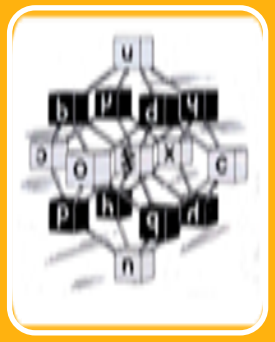
Gestión de la Variabilidad y Líneas de Producto Software

Evolución y Gestión de la Configuración

UNIVERSIDAD D SEVILLA

Parte I

Software Product Lines

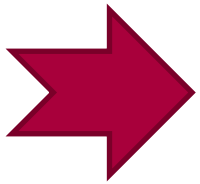


Variability Modelling

Why a new software production paradigm?

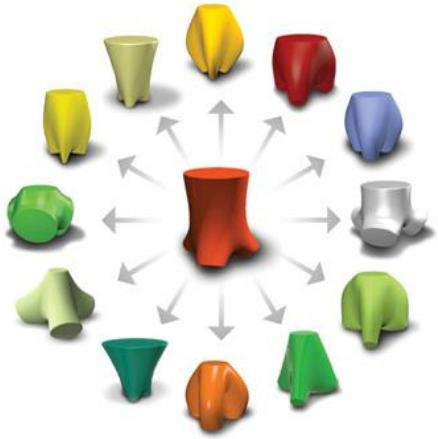
Software product lines

Communicate



Reproduce

Produce



Software product lines

A new software production paradigm

Software Product Line Engineering

Industrial Trends

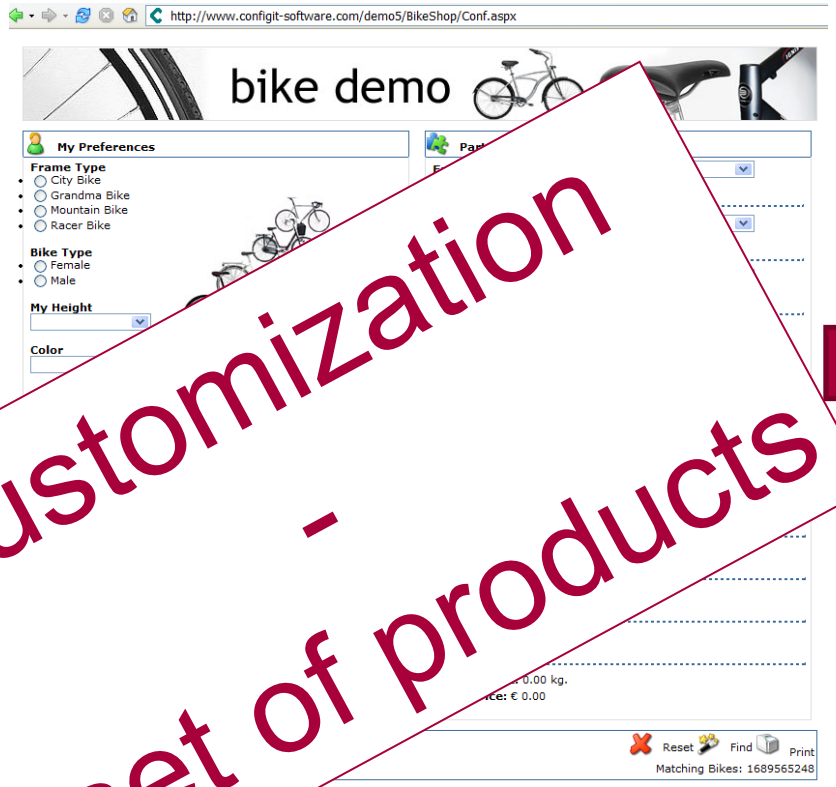
Organizations are evolving	• *Project* Centric Software Engineering • *Product* Centric Software Engineering
Software ***variability*** constantly increasing:	• Variability goes from hardware to software • Variations points grows by thousands
Assets' ***Reuse*** is shifting	• from ad-hoc to ***systematic***

No customization - one product

oduction

acing efficiently a large amount of standardized products

Software product lines



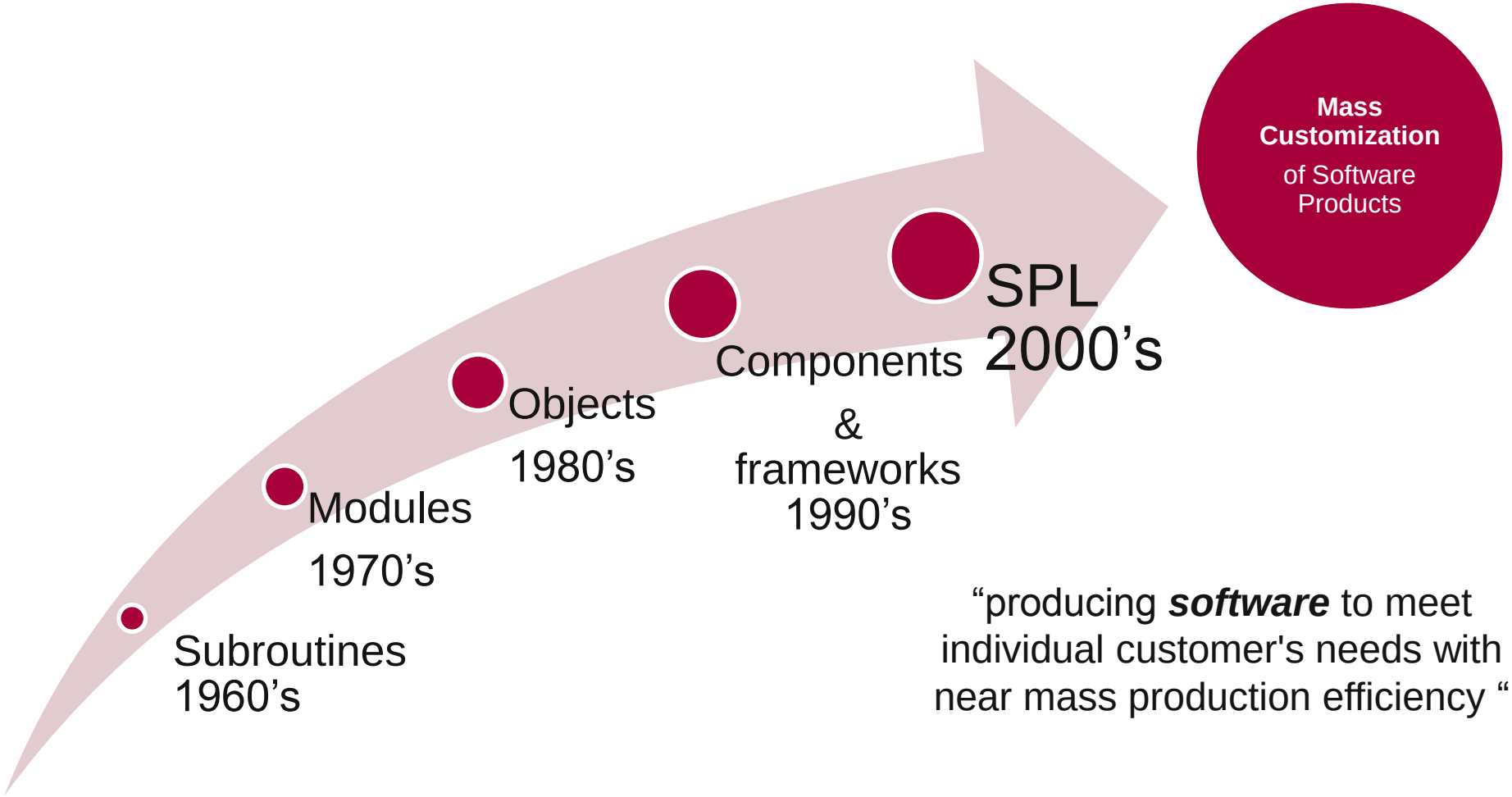
Customization
-
A set of products

Mass customization

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

[Tseng, M.M., Jiao, J. (2001)]

Software product lines



"producing ***software*** to meet individual customer's needs with near mass production efficiency "

Software product lines

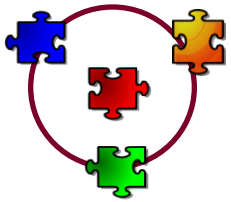
Common features

- Alarm clock
- Calls
- Messaging

Variable features

- Media
- Games
- Connectivity

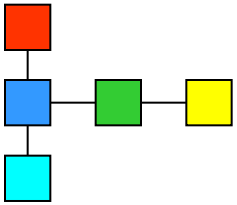
Variability Model



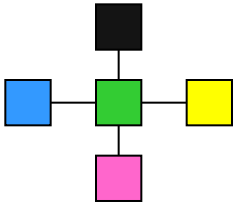
- Documents the variability of SPL
- Enable managing the variability

Software product lines

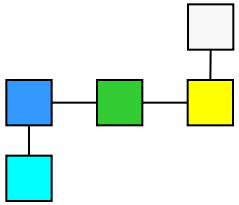
Traditional Approach (*mass production*)



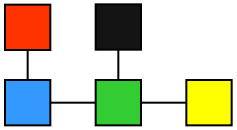
Product 1



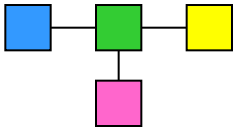
Product 2



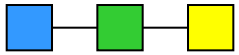
Product 3



Product 4

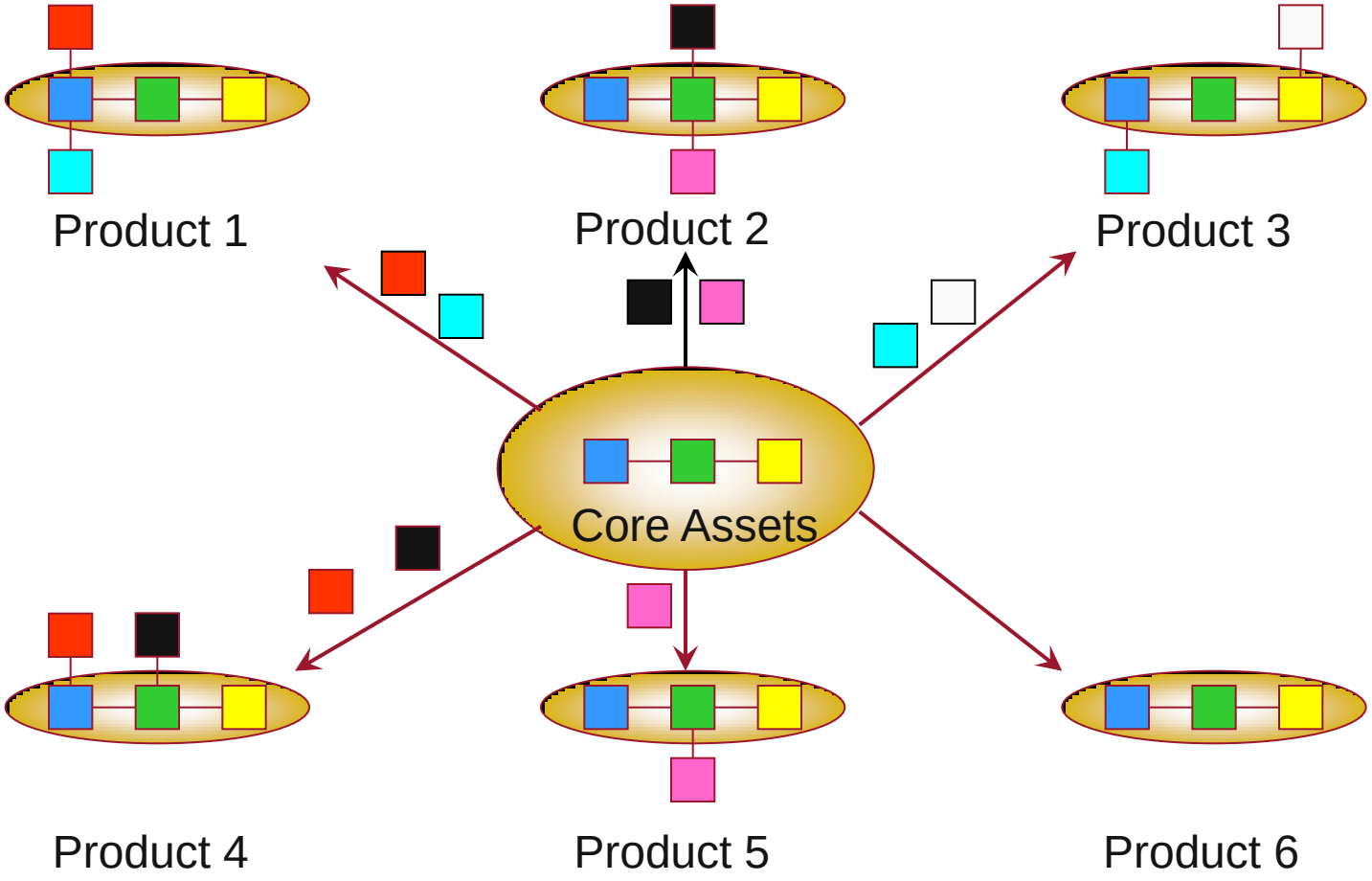


Product 5

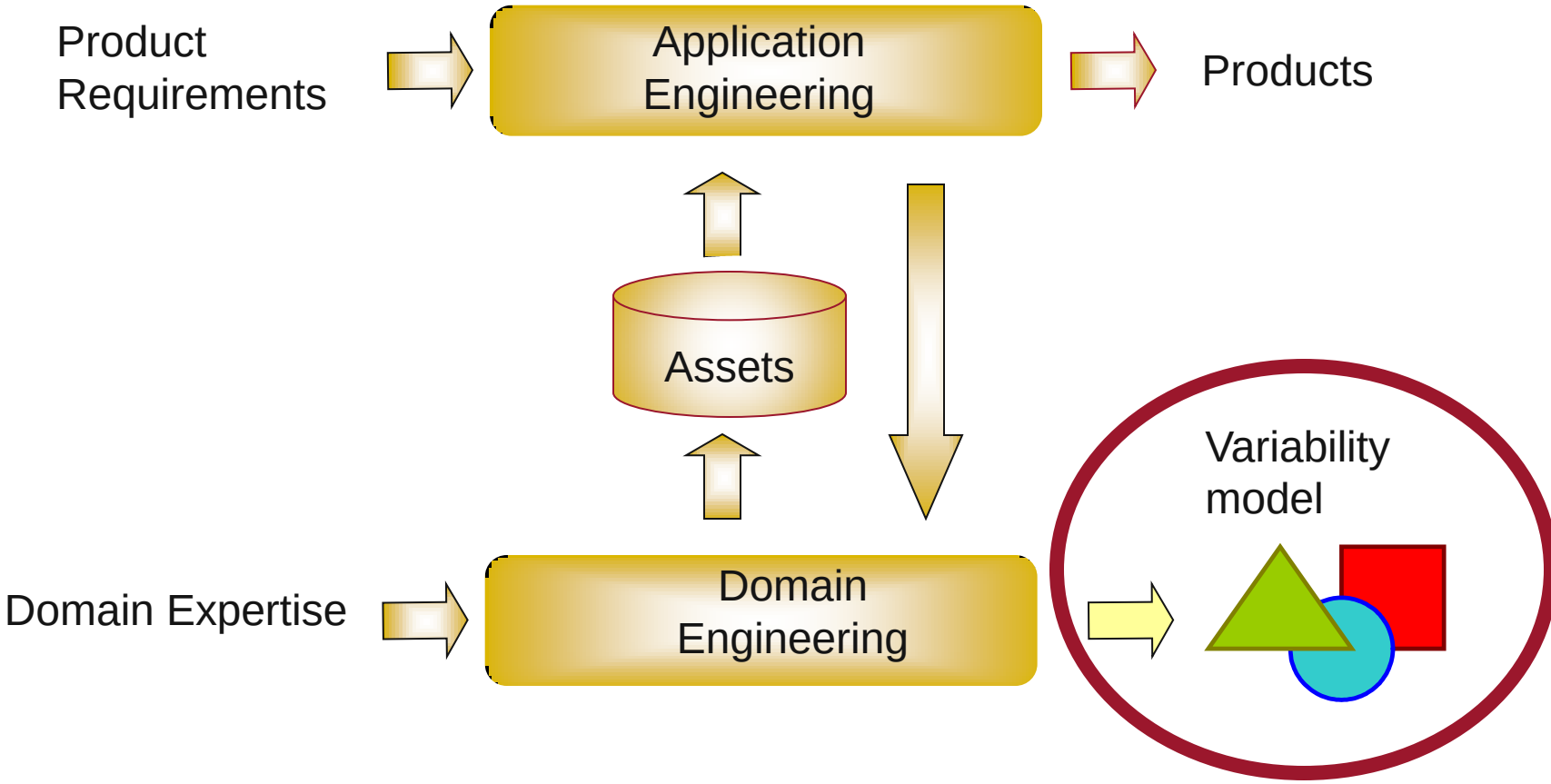


Product 6

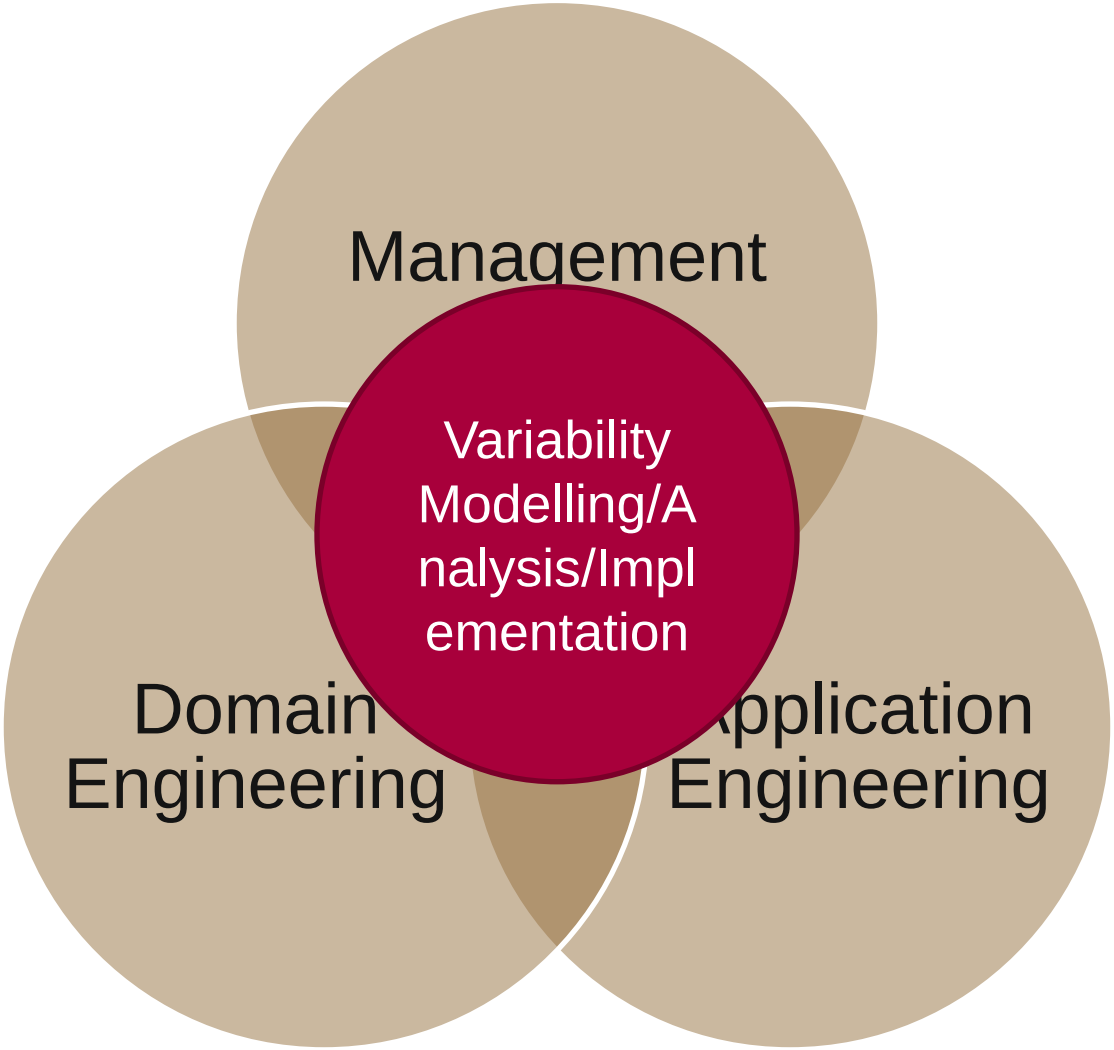
Product Lines Approach (*mass customization*)

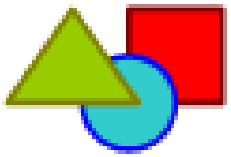


SPL: Activities

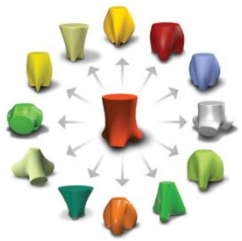


Software product lines



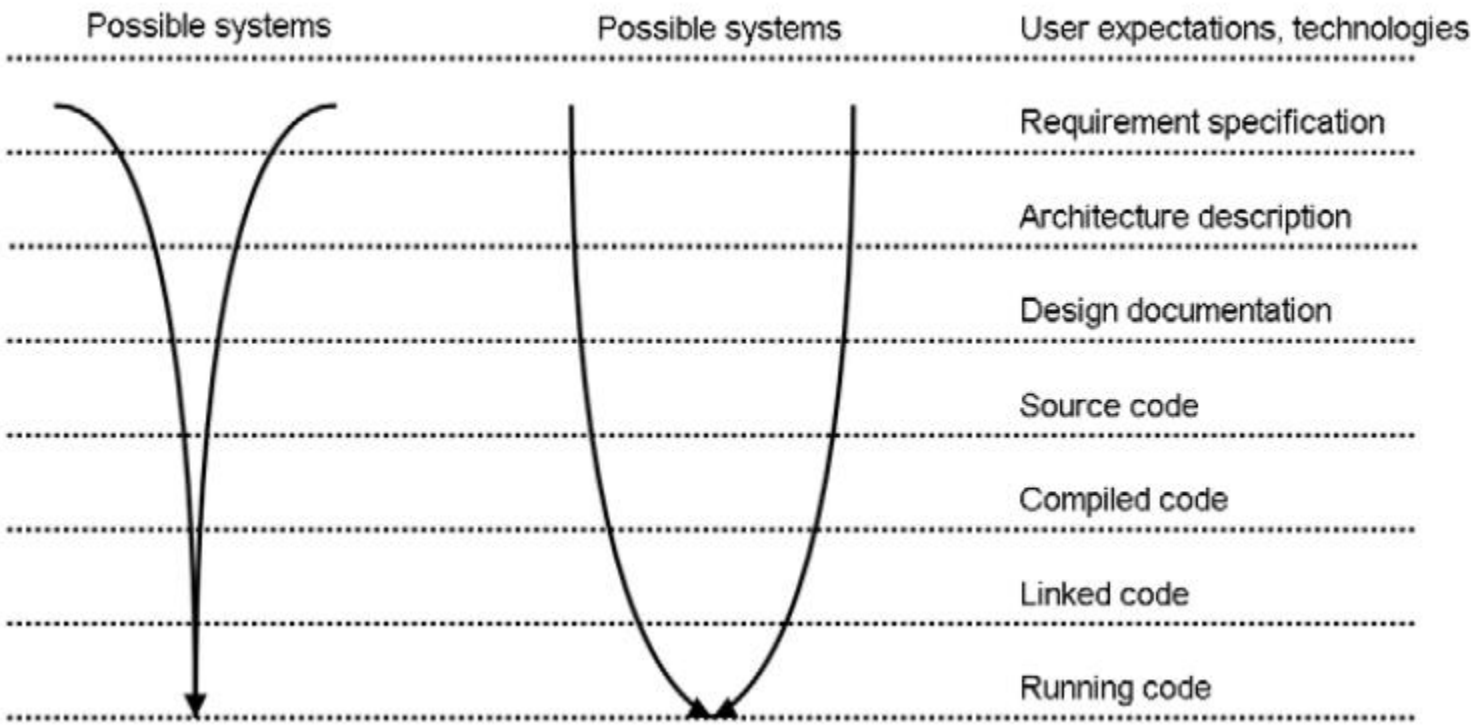


Core assets



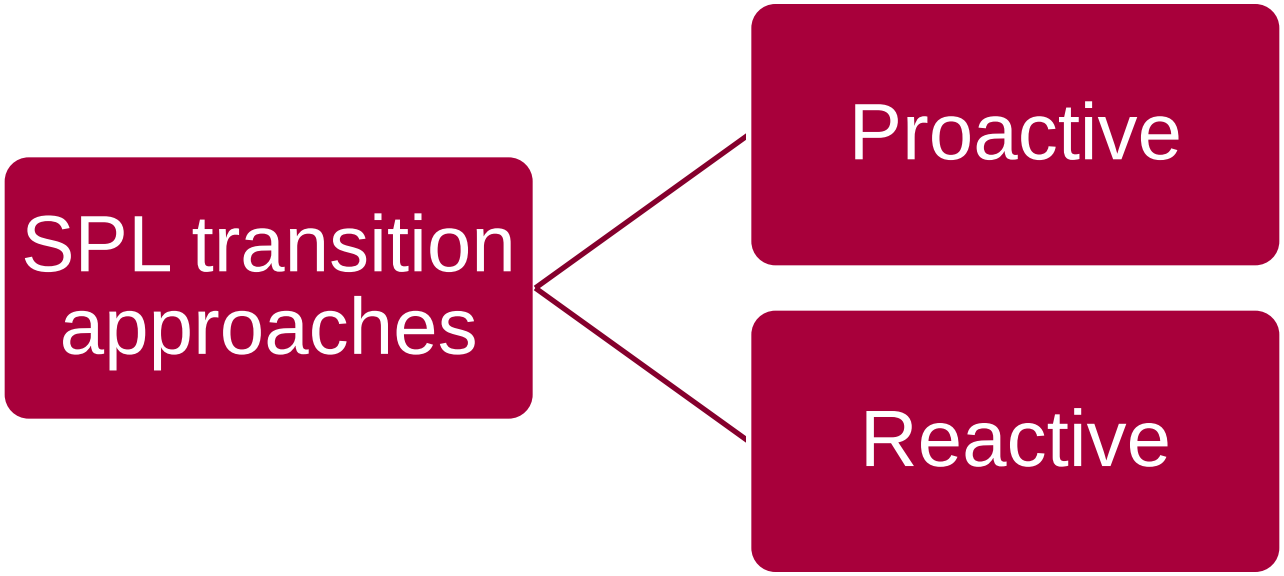
Individual products

SPL metaphors

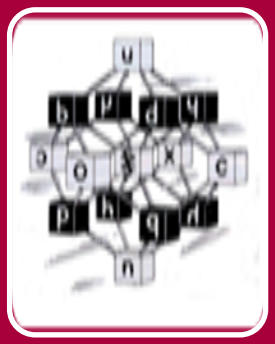


Svahnberg M., van Gurp J., Bosch J., *On the Notion of Variability in Software Product Lines*. Proceedings of IEEE/IFIP Conference on Software Architectures, 2001.

Is SPL incompatible with any software production methodology (e.g. agile approaches)?

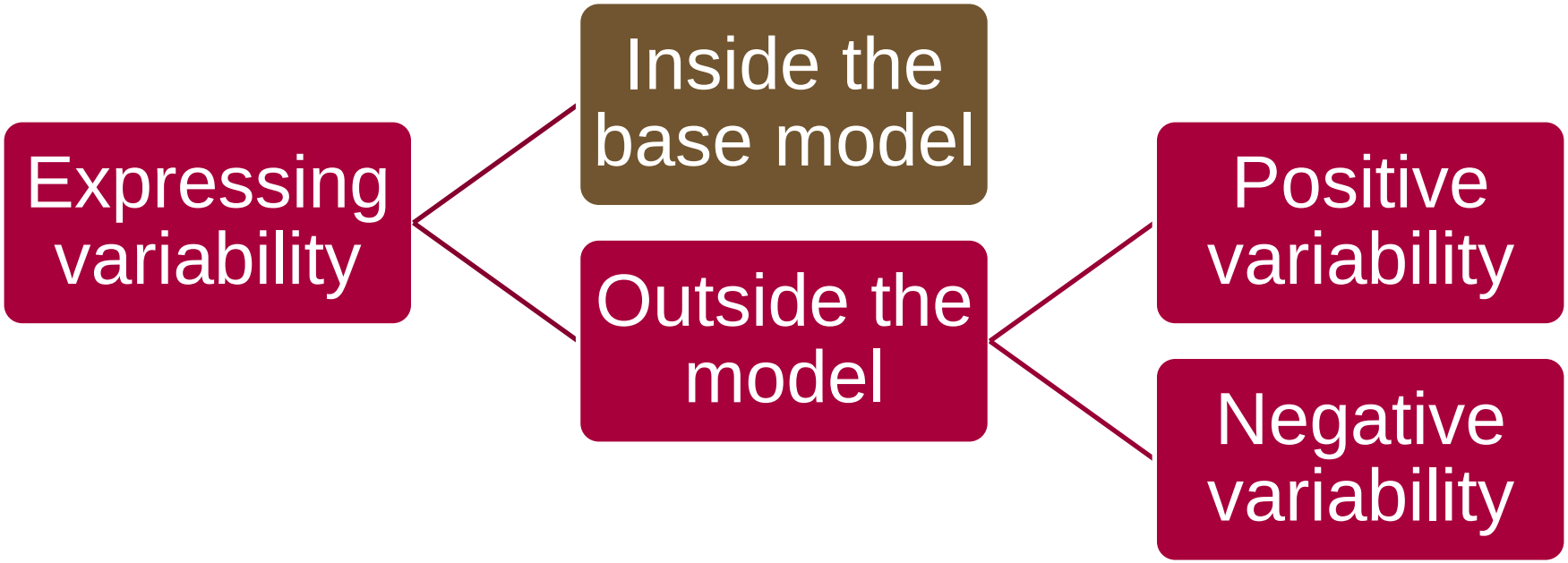

SPL transition approaches
Proactive
Reactive

Software Product Lines



Variability modelling

How to model variability



Inside the model

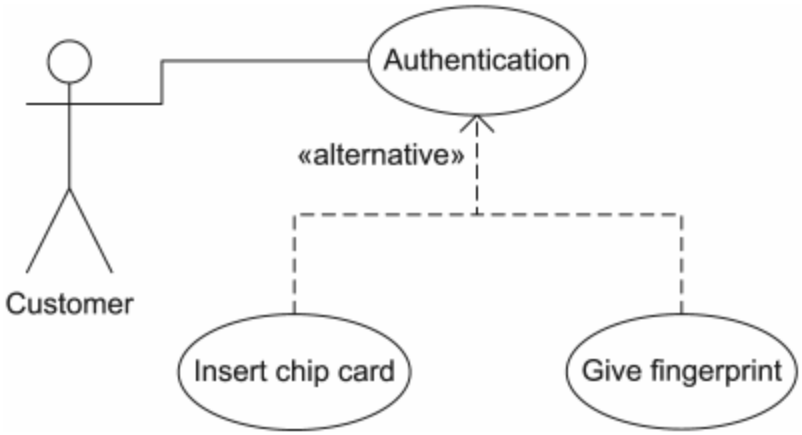


Figure 5: Example of an alternative relationship

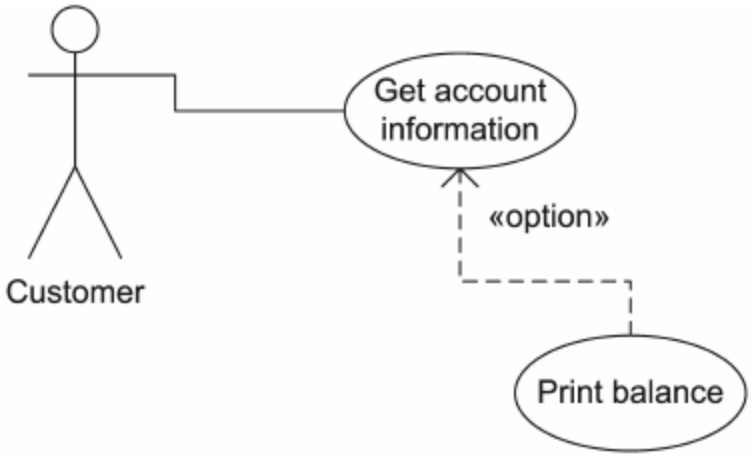
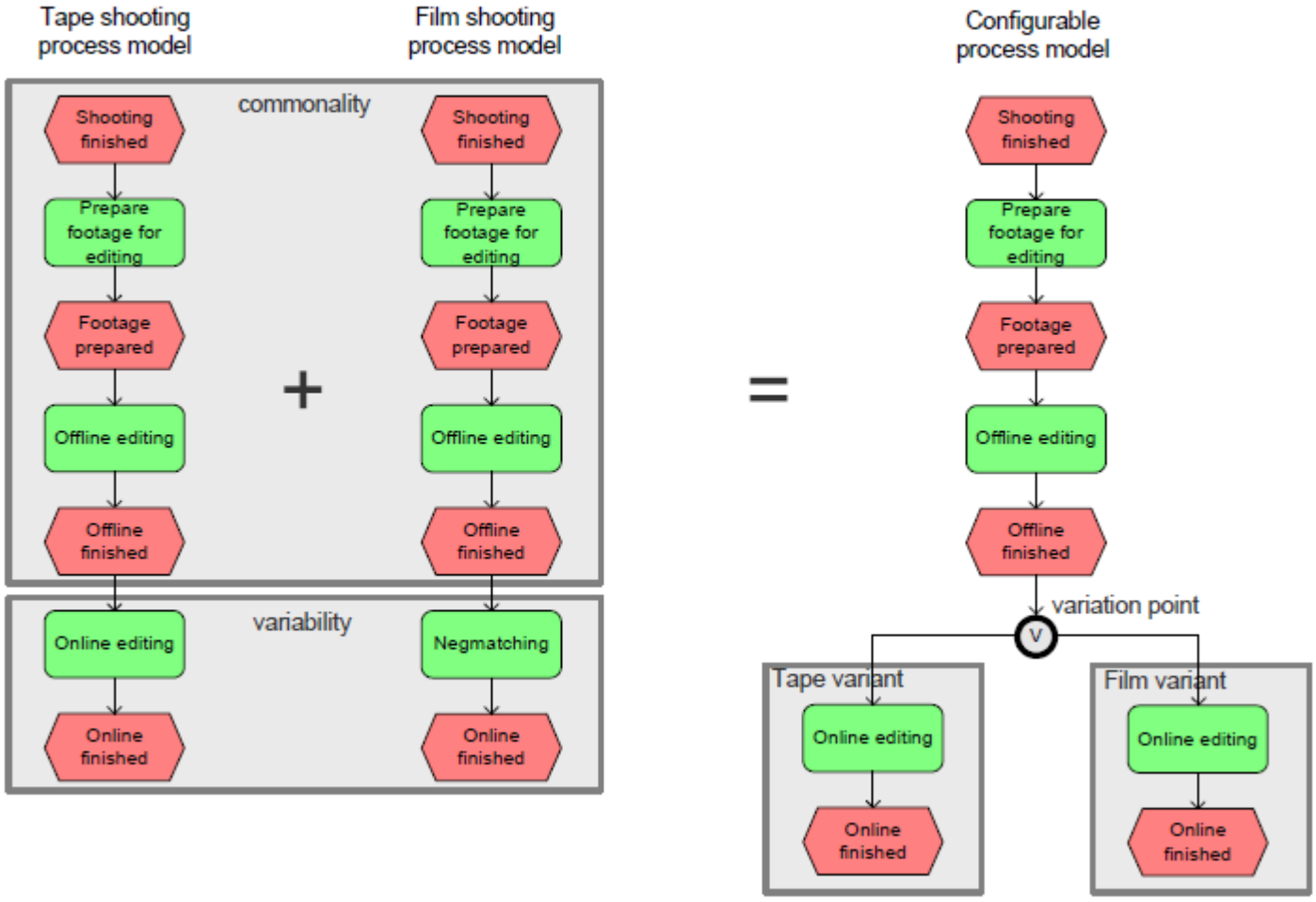


Figure 6: Example of an optional relationship

Taken from Modeling variability by UML use case diagrams by Von der Massen et al.

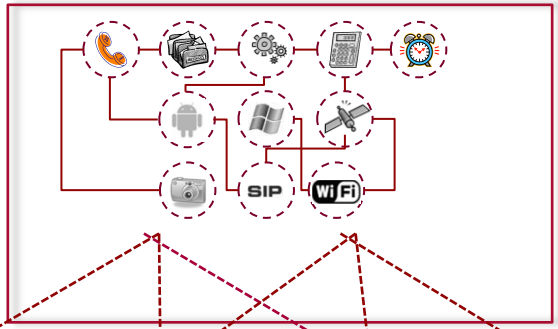
Inside the model

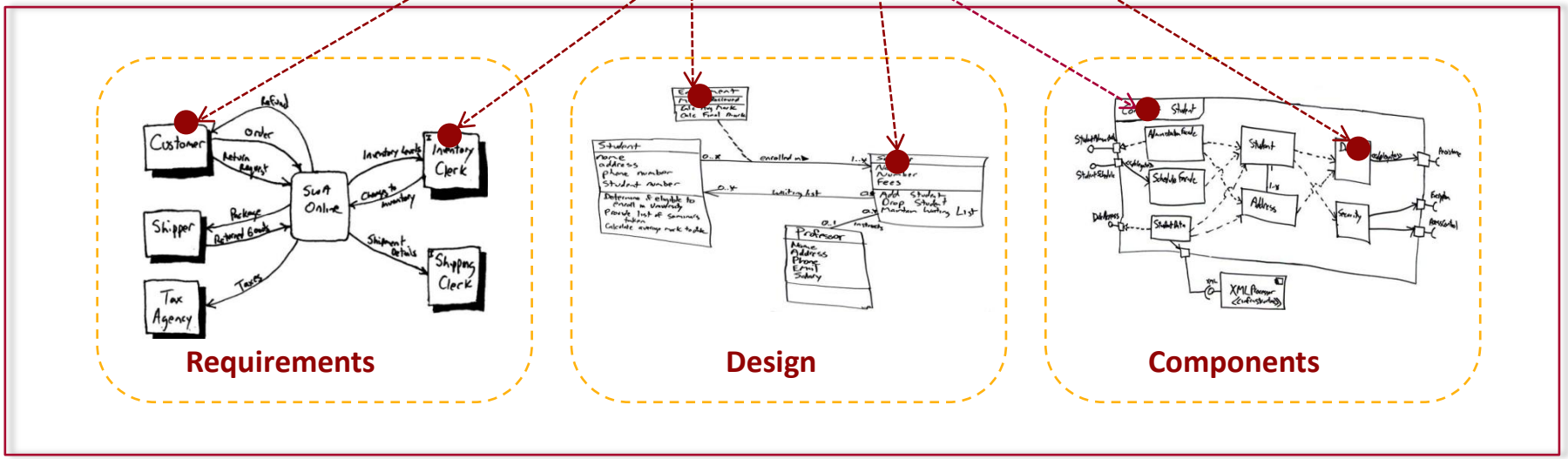


Taken from La Rosa et al. (2010) Configurable multi-perspective business process models. Information Systems

Outside the model

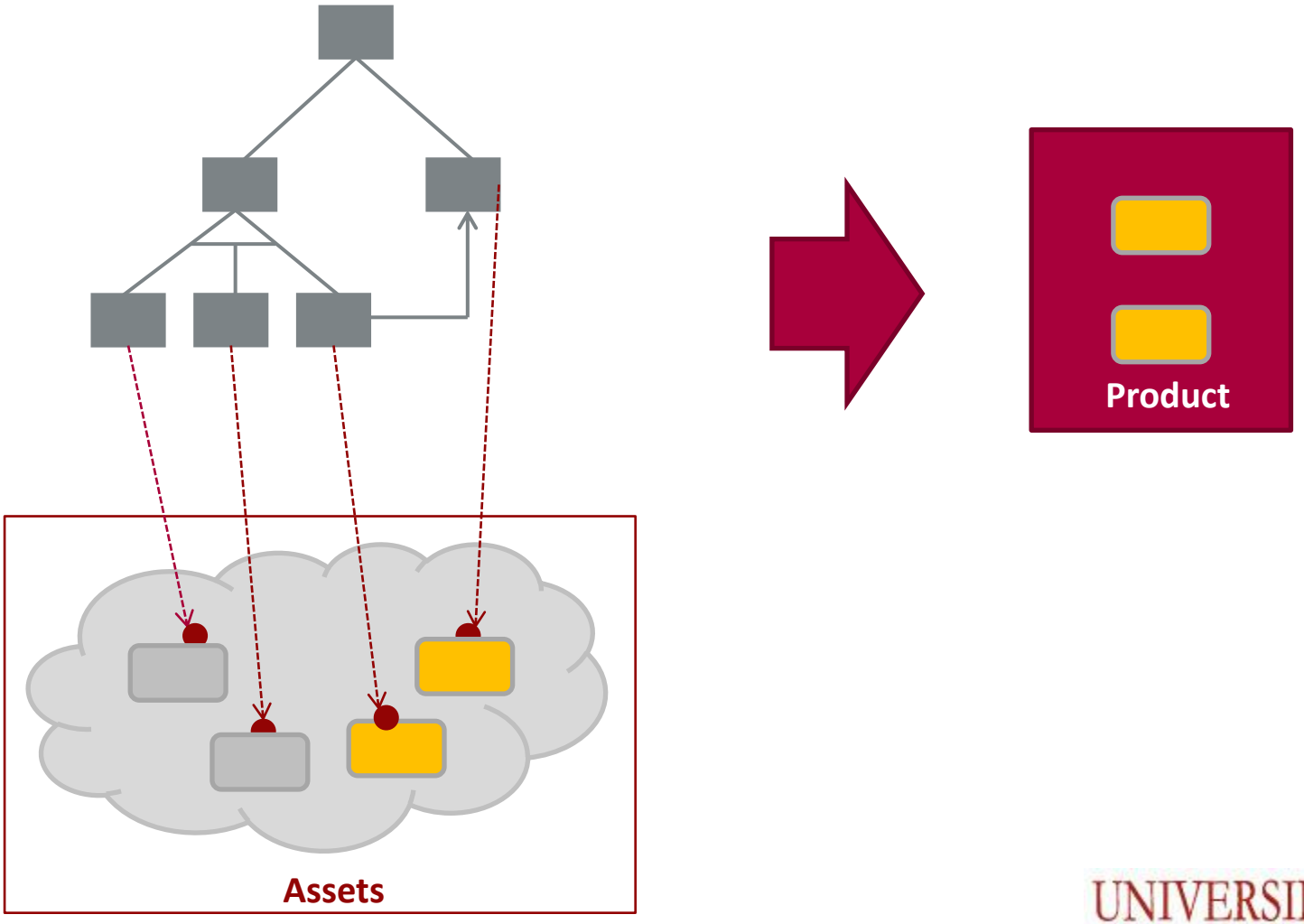
Variability Model



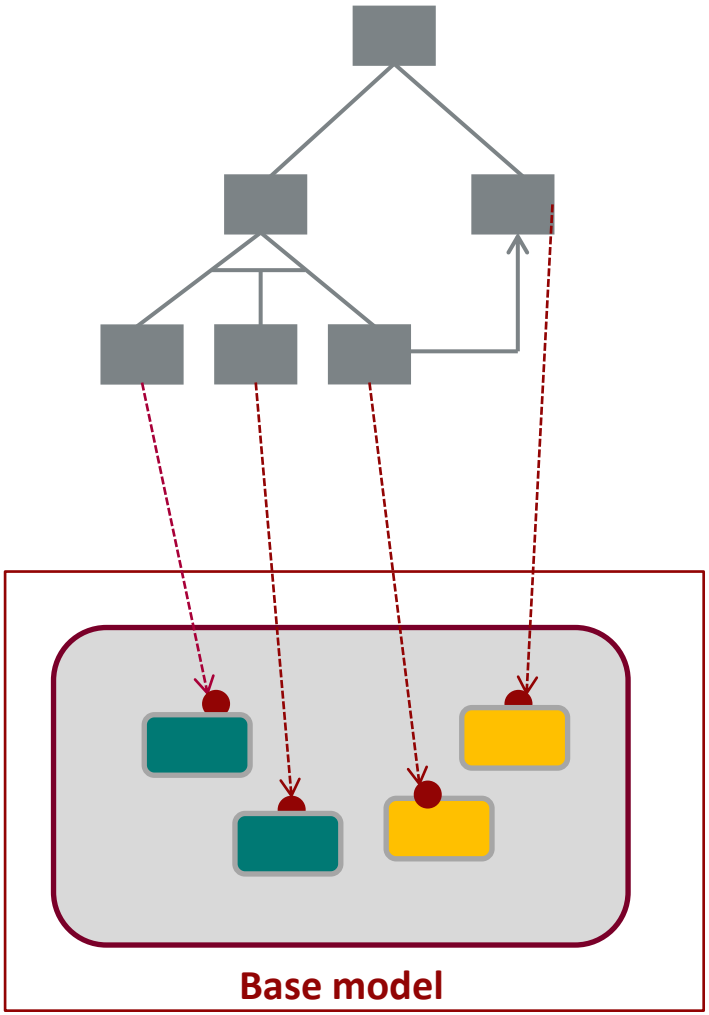


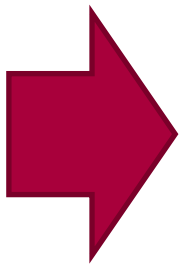
Base models

Positive variability

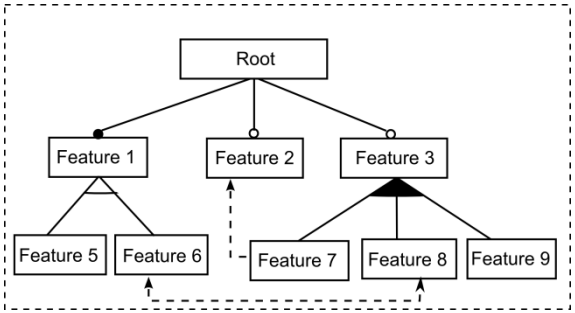


Negative variability

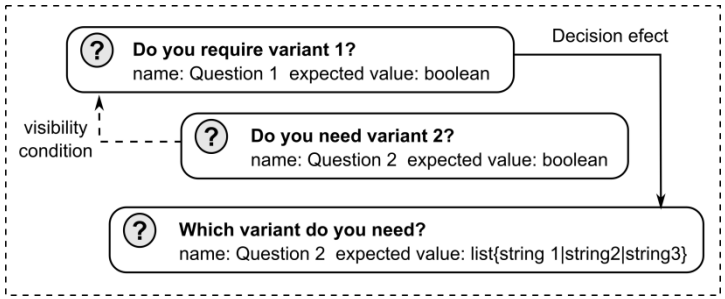




How to model variability



Feature modelling

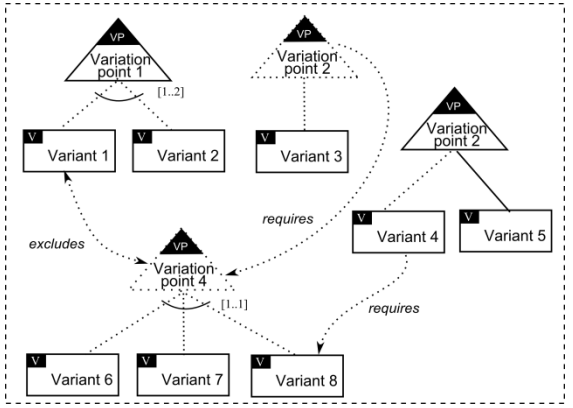


Decision modelling

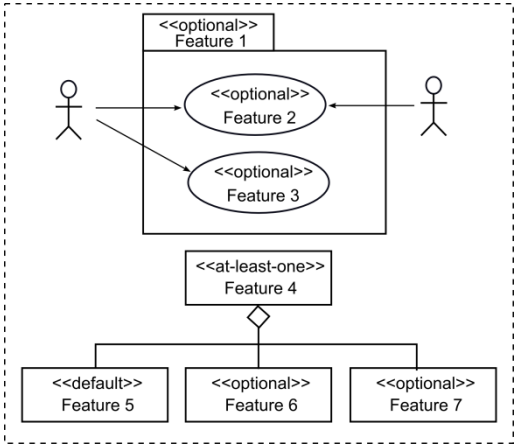
Techniques

Ad-hoc solutions: tables, textual docs, ...

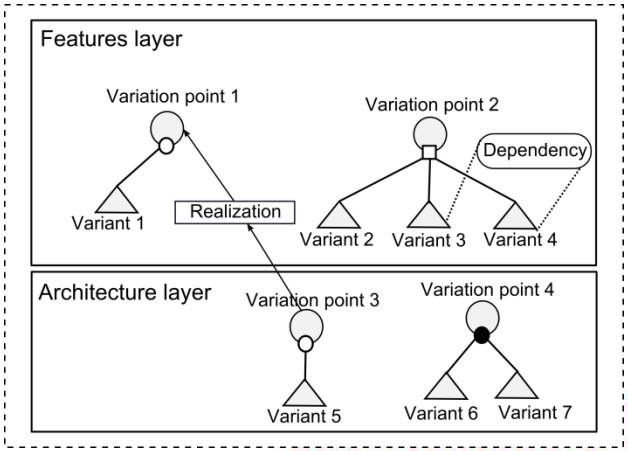
Orthogonal variability modelling



UML-based



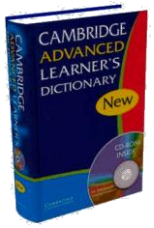
COVAMOF



Ð SEVILLA

How to specify a particular product?

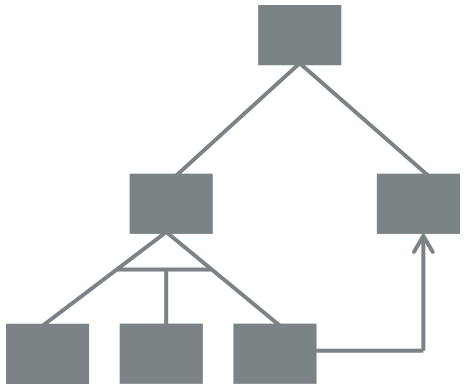
FEATURE



"An important part of something"

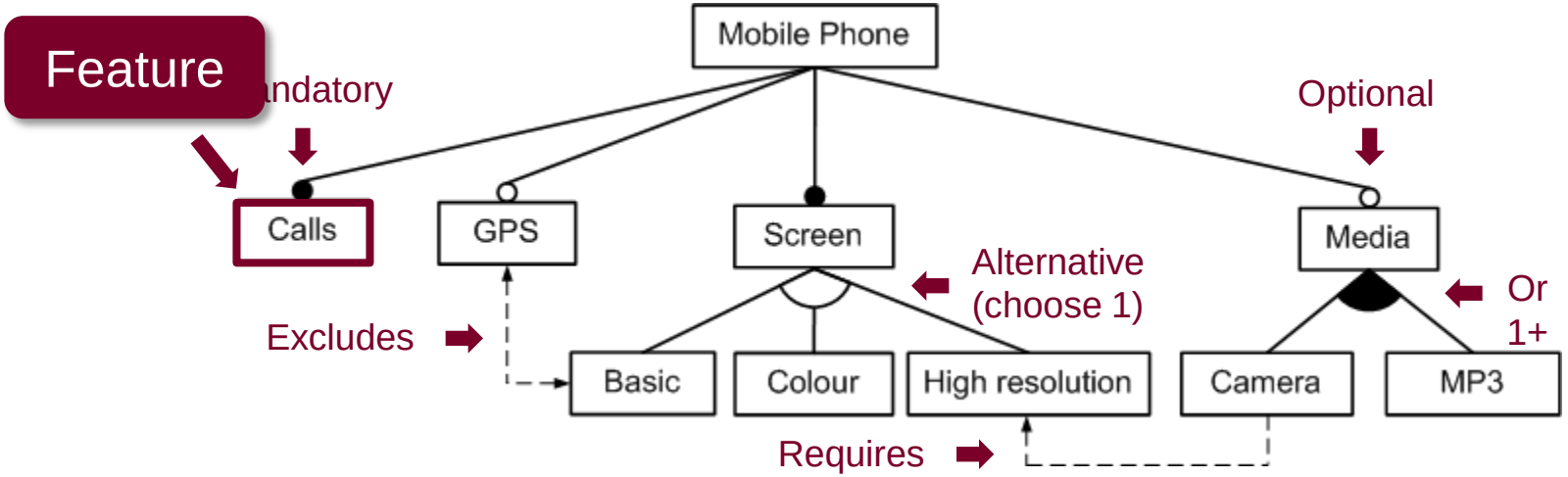
"A prominent or distinctive characteristic of a software system"

How to specify an SPL?

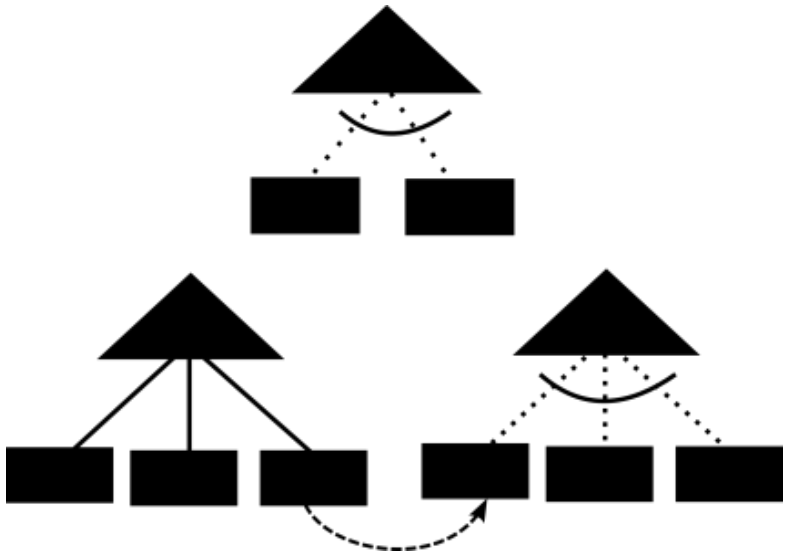


“Feature Model: A hierarchically arranged set of features to represent all possible products of an SPL”

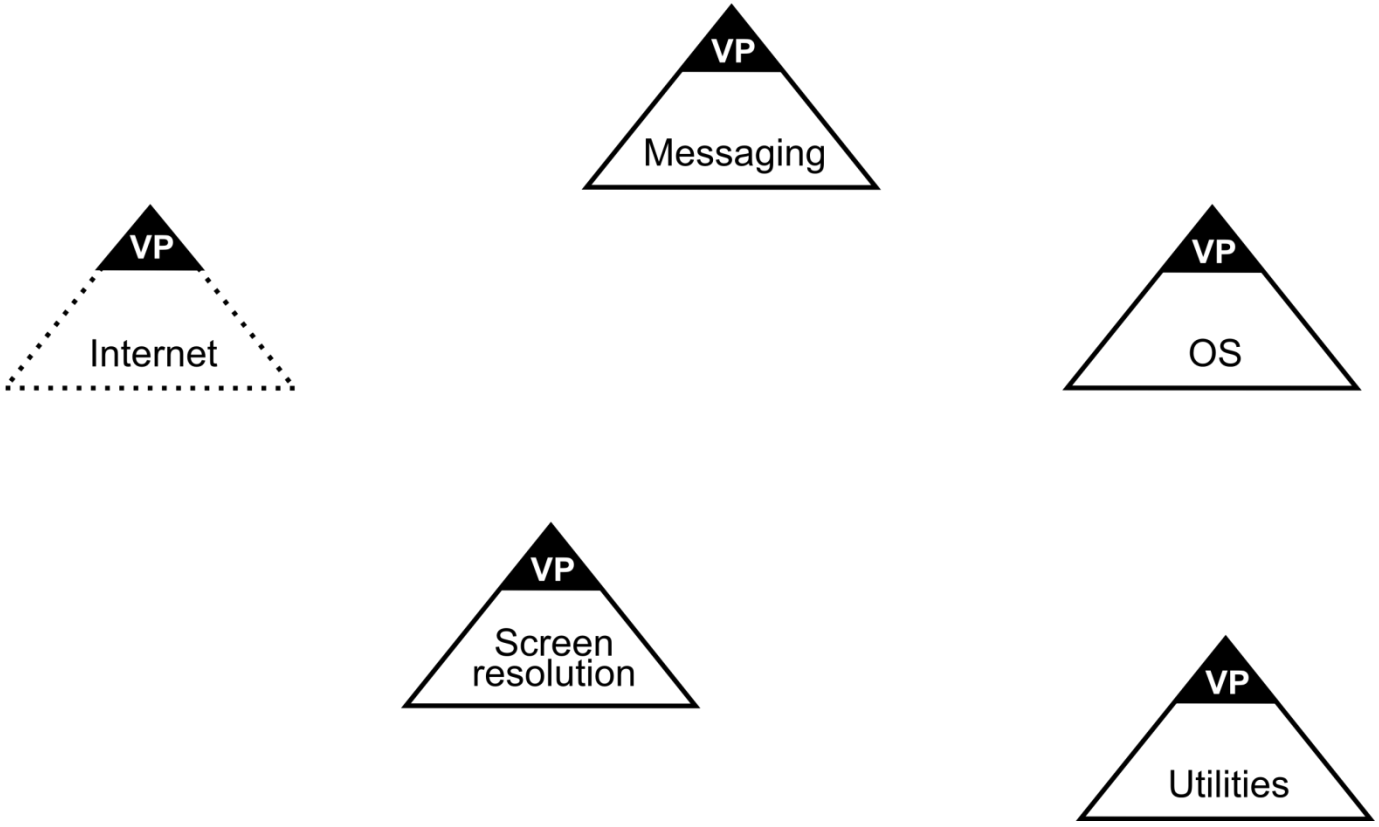
Feature models



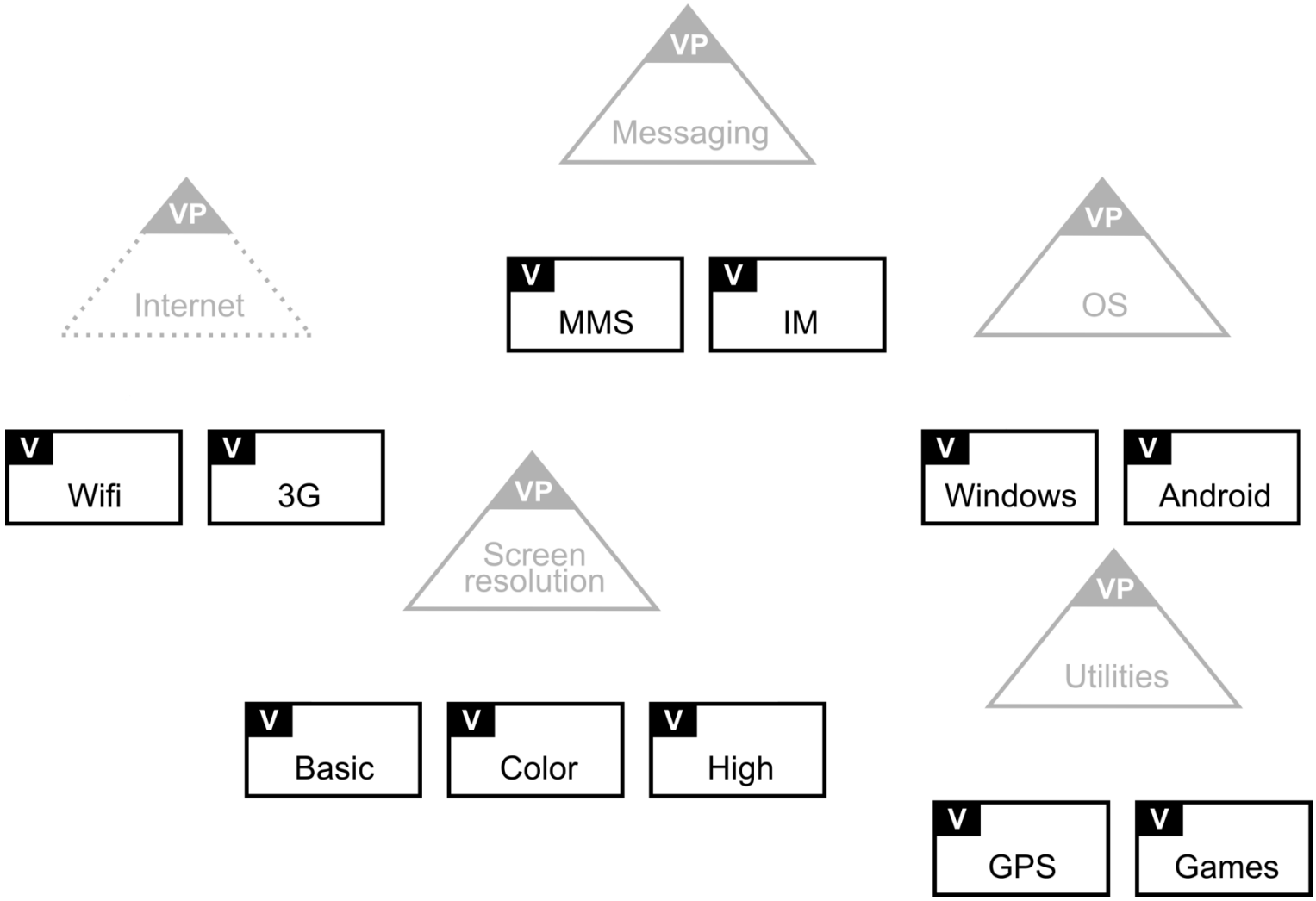
OVM



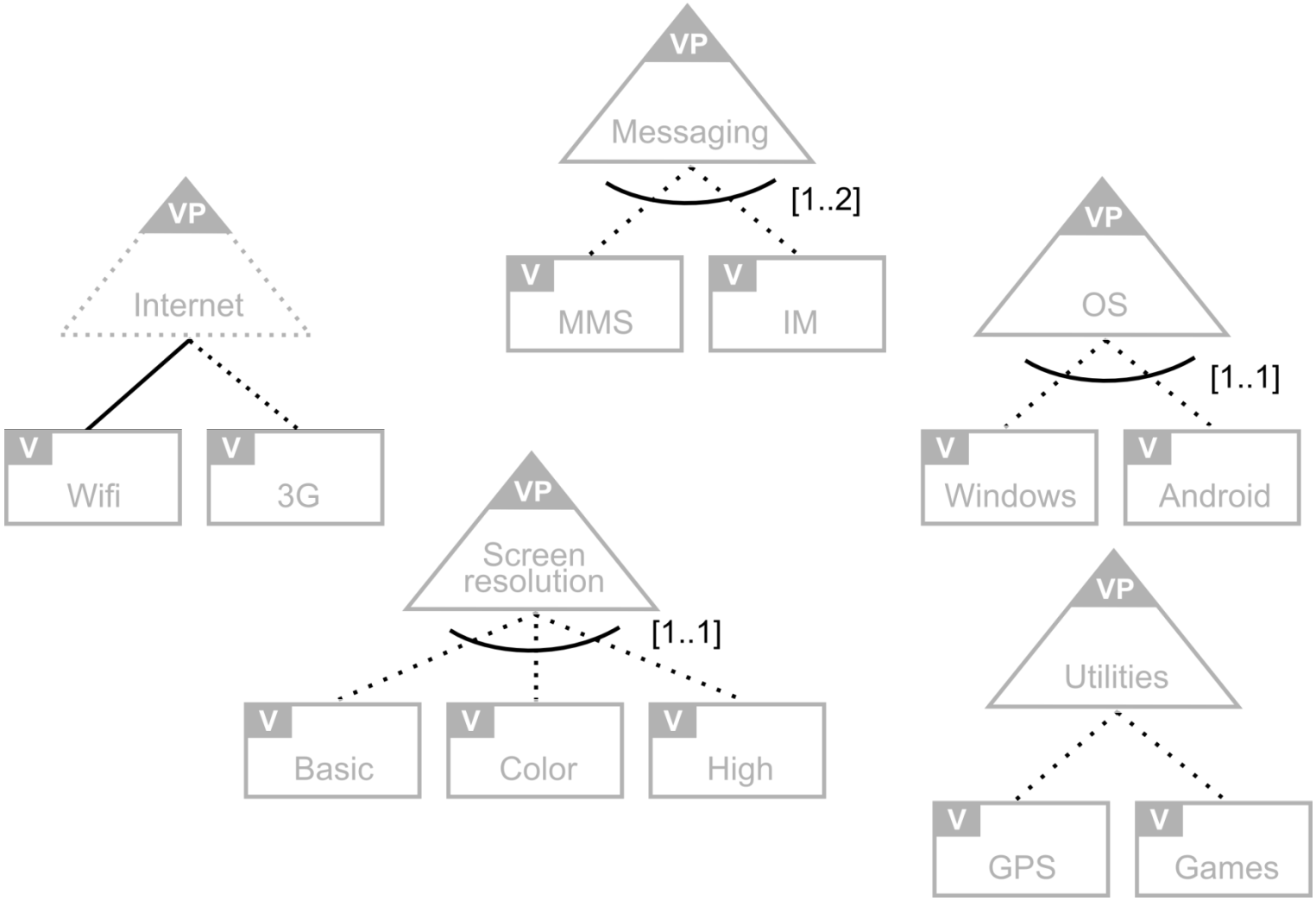
Variation Points



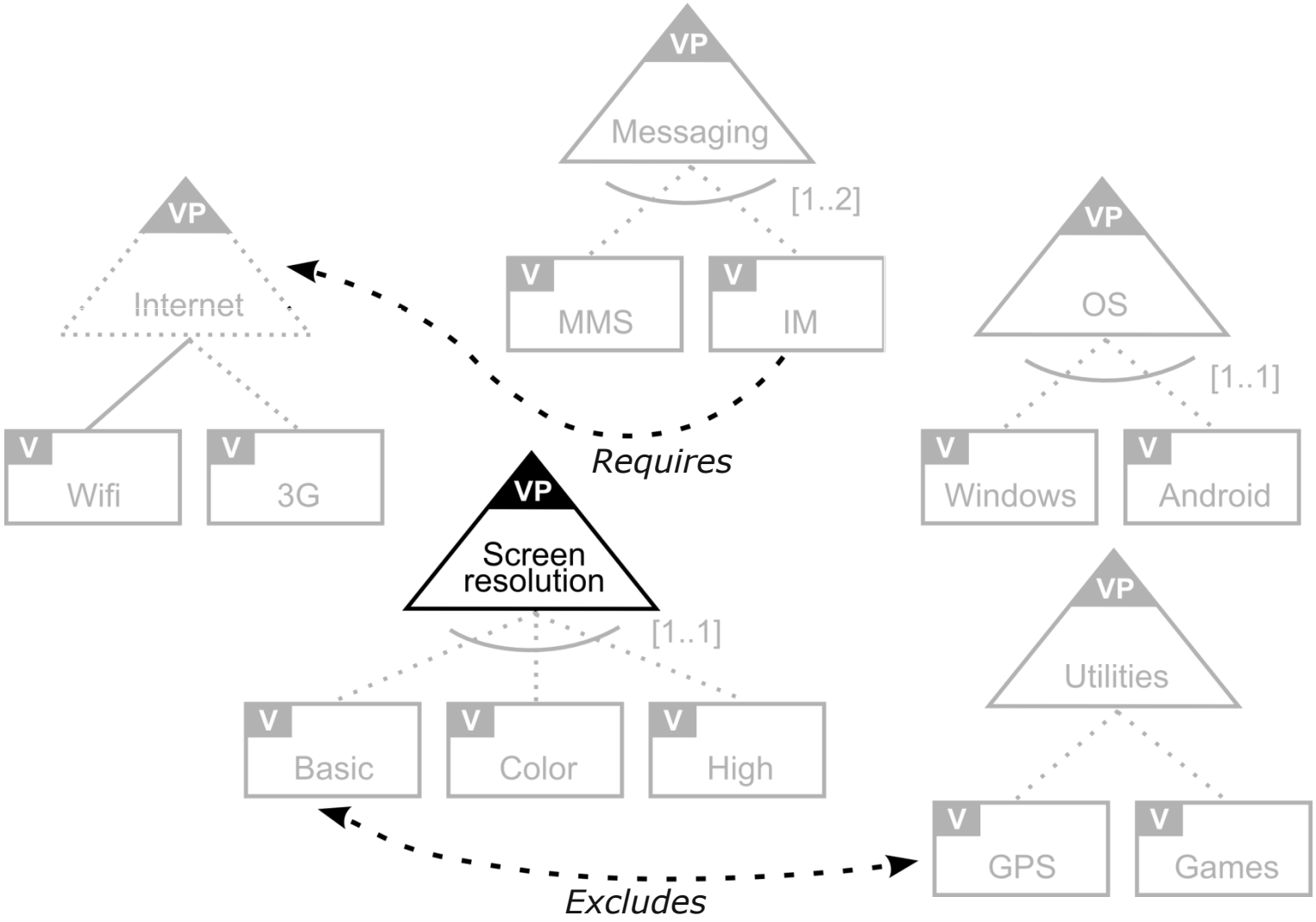
Variants


VP
Messaging
VP
Internet
V
MMS
V
IM
VP
OS
V
Wifi
V
3G
VP
Screen resolution
V
Windows
V
Android
VP
Utilities
V
Basic
V
Color
V
High
V
GPS
V
Games

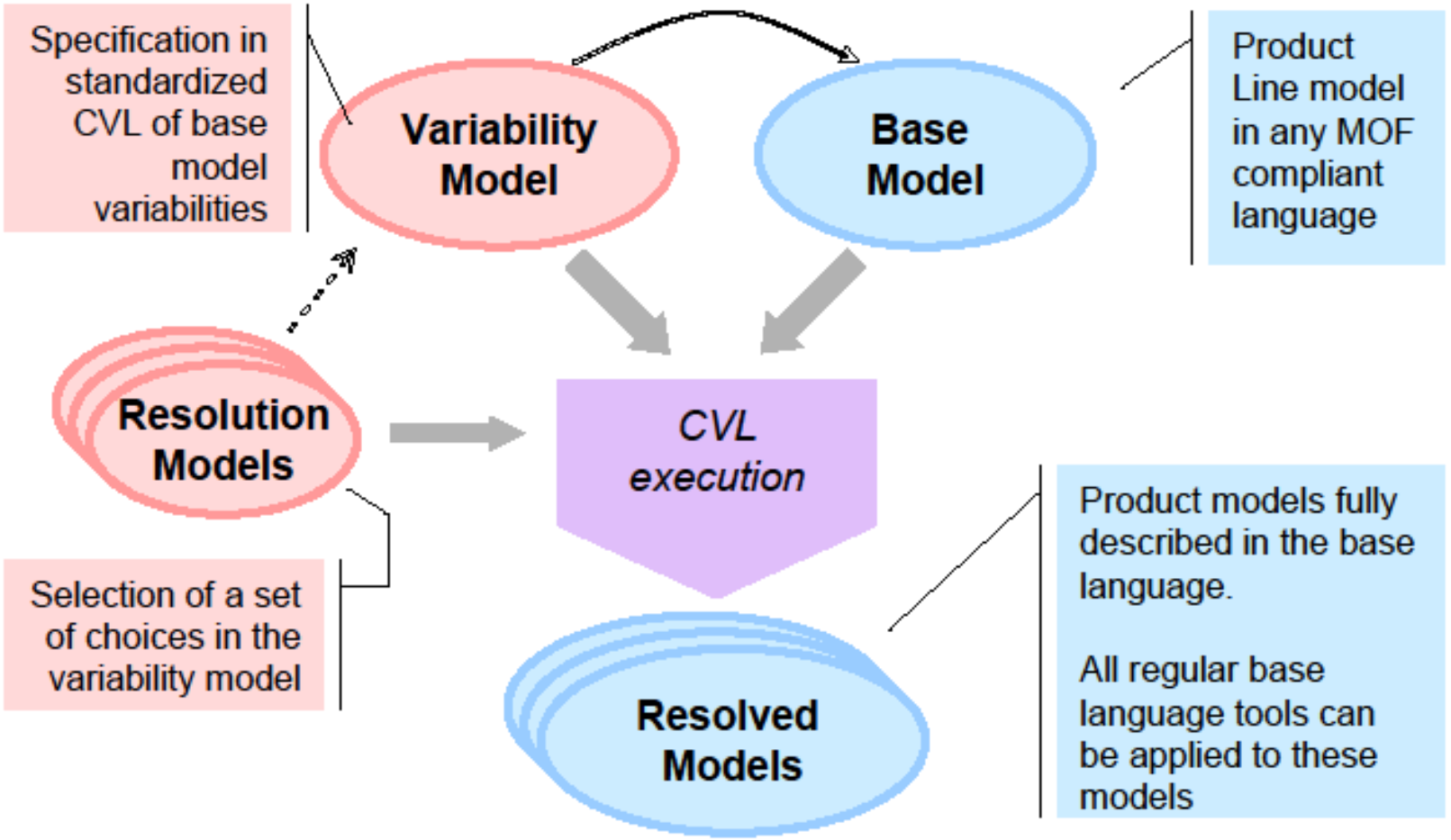
Relationships



Constraints



CVL overview and terms



Taken from CVL Tutorial by Øystein Haugen et al.

UNIVERSIDAD Đ SEVILLA

How do/could you model variability?

Conclusions

- SPL is a *new* software production paradigm
- Variability management is essential

Parte II

UNIVERSIDAD Đ SEVILLA

Variability implementation

- Templates
- Compilación
- Preprocesamiento
- Cargadores de clases
- Modularización

Variabilidad

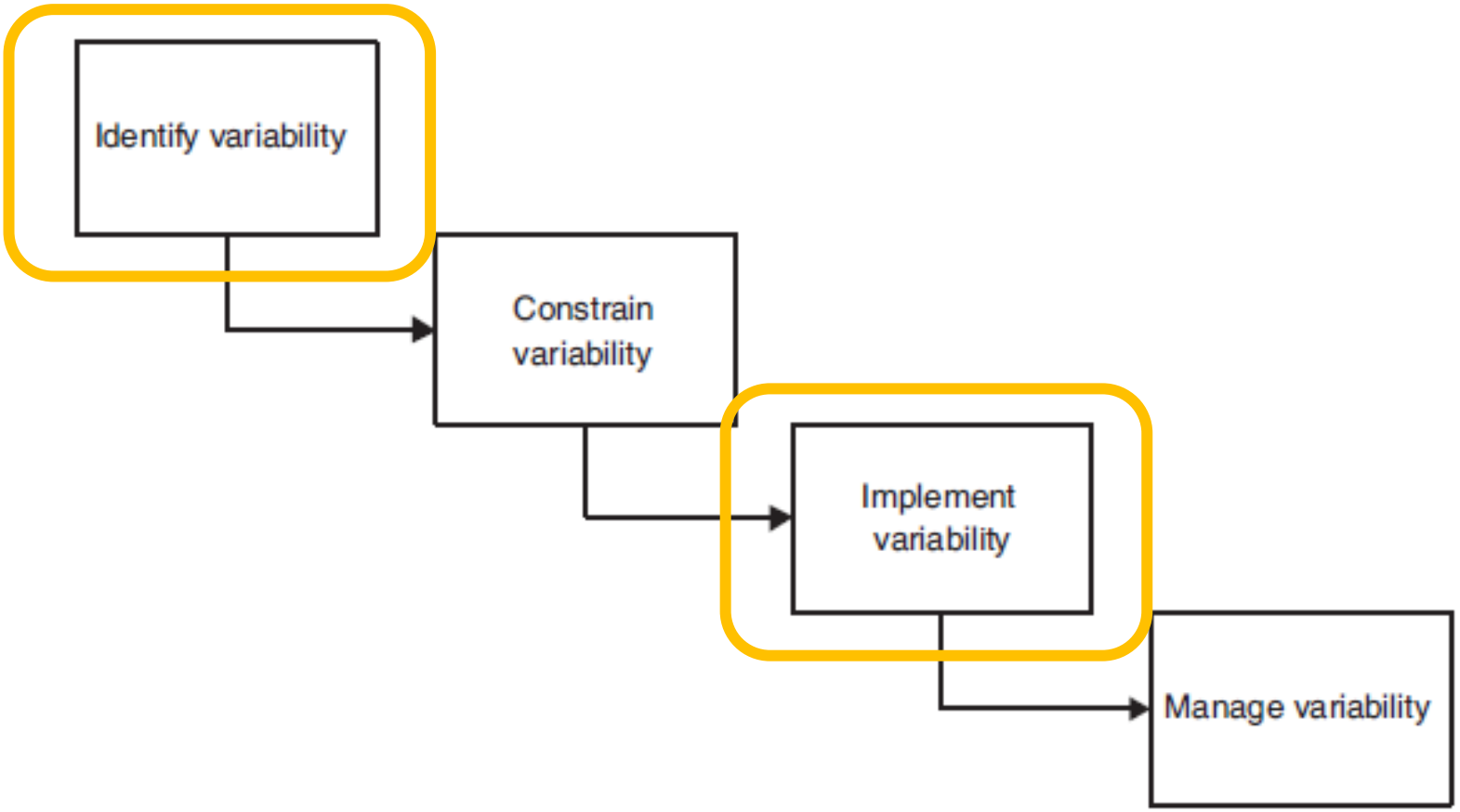
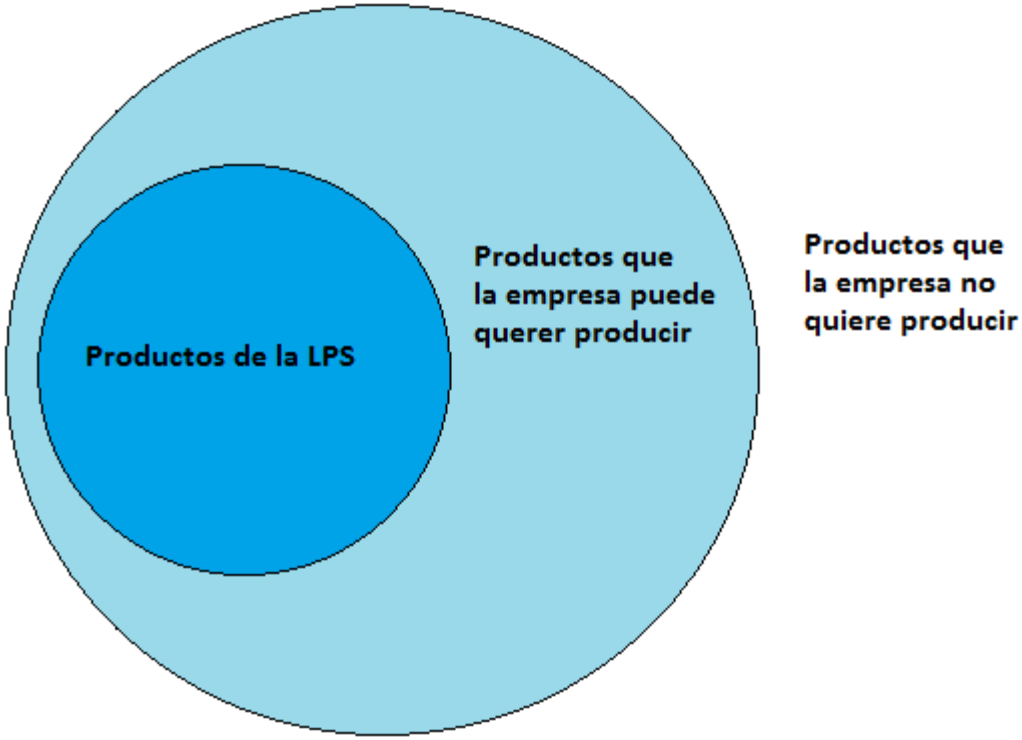


Figure 1. Steps for introducing variability.

El alcance de una SPL



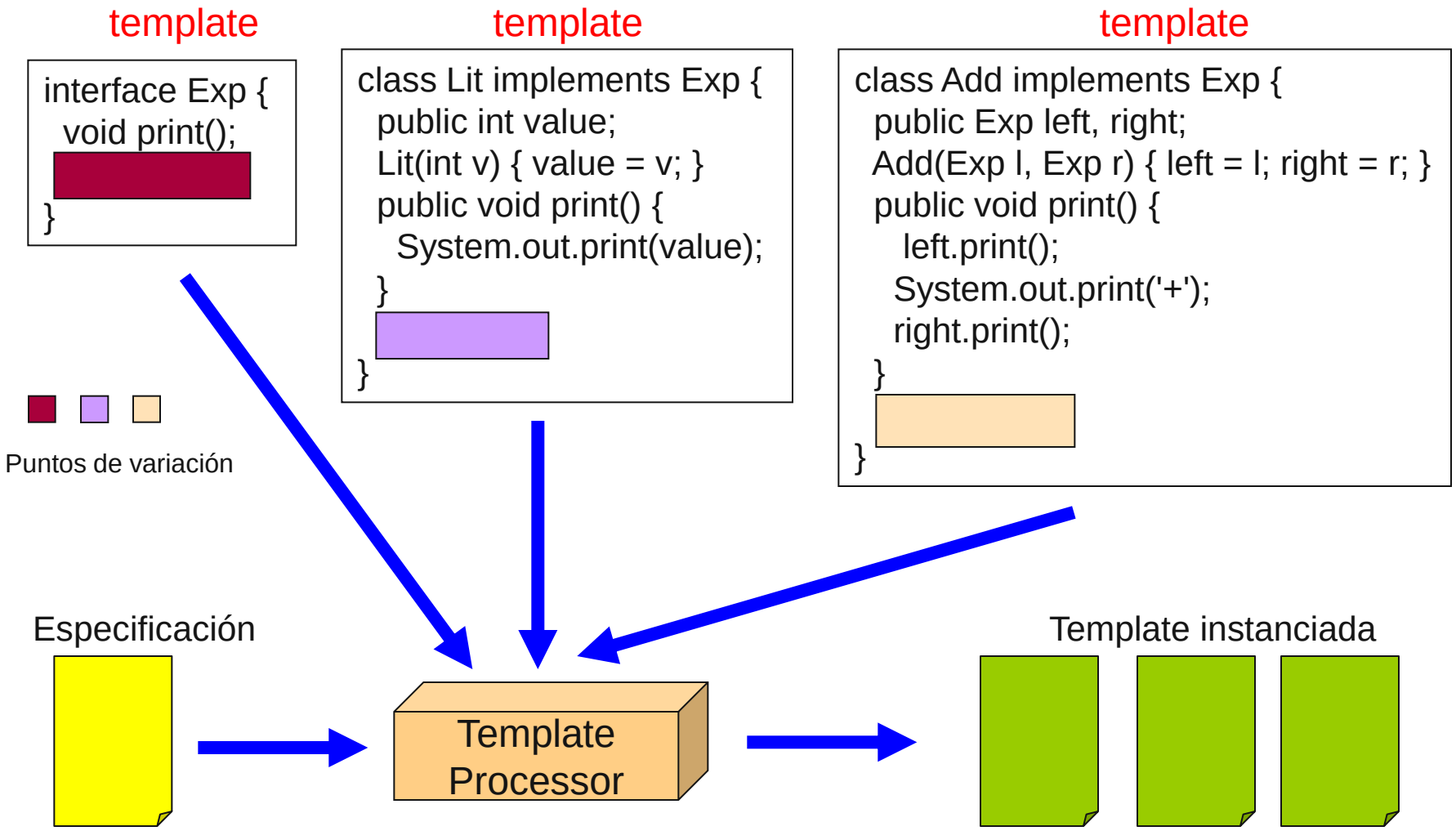
Implementación de la variabilidad

- **Plantillas**
- Compilación
- Preprocesamiento
- Cargadores de clases
- Modularización
- DSL

¿Cómo implementar la variabilidad?

- Técnicas basadas en motores de plantillas (*template engines, frames*)
- Se pueden definir como motores de “copy & paste”
- Distintas alternativas

Proceso general



XVCL

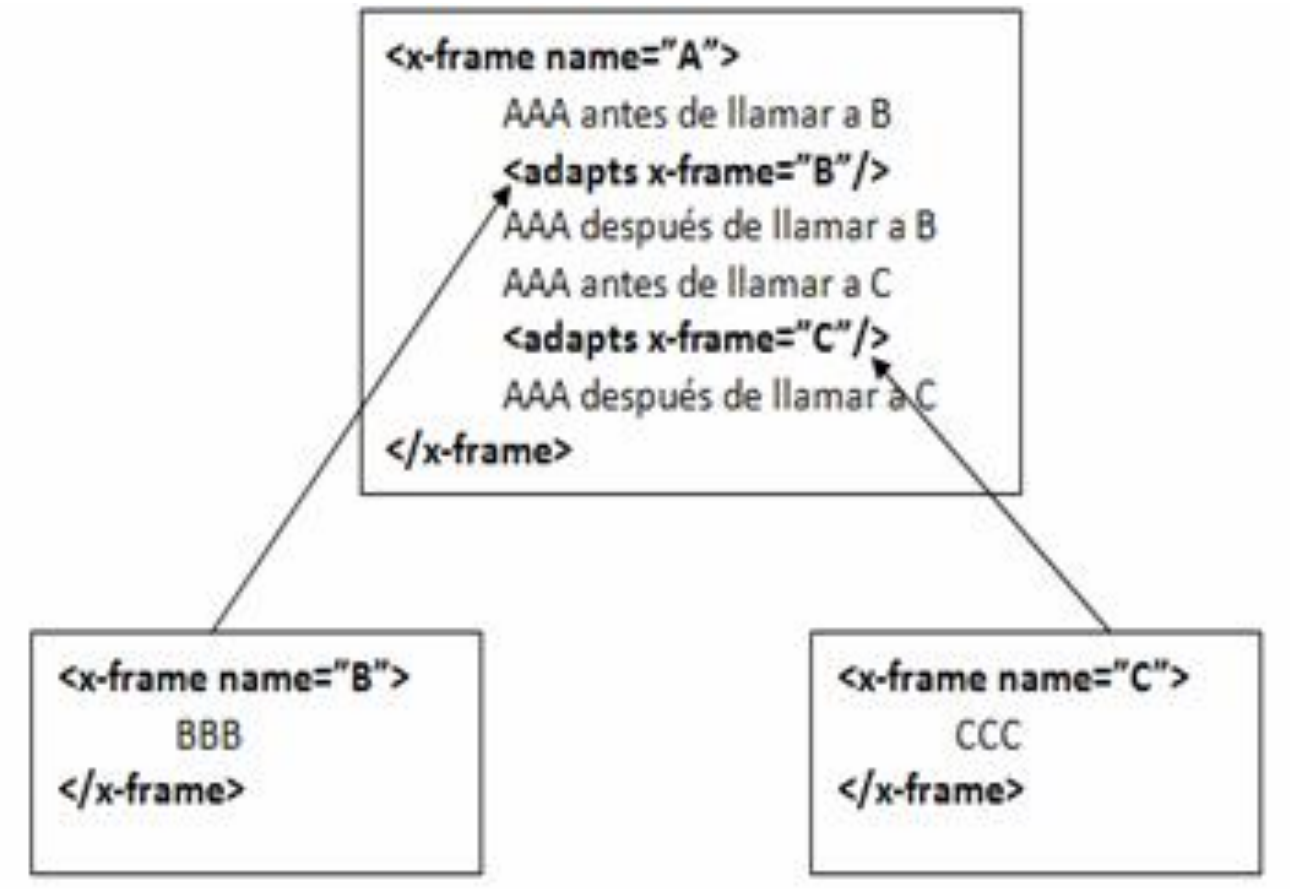
- XML-based Variant Configuration Language
- Stanislaw Jarzabek (Universidad de Singapur)
- Basado en tecnología de marcos (*frames*)
- Complemento a lenguaje de programación convencional

XVCL

- Funcionamiento:
  - X-frames
  - X-framework
  - Puntos de variación
  - Comandos

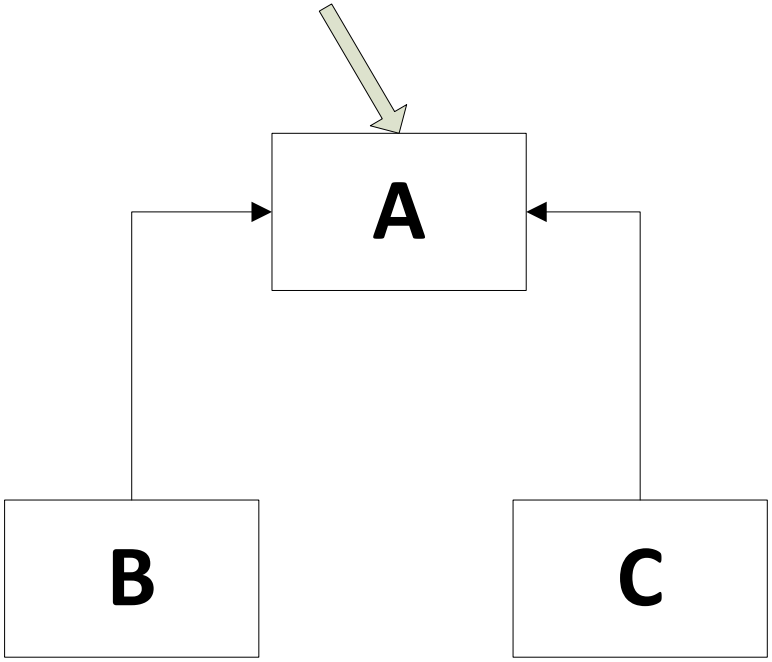
XVCL

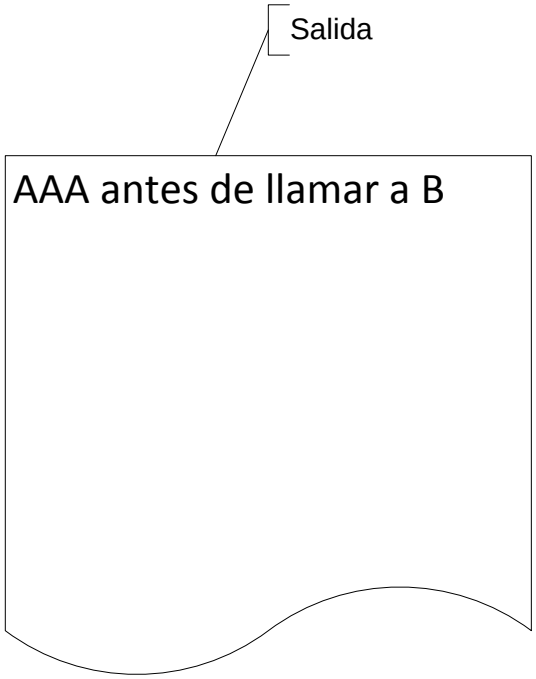
- Tenemos el siguiente x-framework:



XVCL

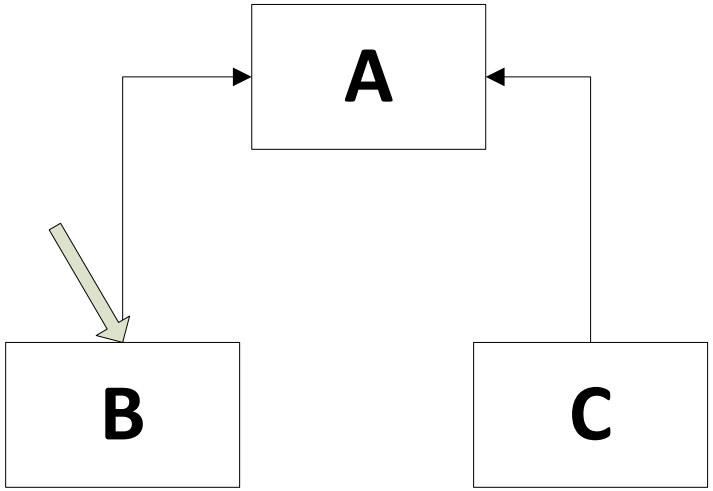
- Ejecución – paso 1





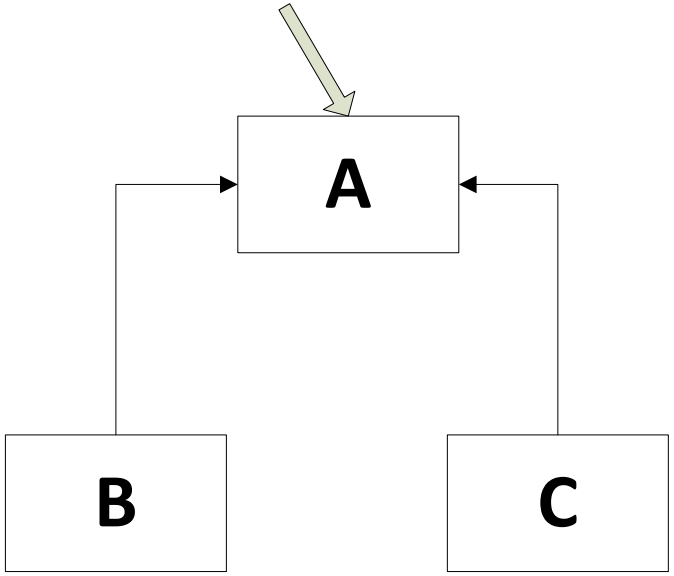
XVCL

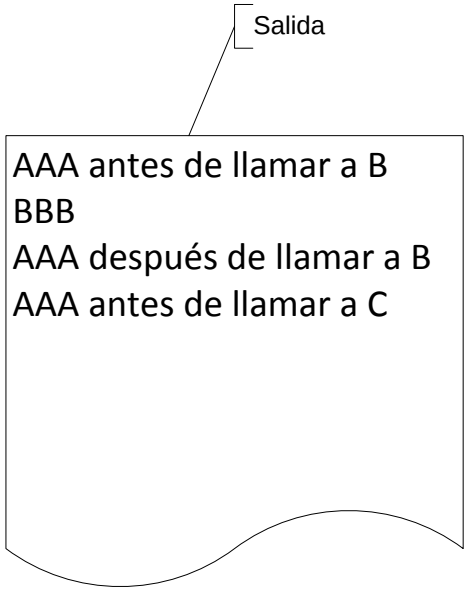
- Ejecución – paso 2



XVCL

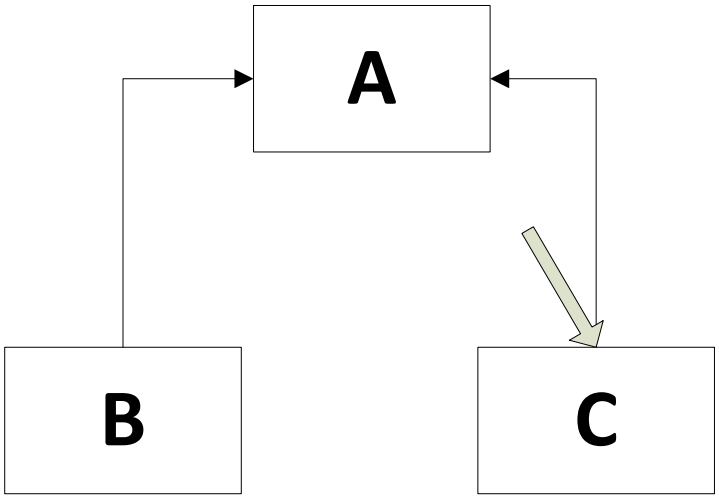
- Ejecución – paso 3

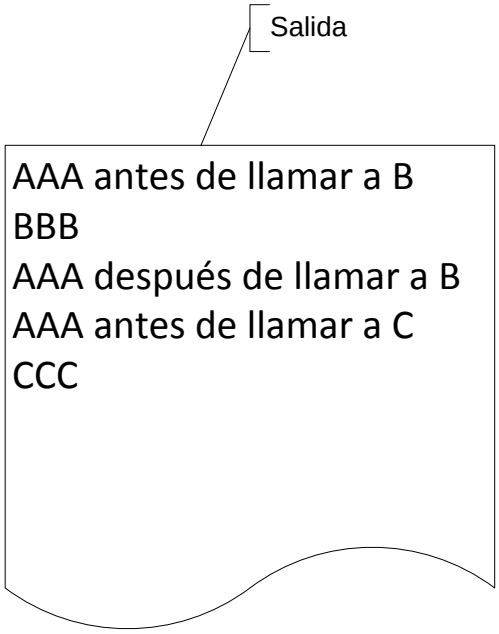




XVCL

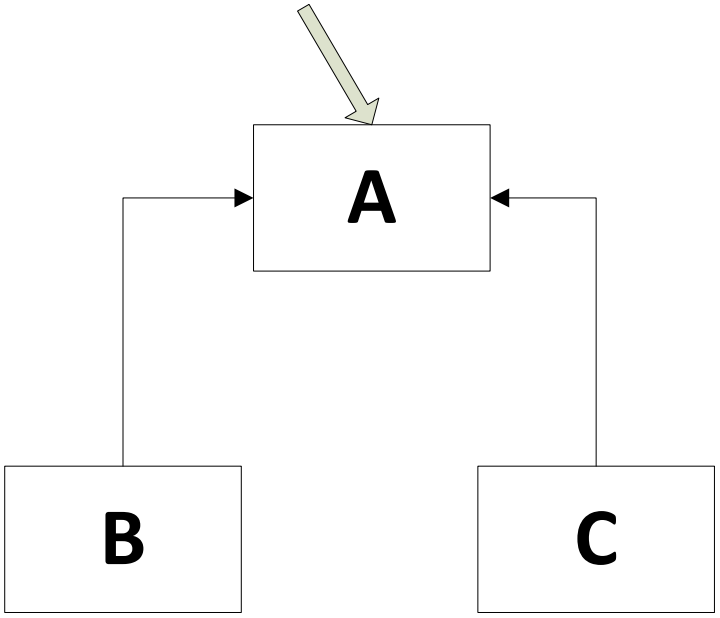
- Ejecución – paso 4

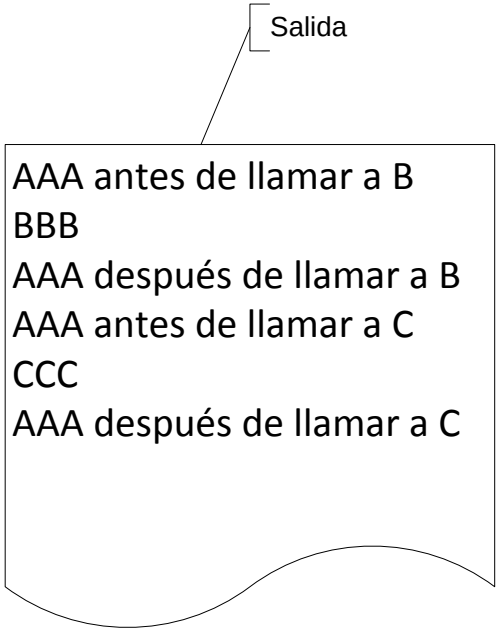




XVCL

- Ejecución – paso 5



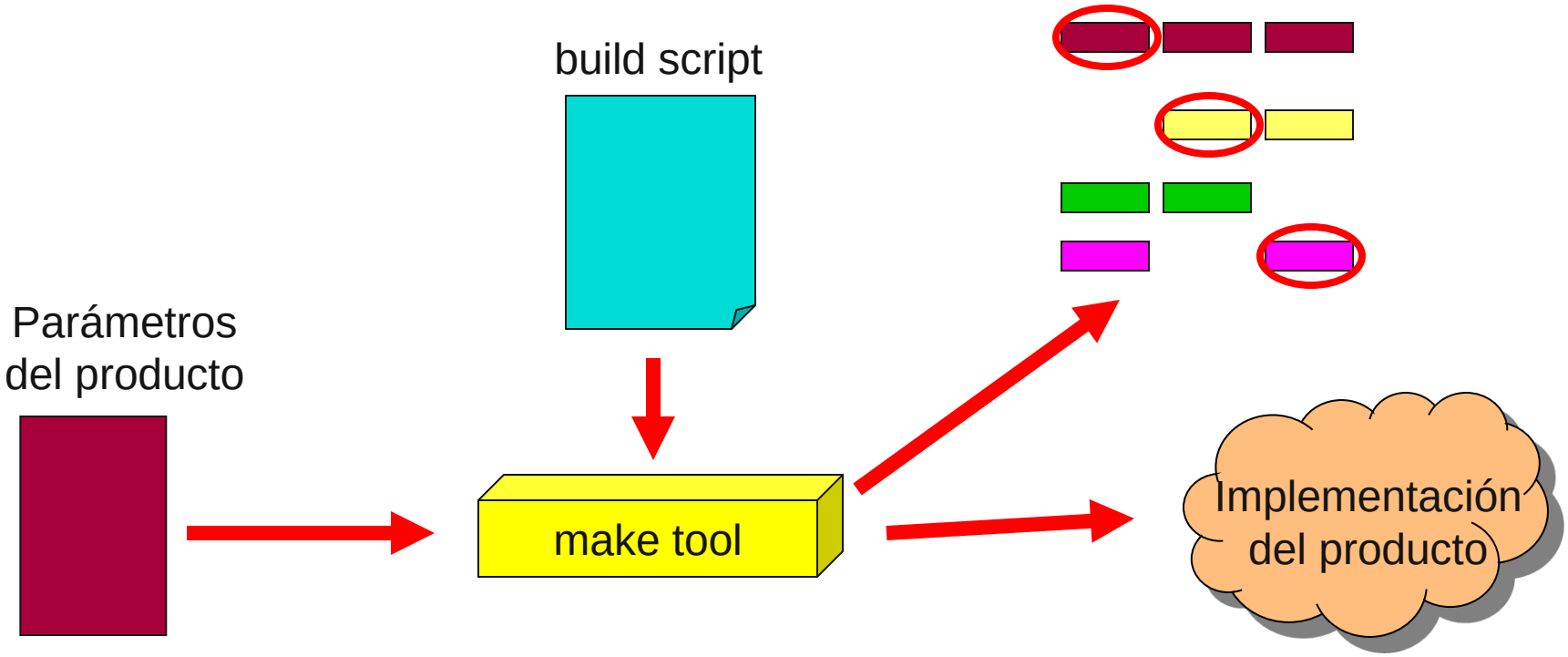


Implementación de la variabilidad

- Plantillas
- **Compilación**
- Preprocesamiento
- Cargadores de clases
- Modularización

Técnicas de compilación

- Algunas herramienta como make, ant, maven,... se pueden usar para implementar la variabilidad en tiempo de compilación



Técnicas de compilación

- Ventajas
  - Son fáciles de implementar y útiles cuándo hay puntos simples de variación

- Desventajas
  - Implementar todas las variantes de una SPL en un solo script puede resultar muy tedioso si no directamente imposible, especialmente cuándo haya muchas variantes

Implementación de la variabilidad

- Plantillas
- Compilación
- **Preprocesamiento**
- Cargadores de clases
- Modularización

Preprocesamiento

- Directivas del compilador para incluir o no ciertos fragmentos de código
- Es la primera y más antigua técnica para soportar variabilidad
- Está soportada por lenguajes como C, pero también se pueden introducir en Java
- Ventajas
  - Simple, viene por defecto en algunos lenguajes y compiladores.
- Desventajas
  - Parecida a las desventajas de las plantillas, incluso más graves.

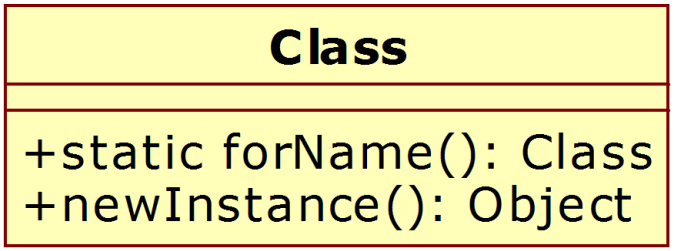
Implementación de la variabilidad

- Plantillas
- Compilación
- Preprocesamiento
- **Cargadores de clases**
- Modularización

Cargadores de clases

- Classloader
  - Forma parte del JRE estándar de Java
  - Se cargan de manera dinámica clases en la JVM

- Se proporciona un cargador de clases por defecto
  - Este cargador se puede refinar y/o extender.




```
Driver dBDriver = (Driver) Class.forName(driverName)
    .newInstance();
DriverManager.registerDriver(dBDriver);
```

Cargadores de clases

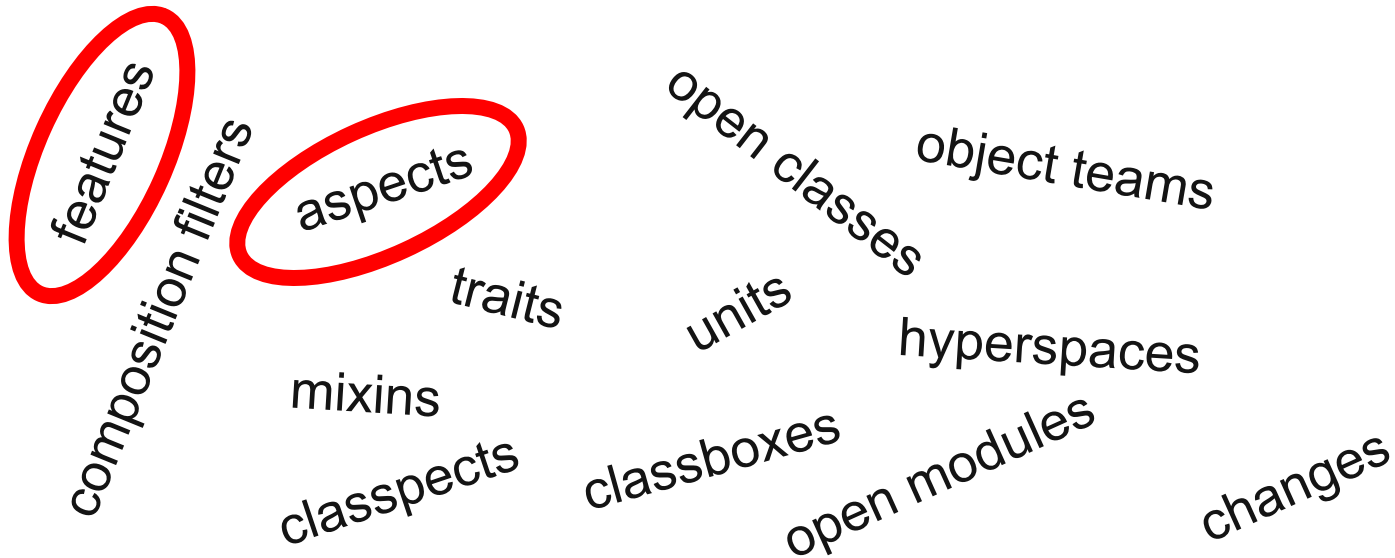
- Ventajas
  - Son una manera flexible de implementar variabilidad en tiempo de ejecución
- Desventajas
  - Implementar un cargador de clases no es una tarea trivial
  - Solo funciona para programas escritos en lenguajes que soporten reflexión
  - La legibilidad del código no es trivial

Implementación de la variabilidad

- Plantillas
- Compilación
- Preprocesamiento
- Cargadores de clases
- **Modularización**

Técnicas de modularización

- Hay muchos lenguajes de programación que se proponen para solventar los problemas de los lenguajes OO
- El punto clave es la granularidad
  - Se parte de la constatanción de que una clase es muy pequeña para construir un sistema si a este sistema hay que añadirle incrementos de funcionalidad.



Bibliografía

JOURNAL OF OBJECT TECHNOLOGY
Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering. ©JOT, 2009

Vol. 8, No. 6, September–October 2009

Virtual Separation of Concerns – A Second Chance for Preprocessors

Christian Kästner, School of Computer Science, University of Magdeburg, Germany
Sven Apel, Department of Informatics and Mathematics, University of Passau,

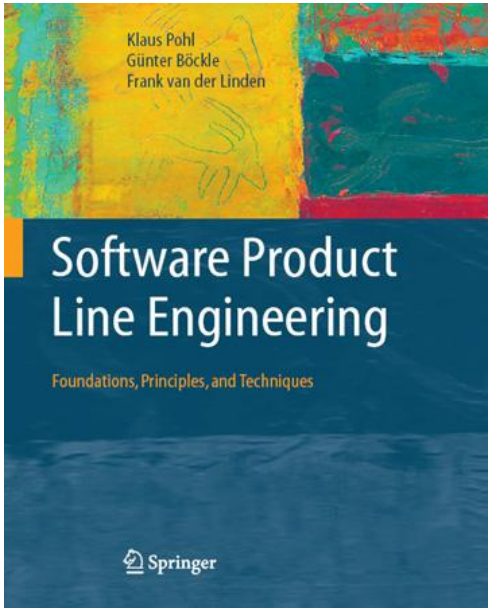
SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2005; **35**:705–754
Published online 1 April 2005 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/spe.652

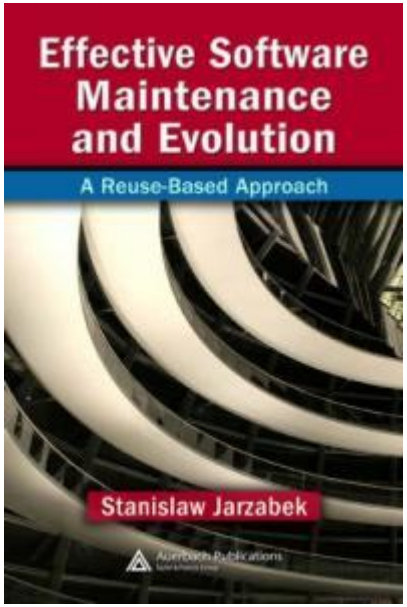
A taxonomy of variability realization techniques‡

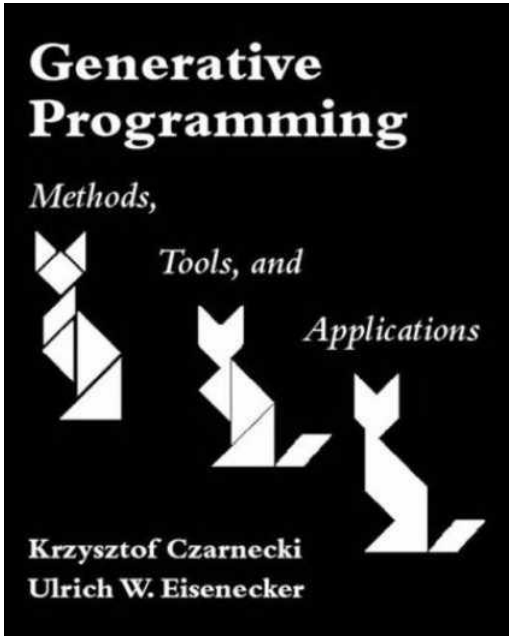
Mikael Svahnberg[1,*,†], Jilles van Gurp[2] and Jan Bosch[3]

UNIVERSIDAD D SEVILLA

Bibliografía







Parte III

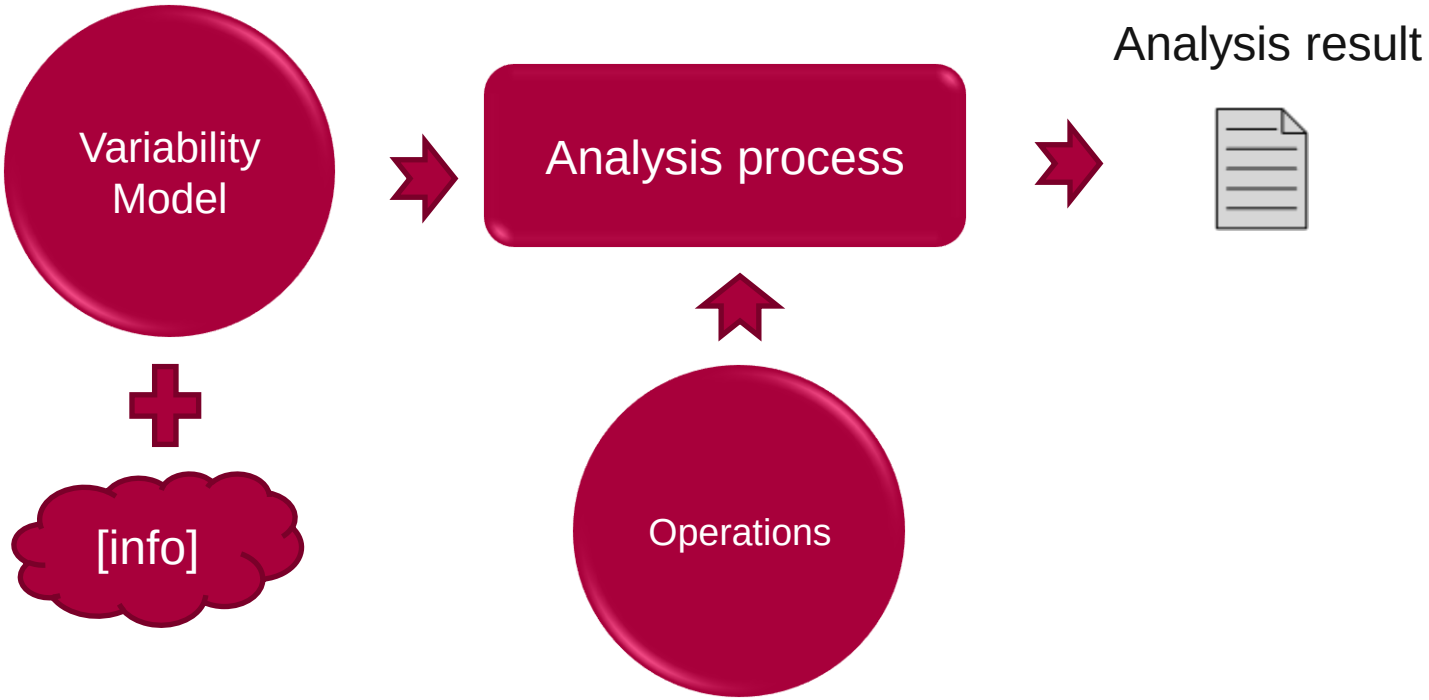
UNIVERSIDAD Ð SEVILLA

How to model variability

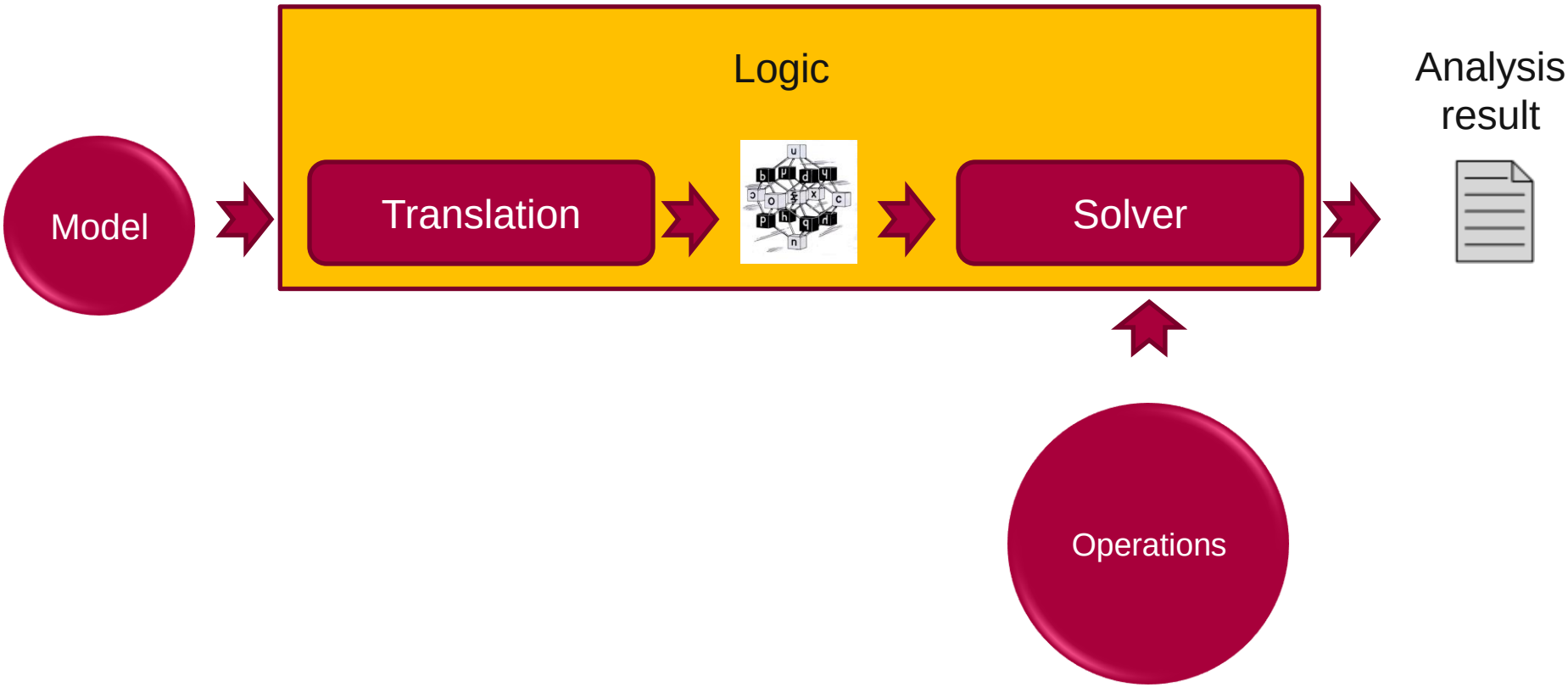
How to automatically analyse variability models

How to automatically analyse variability models

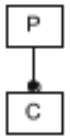
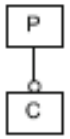
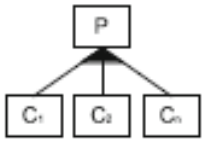
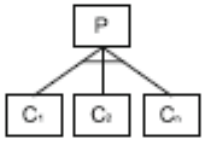
Computer-aided, extraction of useful information from feature models



Analysis process

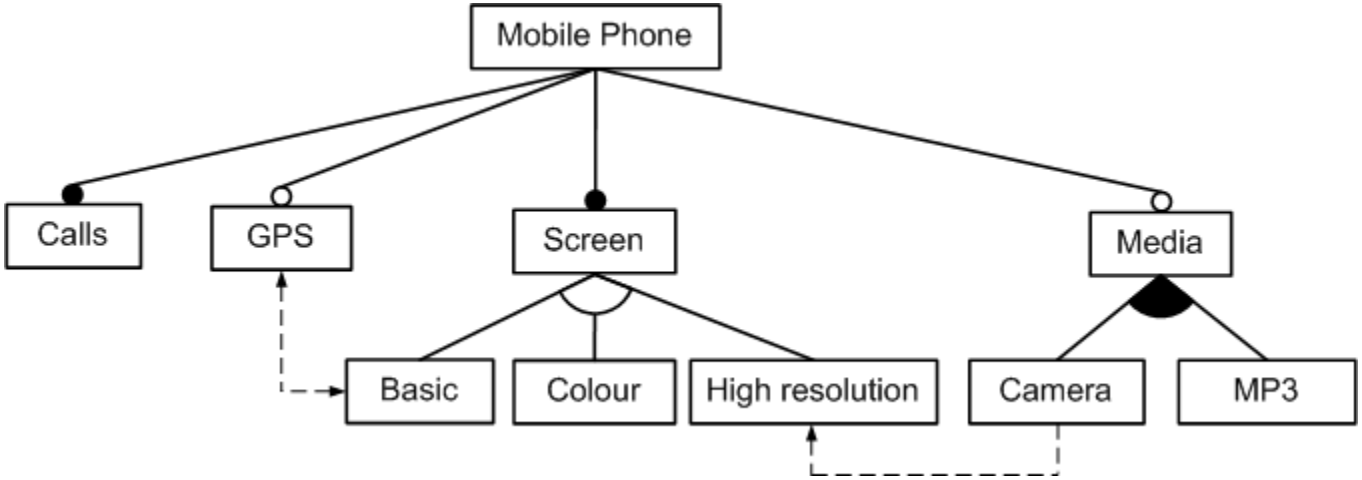


Mapping to Propositional Logic

Relationship		PL Mapping
MANDATORY		$P \leftrightarrow C$
OPTIONAL		$C \rightarrow P$
OR		$P \leftrightarrow (C_1 \vee C_2 \vee \ldots \vee C_n)$
ALTERNATIVE		$(C_1 \leftrightarrow (\neg C_2 \wedge \ldots \wedge \neg C_n \wedge P)) \wedge$ $(C_2 \leftrightarrow (\neg C_1 \wedge \ldots \wedge \neg C_n \wedge P)) \wedge$ $(C_n \leftrightarrow (\neg C_1 \wedge \neg C_2 \wedge \ldots \wedge \neg C_{n-1} \wedge P))$
IMPLIES		$A \rightarrow B$
EXCLUDES		$\neg(A \wedge B)$

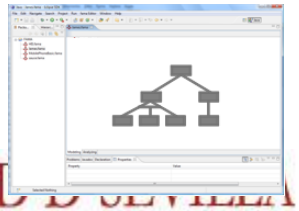
ERSIDAD Ð SEVILLA

Automated analysis of feature models:
Computer-aided extraction of information from FMs

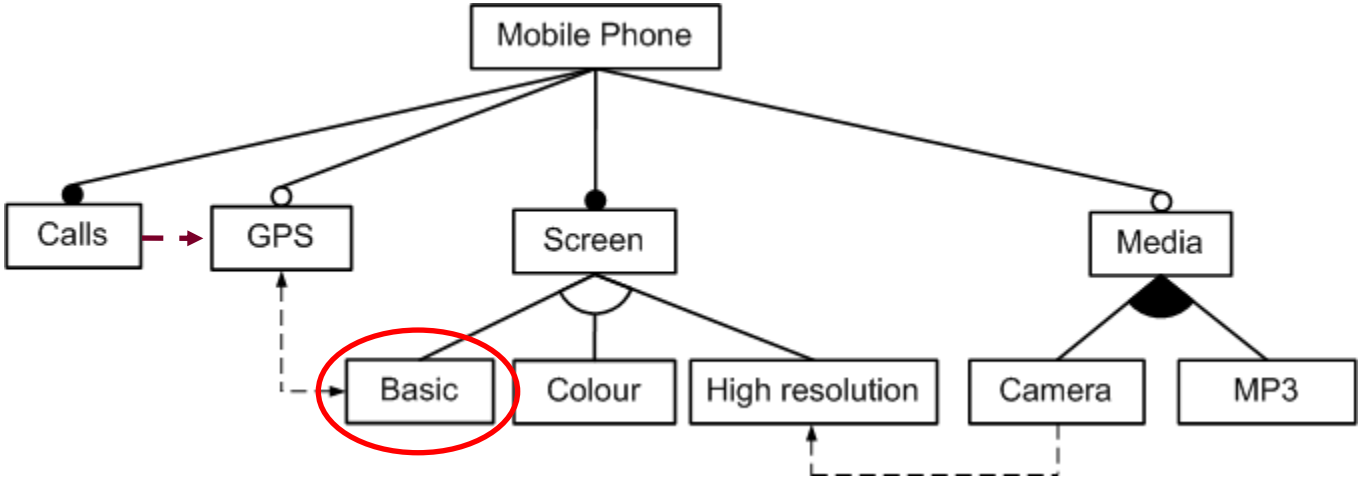


How many products?

14

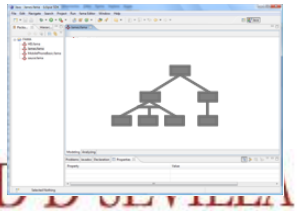


Automated analysis of feature models:
Computer-aided extraction of information from FMs

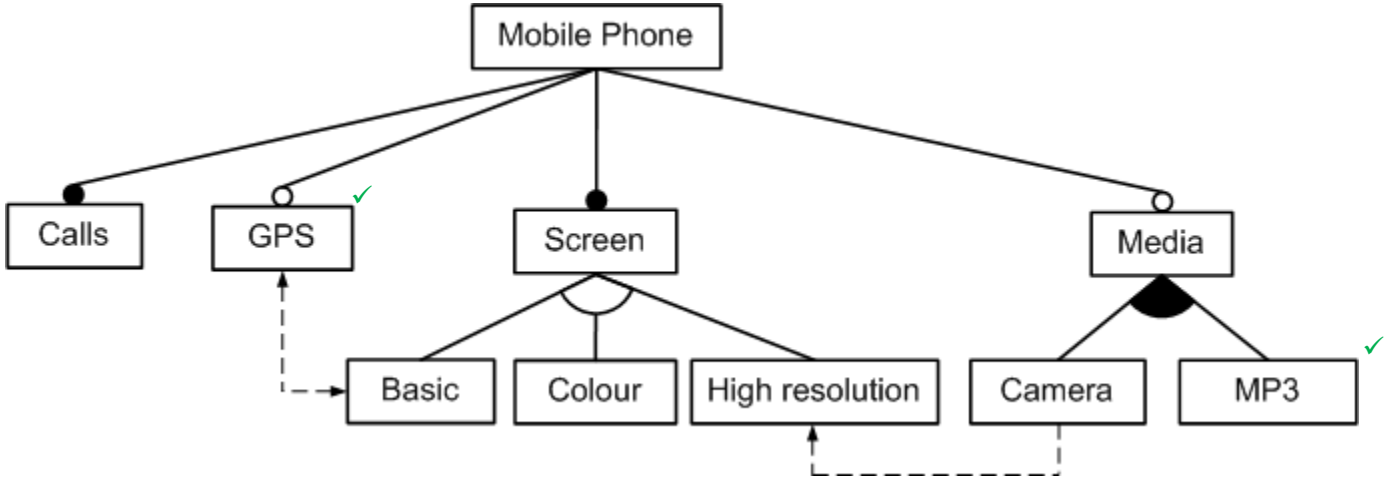


Any error?

Yes, feature "Basic" is dead

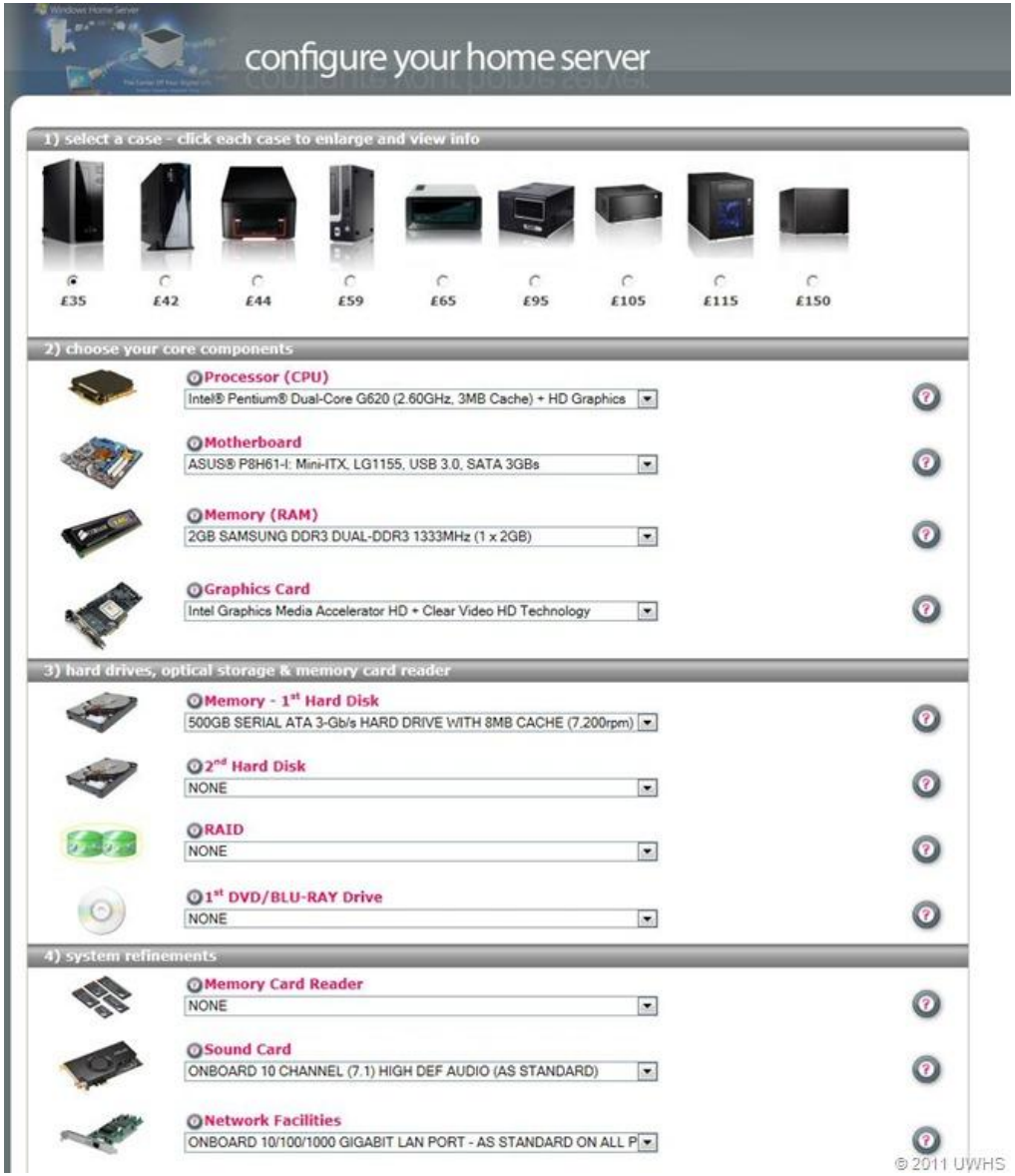


Automated analysis of feature models:
Computer-aided extraction of information from FMs



THE BASE FOR BULDING CONFIGURATORS

Example of a configurator



Automated analysis of SPL:
Computer-aided, extraction of useful information from SPL models

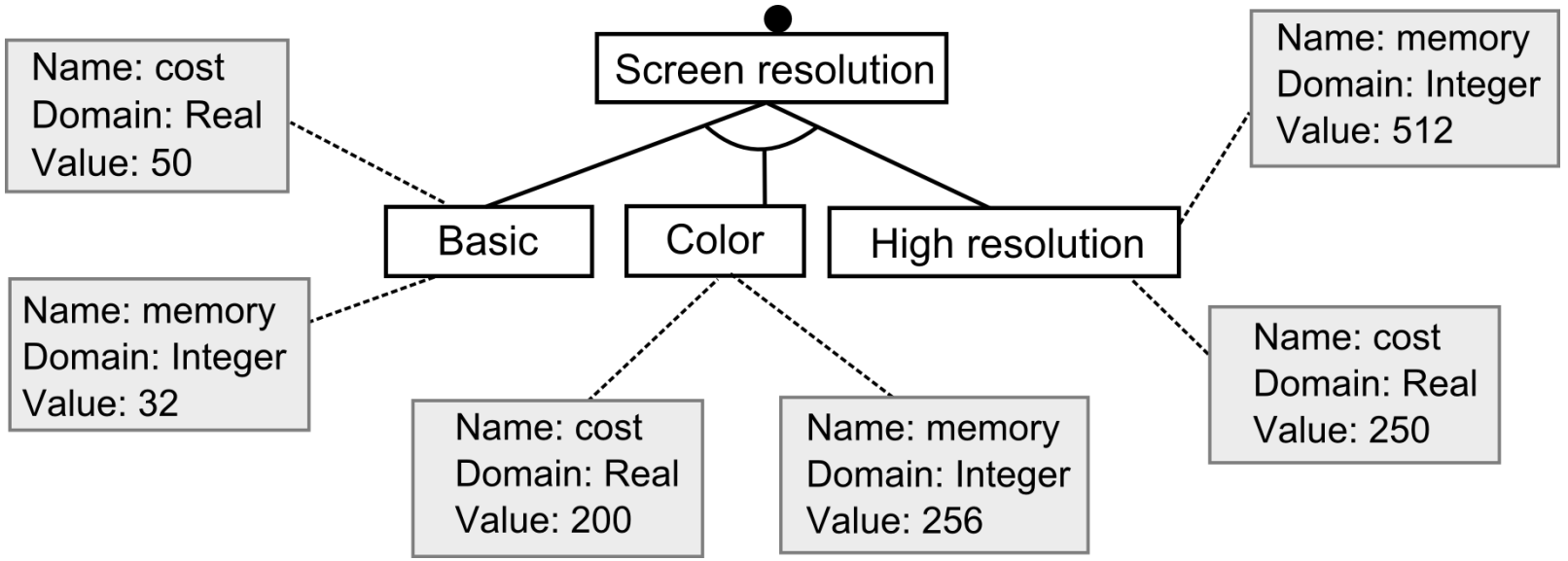
30 analysis operations found

CSP, SAT, BDD, DL, ad-hoc

Performance testing

Formal methods

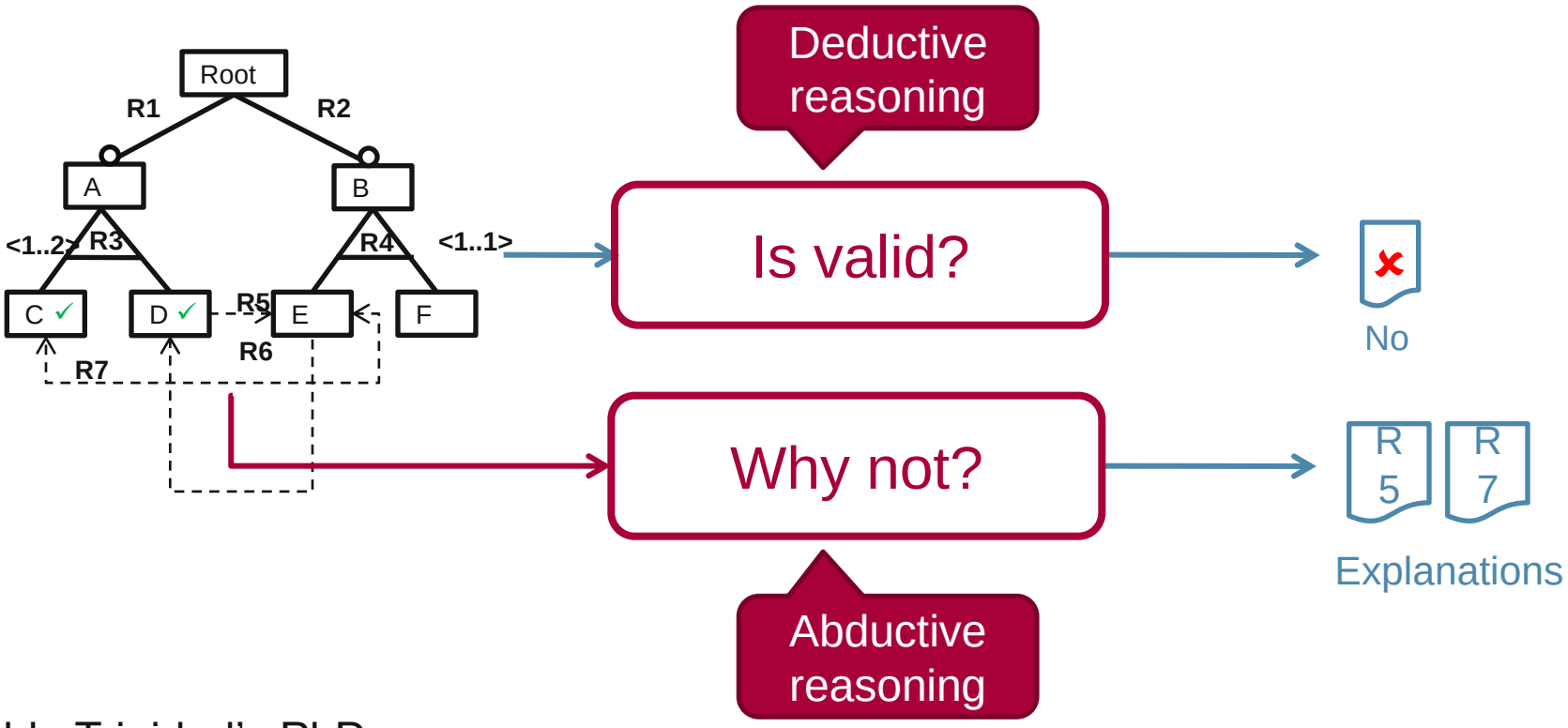
Automated Analysis of Feature Models with attributes



• David Benavides, Pablo Trinidad Martín-Arroyo, Antonio Ruiz Cortés: Automated Reasoning on Feature Models. CAiSE 2005: 491-503

• F Roos-Frantz, D Benavides, A Ruiz-Cortés, A Heuer, K Lauenroth Quality-aware analysis in product line engineering with the orthogonal variability model. Software Quality Journal

Explanations on the Automated analysis of SPL

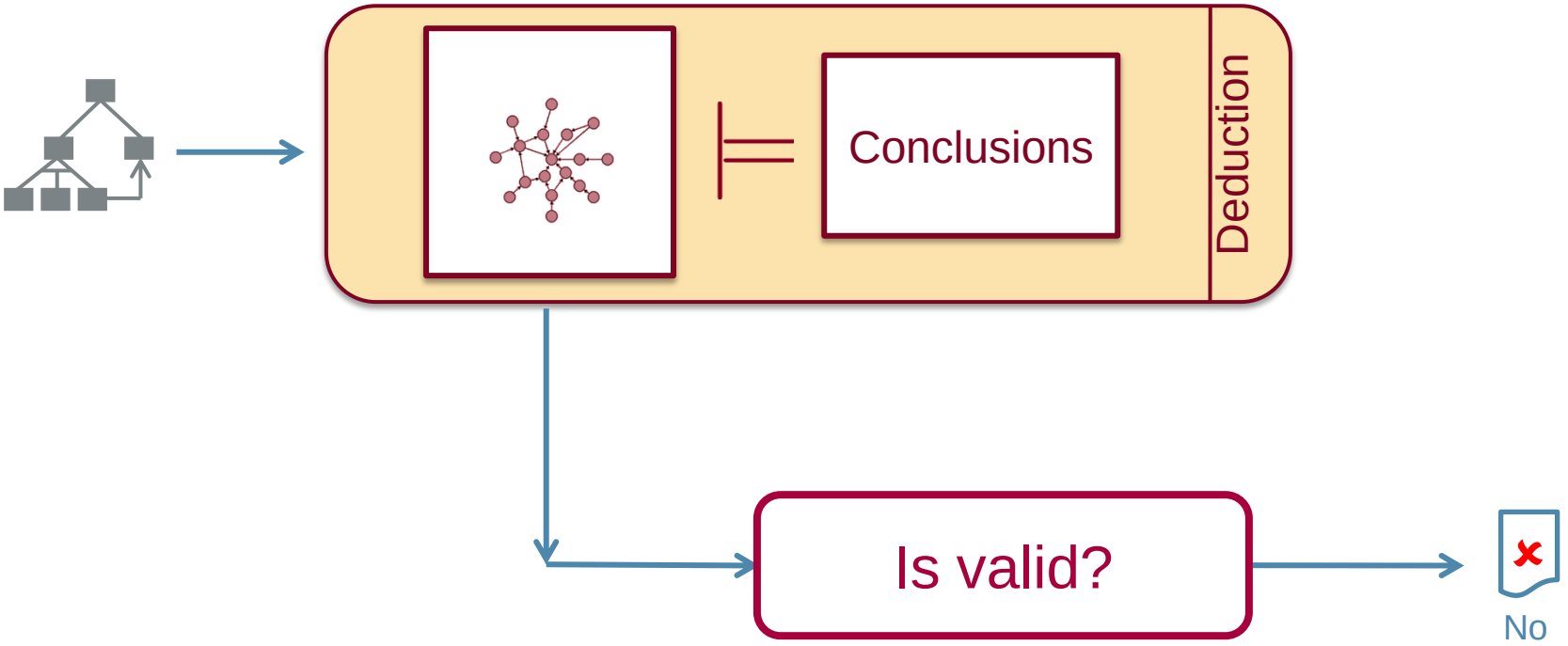


Pablo Trinidad's PhD

Pablo Trinidad, Antonio Ruiz Cortés: Abductive Reasoning and Automated Analysis of Feature Models: How are they connected?. VaMoS 2009: 145-153

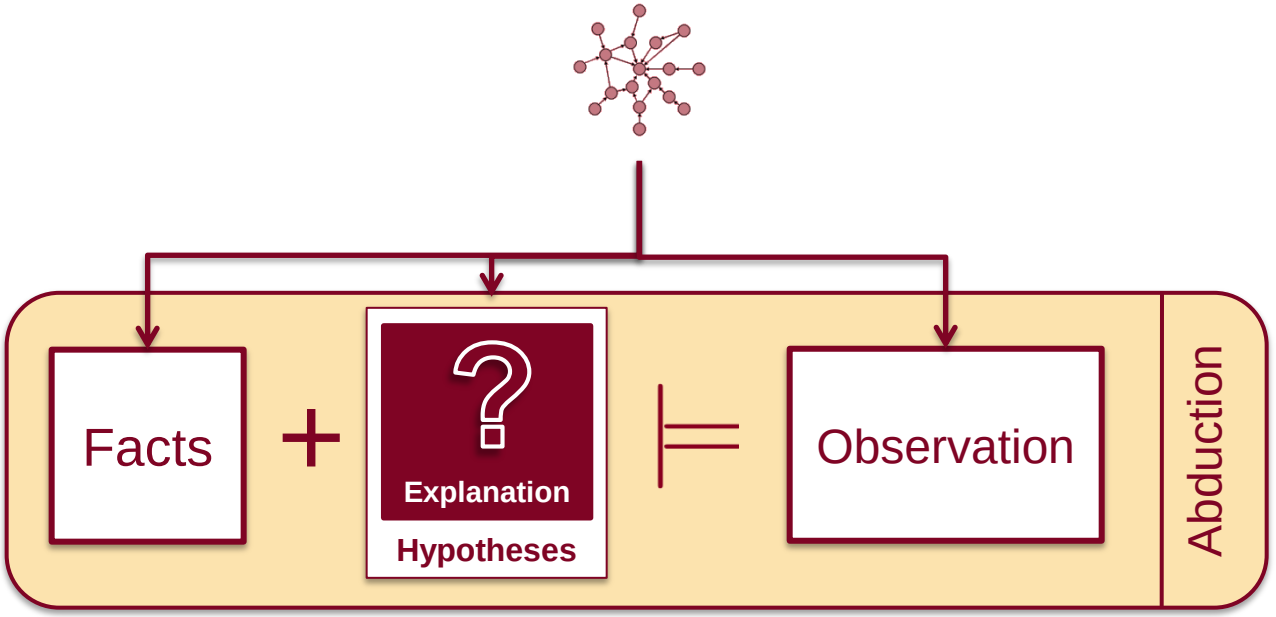
Explanations on the Automated analysis of SPL

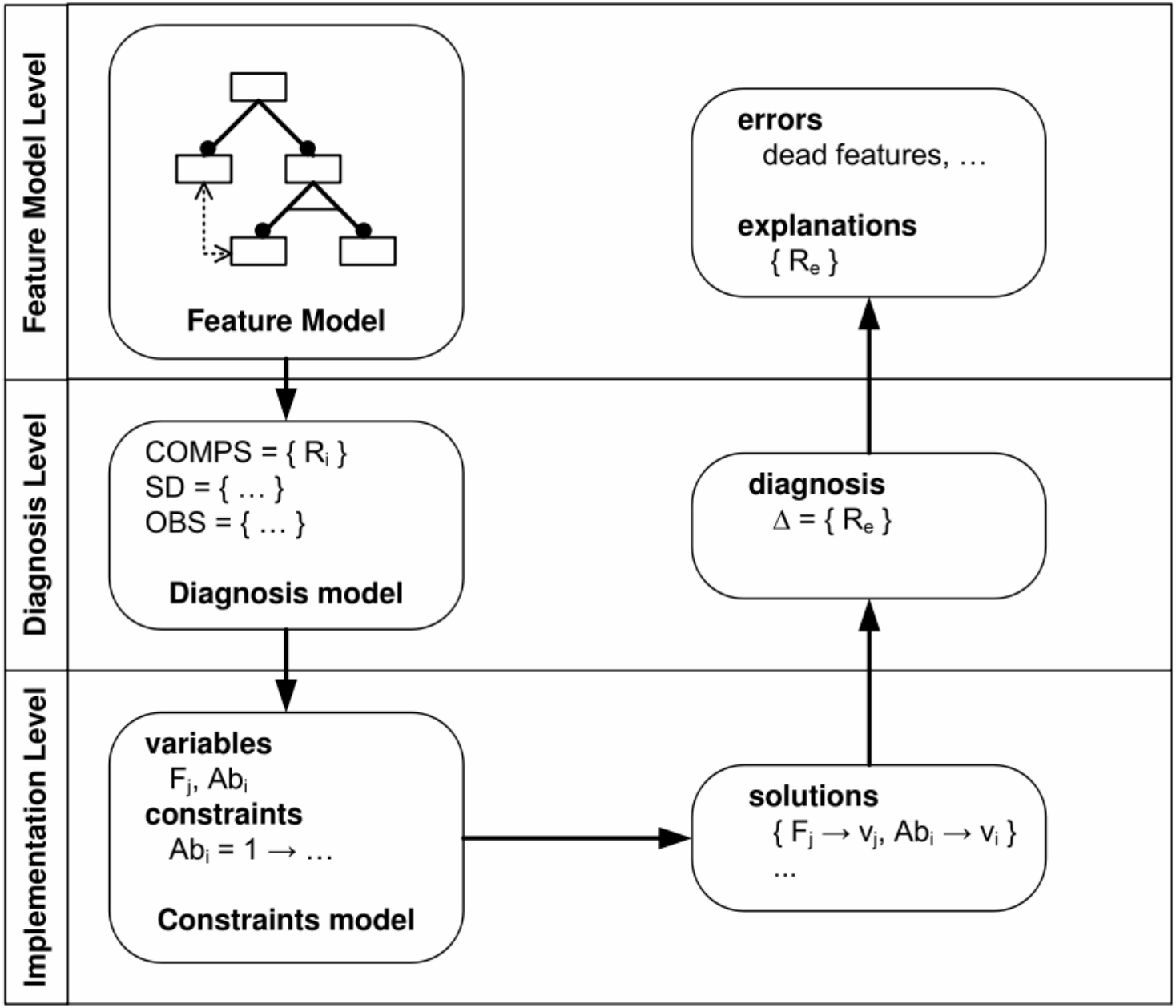
Deductive Reasoning



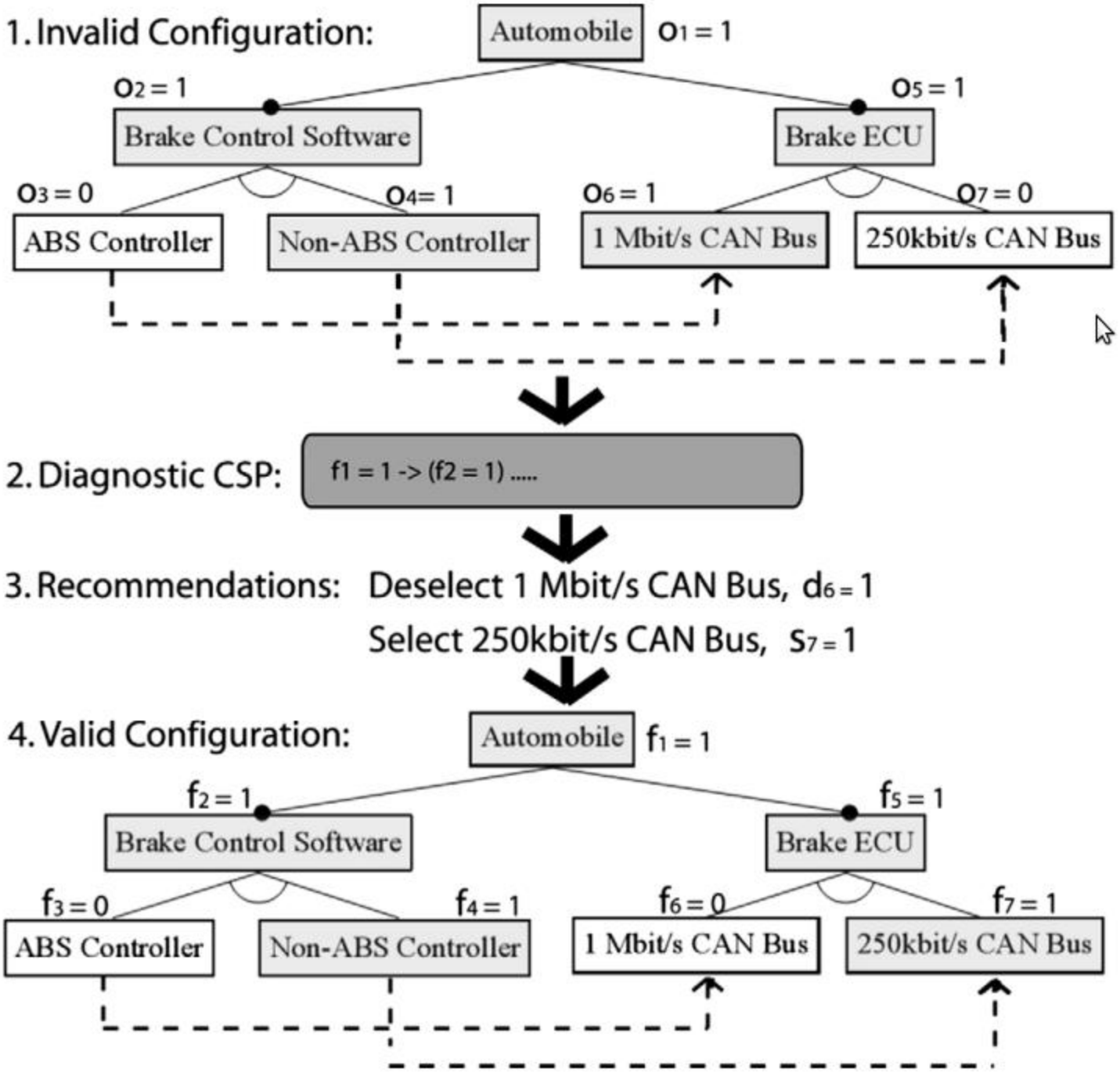
Explanations on the Automated analysis of SPL

Abductive Reasoning




Feature Model Level
Feature Model
errors
dead features, …
explanations
{ R_e }
Diagnosis Level
COMPS = { R_i }
SD = { … }
OBS = { … }
Diagnosis model
diagnosis
Δ = { R_e }
Implementation Level
variables
F_j, Ab_i
constraints
Ab_i = 1 → …
Constraints model
solutions
{ $F_j \rightarrow v_j$, $Ab_i \rightarrow v_i$ }
…

Ch 3.1 with configurations



Software Product Lines Introduction

David Benavides

Department of Computer Languages and Systems
University of Sevilla

UNIVERSIDAD D SEVILLA