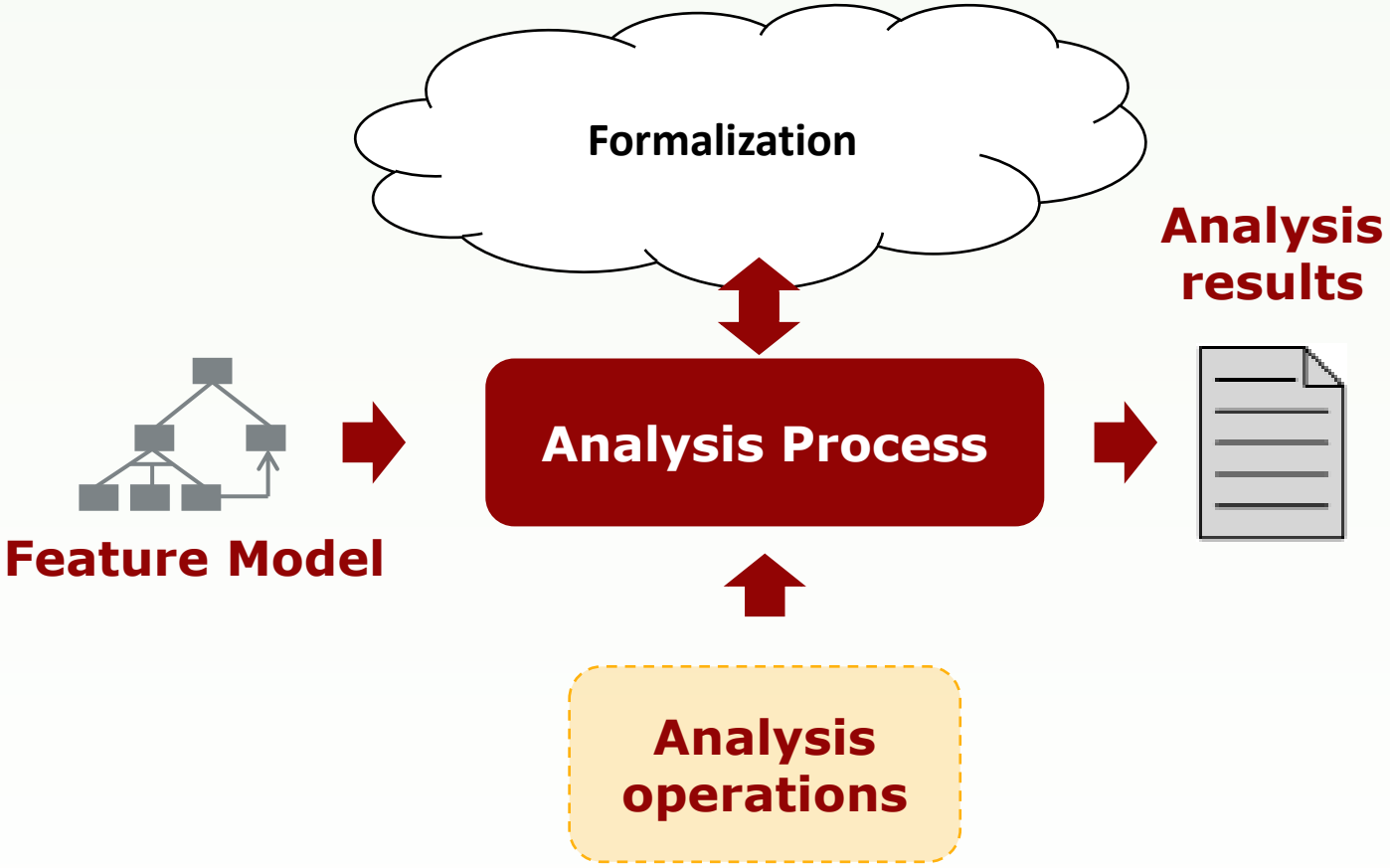
●: support. ◐: partially support. ○: not support.

2. Variability analysis

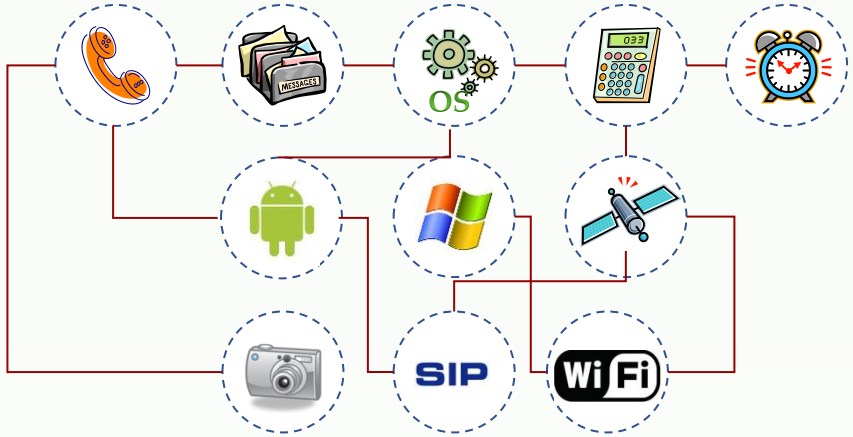
SPL activities for proactive approach



Automated Analysis of feature models (AAFM)



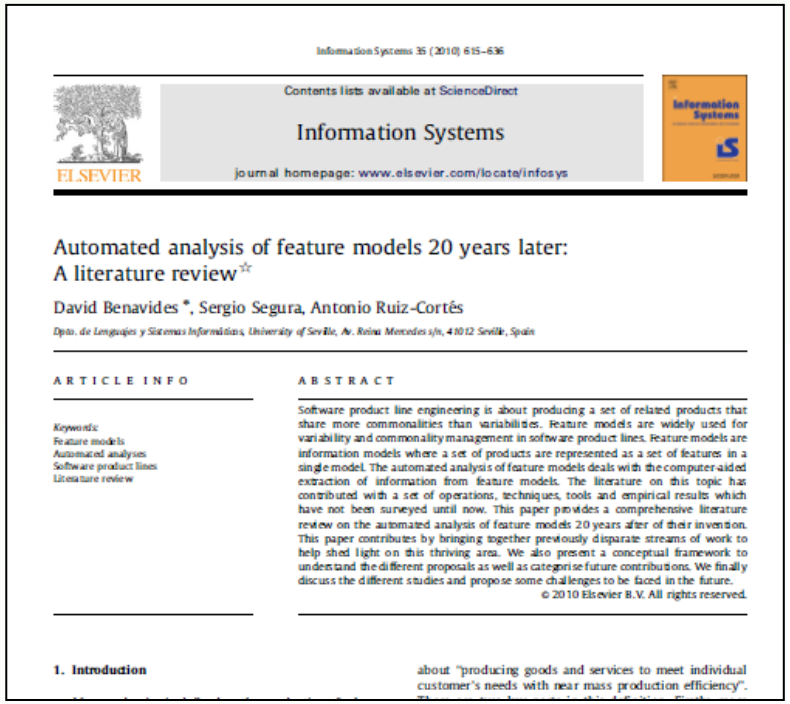
Analysis operations



My app requires WiFi and Bluetooth 5

In how many different devices will it work?

Analysis operations catalog



Information Systems 35 (2010) 615–636

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

Automated analysis of feature models 20 years later: A literature review☆

David Benavides *, Sergio Segura, Antonio Ruiz-Cortés

Dpto. de Lenguajes y Sistemas Informáticos, University of Seville, Av. Reina Mercedes s/n, 41012 Seville, Spain

ARTICLE INFO

Keywords:
Feature models
Automated analyses
Software product lines
Literature review

ABSTRACT

Software product line engineering is about producing a set of related products that share more commonalities than variabilities. Feature models are widely used for variability and commonality management in software product lines. Feature models are information models where a set of products are represented as a set of features in a single model. The automated analysis of feature models deals with the computer-aided extraction of information from feature models. The literature on this topic has contributed with a set of operations, techniques, tools and empirical results which have not been surveyed until now. This paper provides a comprehensive literature review on the automated analysis of feature models 20 years after of their invention. This paper contributes by bringing together previously disparate streams of work to help shed light on this thriving area. We also present a conceptual framework to understand the different proposals as well as categorise future contributions. We finally discuss the different studies and propose some challenges to be faced in the future.

1. Introduction

about "producing goods and services to meet individual customer's needs with near mass production efficiency".

Computing

May 2019, Volume 101, Issue 5, pp 387–433 | Cite as

Automated analysis of feature models: Quo vadis?

Authors | Authors and affiliations

José A. Galindo, David Benavides, Pablo Trinidad, Antonio-Manuel Gutiérrez-Fernández, Antonio Ruiz-Cortés

Tools	Void feat. model	#Products	Dead features	Valid product	All products	Explanations	Refactoring	Optimization	Commonality	Filter	Valid conf.	Atomic sets	False optional	Corrective explan.	Depend. analysis	ECR	Generalization	Core features	Variability factor	Arbitrary edit	Cond. dead feat.	Homogeneity	LCA	Multi-step conf.	Roots features	Specialization	Orthonality	Redundancies	Variant features	Wrong cardinal.
Glencoe	●	●	●	●	○	●	○	○	●	○	●	●	●	●	●	●	○	●	●	○	●	●	○	●	○	○	●	●	●	●
SPLOT	●	●	●	●	○	○	○	○	●	○	●	○	○	○	○	●	○	●	●	○	○	○	○	●	○	○	●	○	○	●
FaMa	●	◐	●	●	◐	●	○	○	◐	○	◐	○	○	●	●	○	○	◐	◐	○	○	○	○	○	○	○	○	○	●	○
Clafer	●	◐	○	●	◐	○	○	◐	○	○	◐	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
FeatureIDE	●	◐	●	●	◐	●	○	○	○	○	●	●	●	●	●	●	○	●	◐	○	●	○	○	●	○	○	●	●	●	●
pure::variants	●	◐	○	●	○	●	○	○	○	○	●	○	○	●	●	○	○	○	○	○	○	○	○	●	○	○	●	○	○	●

●: support. ○: not support. ◐: support with limitations or not scale for large models.

Automated analysis

Formalization
(SAT, #SAT, BDD, SMT, CSP,…)

Propositional Logic (SAT)

	Relationship	PL Mapping	Mobile Phone Example
MANDATORY	P — C	$P \leftrightarrow C$	MobilePhone ↔ Calls MobilePhone ↔ Screen
OPTIONAL	P — C	$C \rightarrow P$	GPS → MobilePhone Media → MobilePhone
OR	P: C_1, C_2, C_n	$P \leftrightarrow (C_1 \vee C_2 \vee \ldots \vee C_n)$	Media ↔ (Camera ∨ MP3)
ALTERNATIVE	P: C_1, C_2, C_n	$(C_1 \leftrightarrow (\neg C_2 \wedge \ldots \wedge \neg C_n \wedge P)) \wedge$ $(C_2 \leftrightarrow (\neg C_1 \wedge \ldots \wedge \neg C_n \wedge P)) \wedge$ $(C_n \leftrightarrow (\neg C_1 \wedge \neg C_2 \wedge \ldots \wedge \neg C_{n-1} \wedge P))$	(Basic ↔ (¬Color ∧ ¬Highresolution ∧ Screen))∧ (Color ↔ (¬Basic ∧ ¬Highresolution ∧ Screen))∧ (Highresolution ↔ (¬Basic ∧ ¬Color ∧ Screen))
IMPLIES	A ⇢ B	$A \rightarrow B$	Camera → Highresolution
EXCLUDES	A ⇠⇢ B	$\neg(A \wedge B)$	¬(GPS ∧ Basic)

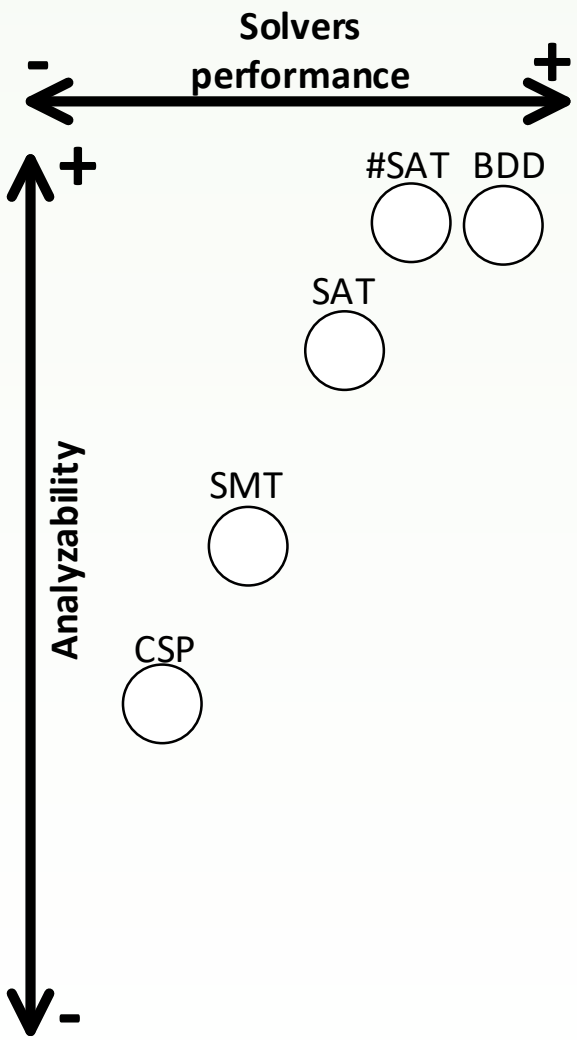
Constraint Programming (CSP)

	Relationship	CSP Mapping	Mobile Phone Example
MANDATORY	P — C	P = C	Mobilephone = Calls Mobilephone = Screen
OPTIONAL	P — C	if (P = 0) C = 0	if (Mobilephone = 0) GPS = 0 if (Mobilephone = 0) Media = 0

OR	P: C1, C2, Cn	if (P > 0)
		Sum (C1, C2,...Cn) in {1..n}
		else
		C1 = 0, C2 = 0,...., Cn = 0
			if (Media > 0)
			Sum (Camera, MP3) in {1..2}
			else
			Camera = 0, MP3 = 0

ALTERNATIVE	P: C1, C2, Cn	if (P > 0)
		Sum (C1 ,C2,...Cn) in {1..1}
		else
		C1 = 0, C2 = 0,...., Cn = 0
			if (Screen > 0)
			Sum (Basic, Colour, High resolution) in {1..1}
			else
			Basic = 0, Colour = 0, High resolution = 0

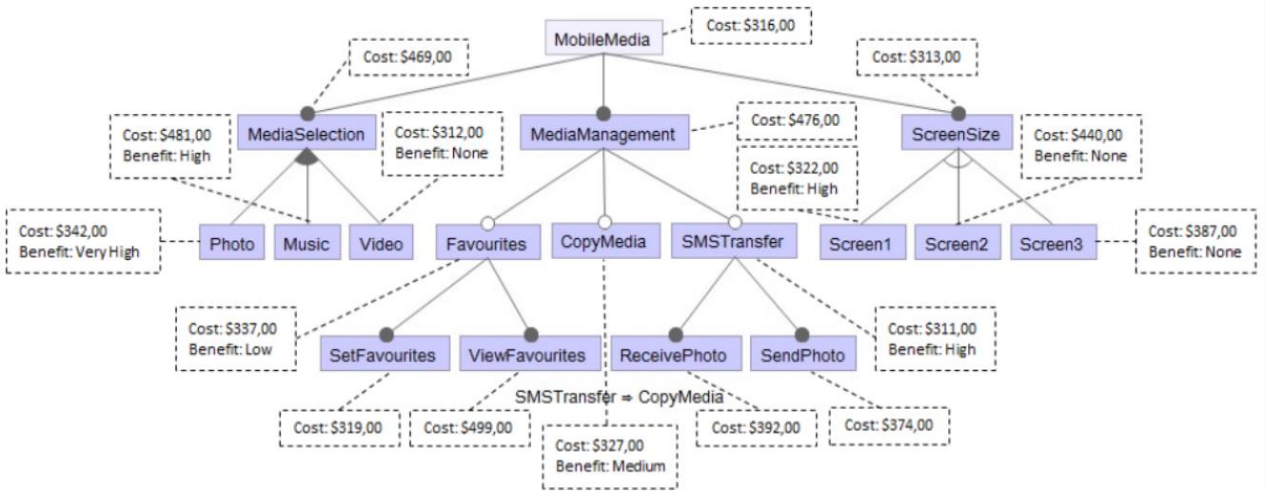
REQUIRES	A ⇢ B	if (A > 0) B>0	if (Camera > 0) High resolution > 0
EXCLUDES	A ⇠⇢ B	if (A > 0) B = 0	if (GPS > 0) Basic = 0

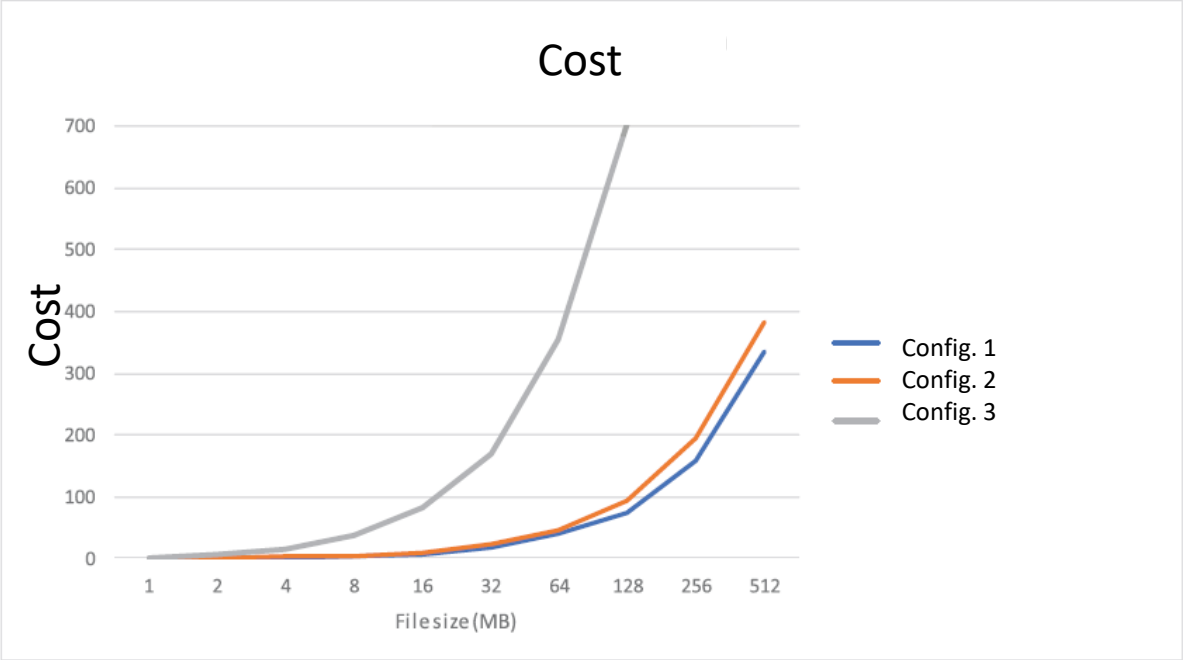


Advanced operations

- **Optimization of FMs configurations**
  - Feature atributes
    - Cost
    - Performance
    - Energy consumption
    - …
  - Goal: Obtaining optimum configurations based on a quality criteria.

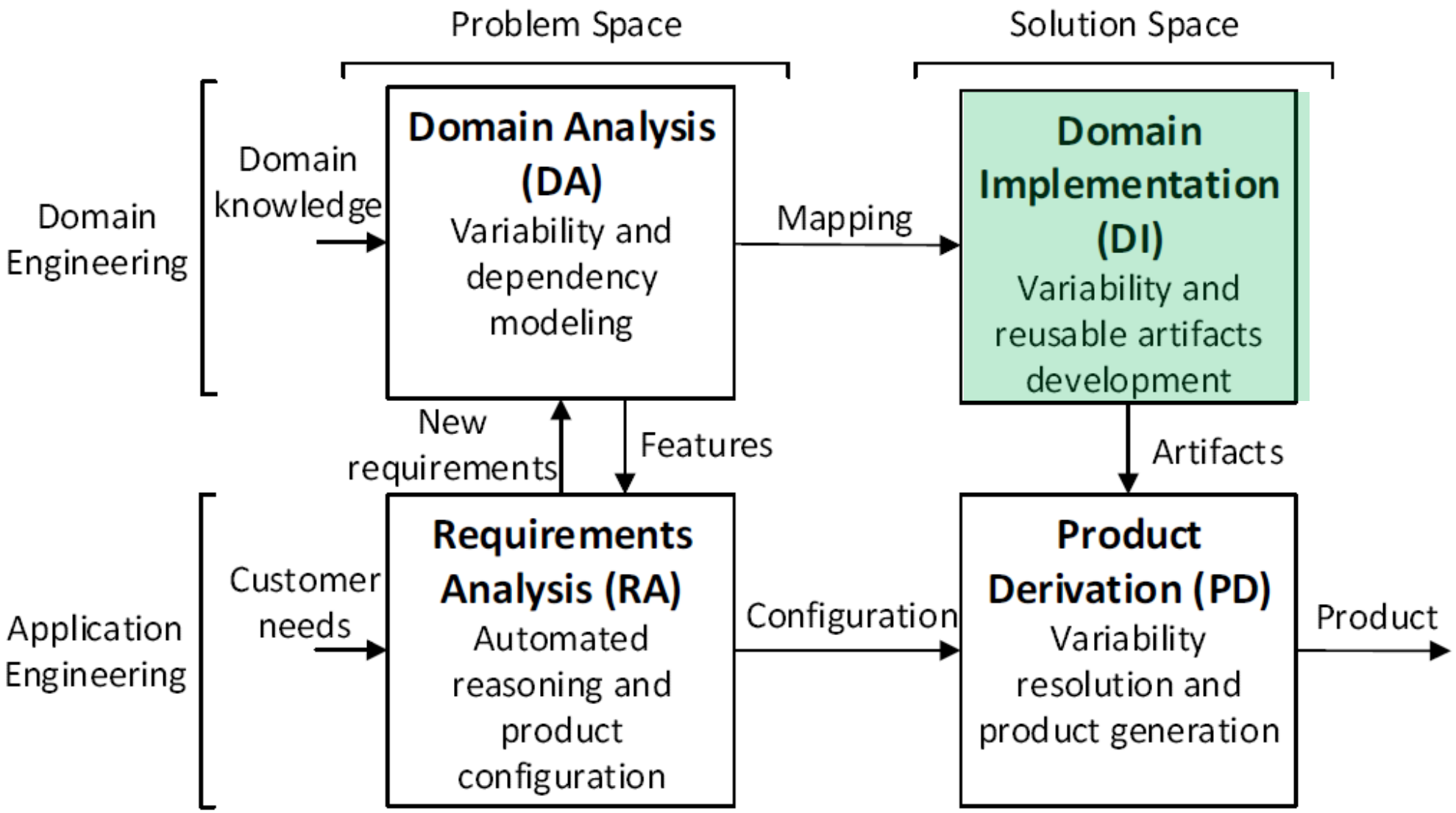
- **Sampling configurations**
  - Sampling techniques:
    - Uniform random sampling (URS)
    - Latin hypercube sampling (LHS)
    - Statistical recursive searching (SRS)
    - …
  - Goal: Not evaluating all possible configurations.





3. Variability implementation

SPL activities for proactive approach



Implementation approaches

Annotation-based approaches

Configuration parameters
Conditional statements (if, switch)
Global configuration parameters
Parameter objects
...

Virtual separation of concerns
Coloring (CIDE)
FeatureMapper
...

Preprocessors
C preprocessor (CPP)
Frames/XVCL
StringTemplate
Spoon
Munge
Antenna
Colligens
fmp2rsm
...

Composition-based approaches

Feature-Oriented Programming
AHEAD, FeatureHouse, Jak, FeatureComposer, GenVoca,...

Aspect-Oriented Programming
AspectJ, AspectC, AspectC++, SpringAOP,...

Variability design patterns
Observer, template method, strategy, decorator,...

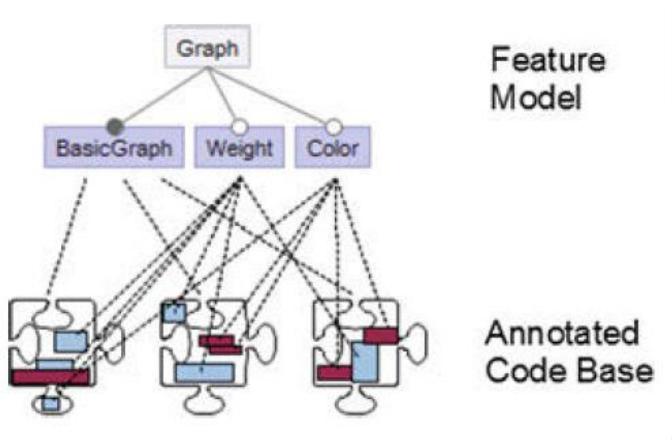
Build systems
Kbuild, make, ant, maven,...

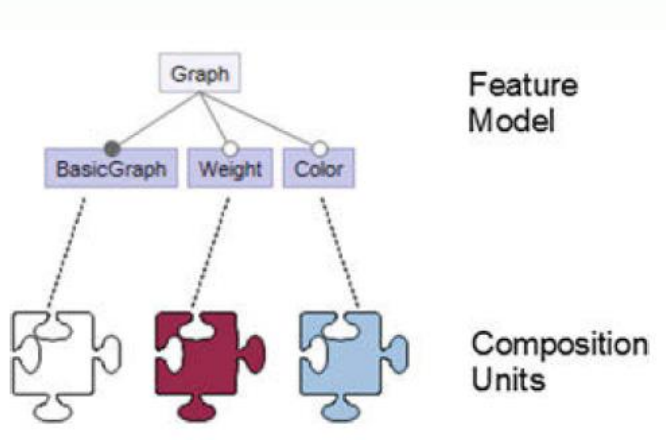
Frameworks
White-box frameworks
Black-box frameworks (plug-ins)

Components and services
Component-based development
Services-oriented architectures

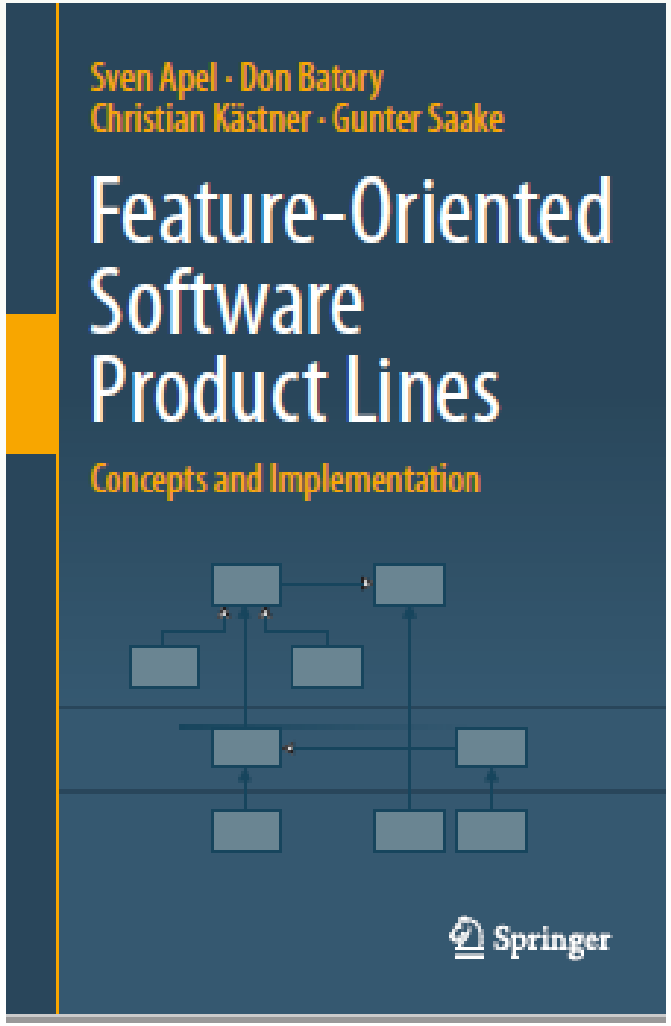
Others
Delta-Oriented Programming
Refactoring feature modules
Context-Oriented Programming
FSTComposer
CVL
...

Version-control systems
Git, Mercurial, CVS, Subversion,...



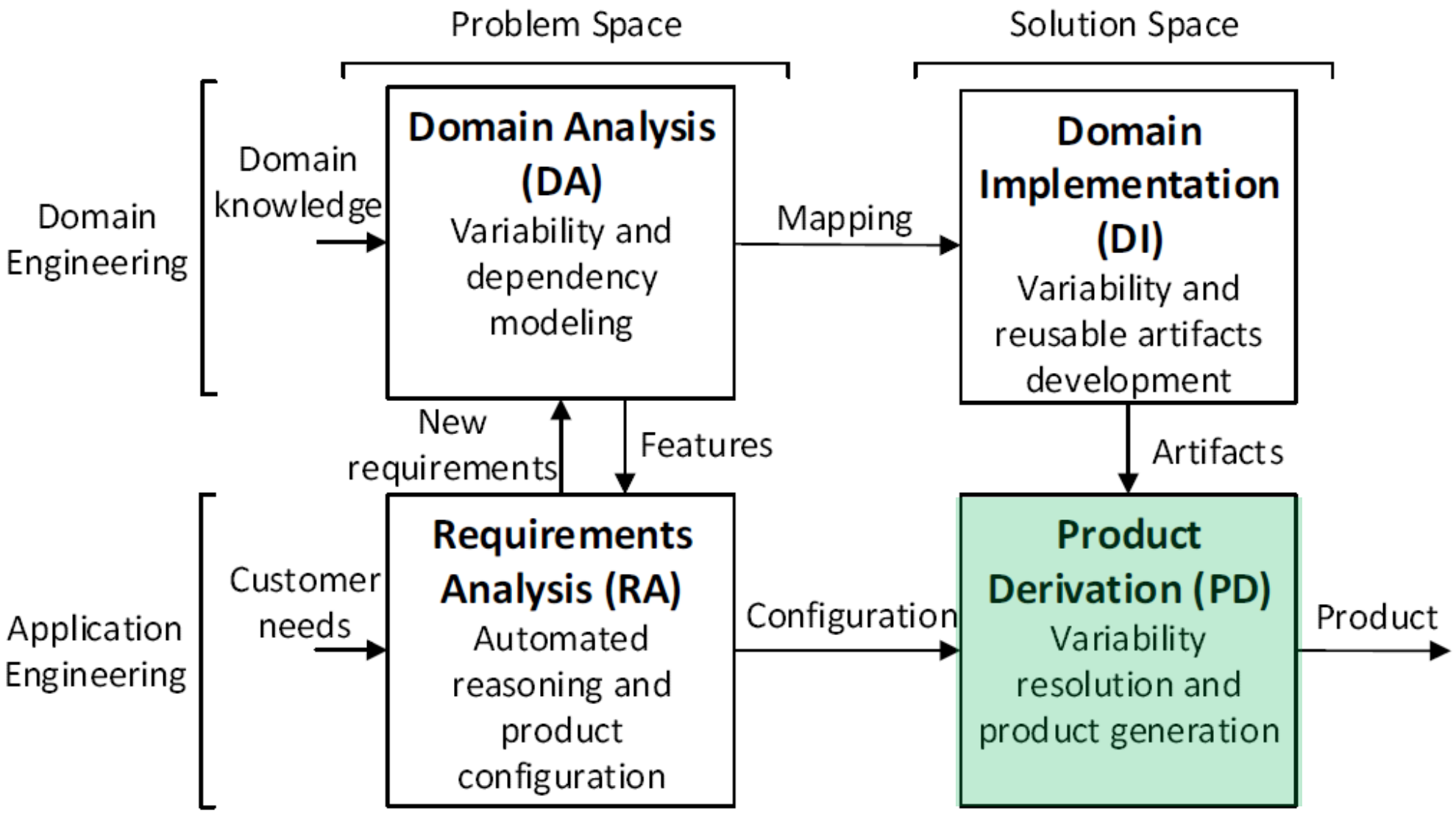


Implementation approaches

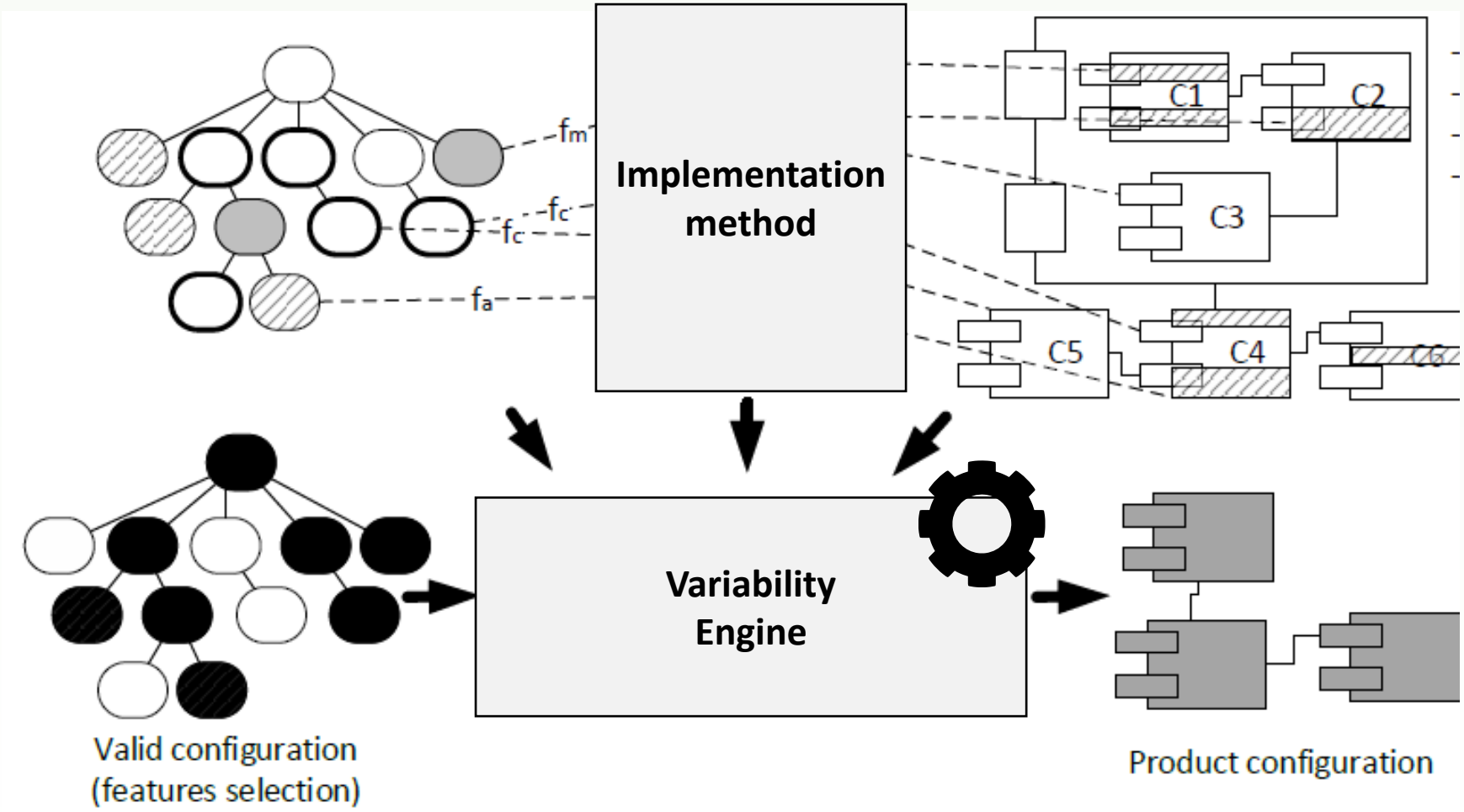


4. Product derivation

SPL activities for proactive approach



Product generation



Variability resolution

Annotation-based approaches

Configuration parameters
Conditional statements (if, switch)
Global configuration parameters
Parameter objects
...

Virtual separation of concerns
Coloring (CIDE)
FeatureMapper
...

Preprocessors
C preprocessor (CPP)
Frames/XVCL
StringTemplate
Spoon
Munge
Antenna
Colligens
fmp2rsm
...

Composition-based approaches

Feature-Oriented Programming
AHEAD, FeatureHouse, Jak, FeatureComposer, GenVoca,...

Aspect-Oriented Programming
AspectJ, AspectC, AspectC++, SpringAOP,...

Variability design patterns
Observer, template method, strategy, decorator,...

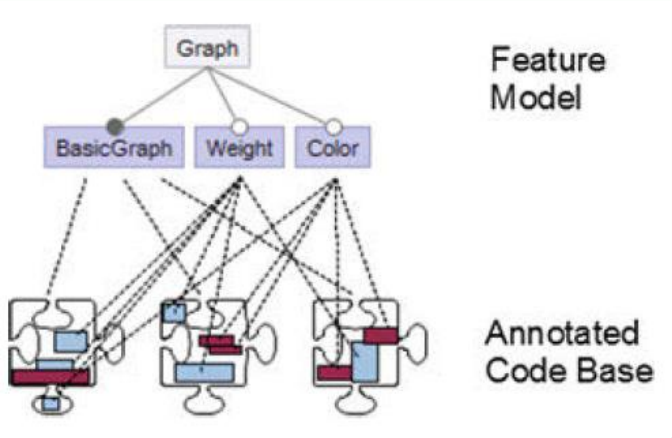
Build systems
Kbuild, make, ant, maven,...

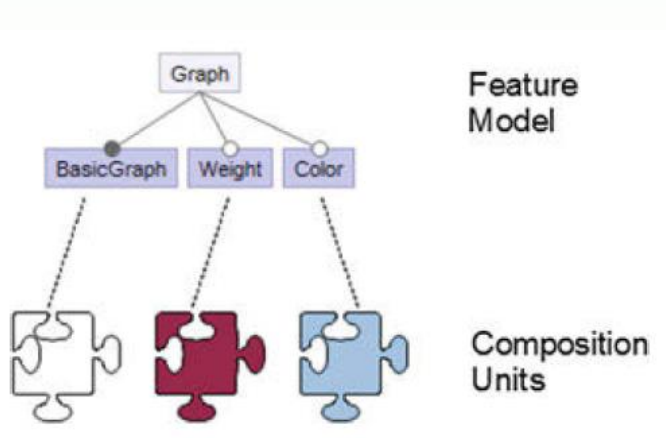
Frameworks
White-box frameworks
Black-box frameworks (plug-ins)

Components and services
Component-based development
Services-oriented architectures

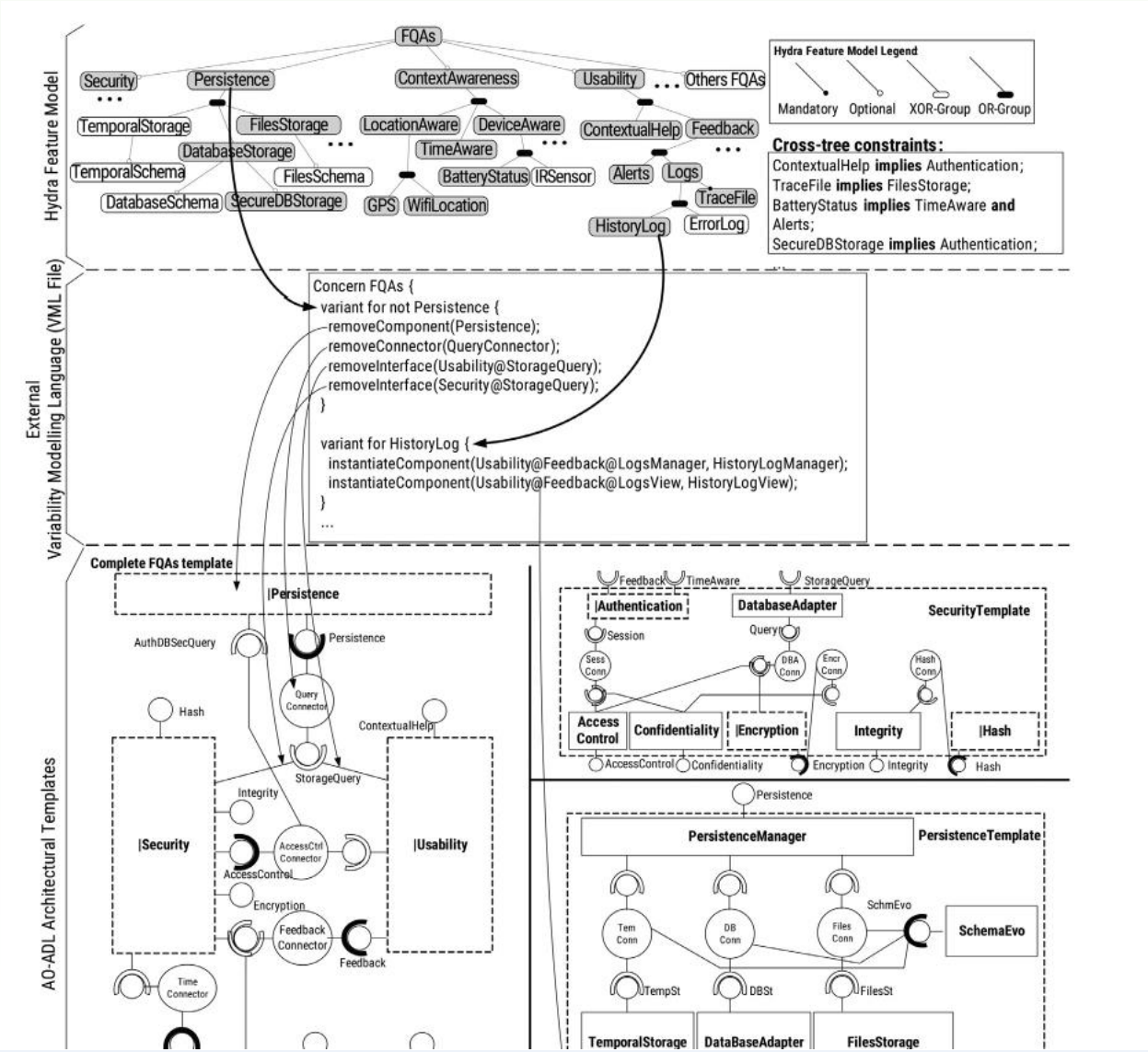
Others
Delta-Oriented Programming
Refactoring feature modules
Context-Oriented Programming
FSTComposer
CVL
...

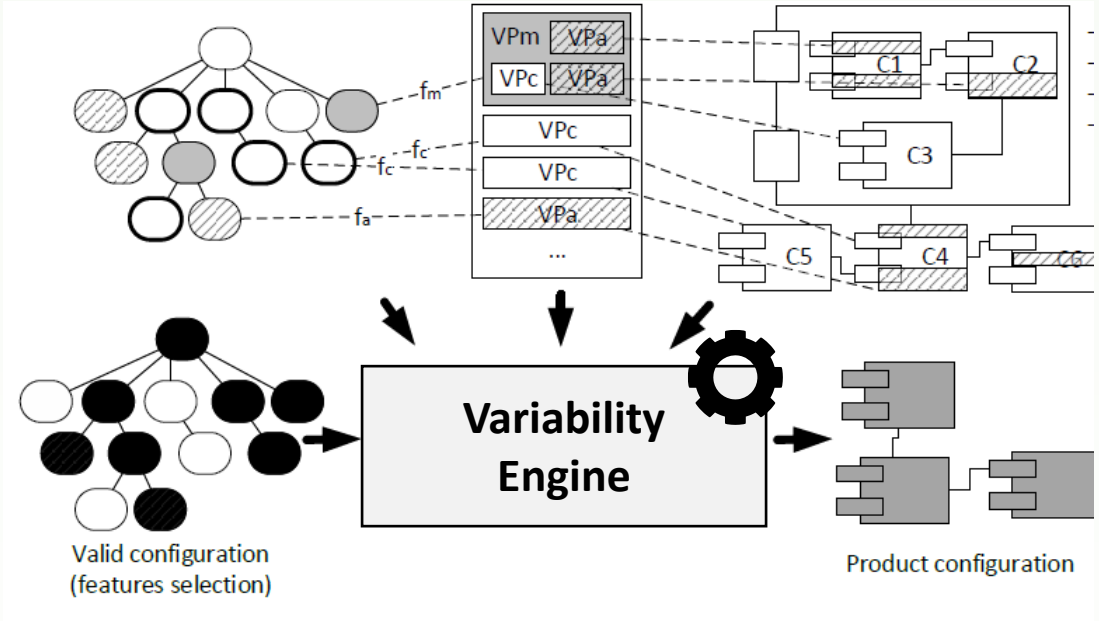
Version-control systems
Git, Mercurial, CVS, Subversion,...





Ortoghonal approach





vEXgine UMA

Extending the Common Variability Language (CVL) Engine: A practical tool

Jose-Miguel Horcas
Universidad de Málaga
Andalucía Tech, Málaga, Spain
horcas@lcc.uma.es

Mónica Pinto
Universidad de Málaga
Andalucía Tech, Málaga, Spain
pinto@lcc.uma.es

Lidia Fuentes
Universidad de Málaga
Andalucía Tech, Málaga, Spain
lff@lcc.uma.es

ABSTRACT
The Common Variability Language (CVL) has become a reference in the specification and resolution of variability in the last few years. Despite the multiple advantages of CVL (orthogonal variability, variability in the last few years. CVL has displaced traditional feature modeling for managing the variability in Software Product Lines (SPLs) approaches [10], especially for architecture-centric approaches. This is partly due to several advantages that CVL has

http://caosd.lcc.uma.es/vexgine/

Questions

- **José Miguel Horcas**
  - https://sites.google.com/view/josemiguelhorcas
  - horcas@lcc.uma.es

  CAOSD group, Universidad de Málaga

  http://caosd.lcc.uma.es/