



Requirements in Software Product Lines

ARLIND BEJTA

Agenda

A framework for Software Product Line Engineering

Generic semantics of feature diagrams



A framework for Software Product Line Engineering

Idea

- ▶ Software Product Line → Software Families
- ▶ Generate members of the same family quickly and efficiently
- ▶ Systematically and considering the economic value

Framework Overview

- ▶ Two Development Processes:

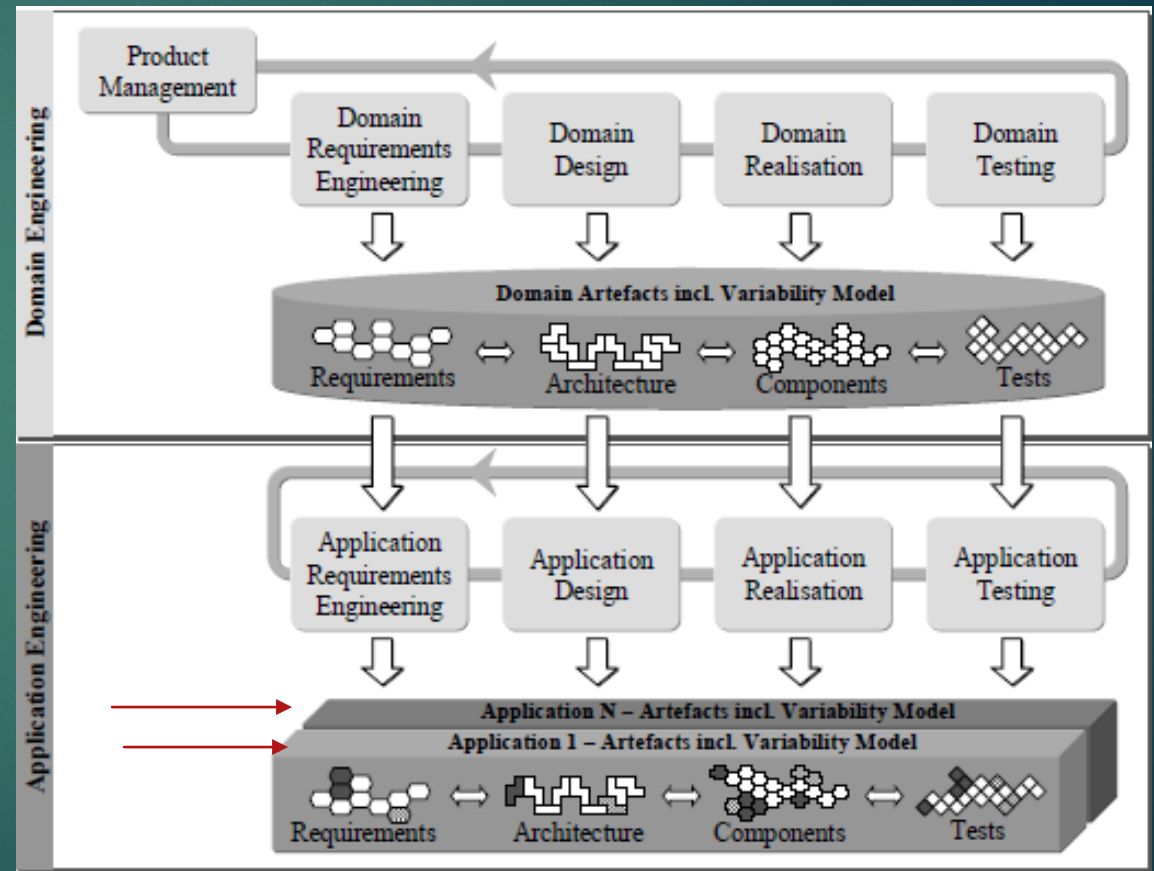
- ▶ Domain Engineering
- ▶ Application Engineering

- ▶ Advantage:

- ▶ Robust platform
- ▶ Tailored application

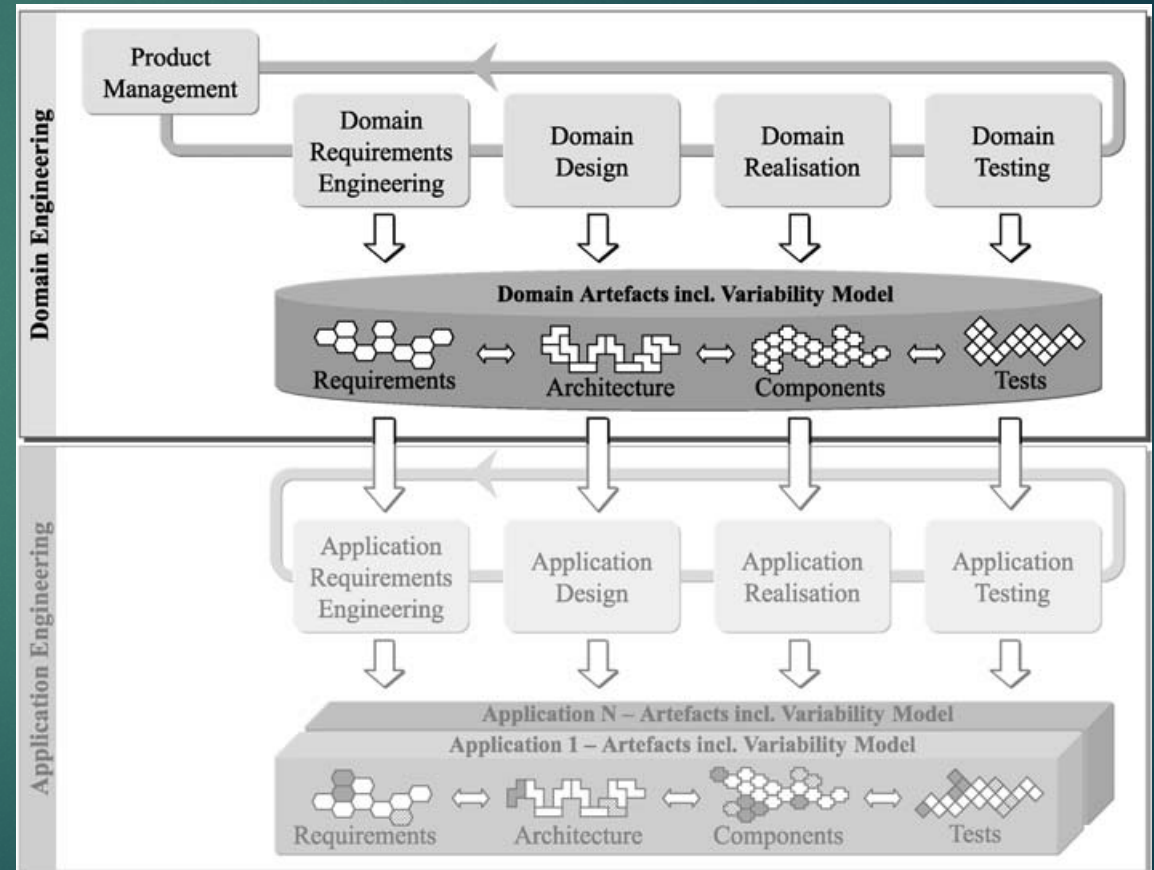
- ▶ Domain Artefacts

- ▶ Application Artefacts



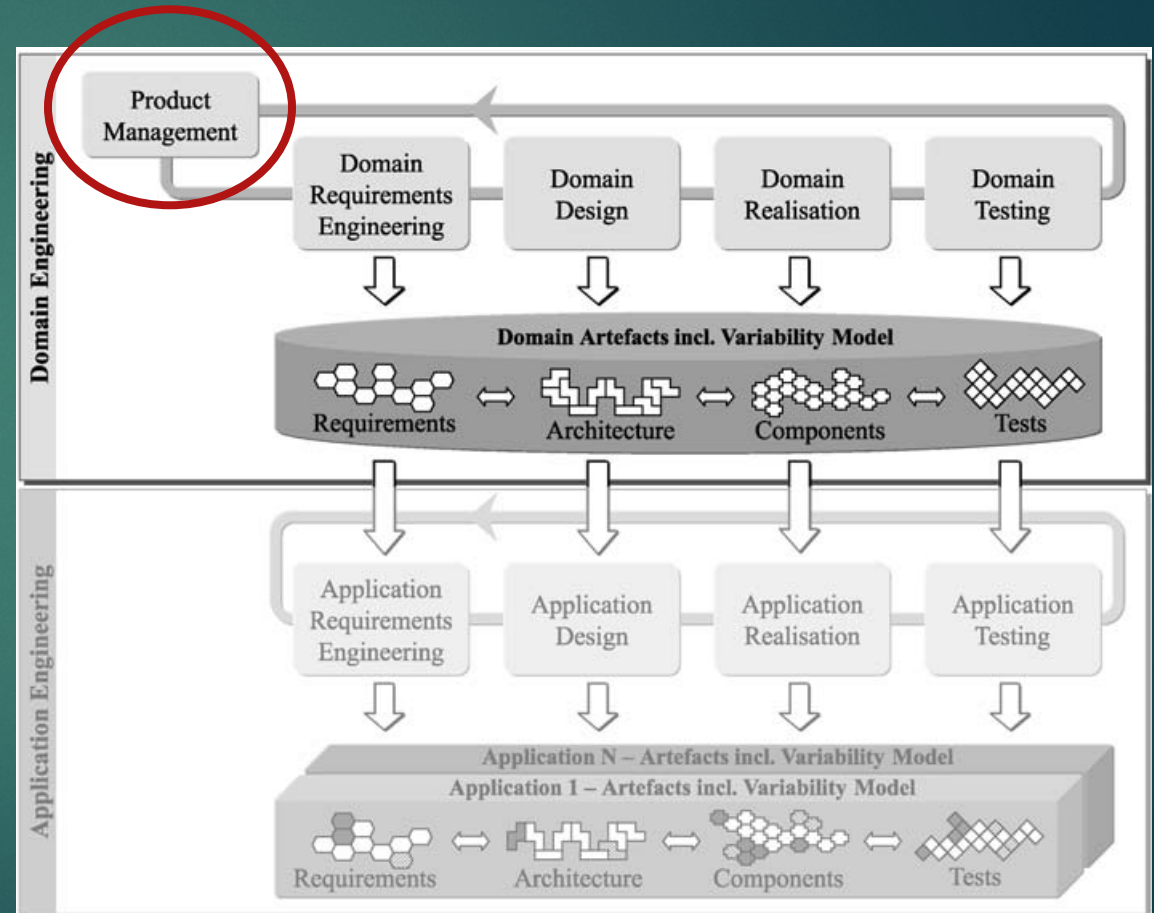
Domain Engineering Process

- ▶ Goals:
 - ▶ Define the commonality and the variability of the software product line.
 - ▶ Define the set of applications the software product line is planned for.
- ▶ Sub processes:
 - ▶ Product Management, Domain Requirements Engineering, Domain Design, Domain Realization, Domain Testing.



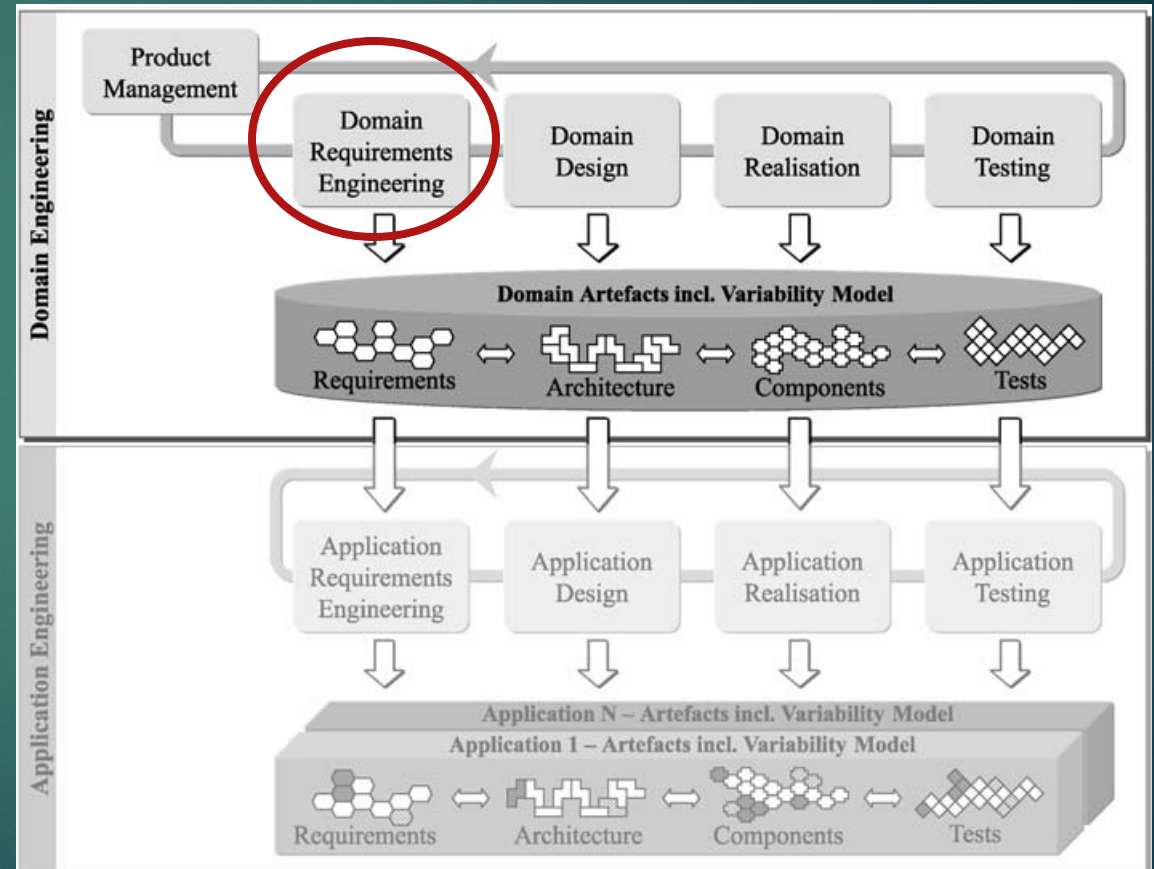
Product Management

- ▶ Input: company goals
- ▶ Output: product roadmap
- ▶ PM for product lines vs. PM for single systems



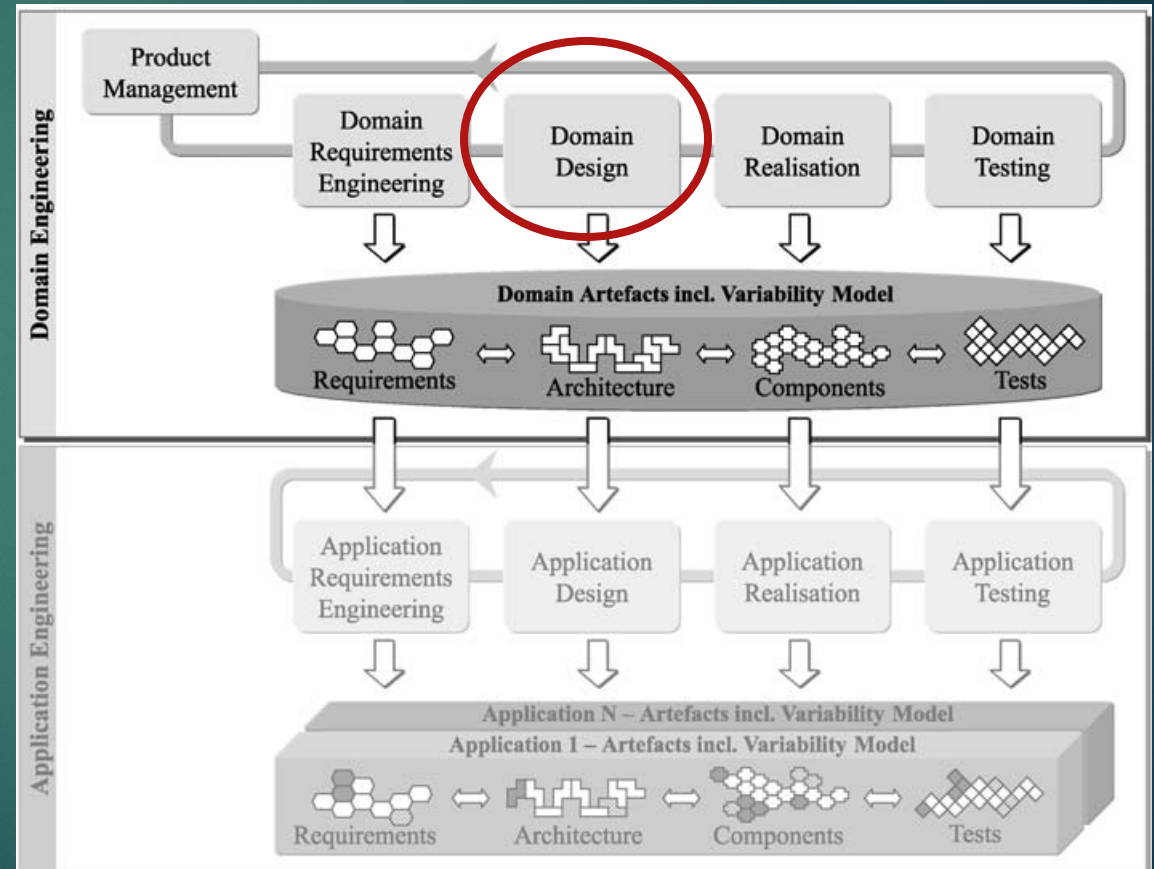
Domain Requirements Engineering

- ▶ Input: product roadmap
- ▶ Output: textual and model-based requirements
- ▶ Differences from single-system engineering:
 - ▶ Common to all application



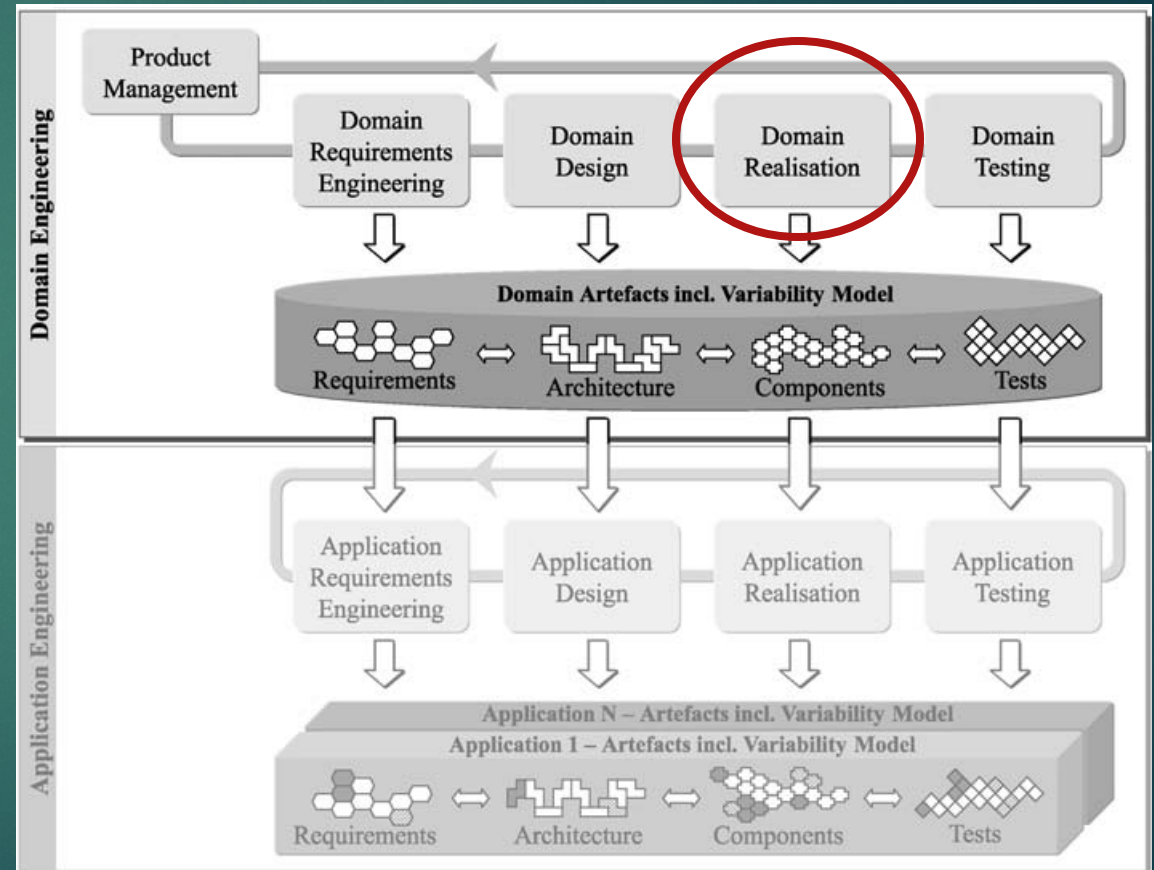
Domain Design

- ▶ Input: domain requirements and the variability model
- ▶ Output: reference architecture and a refined variability model
- ▶ Differences from single-system:
 - ▶ Flexibility consideration
 - ▶ Common rules for the development of specific application



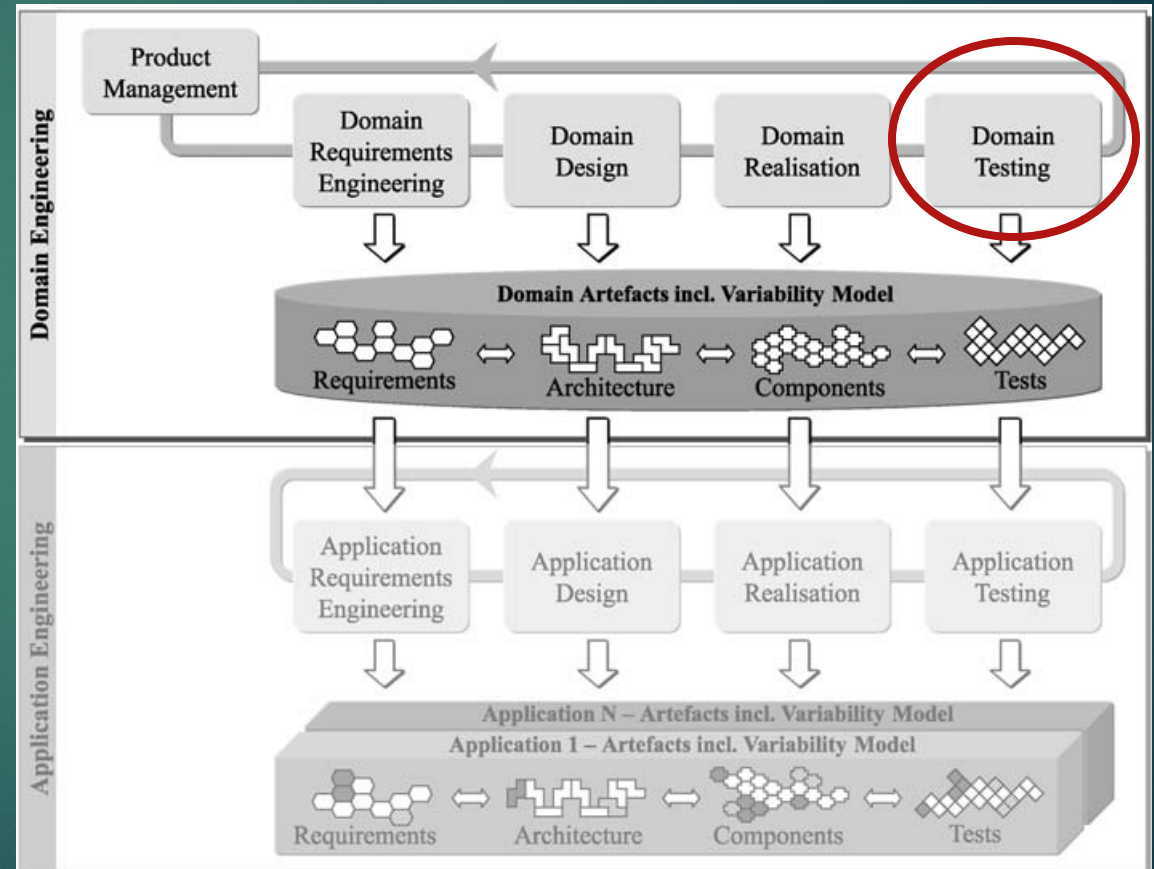
Domain Realization

- ▶ Input: reference architecture
- ▶ Output: detailed design and implementation asses of reusable software components.
- ▶ Differences from single-system:
 - ▶ Components are reused in different contexts



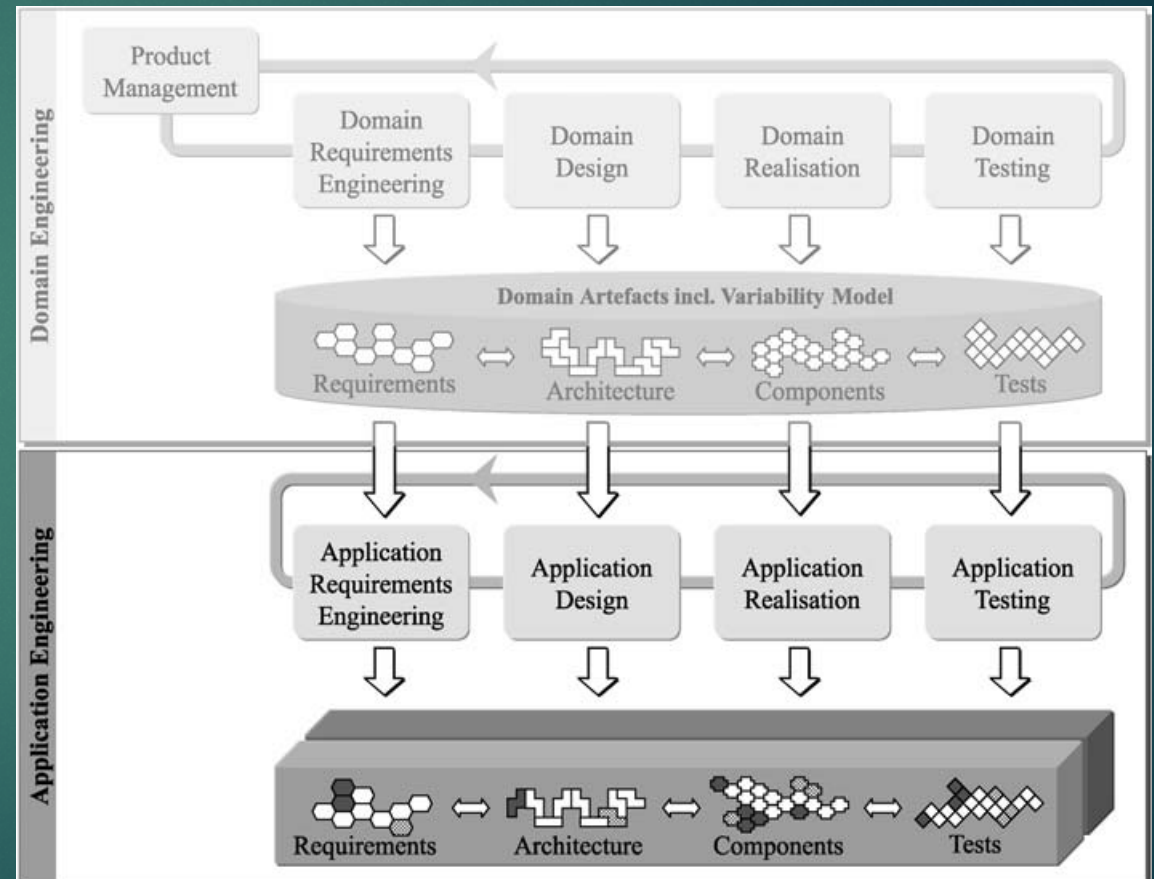
Domain Testing

- ▶ Input: domain requirements, reference architecture, component and interface design, and reusable components
- ▶ Output: test results
- ▶ Differences from single-system:
 - ▶ No running application to be tested in domain testing



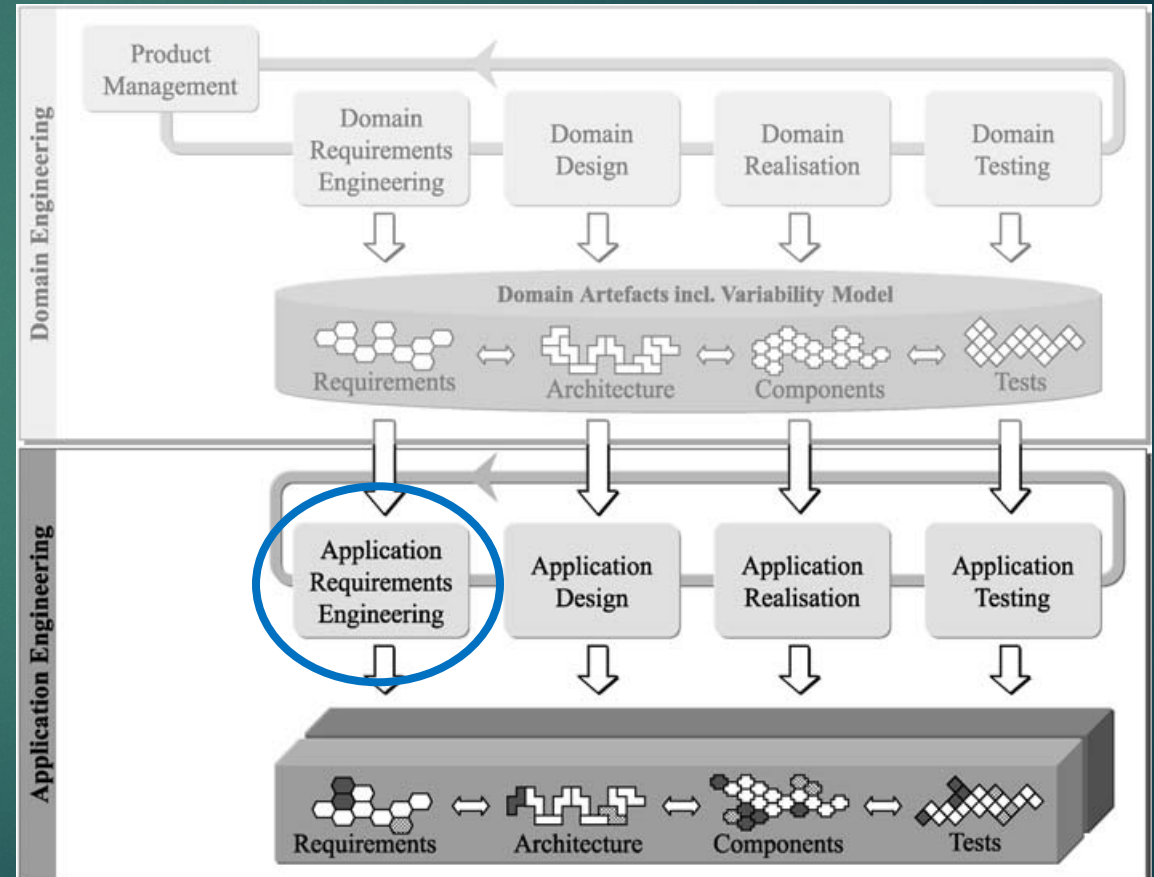
Application Engineering

- ▶ Goals:
 - ▶ Reuse as much as possible of domain assets
 - ▶ Document the application artefacts
 - ▶ Estimate the impacts of the differences between application and domain requirements on architecture, components and tests.
- ▶ Sub processes: application requirements engineering, application design, application realization, and application test



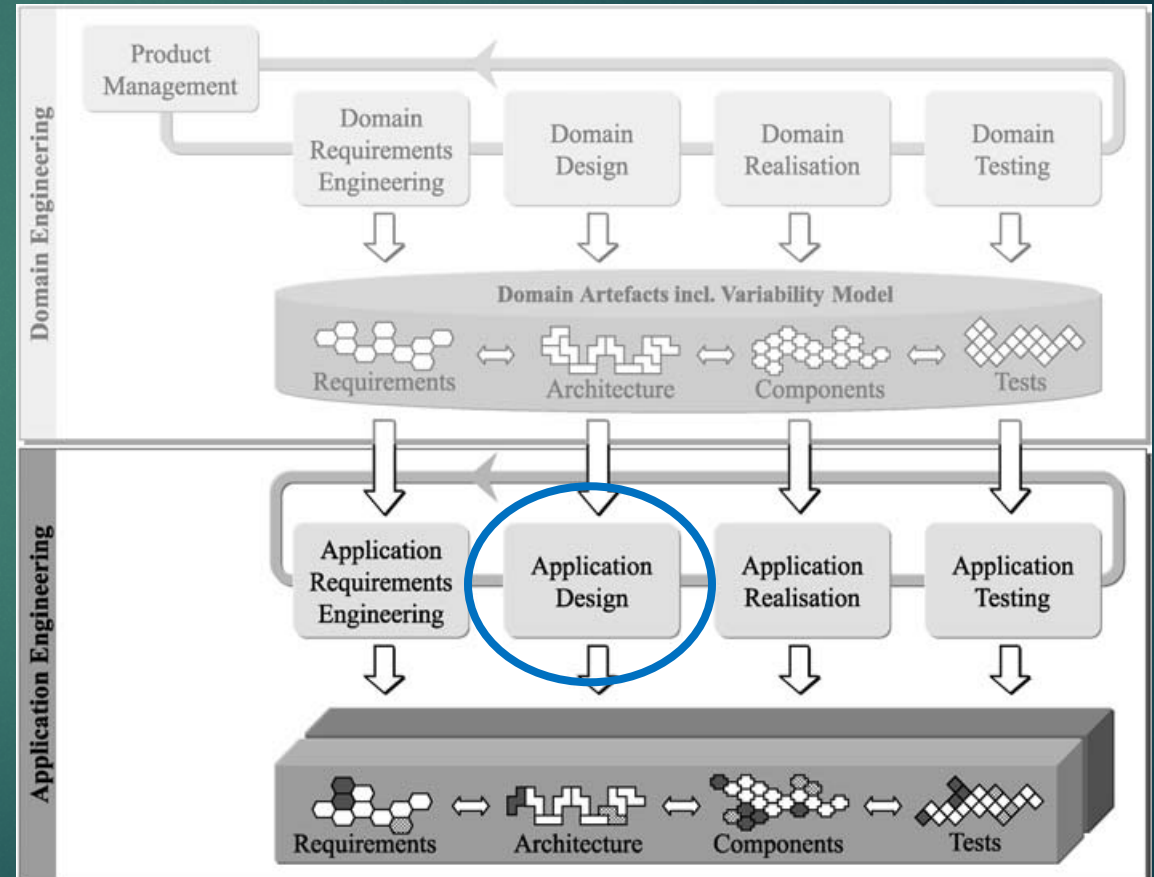
Application Requirements Engineering

- ▶ Input: domain requirements and product roadmap
- ▶ Output: requirements specification for particular application
- ▶ Differences from single-system:
 - ▶ Most of the requirements derive from domain requirements and are not elicited as new.



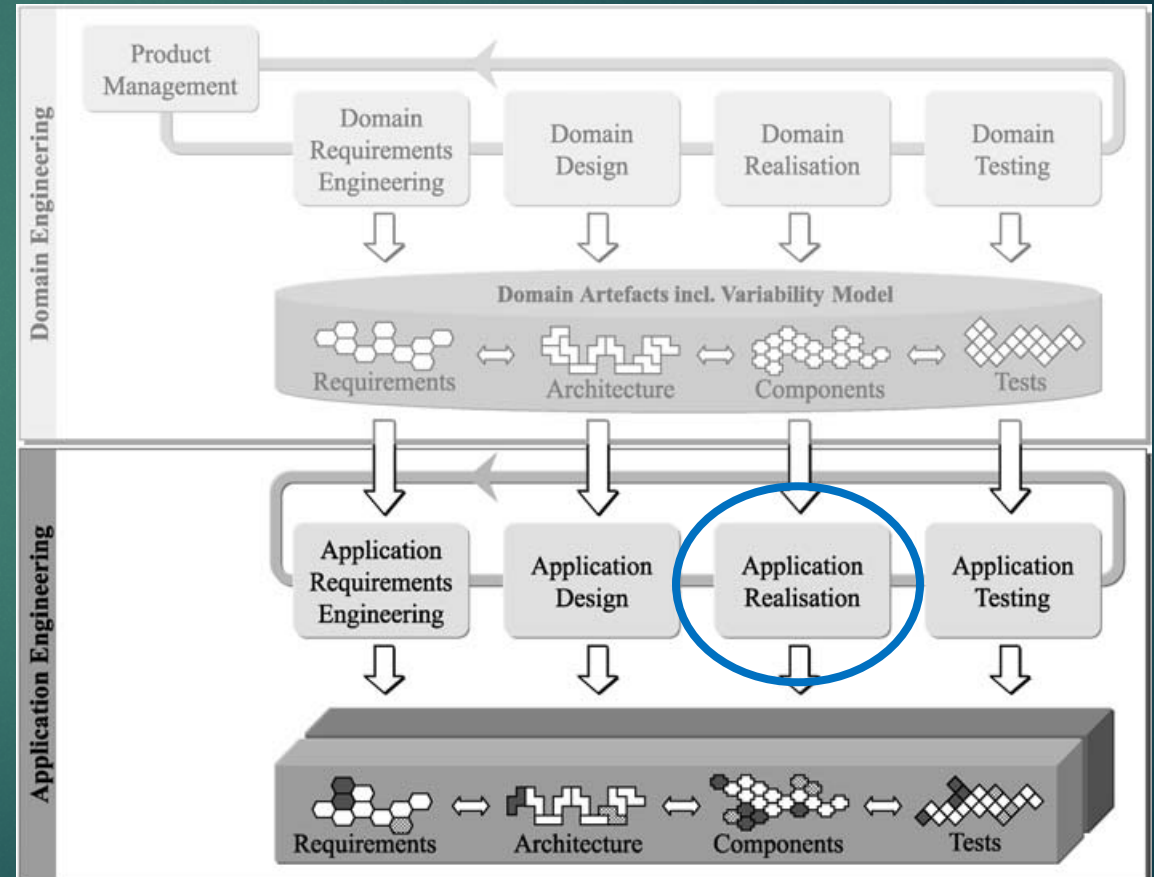
Application Design

- ▶ Input: reference architecture and application requirements specification
- ▶ Output: application architecture for the application at hand
- ▶ Differences from single-system :
 - ▶ Application architecture derives from the reference architecture



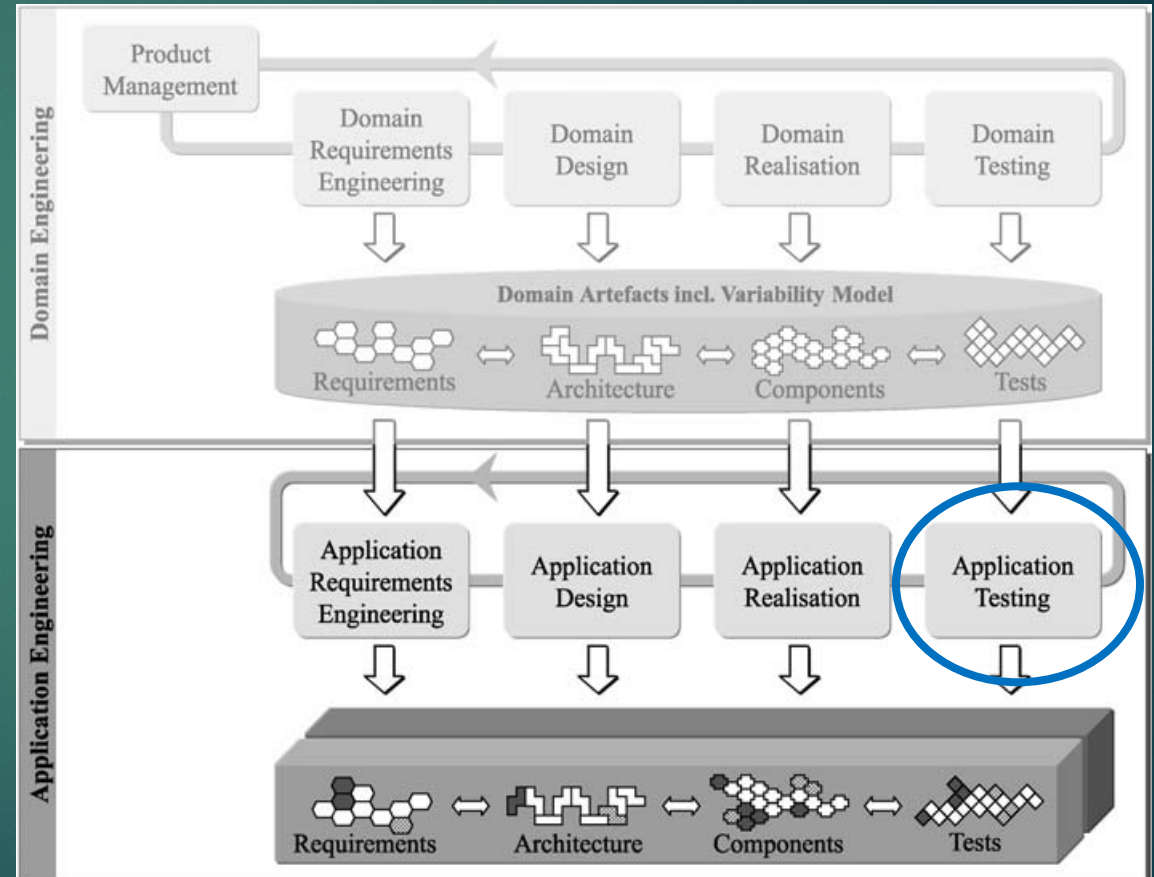
Application Realization

- ▶ Input: application architecture and reusable realization artefacts
- ▶ Output: running application
- ▶ Differences from single-system:
 - ▶ Application-specific realization must fit into the overall structure.



Application Testing

- ▶ Input: different application artefacts
- ▶ Output: test report with results of all tests that have been performed
- ▶ Differences from single-systems:
 - ▶ Many test artefacts derive from platform and are not created anew



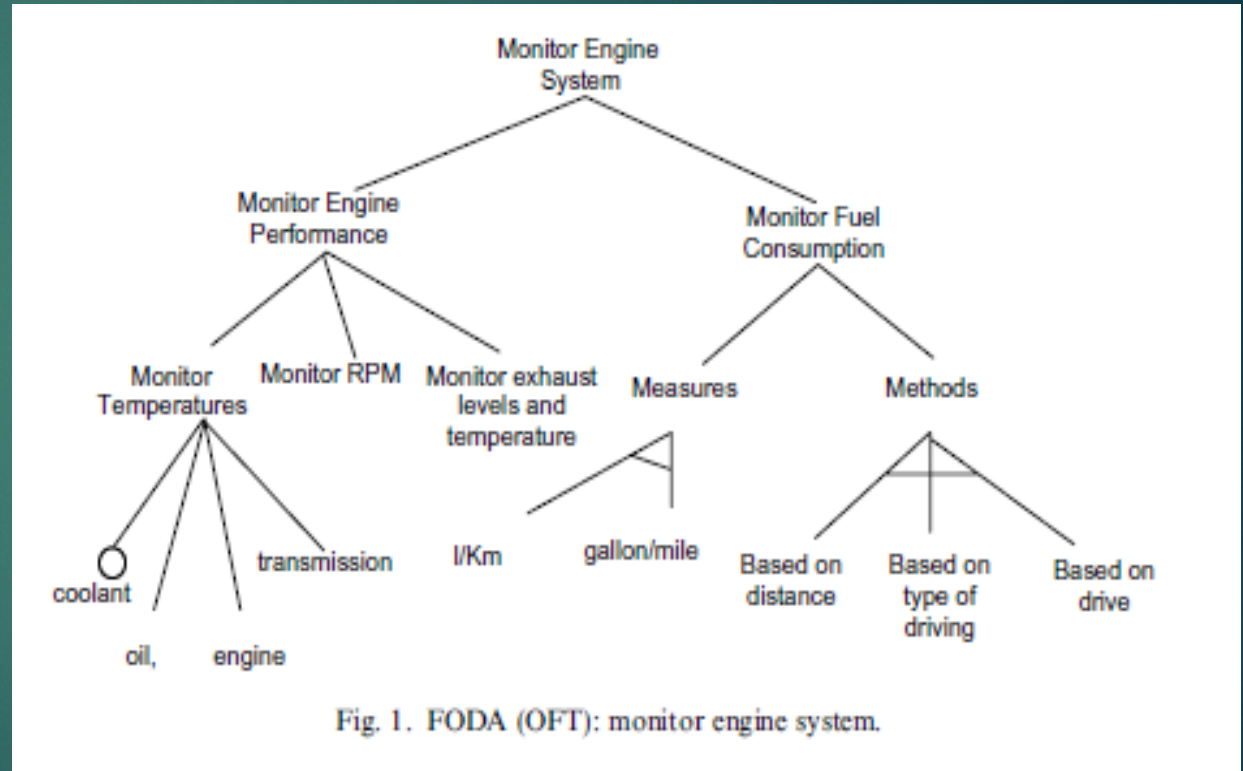
Generic semantics of feature diagrams

Feature Diagrams (FDs)

- ▶ First introduced by Kyo C. Kang as part of the FODA (Feature-Oriented Domain Analysis) method in 1990.
- ▶ Valuable tool to address the complexity of engineering software product lines.
- ▶ FODA is an analysis to identify features in a domain that is covered by a particular Software Product Line.
- ▶ FDs are an essential means of communication between domain and application engineers

FODA FD / OFT

- ▶ FODA features are nodes of a graph represented by strings and various types of edges.
- ▶ *Monitor Engine System* is root feature/concept
- ▶ *Edges – decompose*



OFT extensions

- ▶ FORM (OFD) – Feature-Oriented Reuse Method (Original Feature Diagrams)
 - ▶ Motivation: Enlarge the scope of feature modelling
 - ▶ Can be also DAG (Directed Acyclic Graphs)
- ▶ FeatureRSEB – Combination of FODA and Reuse-Driven Software Engineering Business method.
 - ▶ Use case drive process
 - ▶ DAGs
- ▶ PLUS (Product Line Use case modelling for System and Software engineering)
 - ▶ Based on FeatureRSEB
 - ▶ Combines FD's and uses cases to show a high level view of product family
 - ▶ Tree structure

Free Feature Diagrams (FFDs)

- ▶ Generalization of all the variants of FDs
- ▶ Is a formal framework to formally define, study, or compare FD languages
- ▶ Defined semantics on the basis of the abstract syntax
 - ▶ Conversion from concrete syntax to abstract
 - ▶ 1. mathematical notation (set theory)
 - ▶ 2. meta-model (i.e. UML Class Diagram with OCL constraints)
- ▶ Primitive and compound features (nodes)

Questions?



Describe an experience you have
had with software reuse?

Thank You!