# 40. Beyond Object-Oriented Frameworks: Multi-Stage Frameworking with Model and Component Frameworks

Prof. Dr. Uwe Aßmann

Institut für Software- und Multimediatechnik

Fakultät für Informatik

TU Dresden

11-0.3, 12/27/11

# Literature

## Frameworks with MDA or CBSE

- http://st.inf.tu-dresden.de
- U. Aßmann. Invasive Software Composition. Springer, 2003
- Krzysztof Czarnecki, Simon Helsen, Ulrich Eisenecker. Staged Configuration Through Specialization and Multi-Level Configuration of Feature Models. Software Process Improvement and Practice, special issue of best papers from SPLC04
- Aßmann, Uwe, Johannes, Jendrik, Henriksson, Jakob,  Savga, Ilie, Composition of Rule Sets and Ontologies. Reasoning Web, Second International Summer School 2006, pp. 68-92, 2006, LNCS 4126, Springer
- Uwe Aßmann. Model-driven architecture (MDA) and component-based software development (CBSE). In S. Estok, M.M. Lindwer, G. Gopakumar, and L. Posta, editors, XOOTIC Magazine, number 4126 in XOOTIC Magazine, pages 5-7, Eindhoven, September 2007. XOOTIC.
- Jendrik Johannes and Uwe Aßmann. Concern-based (de)composition of model-driven software development processes. In Dorina C. Petriu, Nicolas Rouquette, and Øystein Haugen, editors, MoDELS (2), volume 6395 of Lecture Notes in Computer Science, pages 47-62. Springer, 2010.
- Jendrik Johannes. Component-Based Model-Driven Software Development. PhD thesis, Dresden University of Technology, December 2010. http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-63986

## Introduction

- Frameworks, where are we going from here?
  - Object-oriented frameworks are just the beginning
  - Translational Frameworks with Model-Driven Architecture (MDA ® OMG) go beyond that
  - Component-based frameworks based on other component models: Component-based Software Engineering
  - Multi-stage frameworking

# The Framework Instantation Problem

## Frameworks with MDA or CBSE

- Frameworks can be quite different. So far we had

| | | |
|---|---|---|
| Abstract classes | with | Inheritance |
| Classes | with | Delegation |
| Generic classes | with | Genericity |
| Role models | with | Role merge of open roles |
| | with | Role layers (mixin layers, ROP) |
| Class collections | with | Framework hook patterns (Pree's Template/Hook role model) |

## There are Other Forms

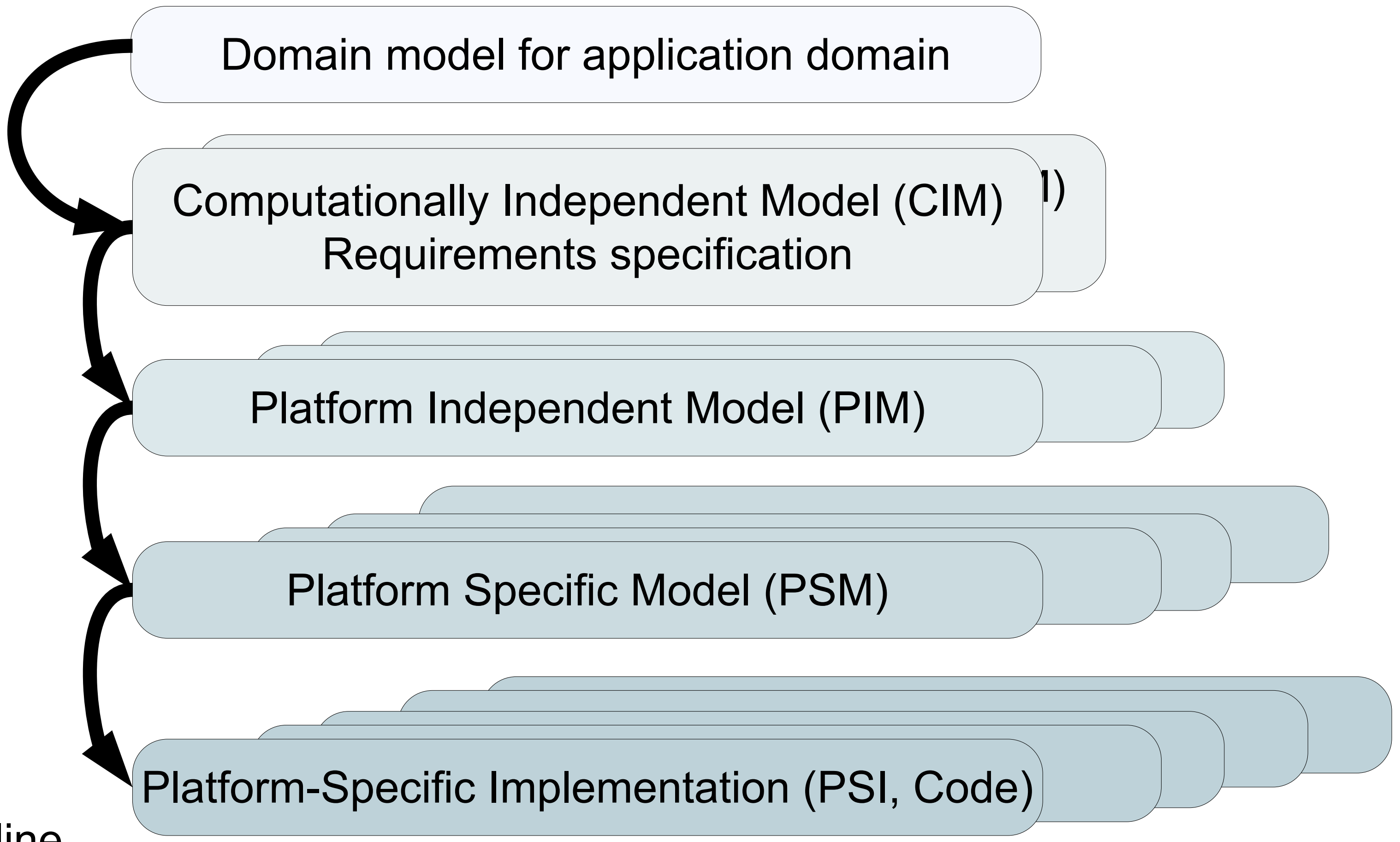
- Model frameworks with transformation (**translational or transformational frameworks**)
  - with weaving (**weaving-based** framework instantiation)
  - How to have reuse of models in MDA *(model frameworks)?*
  - How to realize the MDA components?
- However,
  - Different abstraction levels
    - MDA is about *design/model reuse*
    - Components about *code reuse*
  - Different instantiation mechanisms
    - MDA: translation
    - Components: connection
- Single-stage framework instantiation (normal)
- Multi-stage framework instantiation
  - Extensible multi-stage frameworks
  - Variable multi-stage frameworks

## 40.1 Translational Frameworks (MDA with transformations)

- See also Softwaretechnogie-II

# MDA Describes Product Lines

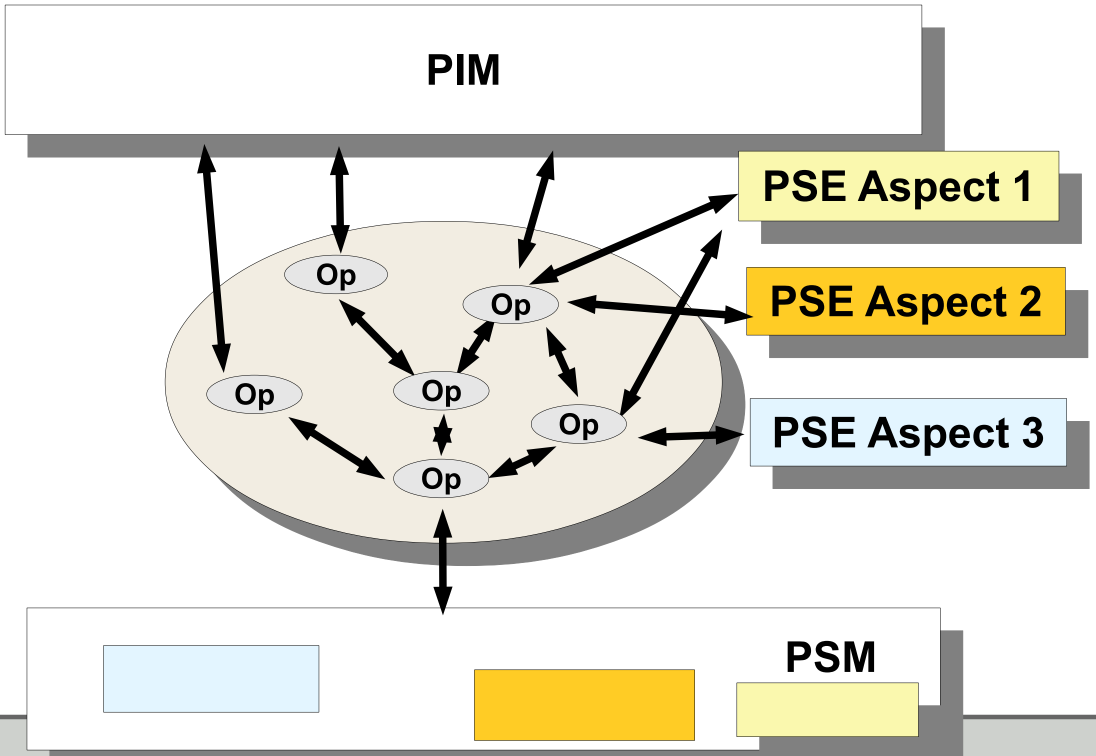
- The platform stack is a *translational model framework*


Domain model for application domain
Computationally Independent Model (CIM)
Requirements specification
Platform Independent Model (PIM)
Platform Specific Model (PSM)
Platform-Specific Implementation (PSI, Code)

The products
of the product line

# MDA with Aspect Mappings

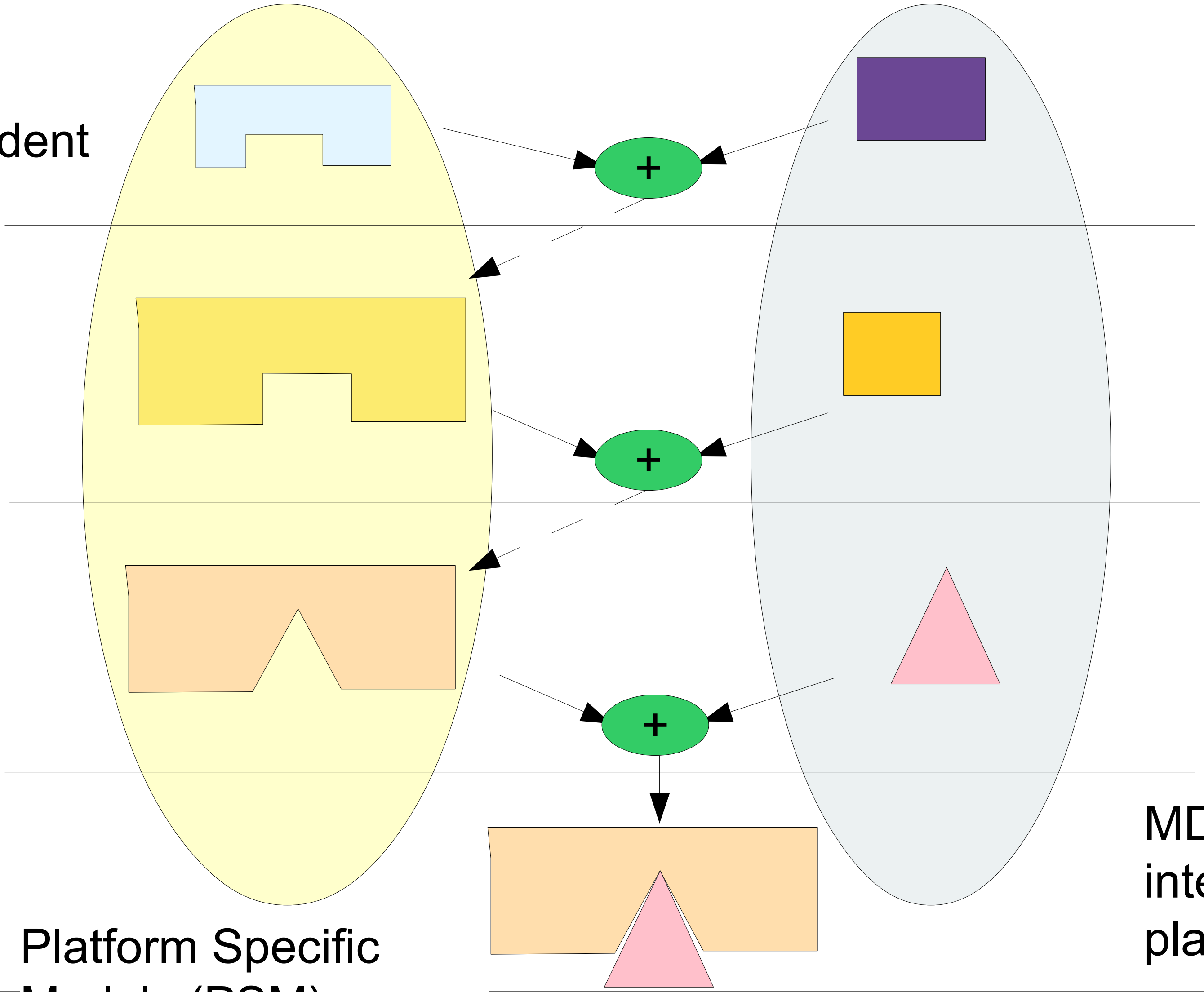
- Describe *platform specific extension (PSE)* as *aspects:*
  - The PIM is the *core, the* PSM the *weaved system*
  - The model mapping becomes an *model aspect weaver*
  - MDA uses transformations to implement the weaving



# A Weaving Architecture: MDA

Platform Independent Models (PIM)

Platform Specific Extensions (PSE)



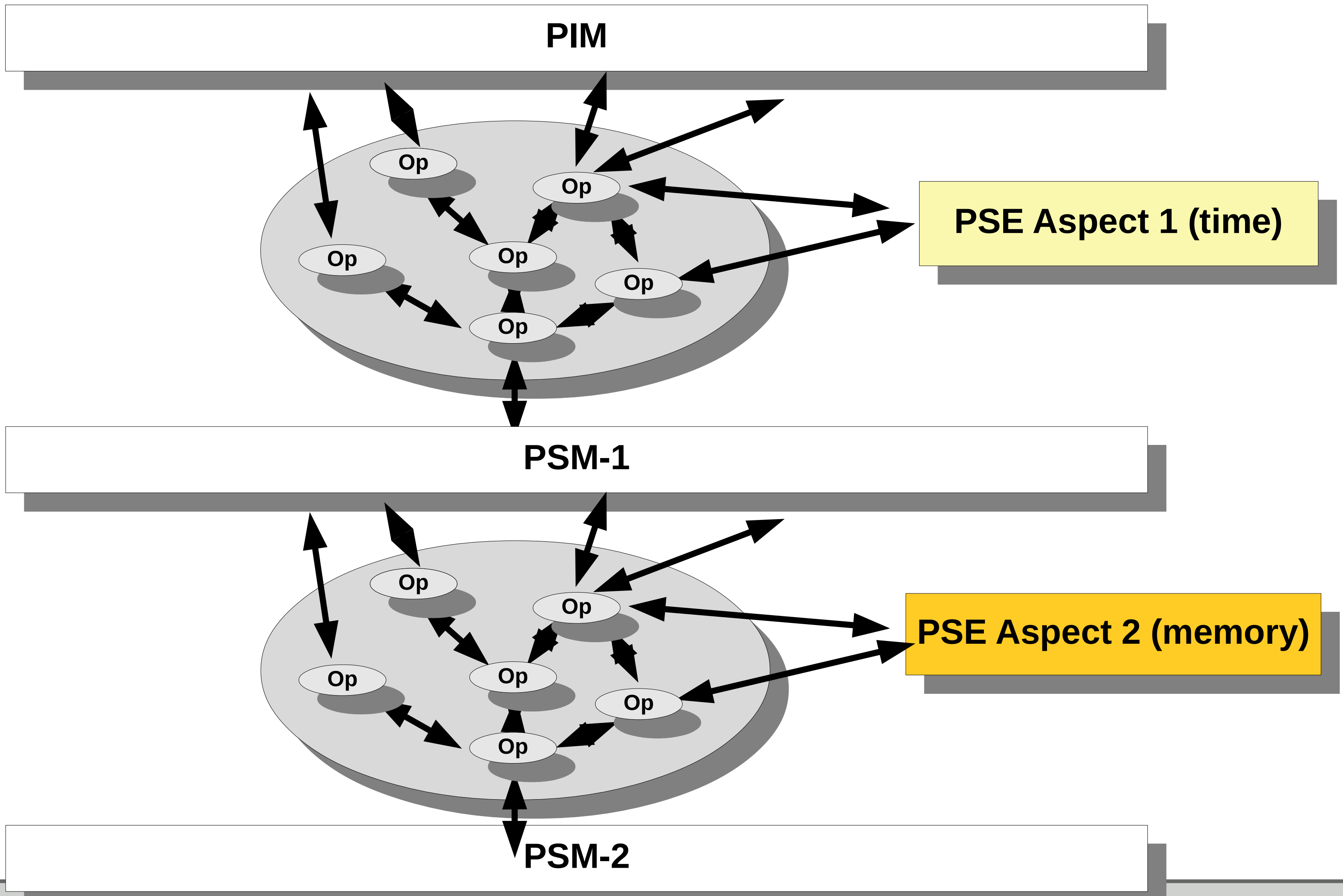
MDA weavers integrate platform variants

Platform Specific Models (PSM)

# Aspects Important for Embedded Systems

- Life-time aspects in MDA
  - How long does an object live?
- Resource aspects in MDA
  - Representation of collections (SetL)
- Middleware aspects
  - Representation of connectors
  - Transactions, persistency, ...
- Real-time aspects in MDA
  - Profile info
  - OCL-RT info
- Quality aspects in MDA
  - Contracts
  - Security

# MDA With Several Layers for Resource-Constrained Systems

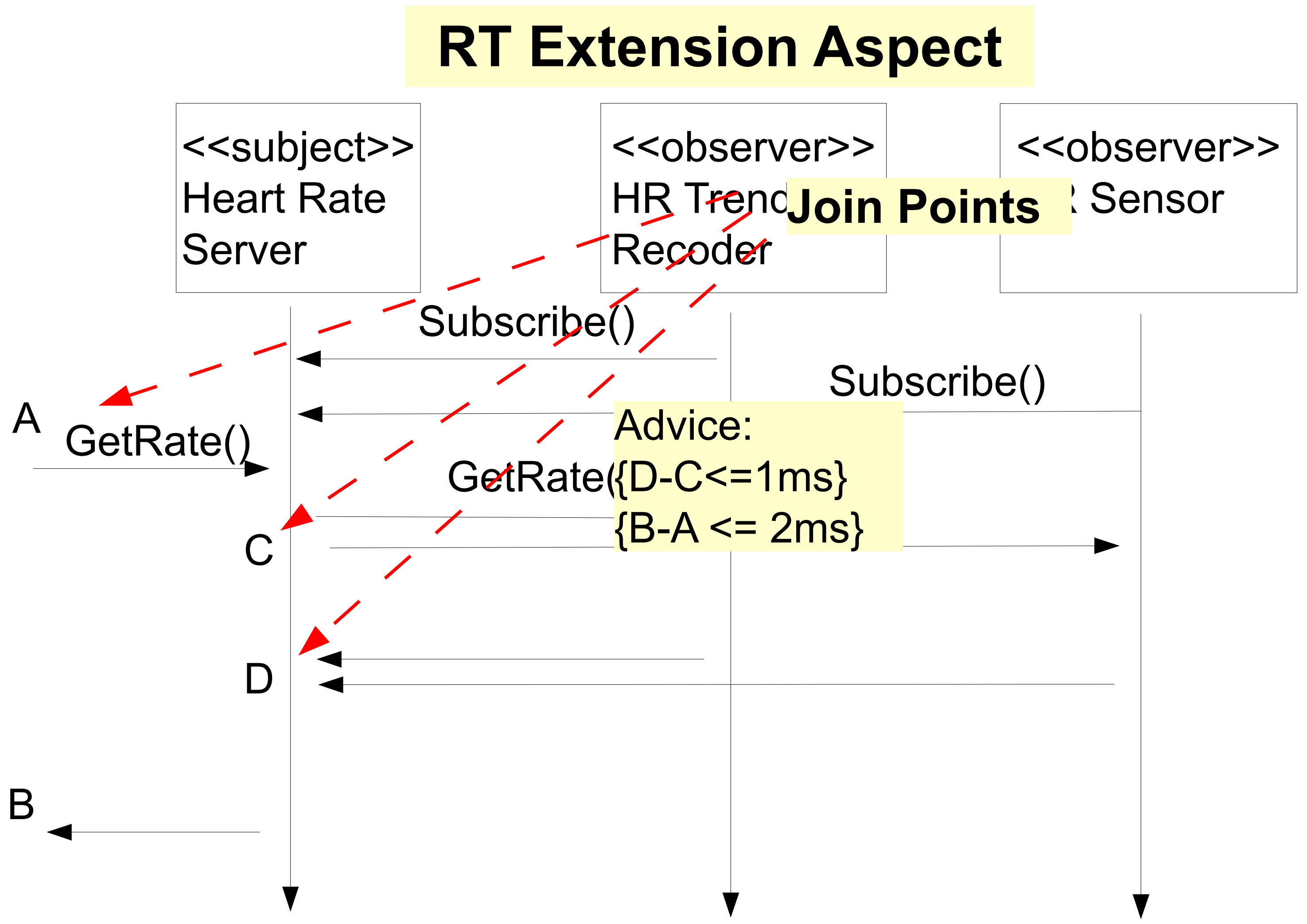


# Example: MDA for RT-UML

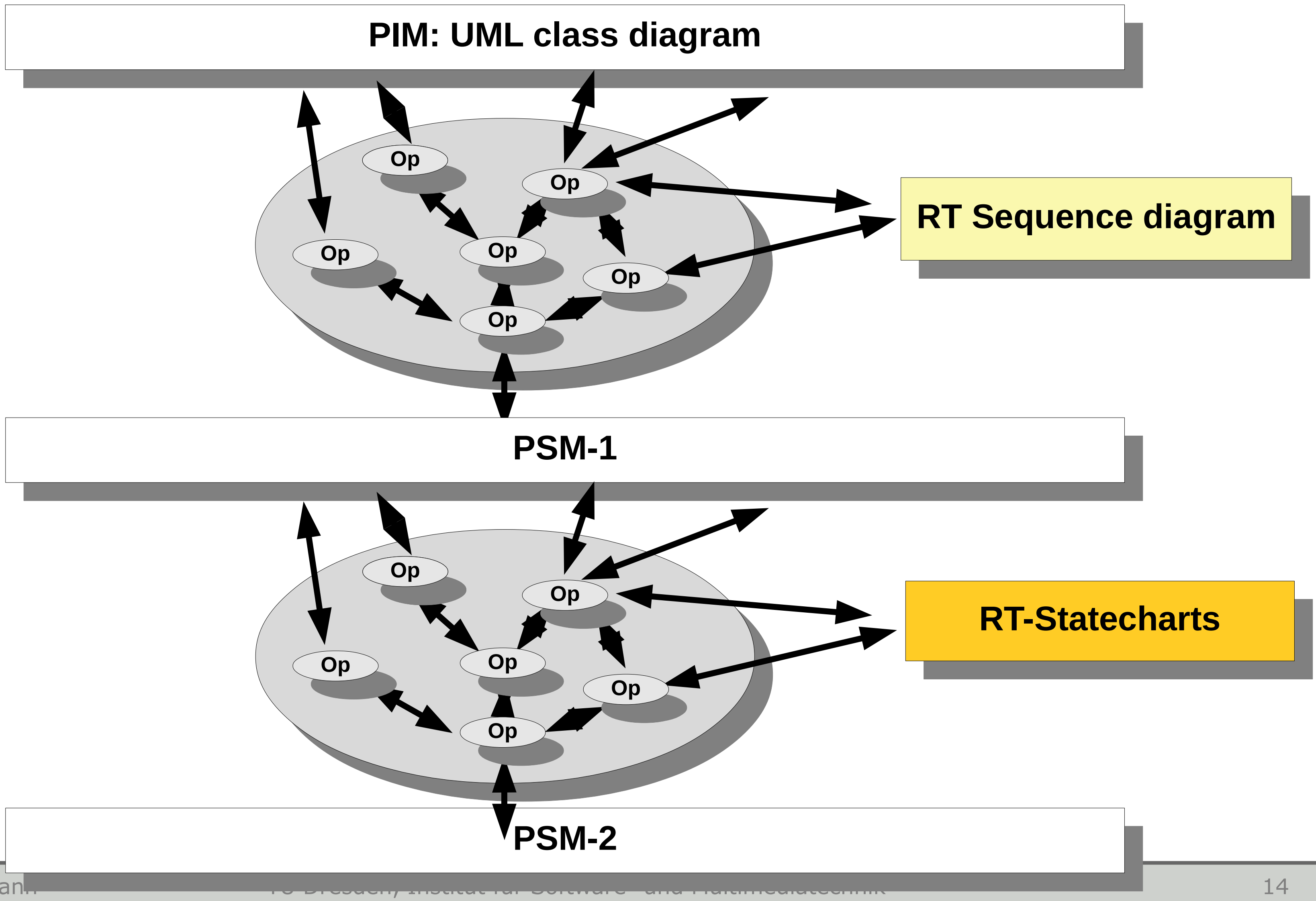
- EU project *High Integrity Distributed Object-Oriented Real-Time Systems (HIDOORS)* http://www.hidoors.org
- German BMBF project SuReal http://www.sureal-projekt.org

- MDA for RT-UML
  - Realtime sequence diagrams (MSC)
  - UML realtime statecharts
- Mapping to timed automata of Uppaal model checker

## Ex. RT Sequence Diagram (UML)



# Ex. RT-SD und RT-Statecharts are Platform Specific Aspects



## What MDA Really Is About

- MDA is about *weaving platform-extension model aspects*

- Problem:
  - We need weavers for every level
  - With different modeling languages
  - Who will build them?

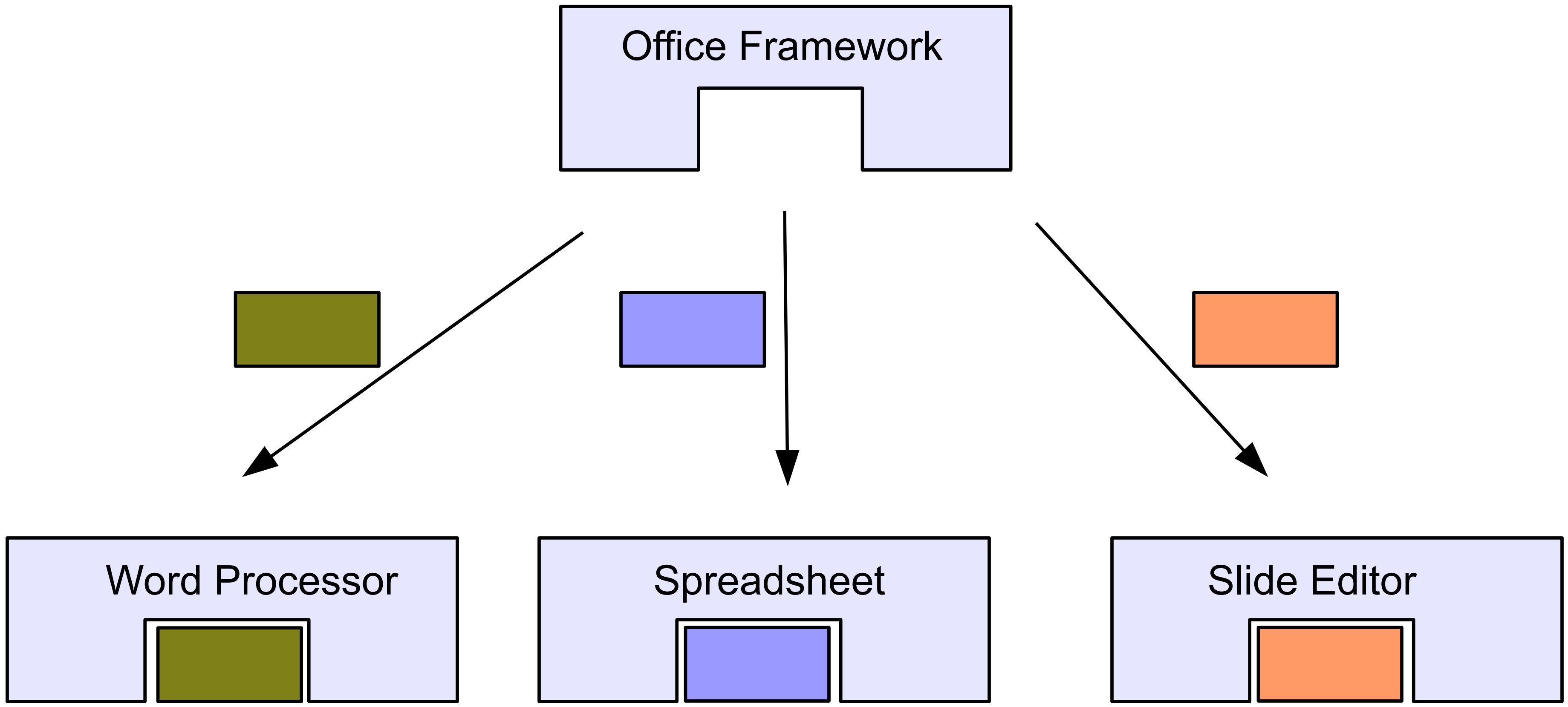
# MDA is about Concern Separation

- MDA is **not about platforms**
- MDA is a **multi-stage model framework** approach with subcategories
  - Variability-based MDA (for variation)
    - Templates (mappings are parameterizations)
    - Modules
    - Variablility frameworks
    - Connector-based frameworks (mappings are refinements)
  - Extensible MDA (for unforeseen extension)
    - Views (mappings are extensions)
    - Aspects
    - Extensible frameworks
  - Translational MDA

## 40.2 Frameworks with other Component Models (Frameworks with CBSE)

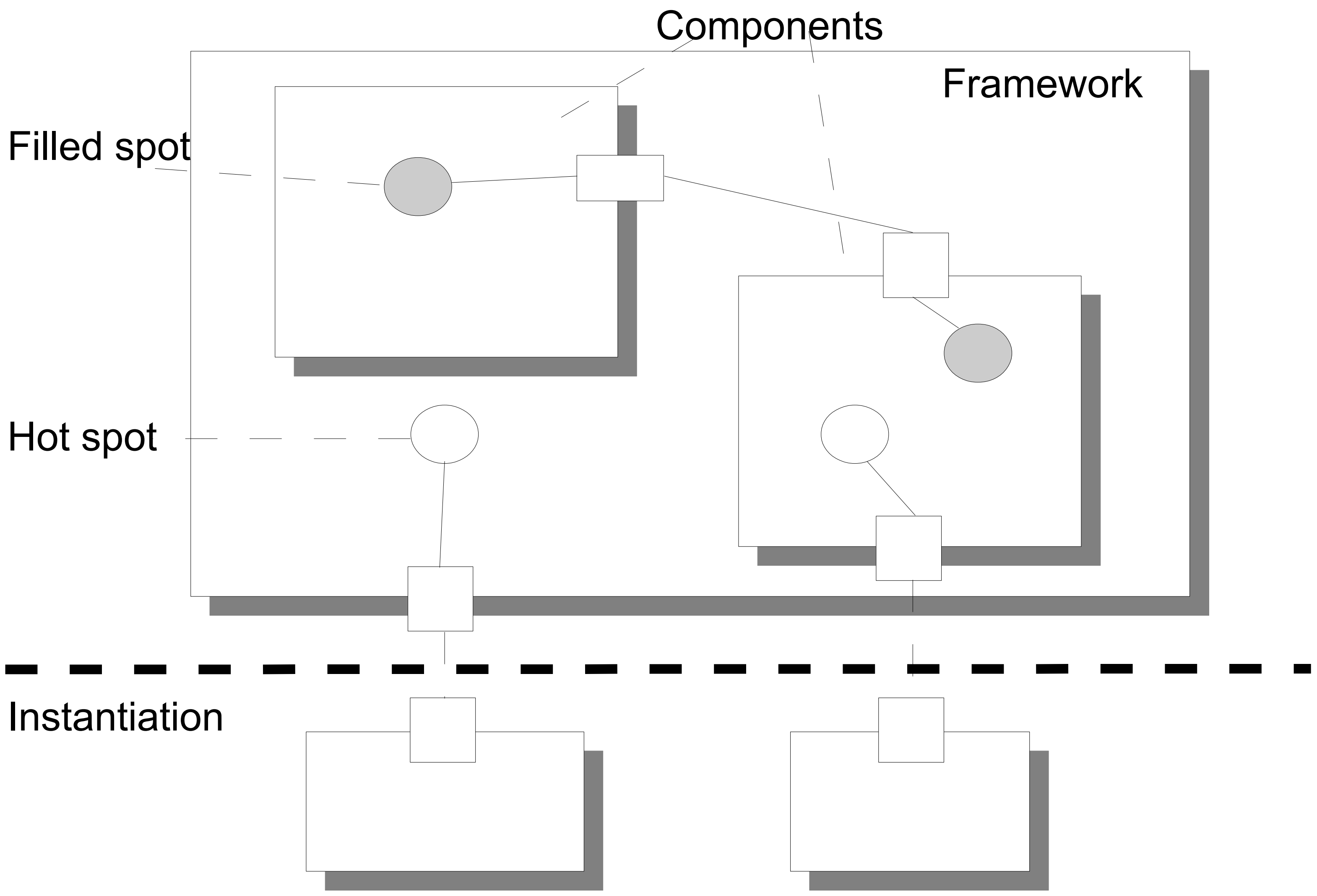
# Frameworks, Components, and Products

- Planning several products from one common code base, but several variant parameterizations



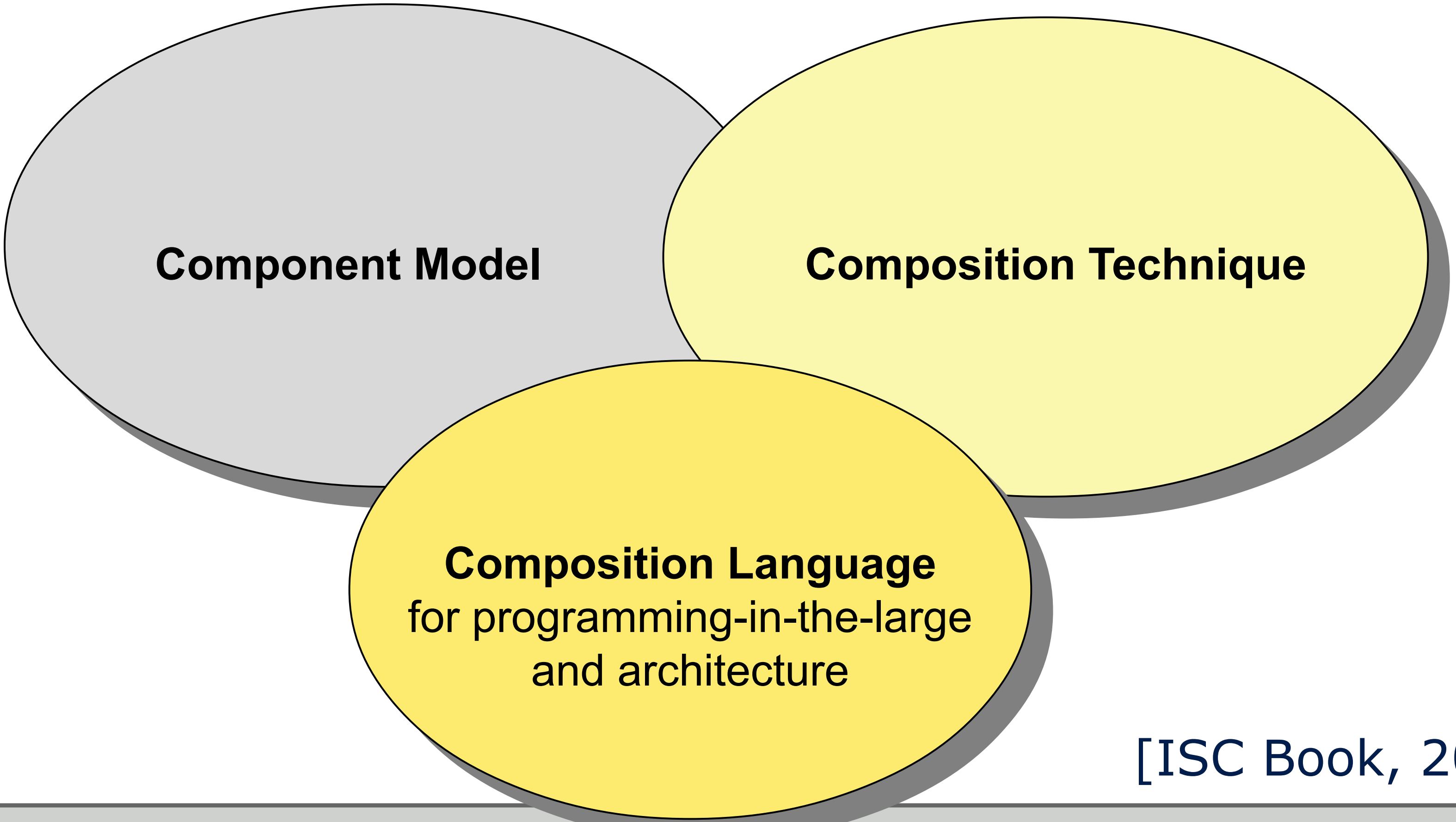
# Frameworks are Larger Components

**Frameworks with MDA or CBSE**



# Software Composition Systems

- A composition system has



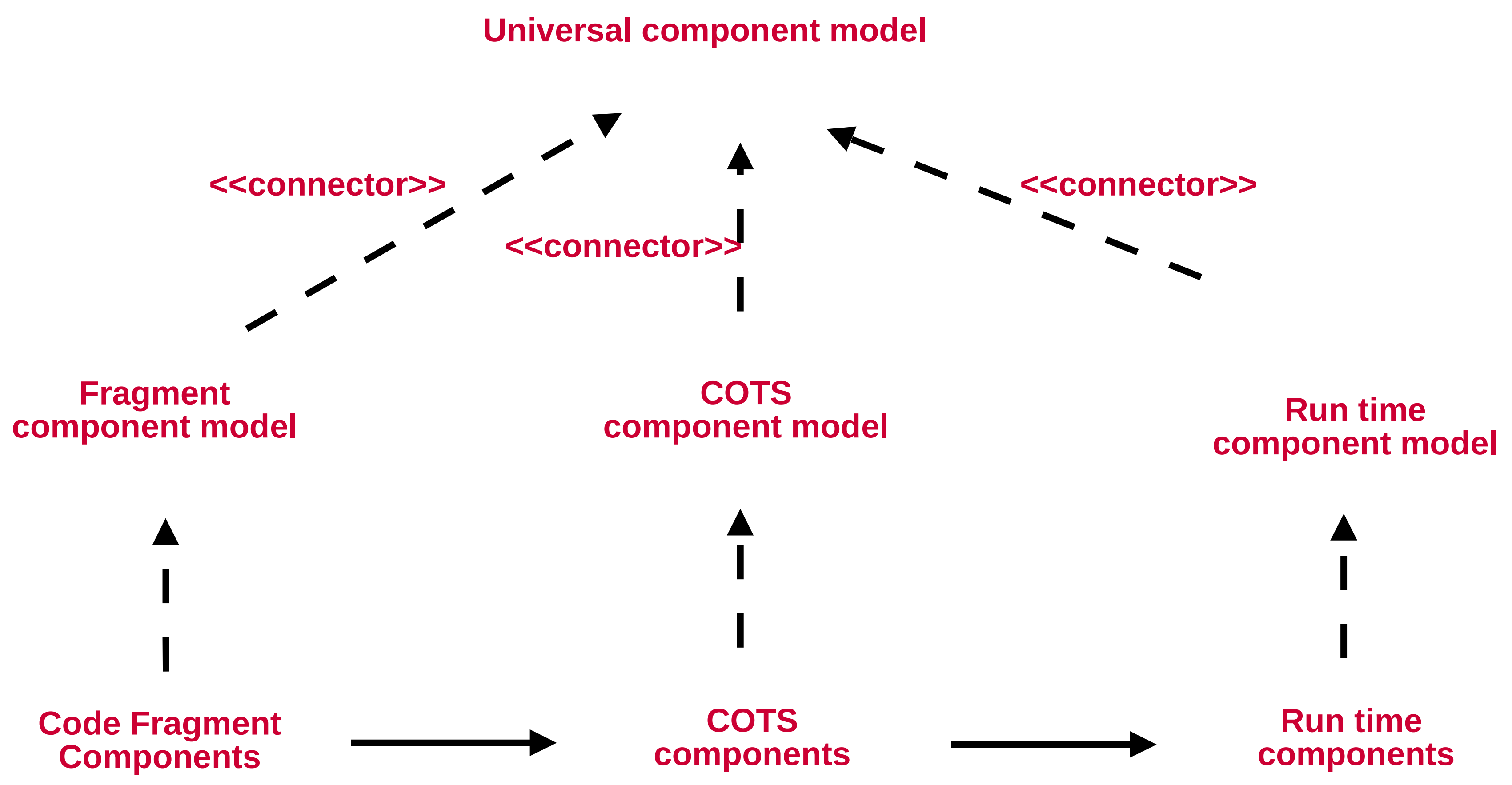
[ISC Book, 2003]

## The Ladder of Composition Systems

| | | | |
|---|---|---|---|
| Grey | **Software Composition Systems** | **Composition Language** | ***Invasive Composition Piccola*** |
| | **Aspect Systems** | **Aspect Separation Crosscutting** | ***Aspect/J Composition Filters*** |
| | **View Systems** | **Merge Operator** | ***Hyperslices Role modeling*** |
| | **Architecture Systems** | **Architecture as Aspect** | ***Darwin ACME*** |
| Black | Classical Component Systems | **Standard Components** | ***.NET CORBA Beans EJB*** |
| | **Object-Oriented Systems** | **Objects as Run-Time Components** | ***C++ Java*** |
| | **Modular Systems** | **Modules as Compile-Time Components** | ***Modula Ada-85*** |

# Component Models at Different Composition Times

## Frameworks with MDA or CBSE



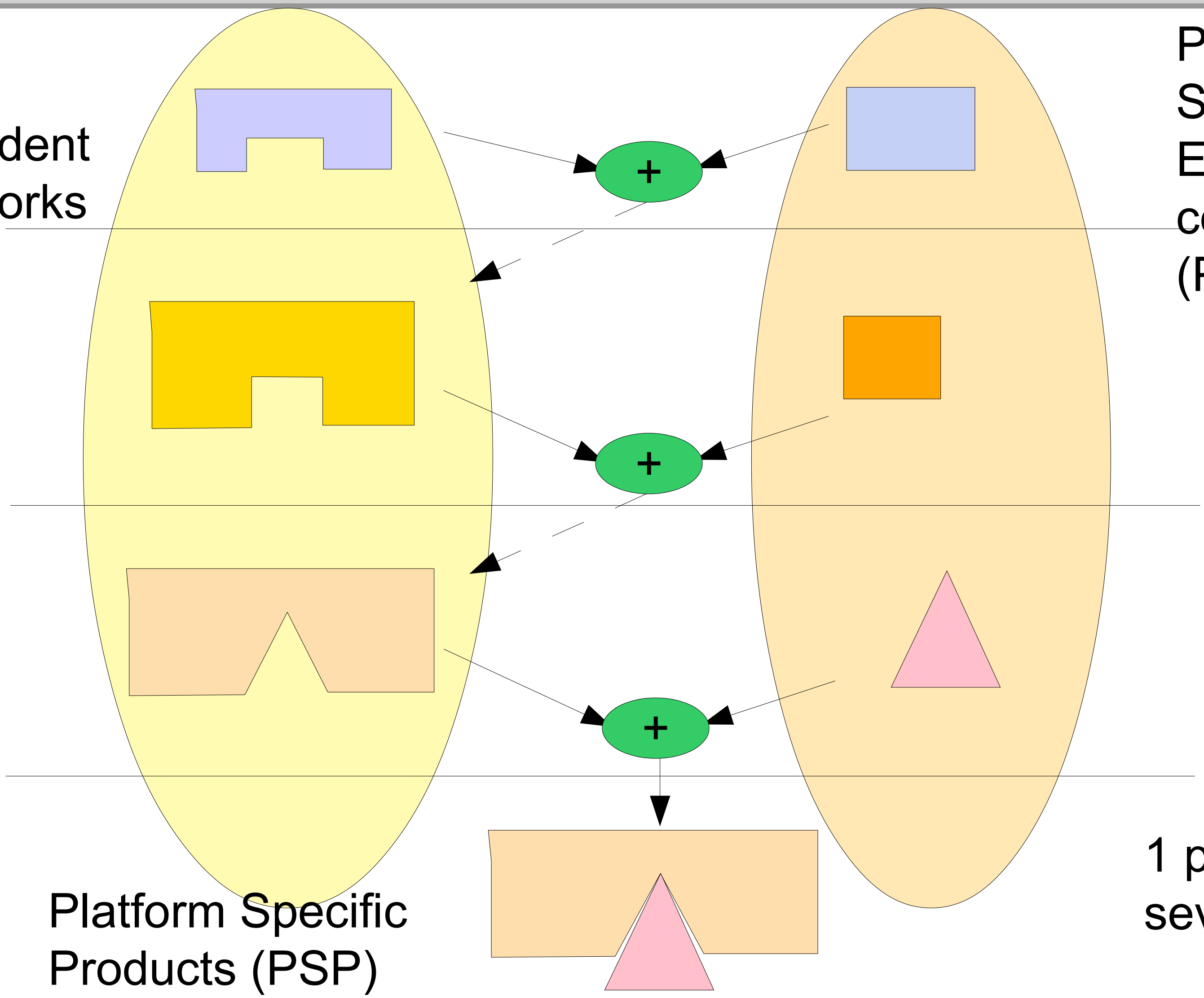
## 40.3 Frameworks on Multiple Stages (Staged Frameworks)

# Frameworks in Stages

## Frameworks with MDA or CBSE

Platform Independent Frameworks (PIF)

Platform Specific Extensions components (PSE)



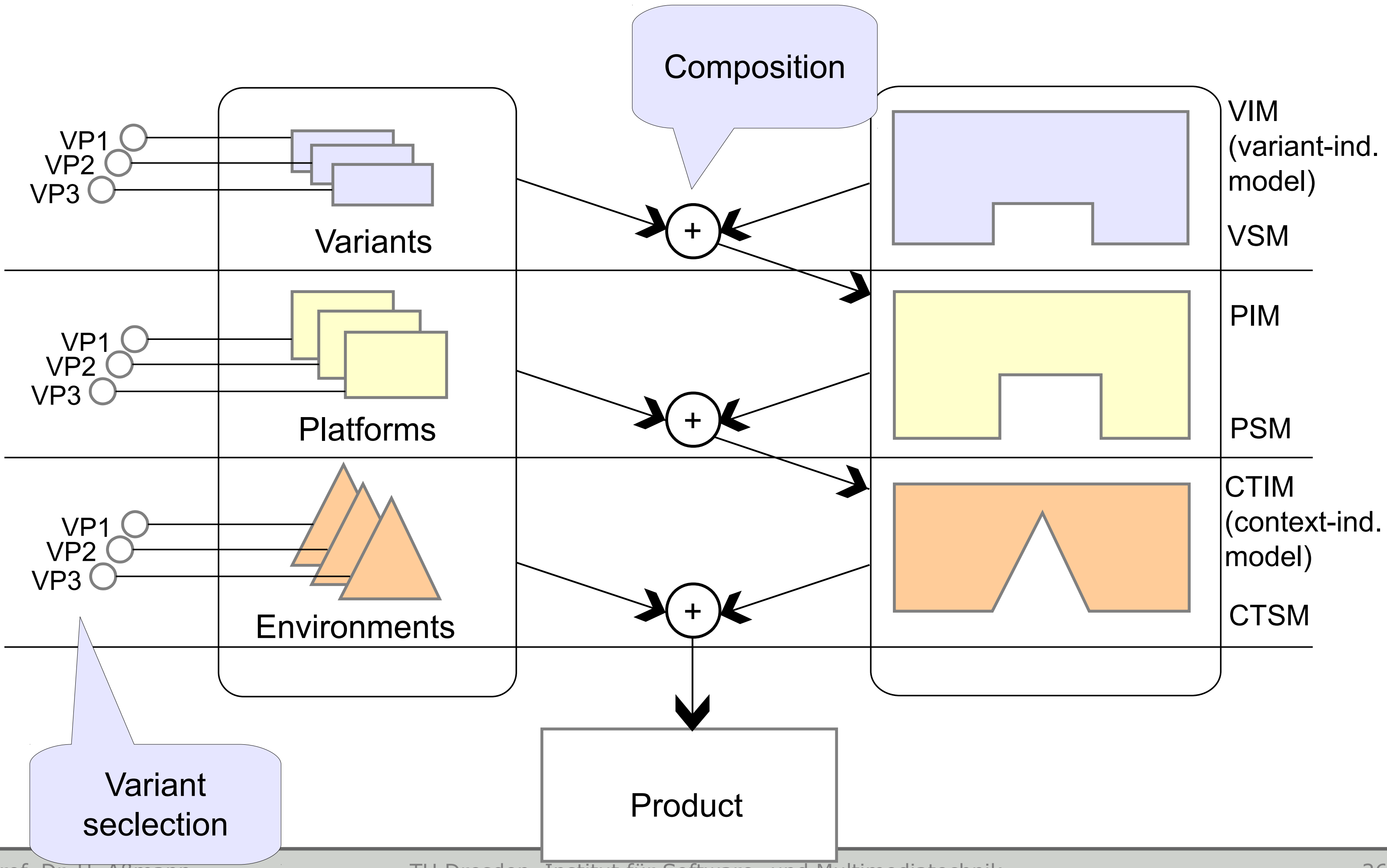
Platform Specific Products (PSP)

1 phase or several

# Multi-Staged MDSD-Frameworks

- Normally appear in Product Line Engineering where you have
  - variabilities on each stage
  - different modeling languages, component systems and composition languages on various stages
  - different instantiation mechanisms per stage
- Every stage produces the software artifacts used for the next stage
- The composers are driven by concrete variant selections on every stage
- Variant selection on stage n may affect variant space on stage n-1

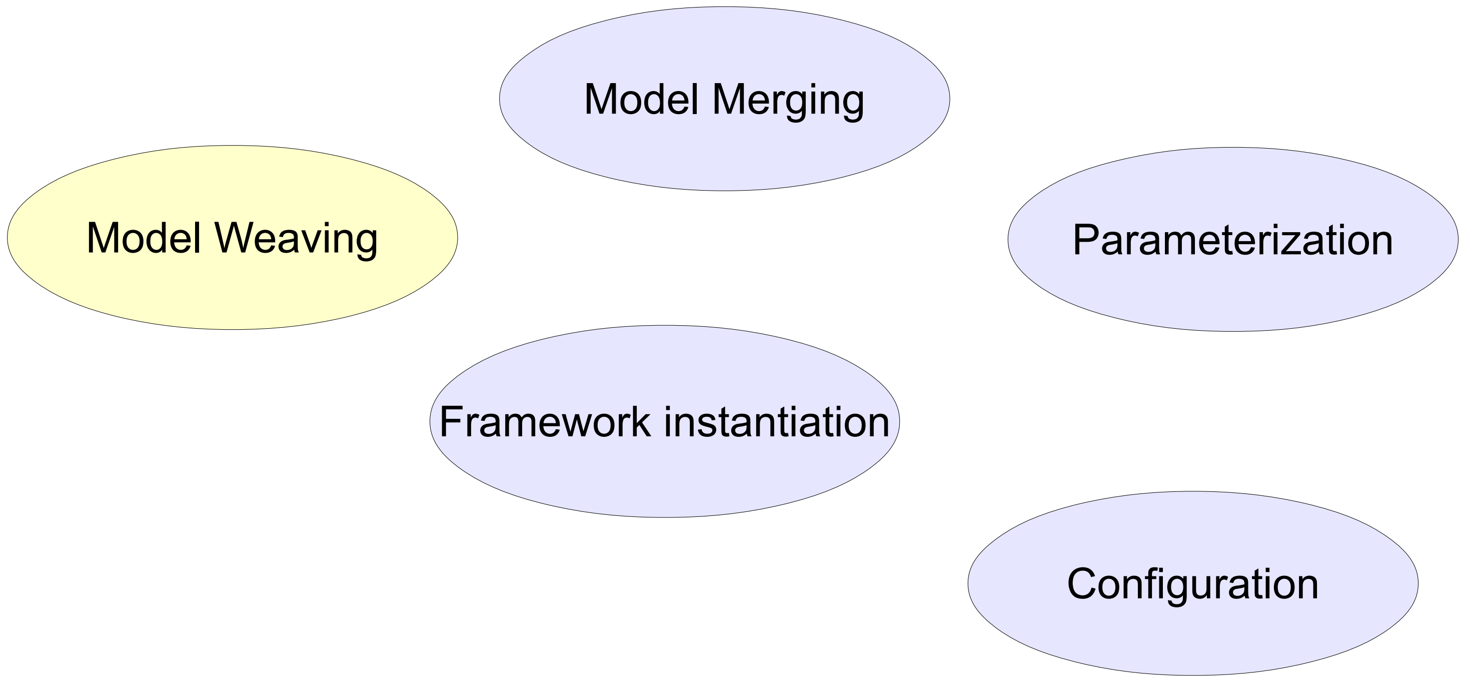
# Example for a Multi-Staged MDSD Framework



# Variablity in composition techniques

## Frameworks with MDA or CBSE

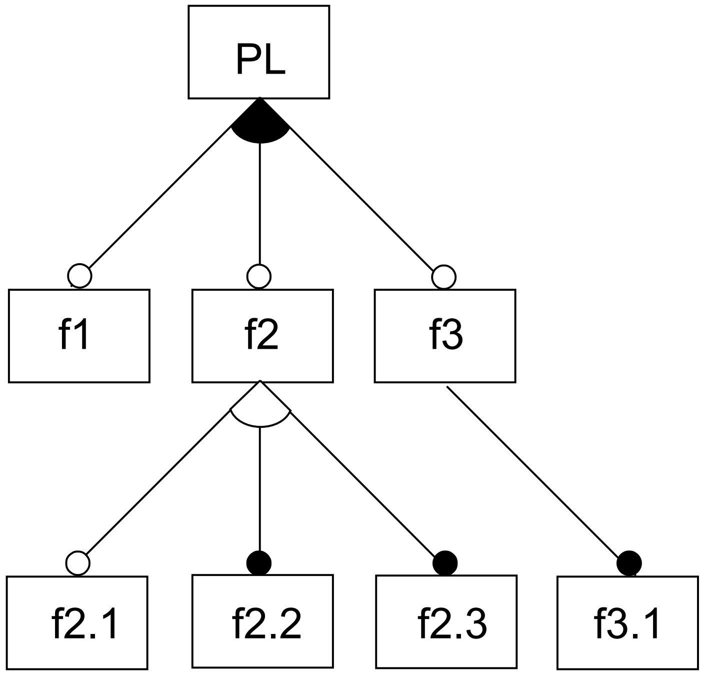
- Possible composition operators / composers
- $\oplus$
- A composer is used to transform software artifacts from stage n to stage n-1



# Modeling Variability in Product Lines

- Feature Models
  - Feature models from FODA are used to express variabilities in Product Lines

- Possiblities to model
  - alternative,
  - mandatory, and
  - optional features



# Subtractive Variant Selection

- **Subtractive approach** of modeling the variant space (the *integrated* VIM)
  - Model all variants in one model and remove elements based on absence of variant selection in feature model
- Used technique: model bridges, weaving models
  - Pros:
    - no need for links between artifacts
    - short cognitive distance
  - Cons:
    - conflicting variants can't be modeled correctly
    - huge and inconcise models

# Additive Variant Selection

- **Additive approach** of modeling the variant space (the *split* VIM)
  - Model all variants in external fragment models and compose them with a core model based on the presence of variant selection in feature model
- Used technique: markers based on e.g. stereotypes
  - Pros:
    - conflicting variants can be modeled correctly
    - small core and many concise variant fragments
  - Cons:
    - traceability problems
    - increased overhead in linking the different fragments

## Challenges in the design of staged MDSD-Frameworks

- Multi-staged variant interaction
- Finding a fixed and well-scoped number of stages
- Variant management for large variation spaces
- Finding the right order of variant selections
  - will be addressed within the feasiPLe BMBF project :-)
  - SAP, Pure Systems, TUD, Misys Dresden
  - Jobs & theses available!

## 40.4 Stability/Change Analysis (SCA) a la Parnas

Old Tales About Change-Oriented Design

Parnas information hiding principle

for change-oriented design

## Modules (Information-Hiding-Based Design a la Parnas)

- Every module hides the an important design decision behind a well-defined interface which does not change when the decision changes.

We can attempt to define our modules “around” *assumptions which are likely to change*.

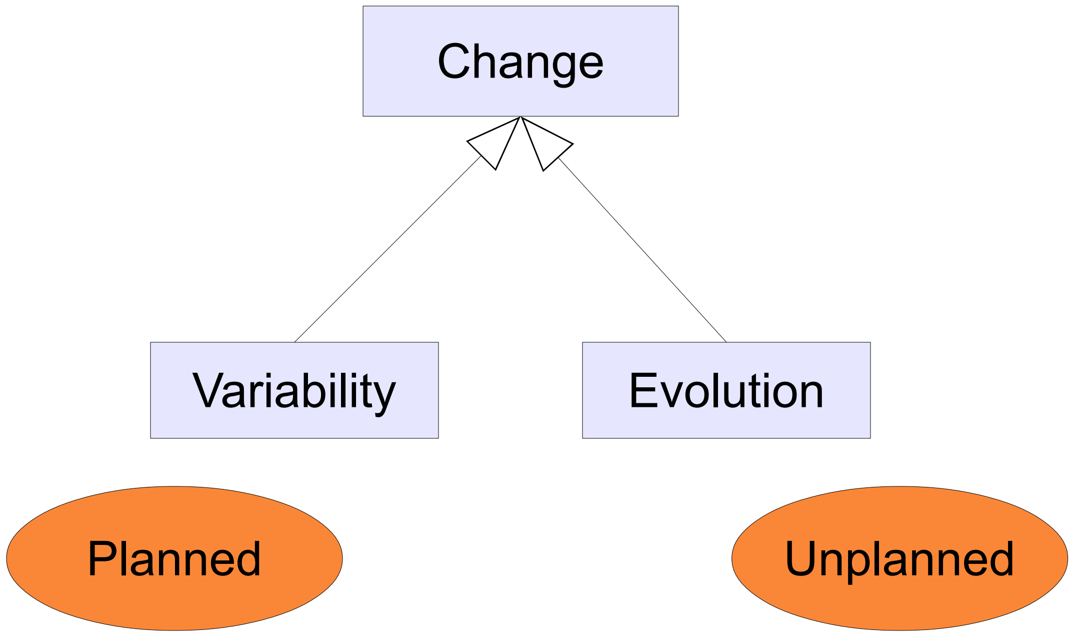
One then designs a module which “hides” or contains each one.

Such modules have rather abstract interfaces which are relatively unlikely to change.

# Commonality-Variability Analysis (CVA)

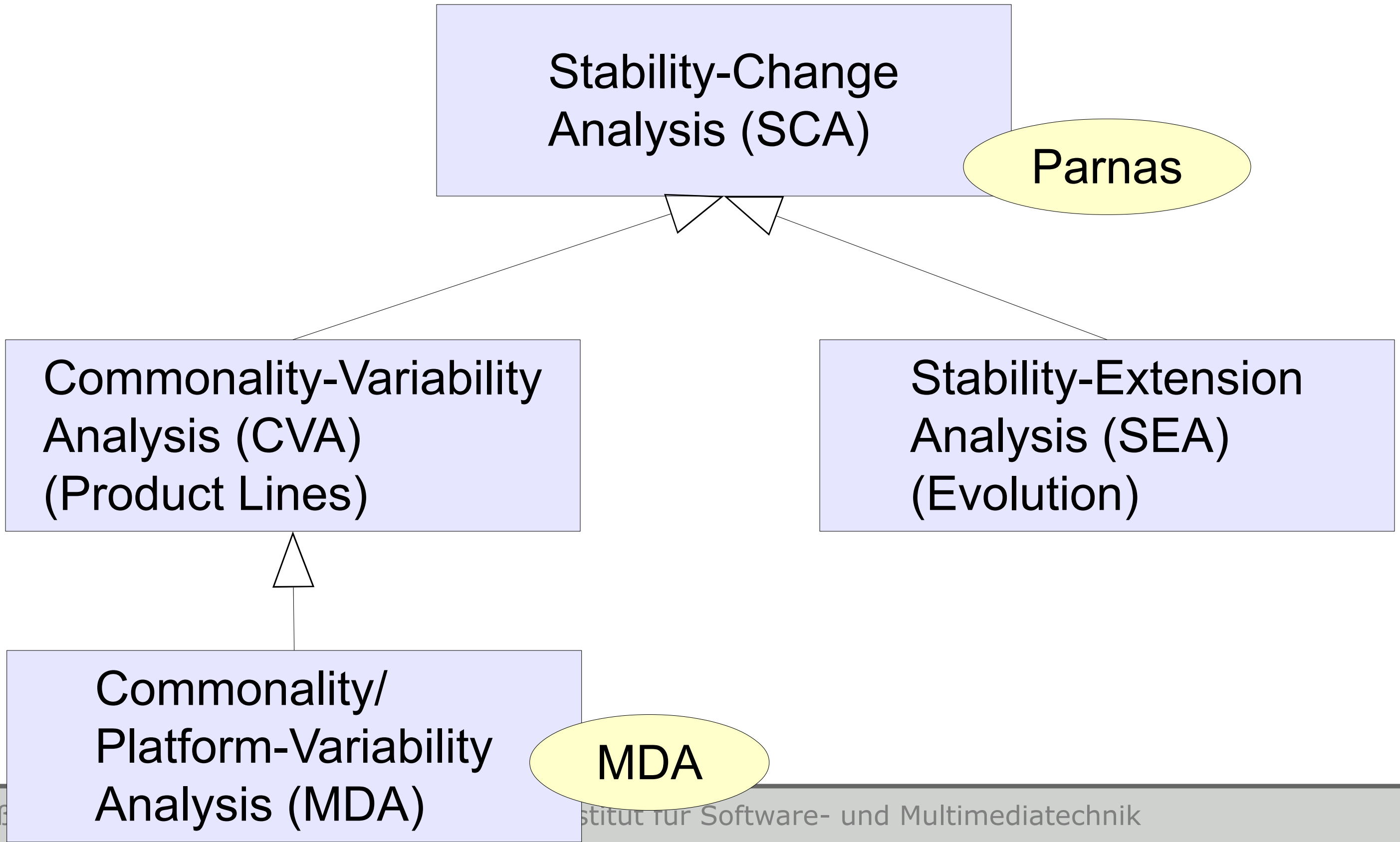
- The Parnas principle has been refined in **Commonality-Variability Analysis**
- Finding common assets for a product line
  - Separating them from variable assets
- CVA can be realized with many technologies

## What is Change?



# SCA and its subclasses

- [Parnas 72] tells us that commonality/variability analysis (CVA) is just a special case of *Stability/Change Analysis (SCA)*
- Variability is *preplanned,* evolution is *unforeseen*


Stability-Change Analysis (SCA)
Parnas
Commonality-Variability Analysis (CVA) (Product Lines)
Stability-Extension Analysis (SEA) (Evolution)
Commonality/ Platform-Variability Analysis (MDA)
MDA

# Planning Variations - Unforeseen Extensions

- Planning relies on *variation points*
  - Templates with slots
  - Frameworks with Pree-like metapatterns

- Extension relies on *extension points*
  - Cores with *hooks* or *joinpoints*
  - Cores with extension *views*

## 40.5 Framework Evolution with Stability/Change Analysis (SCA)

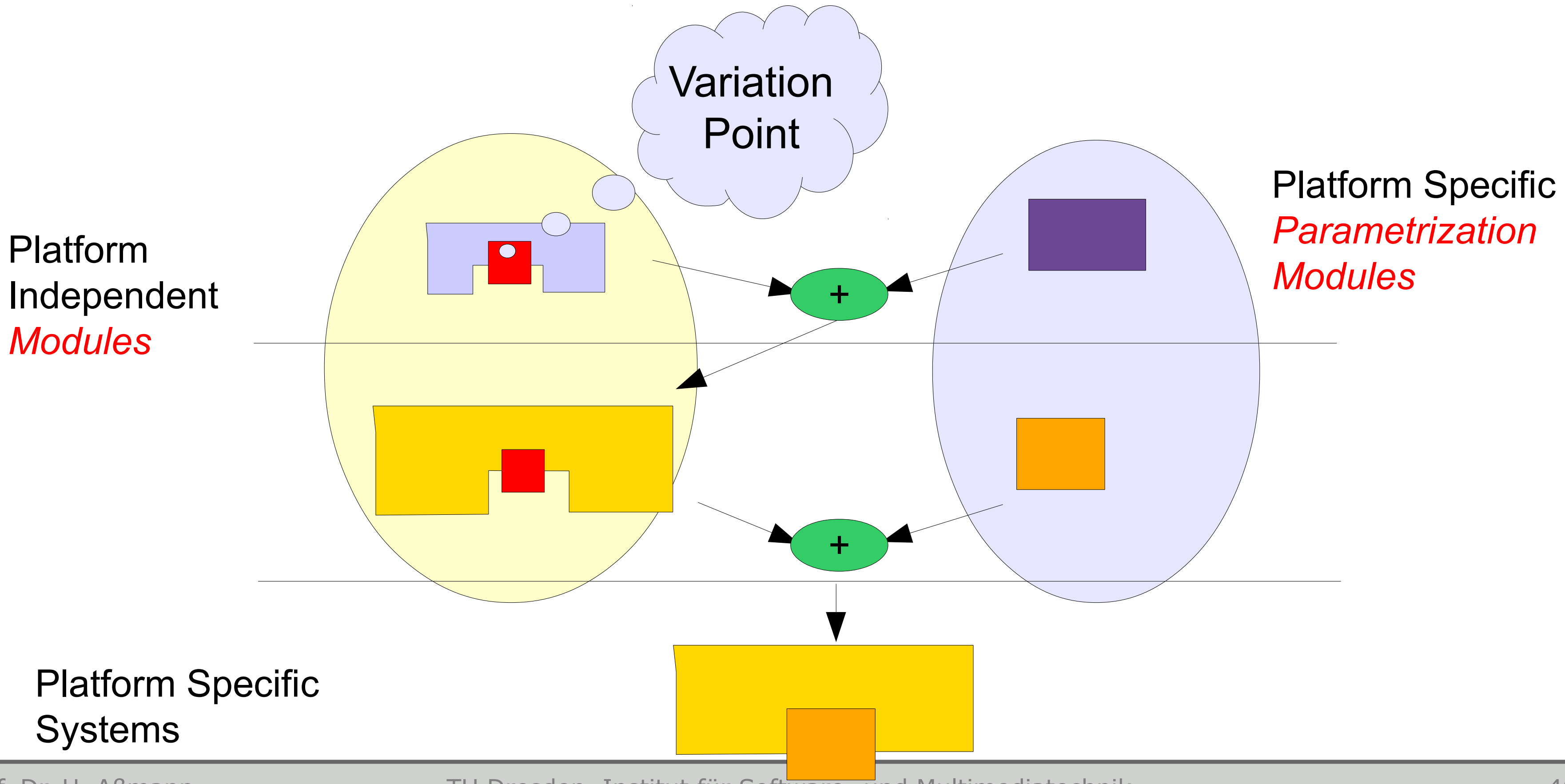
Planned and Unplanned MDA

## Problem

- Parnas requires *interfaces* between modules
  - which are not in MDA which relies on *implicit extension points (implicit hooks on models)*
- Can we extend the Parnas' principle to a parametric MDA?
  - For planned variants
  - Needs explicit composition interfaces between PIM and PSM

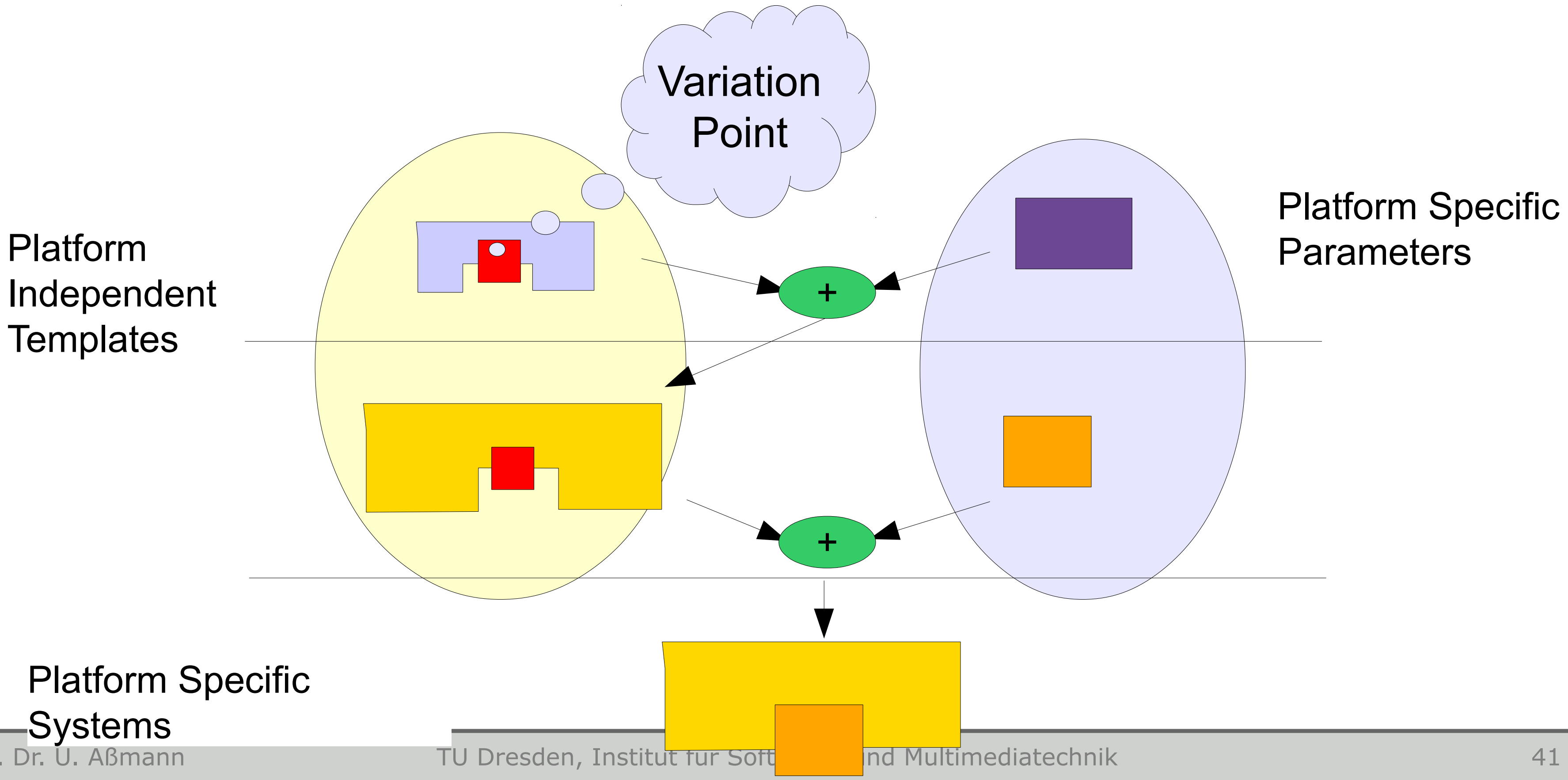
# Planned MDA with Modules (Original Idea of Parnas)

- Modules *parameterize* in *planned ways*
  - Module interface is composition interface
  - but not with crosscutting


Variation Point
Platform Independent *Modules*
Platform Specific *Parametrization Modules*
+
+
Platform Specific Systems

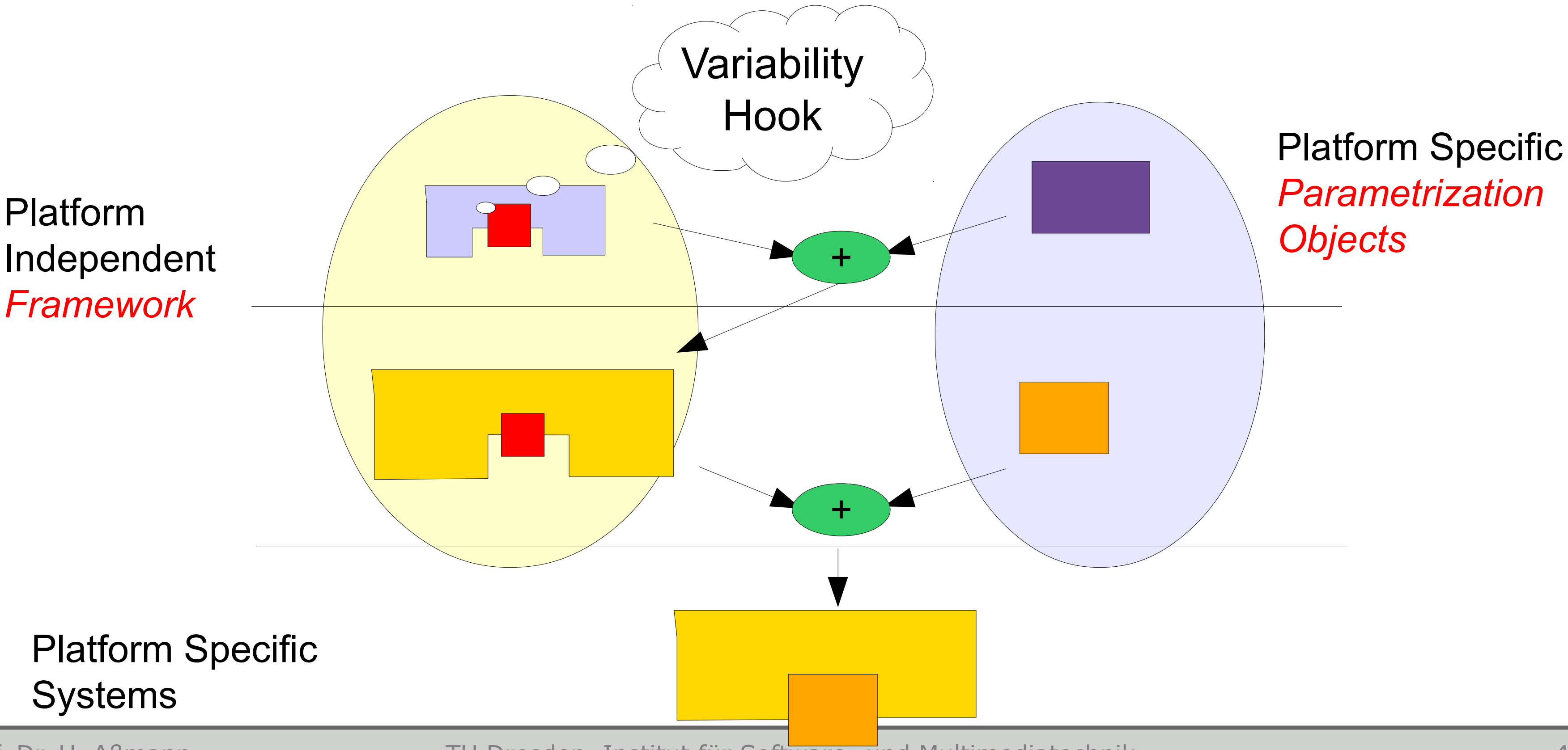
# Planned MDA (MDA with Generic Templates)

- Templates *parameterize* in *planned ways*
  - Template interface is composition interface
  - but not with crosscutting


Variation Point
Platform Independent Templates
Platform Specific Parameters
Platform Specific Systems
+
+

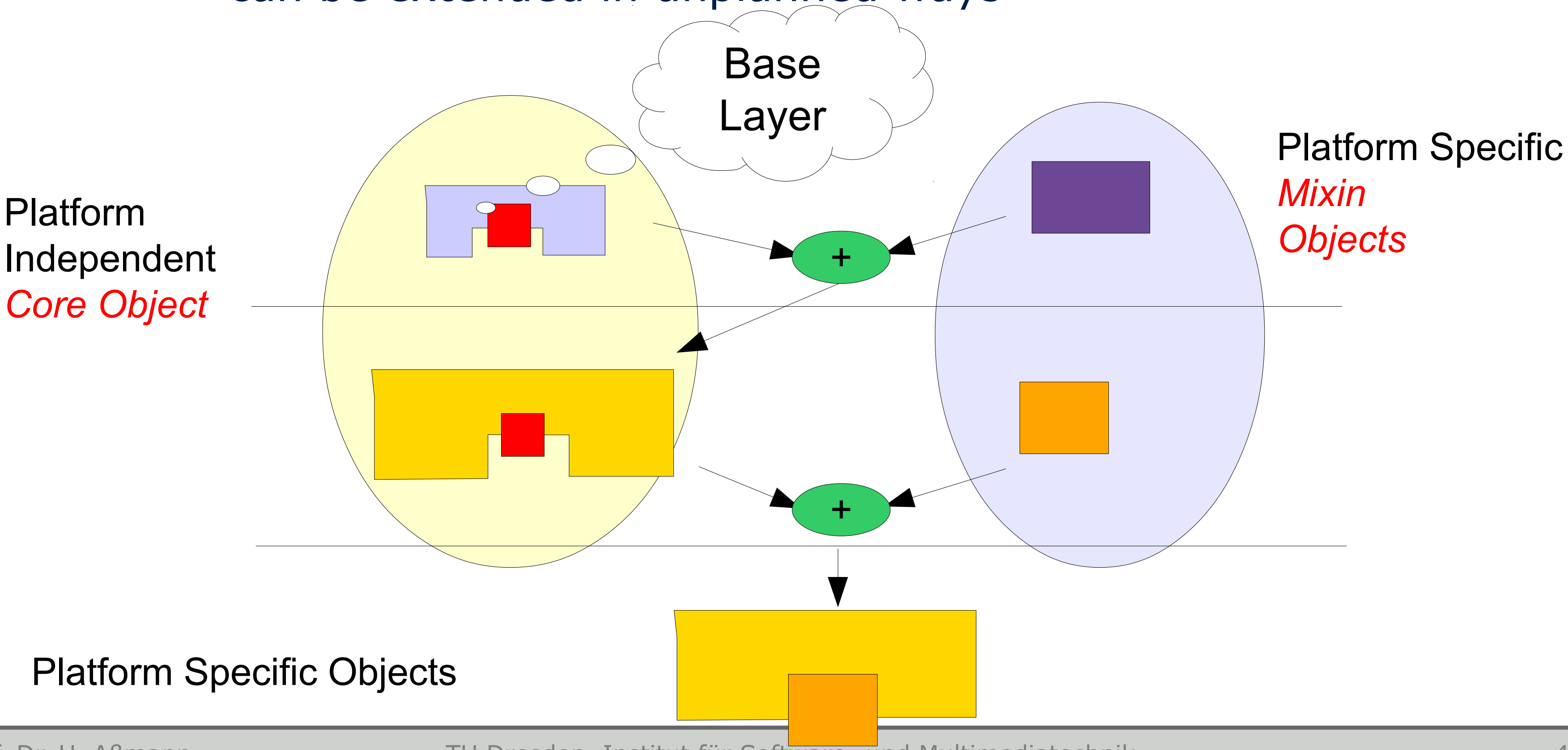
# Planned MDA with Code Frameworks

- Code Frameworks can be parameterized in planned ways
  - Variability design patterns form the interface
  - Variability hooks


Variability Hook
Platform Independent *Framework*
Platform Specific *Parametrization Objects*
+
+
Platform Specific Systems

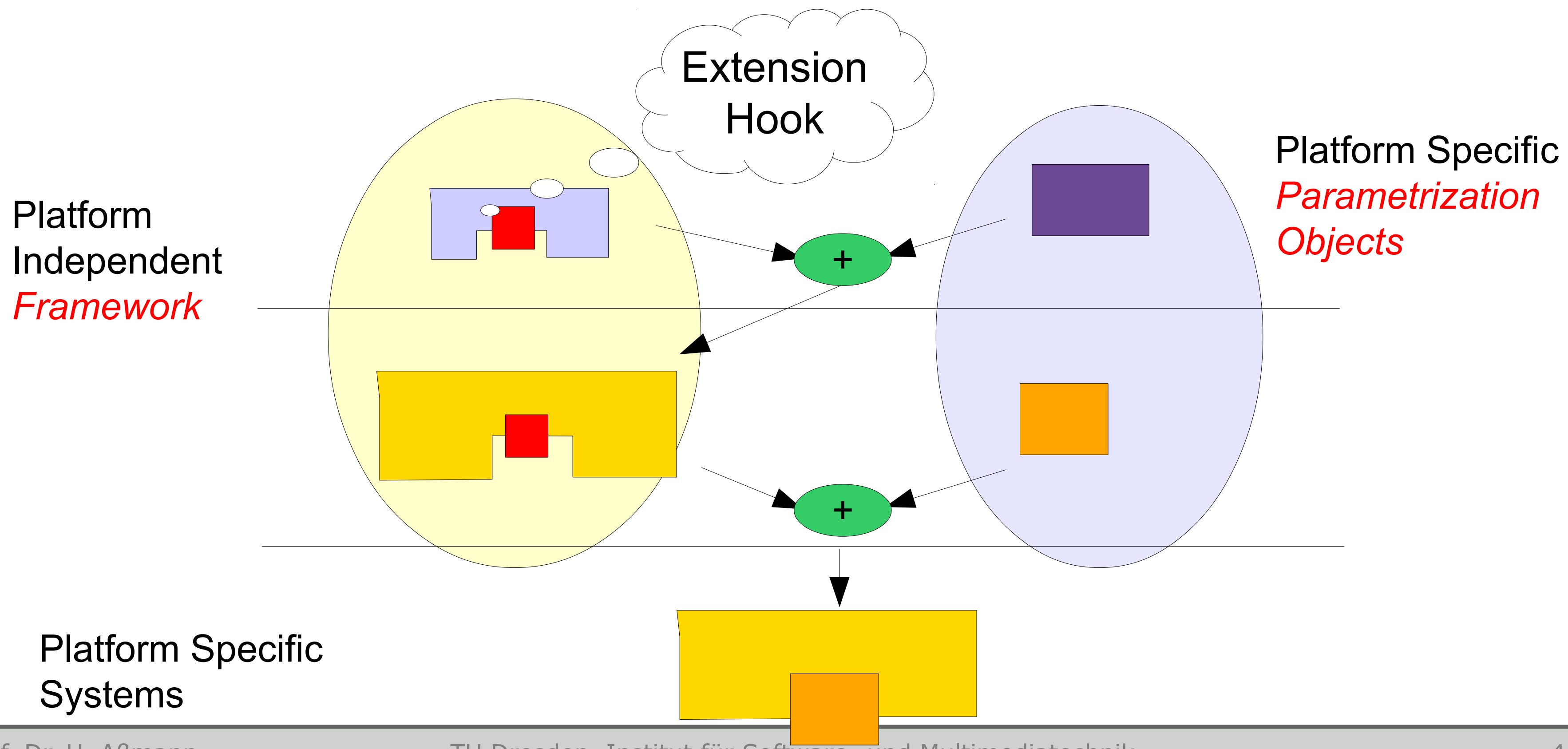
# Planned MDA with Role/Mixin Layers

- Role layers (mixin layers)
  - can be parameterized in planned ways
    - Variability by mixin or role-object composition
  - can be extended in unplanned ways



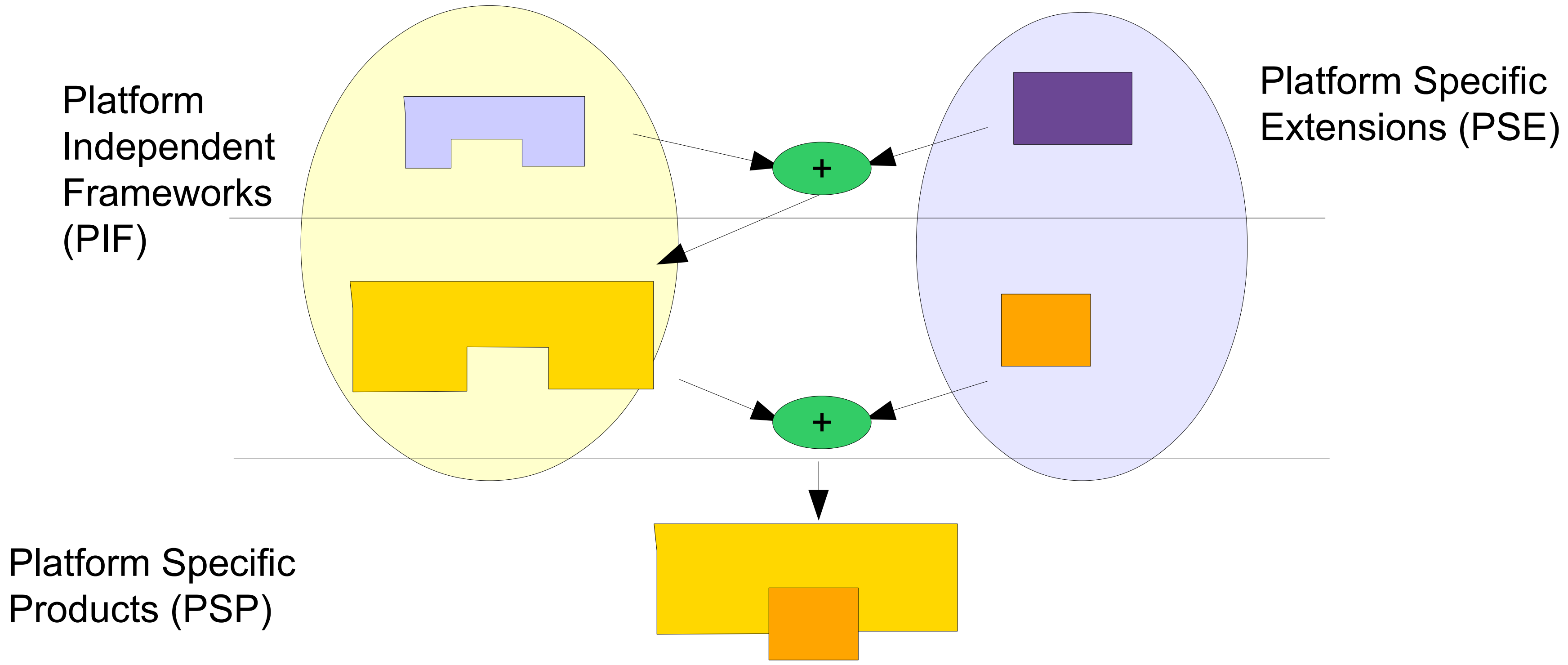
# Unplanned MDA with Code Frameworks

- Code Frameworks can be extended in unplanned ways
  - Extensibility design patterns form the interface
- n-T—H, T<=H framework hooks (extension points)


Extension Hook
Platform Independent *Framework*
Platform Specific *Parametrization Objects*
+
+
Platform Specific Systems

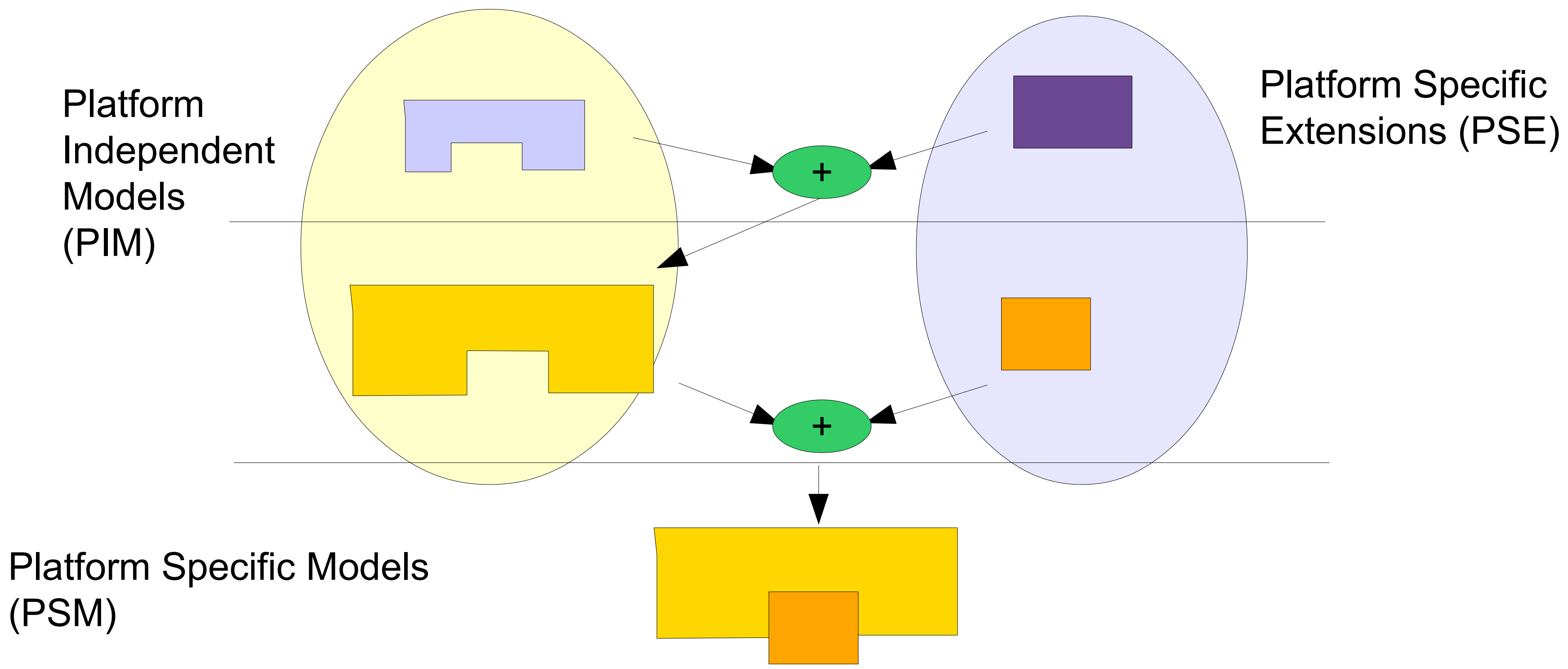
# Unplanned Extensible Eclipse

- Eclipse plugins *extend* in *unforeseen ways*
- n-T—H, T<=H framework hooks (extension points)


Platform Independent Frameworks (PIF)
Platform Specific Extensions (PSE)
+
+
Platform Specific Products (PSP)

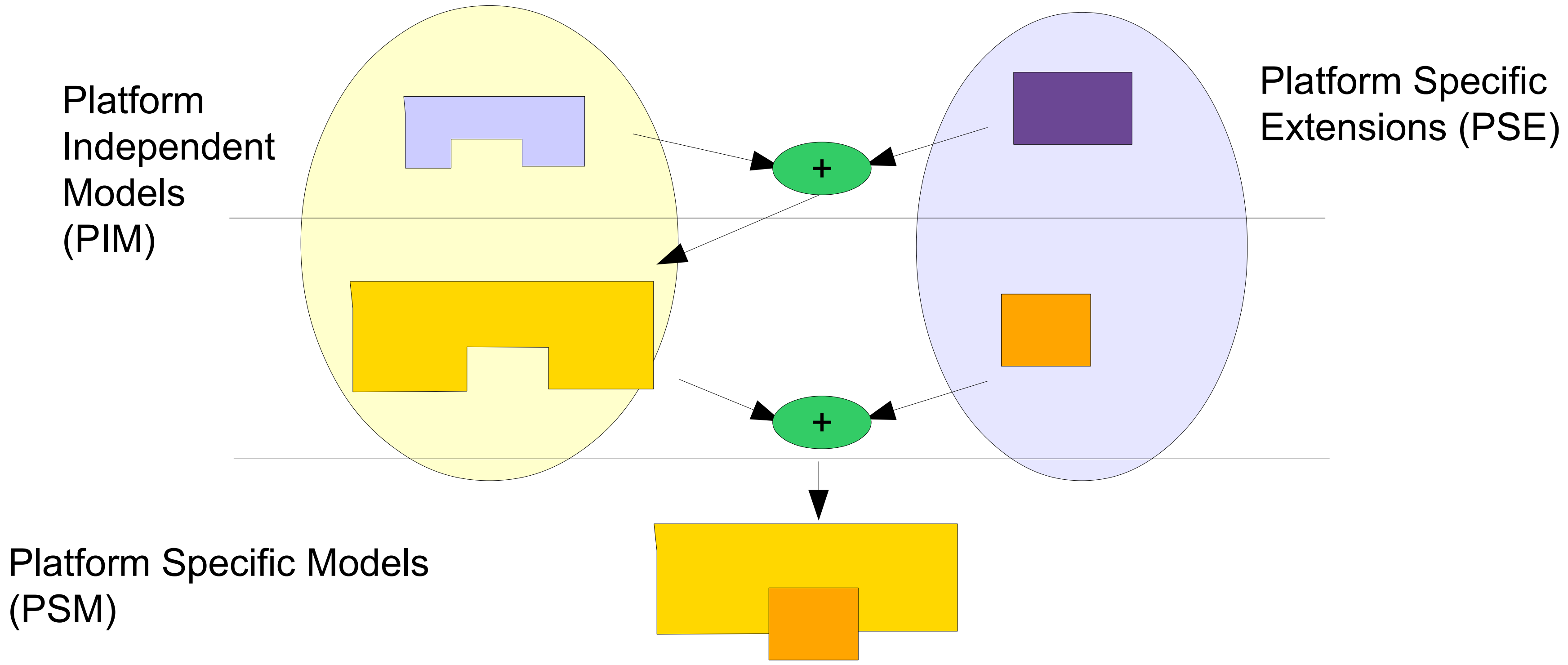
# Unplanned Extensible MDA

- Aspect-model weavers *extend* in *unforeseen ways*
- Using transformations


Platform Independent Models (PIM)
Platform Specific Extensions (PSE)
+
+
Platform Specific Models (PSM)

# Unforeseeable Extensible MDA

- Hyperslices *extend* in *unforeseen ways,*
  - but not with crosscutting


Platform Independent Models (PIM)
Platform Specific Extensions (PSE)
+
+
Platform Specific Models (PSM)

# Planning Variations - Unforeseen Extensions

## Frameworks with MDA or CBSE

- Planning relies on *variation points*
- MDA with
  - templates
  - modules
  - blackbox component models
  - variable frameworks
  - Role layers

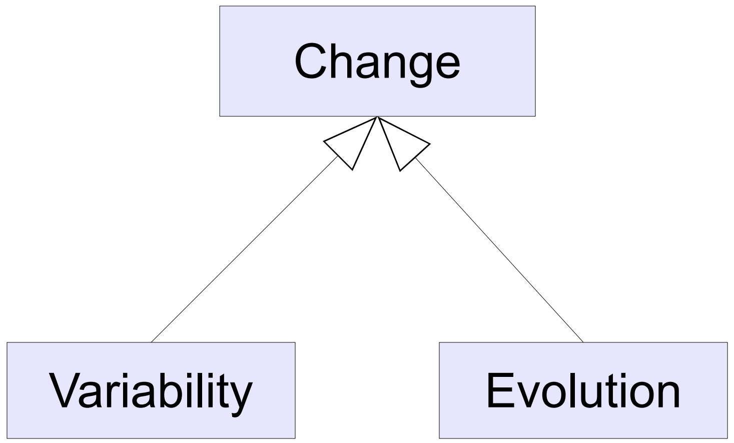
- Extension relies on *extension points*
- MDA with
  - extensible frameworks
  - role layers
  - views
  - aspects

# Variation and Extension Facets

- We can distinguish several facets of variation and extension

- Time of instantiation (binding time)
- Time of exchange
- Granularity of binding point
- Contract on extension

# Change



## Multi-Stage Frameworking

- Multi-stage frameworking generalizes the framework approach of this course
  - Concrete composition system does not matter
  - Modeling or CBSE approach, applicable to all component models

- All composition systems possible:
- Greybox composition
  - PIM and PSE are *greybox components*
- Blackbox composition
  - PIM and PSE are *blackbox components*
  - Without unforeseen extension – everything is planned
  - You can do it in C, but you must plan carefully

# Conclusion: It's Your Choice!

- Multi-staged frameworking can be employed for many different variability technologies
  - Both for CVA and SCA!
  - With many different CBSE technologies
  - MDA is multi-staged frameworking with translational model frameworks

- Multi-staged frameworking for commonality/variability-based design can be *planned*
  - Using a composition interface with declared variation points
- Multi-staged frameworking for stability/extension-based design can be done for *unforeseen extensions*
  - using a composition interface with *implicit hooks (join points)*

# The End