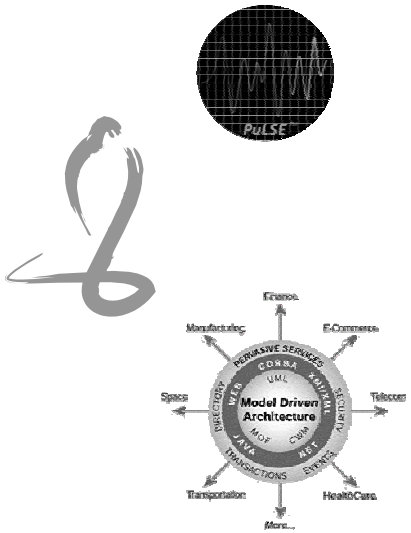
Model-driven Product Line Architectures - Part I: SPL, CBD, and MDA

NoDE, Erfurt

September 22, 2003



Fraunhofer Institut Experimentelles Software Engineering

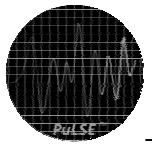
Stefan Kettemann
Dirk Muthig

Sauerwiesen 6
D-67661 Kaiserslautern
Germany

Tutorial Organization (1/2)

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

- *Part I*
  - Introduction (10 min)
    - → Who we are and why we are here
    - → Who you are and why you are here
  - Background (30 min)
    - → KobrA Method
    - → PoLITe project
  - Component Modeling and Frameworks (25 min)
  - Model-driven Component Implementation (25 min)

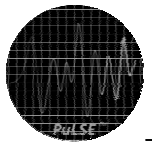


Copyright © Fraunhofer IESE 2003

Tutorial Organization (2/2)

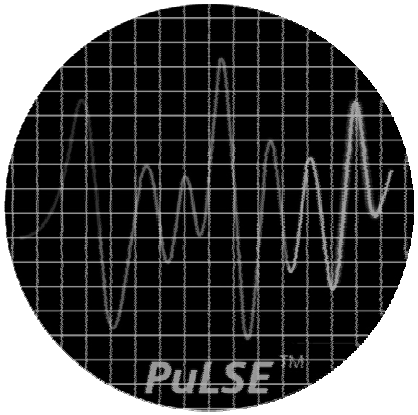
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

- *Part II*
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - J2EE (30 min)
  - COM (10 min)
  - .NET (10 min)
  - Variability Management (10 min)
  - Conlusions and Discussion (15 min)

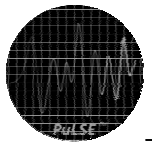


Copyright © Fraunhofer IESE 2003

Part I
Introduction
Background
Component Modeling
Component Implementation



Introduction



Copyright © Fraunhofer IESE 2003

The Fraunhofer Institute for Experimental Software Engineering

Part I

Introduction

Background

Component Modeling

Component Implementation

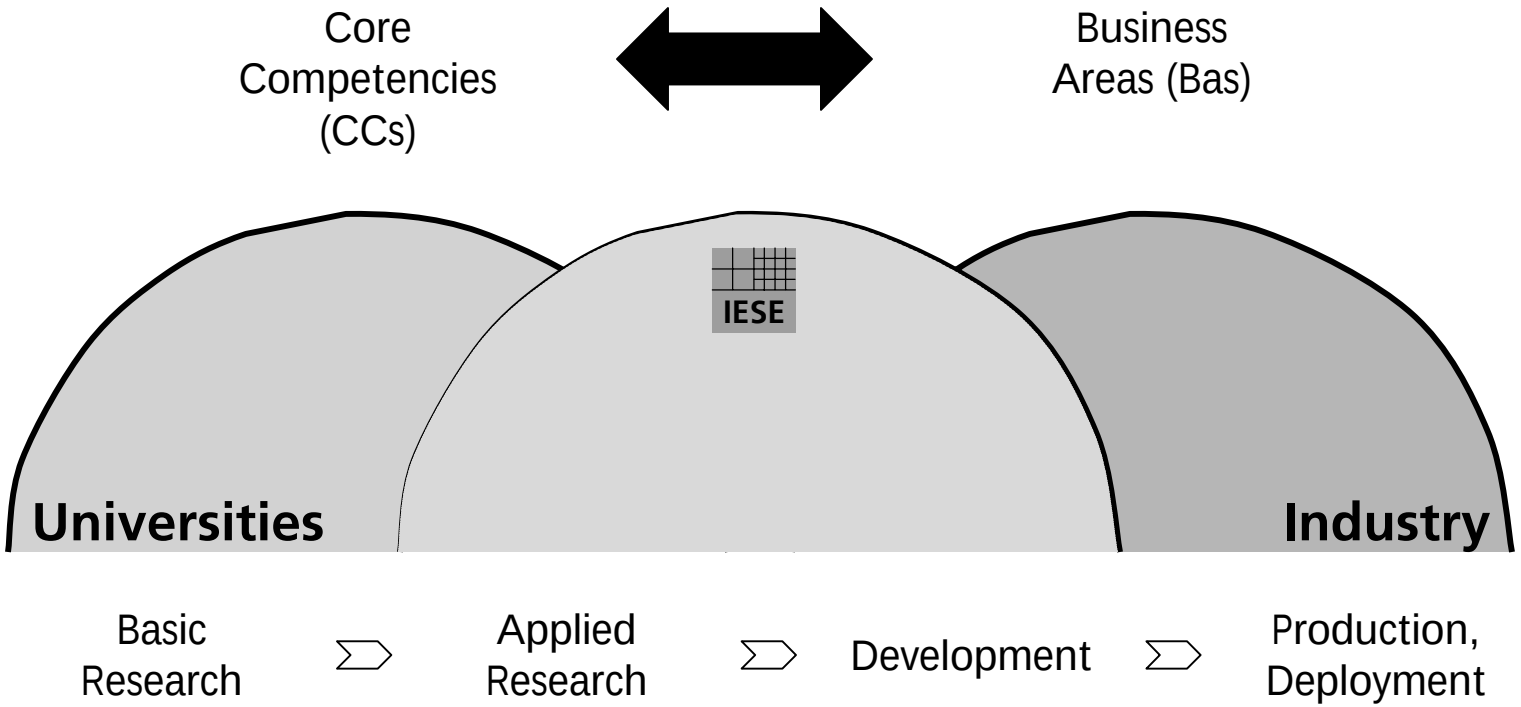




Copyright © Fraunhofer IESE 2003

Technology Transfer

Part I
Introduction
Background
Component Modeling
Component Implementation

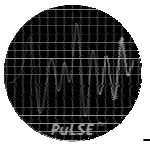


Copyright © Fraunhofer IESE 2003

IESE Core Competences (1/2)

Part I
Introduction
Background
Component Modeling
Component Implementation

- **Experimentation (EXP)**
  (Experimental designs, QIP/GQM/EF, Analysis techniques)
- **Requirements and Usability Engineering (RUE)**
  (Requirements Engineering, Usability Engineering)
- **Component-based Software Engineering (CBE)**
  (Component-based software development mainly in the field of embedded and realtime systems)
- **Software Product Lines (SPL)**
  (Architecture recovery, Architecture development & evaluation, Scoping & modeling of domains & decisions)



Copyright © Fraunhofer IESE 2003

IESE Core Competences (2/2)

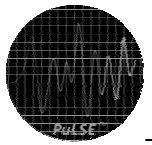
Part I
Introduction
Background
Component Modeling
Component Implementation

- **Quality and Process Engineering (QPE)**
  (Goal-oriented measurement, Descriptive process modeling, Goal-oriented product/process assessment, Subcontracting, COTS Acquisition, Risk Management)

- **Systematic Learning & Improvement (SLI)**
  (SPI planning & setup, EF buildup and Organizational Learning)

- **Certifiable Education & Training in SE (CET)**
  (Evaluation and Certification, Technology-enabled Learning, Simulation-based learning and decision support)

- **IT Security (ITS)**
  (Security Assessments, EF for Security, Automated checking of infrastructures)

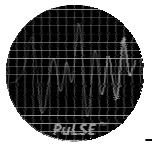


Copyright © Fraunhofer IESE 2003

Vision (CC SPL)

Part I
Introduction
Background
Component Modeling
Component Implementation

- Product Line Engineering is generally recognized as the foundation of successful software organizations.
  - PLE is the default approach in practice rather than an exception.
- PuLSE technology is well-known to industry as the key enabler of successful product line engineering in practice.
- PuLSE technology is recognized in the research community as a representation of the state-of-the-art in product line engineering.
- PuLSE team members are known as excellent software/product-line engineers for adding value to software organizations by bringing in software product-line-engineering, as well as single-system-engineering, expertise.



Copyright © Fraunhofer IESE 2003

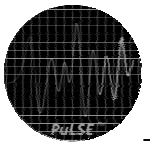
Mission (CC SPL)

Part I
Introduction
Background
Component Modeling
Component Implementation

SPL supports organization in achieving an optimally efficient software development and maintenance approach.
The approach plans and manages related products of an organization as one entity, a software product line.

The SPL approach covers all stages of the software life cycle, as well as most starting points (in different BAs).
Hence, SPL supports

- The proactive planning, assessment, and management of the approach from a business-value perspective,
- The systematic construction, maintenance, and evolution of a product line infrastructure, and
- The integration of existing artifacts into the product line for the future through the recovery of knowledge, architectures, or components.

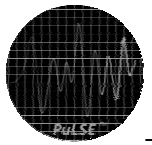


Copyright © Fraunhofer IESE 2003

Schedule

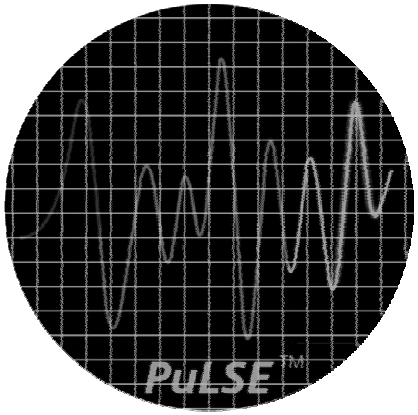
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

- *Part I*
  - Introduction (10 min)
    - → Who we are and why we are here
    - → Who you are and why you are here
  - Background (30 min)
    - → KobrA Method
    - → PoLITe project
  - Component Modeling and Frameworks (25 min)
  - Model-driven Component Implementation (25 min)

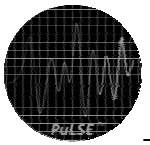


Copyright © Fraunhofer IESE 2003

Part I
Introduction
Background
Component Modeling
Component Implementation



Background

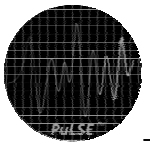


Copyright © Fraunhofer IESE 2003

Motivation (1/2)

Part I
Introduction
Background
Component Modeling
Component Implementation

- Challenges for software-developing organizations today
  - Need for reducing cost, effort, and time-to-market
  - Complexity and software size increases
  - Increasing request for quality software and individually customized products
- Reuse is a promising solution
  - Reusing existing software saves time and effort
  - Reusing approved software brings quality
  - Focusing on new functionality enables realization of individual requirements

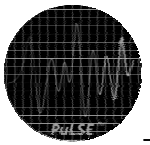


Copyright © Fraunhofer IESE 2003

Motivation (2/2)

Part I
Introduction
Background
Component Modeling
Component Implementation

- How can complex software systems be managed and maintained?
  - divide and conquer
    → component-based development
- How can components be made truly reusable?
  - Implementing a model using EJB, CORBA or COM+ does not automatically make it reusable
    → product-line engineering
- How can components play a major role throughout the whole software life cycle?
  - today, component => EJB, CORBA, COM+
    → model-driven architectures

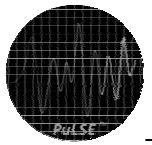


Copyright © Fraunhofer IESE 2003

Component-Based Software Development

Part I
Introduction
Background
Component Modeling
Component Implementation

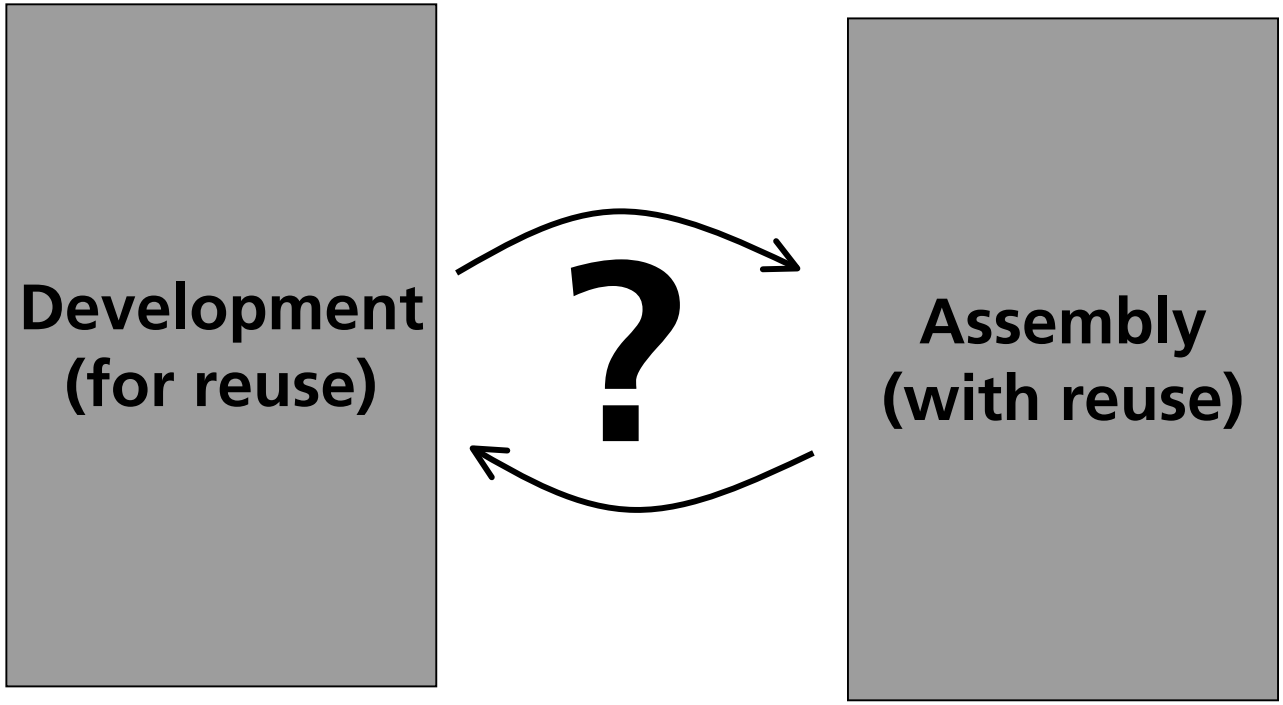
- The development of software applications by the assembly of prefabricated, self-contained software building blocks
- Expected benefits -
  - greater reuse
  - enhanced quality
  - rapid application development
  - COTS Marketplace
  - More cost effective software development and maintenance



Copyright © Fraunhofer IESE 2003

Component-based Development

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



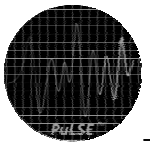
Copyright © Fraunhofer IESE 2003

Development (for reuse)

Part I
Introduction
Background
Component Modeling
Component Implementation

- Techniques
  - Generalization
  - Parameterization
- Process
  - Based on individuals
  - Based on personal experience
  - Based on implicit knowledge

➢ UNSYSTEMATIC definition of generalized components



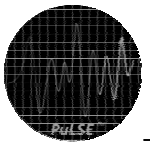
Copyright © Fraunhofer IESE 2003

Assembly (with reuse)

Part I
Introduction
Background
Component Modeling
Component Implementation

- Modularization of applications to be built
  - Based on individual application engineers
  - Based on personal experience
  - Based on implicit knowledge
- Identification of reuse candidates
  - Based on individual application engineers
  - Based on personal experience
  - Based on implicit knowledge
- Integration supported by component technologies

➢ UNSYSTEMATIC reuse of components



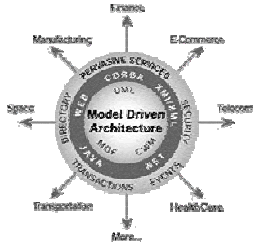
Copyright © Fraunhofer IESE 2003

Challenges for Component-based Development

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

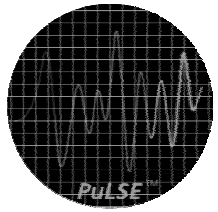
- How can components play a bigger role in "front end" development activities?
  - today, component => EJB, CORBA, COM+

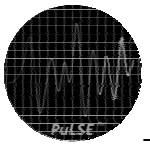
    → model driven architectures



- How can components be made truly reusable?
  - Implementing a model using EJB, CORBA or COM+ does not automatically make it reusable

    → product-line engineering

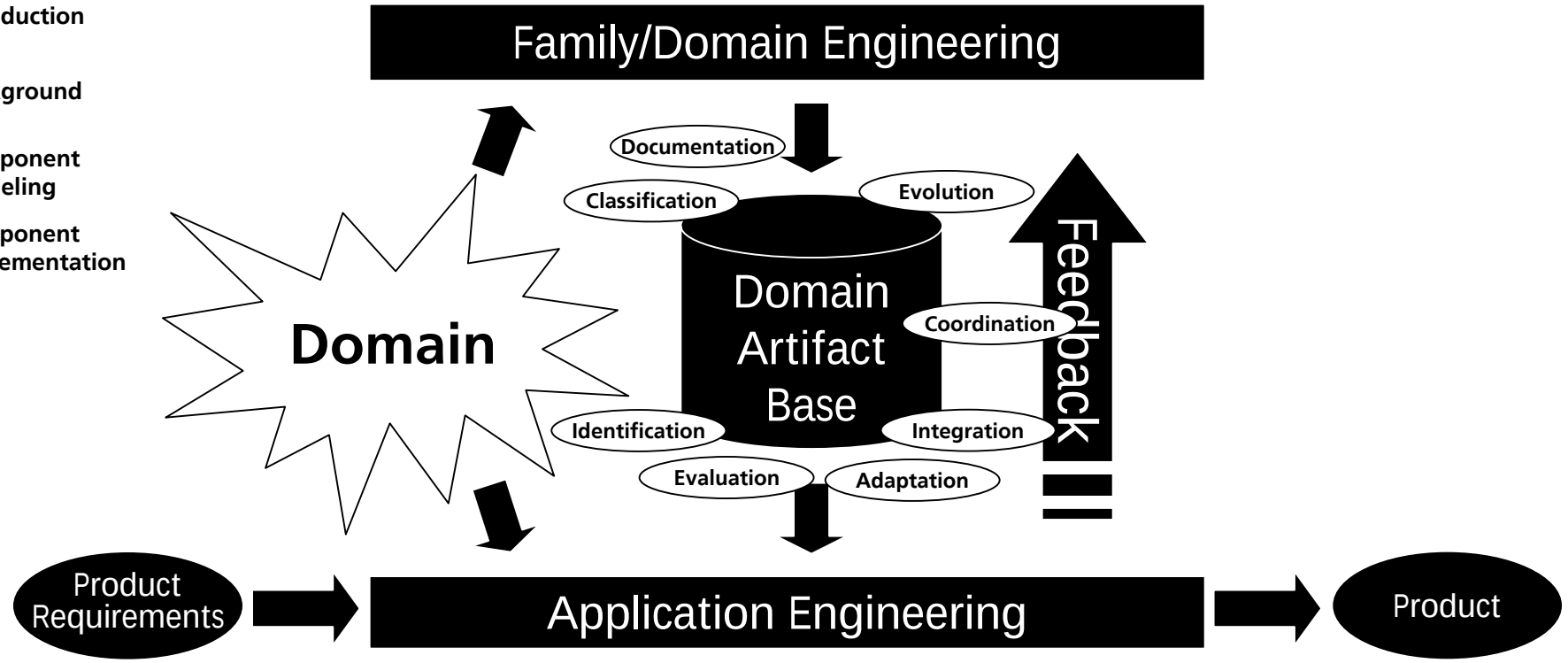




Copyright © Fraunhofer IESE 2003

Domain Engineering

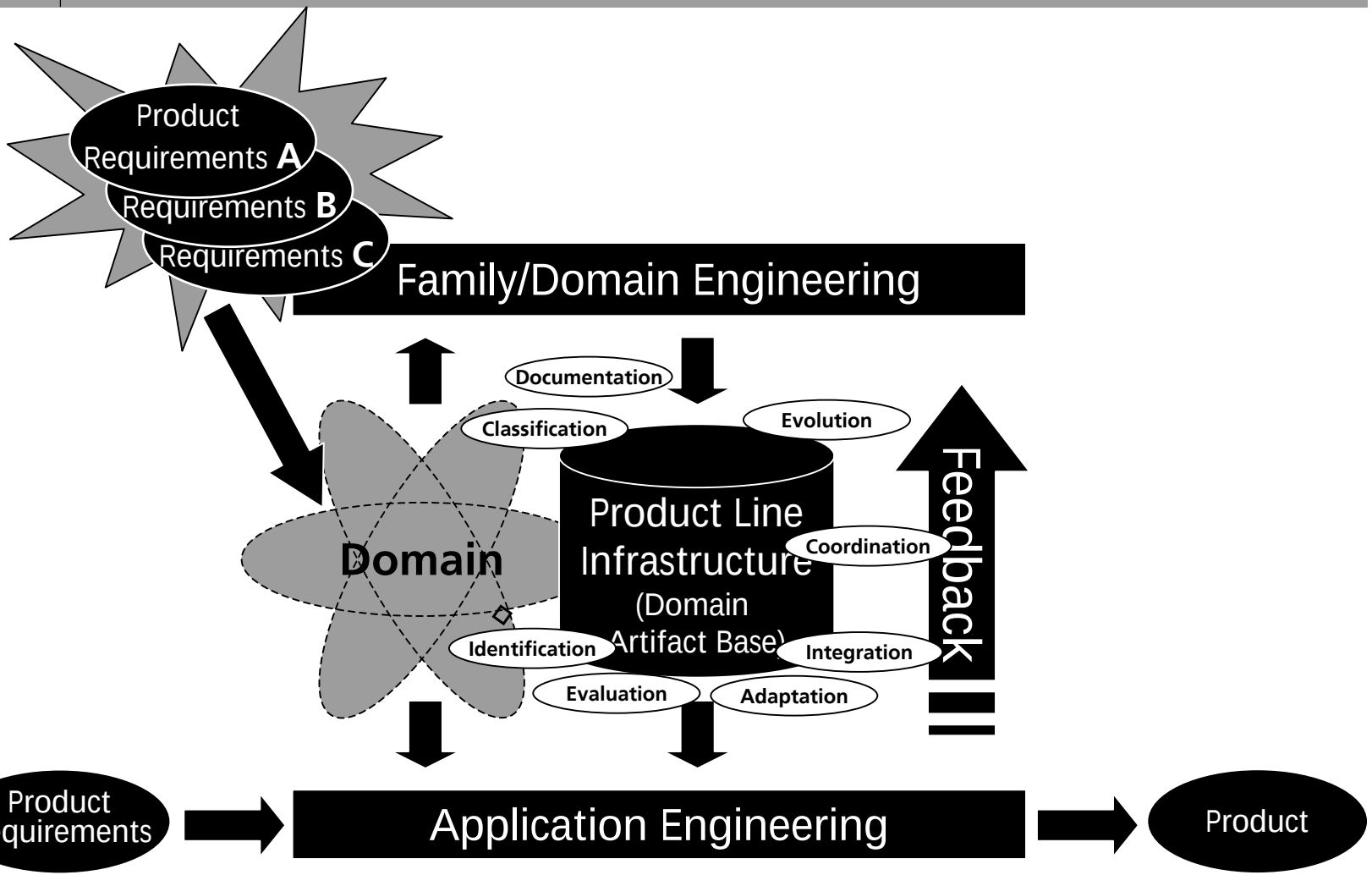
Part I
Introduction
Background
Component Modeling
Component Implementation



Copyright © Fraunhofer IESE 2003

Product Line Engineering

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



Copyright © Fraunhofer IESE 2003

Definition: Software Product Line

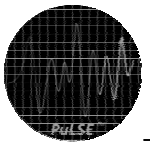
Part I
Introduction
Background
Component Modeling
Component Implementation

- " We call a family of products designed to take advantage of their common aspects and predicted variabilities a product line."

  Lai & Weiss

- "A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way."

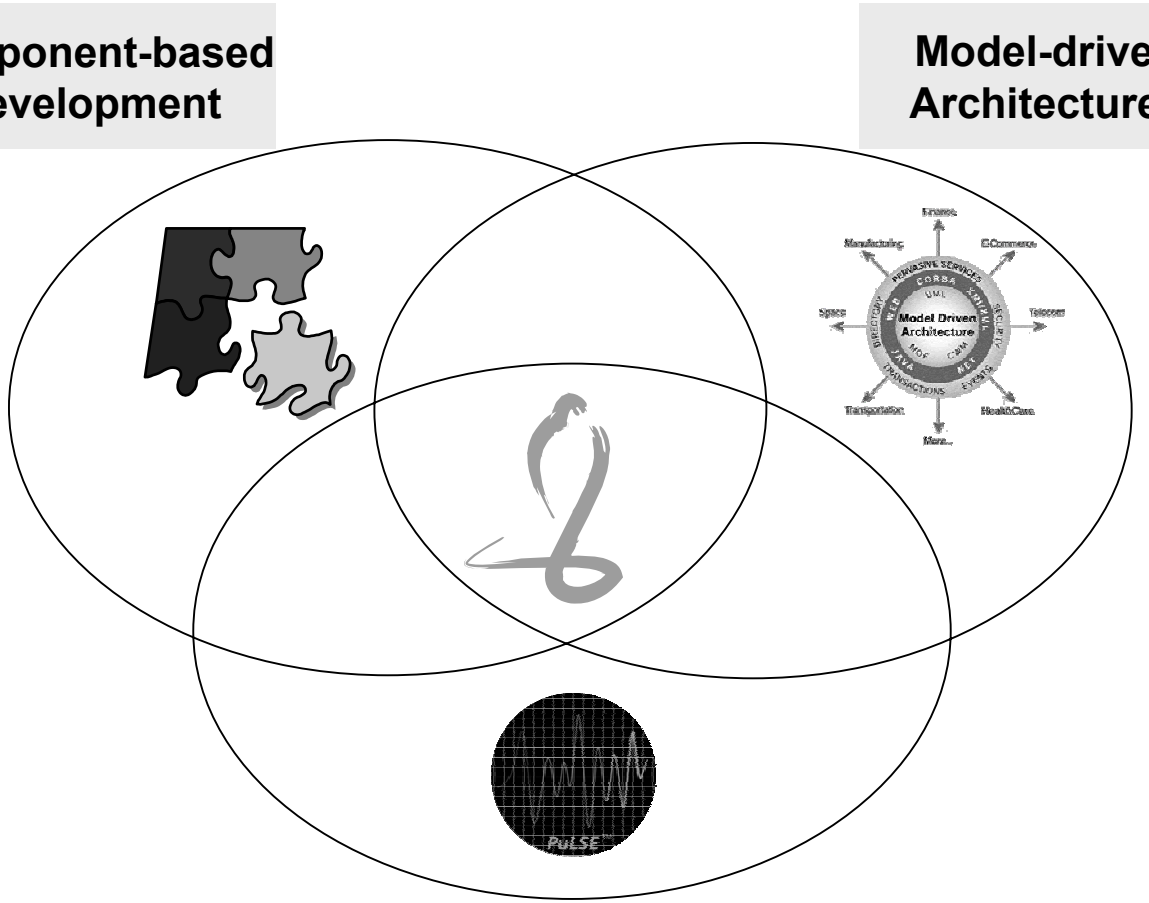
  Clements & Northrop



Copyright © Fraunhofer IESE 2003

Foundations of the KobrA Method

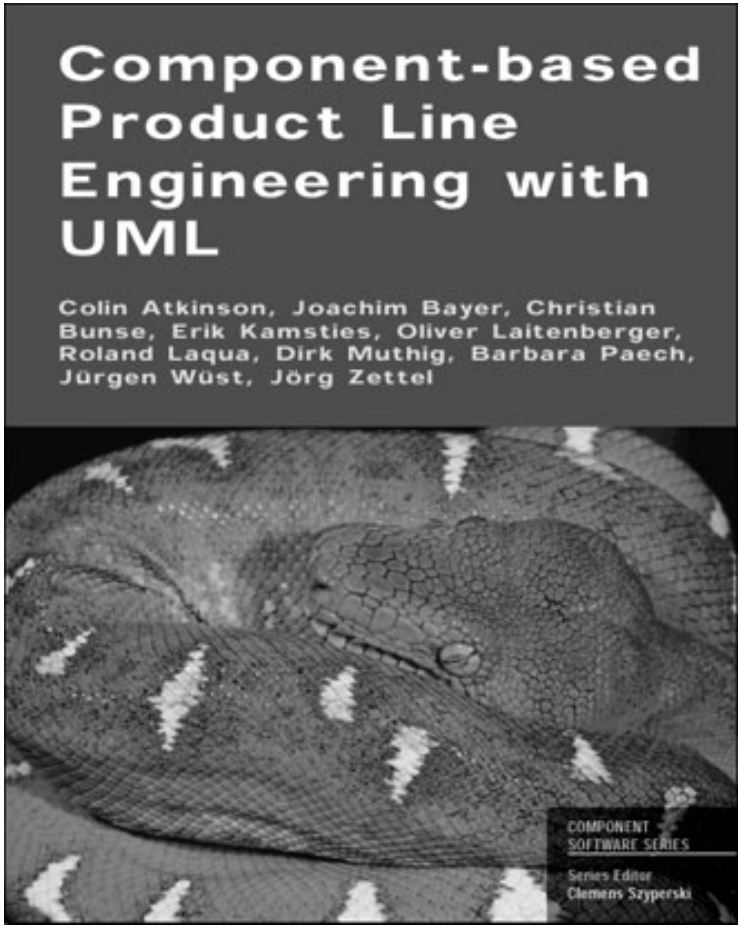
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



Copyright © Fraunhofer IESE 2003

KobrA Book

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



464 pages
1st edition (November 2, 2001)
Component Series
Edited by Clemens Szyperski
Addison-Wesley Pub Co;
ISBN: 0-201-73791-4

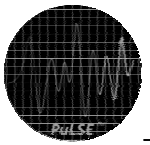
ADDISON-WESLEY

Copyright © Fraunhofer IESE 2003

Properties of a Good Method

Part I
Introduction
Background
Component Modeling
Component Implementation

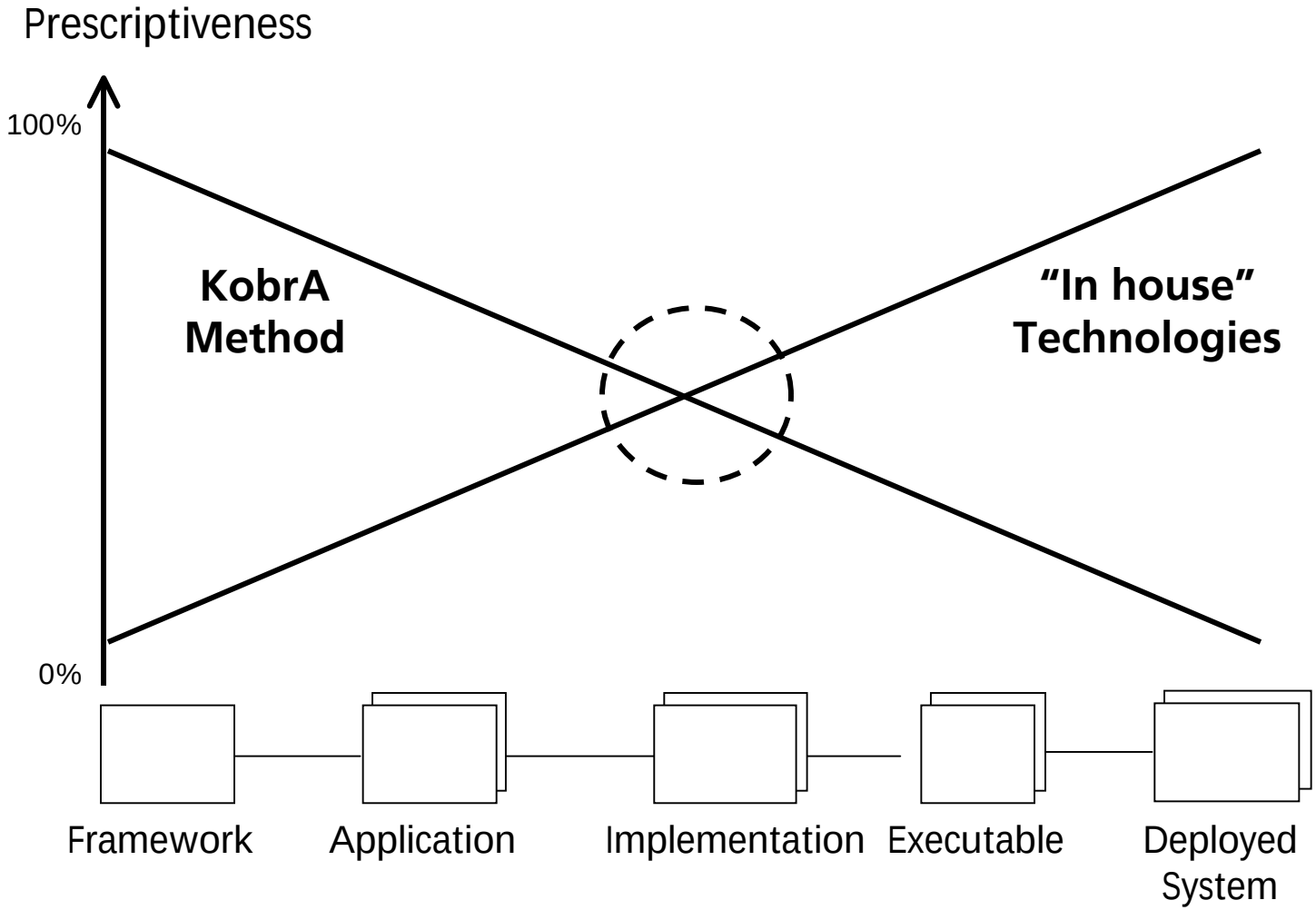
- Simple
  - → Minimal set of concepts
  - → No redundancy (different words for the same thing)
- Systematic
  - → Methodological, step-wise process
  - → rigorous
- Prescriptive
  - → Should try to say what you should do not what you may do
- Flexible
  - → Should be useable in as wide a range of circumstances as possible
- Scaleable
  - → Should be applicable in the same way to large and small systems



Copyright © Fraunhofer IESE 2003

Method Prescriptiveness

Part I
Introduction
Background
Component Modeling
Component Implementation



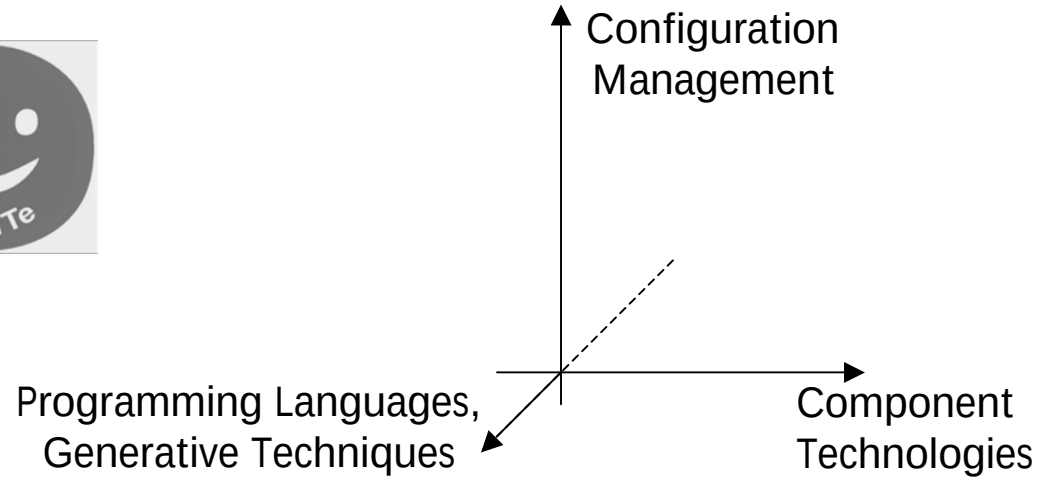
Copyright © Fraunhofer IESE 2003

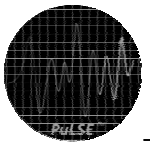
Context: The PoLITe Project

Part I
Introduction
Background
Component Modeling
Component Implementation

- PoLITe: Product Line Implementation Technologies
  - Research partner: FH Zweibrücken (Eisenecker)
- Three dimensions of implementation approaches:





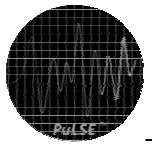


Copyright © Fraunhofer IESE 2003

Schedule

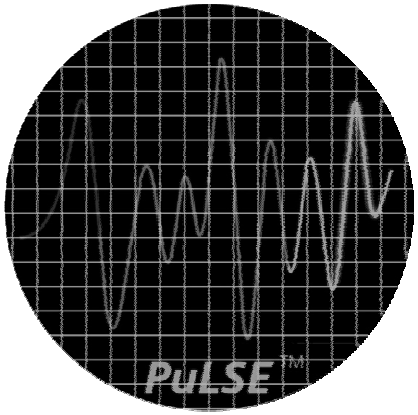
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

- *Part I*
  - Introduction (10 min)
    - → Who we are and why we are here
    - → Who you are and why you are here
  - Background (30 min)
    - → KobrA Method
    - → PoLITe project
  - **Component Modeling and Frameworks (25 min)**
  - Model-driven Component Implementation (25 min)

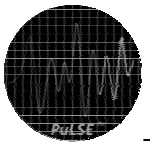


Copyright © Fraunhofer IESE 2003

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



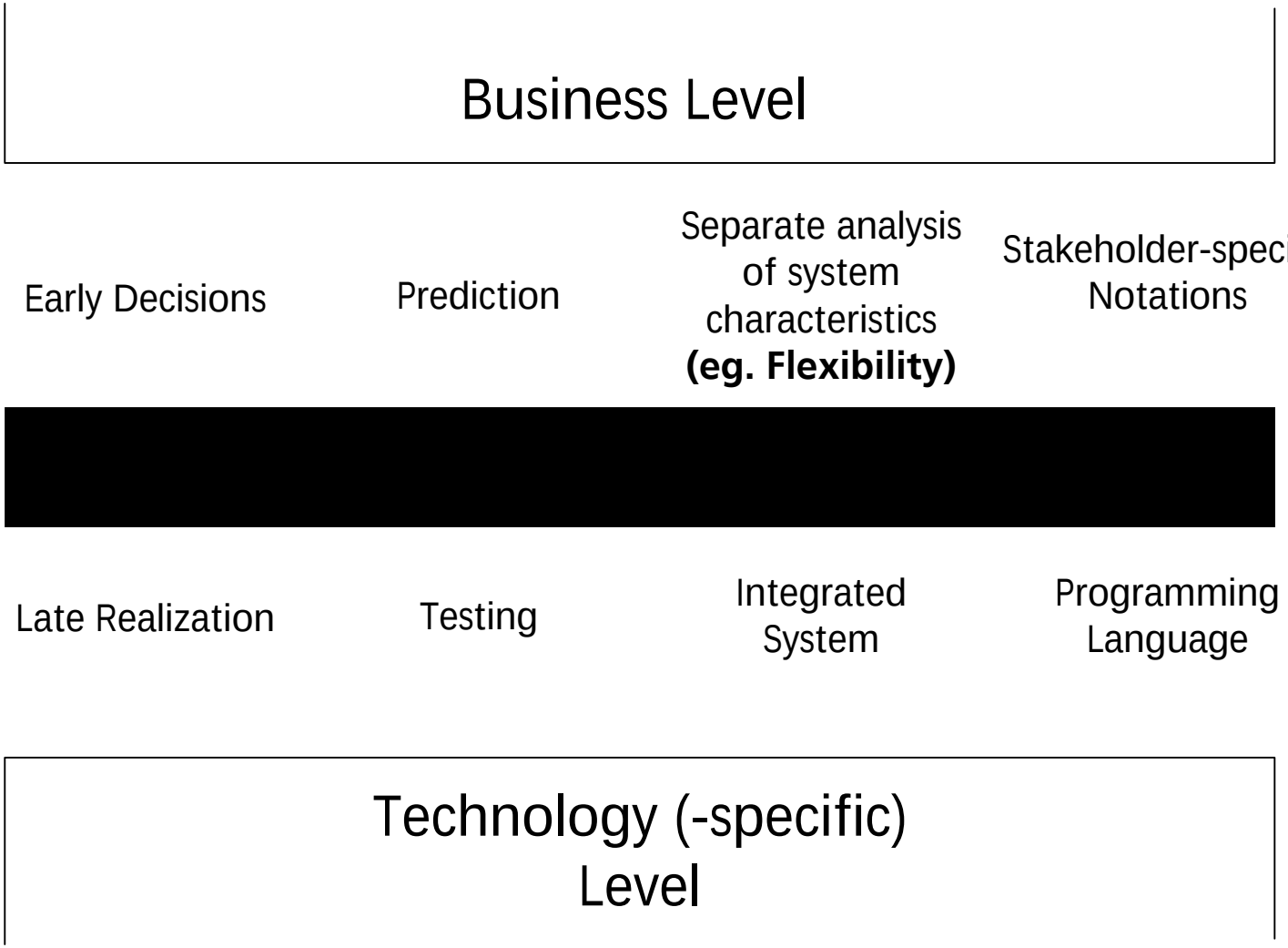
Component Modeling and Frameworks



Copyright © Fraunhofer IESE 2003

Architectural Level

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

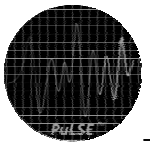


Copyright © Fraunhofer IESE 2003

Modeling Principles

Part I
Introduction
Background
Component Modeling
Component Implementation

- Uniformity
  - all behavior rich elements should be viewed as components, including a (sub)system
  - component assembly = component development
- Parsimony
  - minimal set of concepts (no redundancy)
- Locality
  - all models should be local to a component
- Encapsulation
  - component specifications (what) must be separated from component realizations (how)

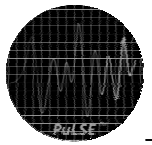


Copyright © Fraunhofer IESE 2003

The Principle of Locality

Part I
Introduction
Background
Component Modeling
Component Implementation

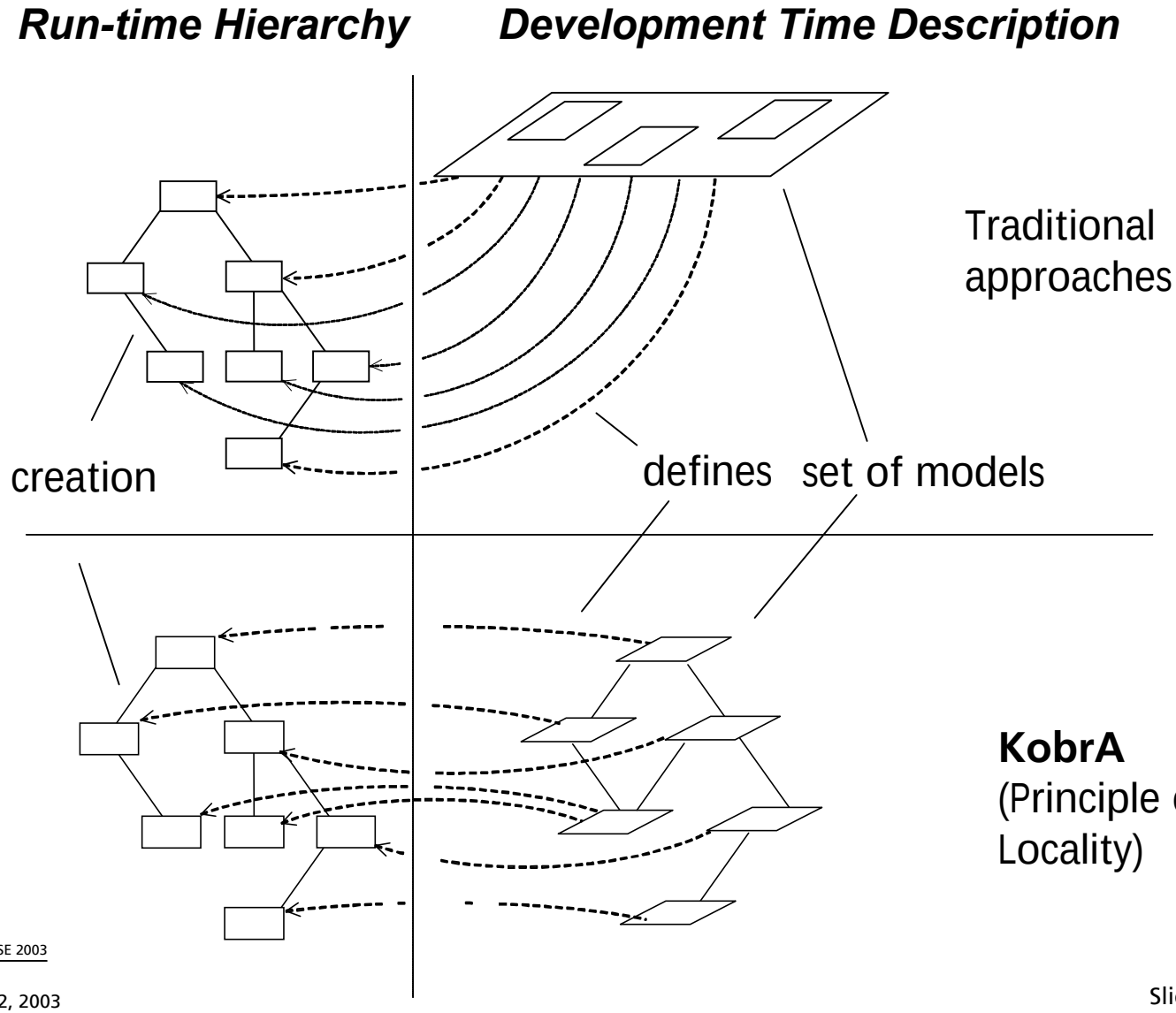
- Run-time composition structures are traditionally described within global models
  - inhibits reuse
  - non-component oriented
- The **principle of locality** states that -
  - content and organization of software artifacts should be relative to a single component
- Global artifacts
  - should be used as little as possible
  - should be generated recursively



Copyright © Fraunhofer IESE 2003

Applying the Principle of Locality

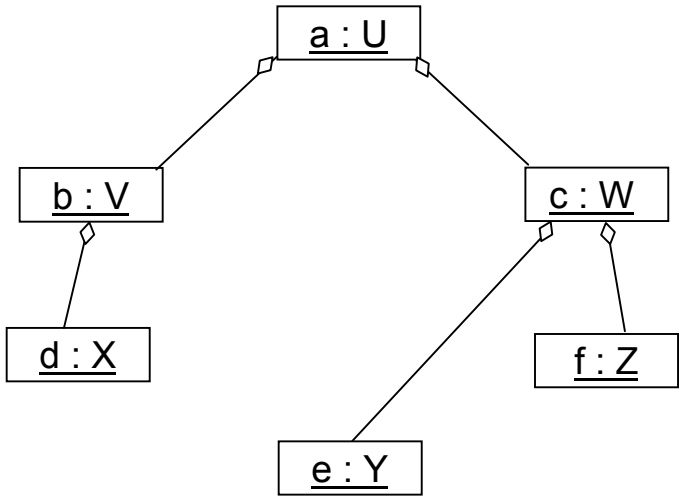
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



Copyright © Fraunhofer IESE 2003

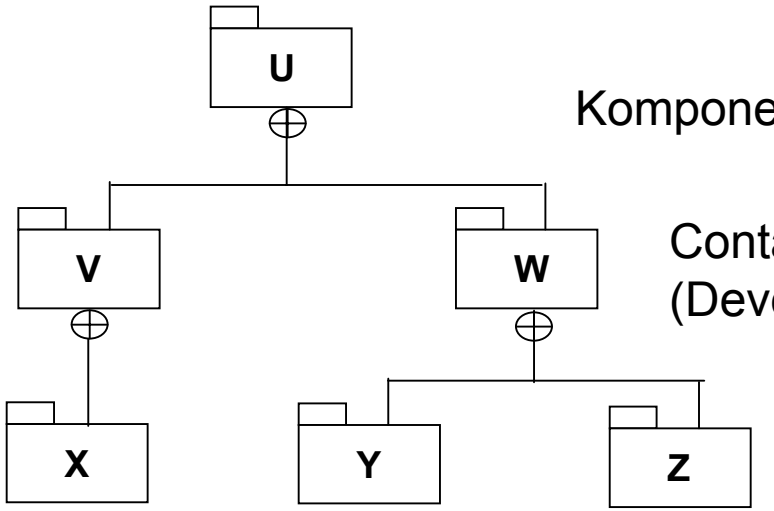
Aligning Composition and Containment Trees

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



Komponent instances

Composition Tree
(Run Time)



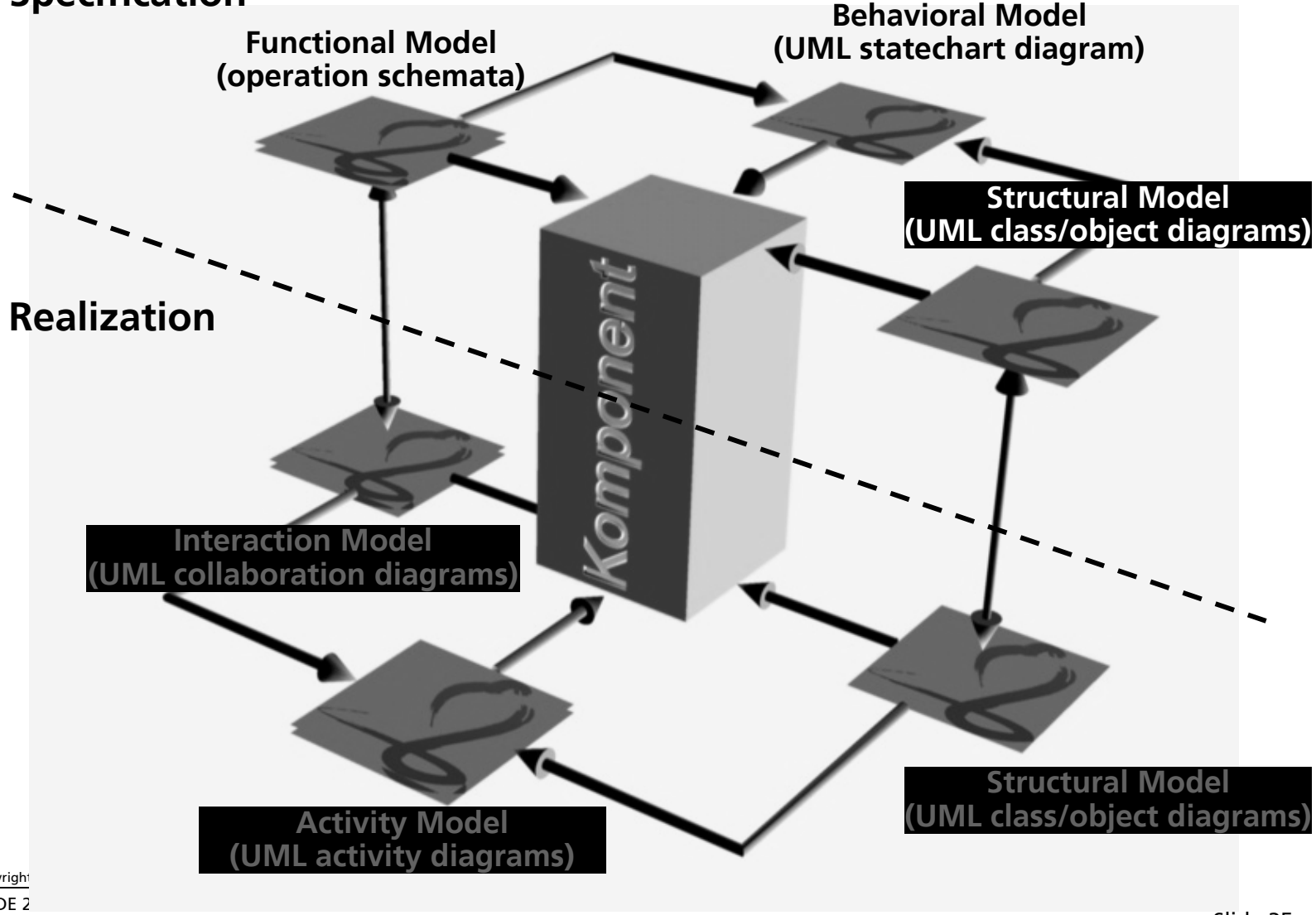
Komponents (i.e. types)

Containment Tree
(Development Time)

Copyright © Fraunhofer IESE 2003

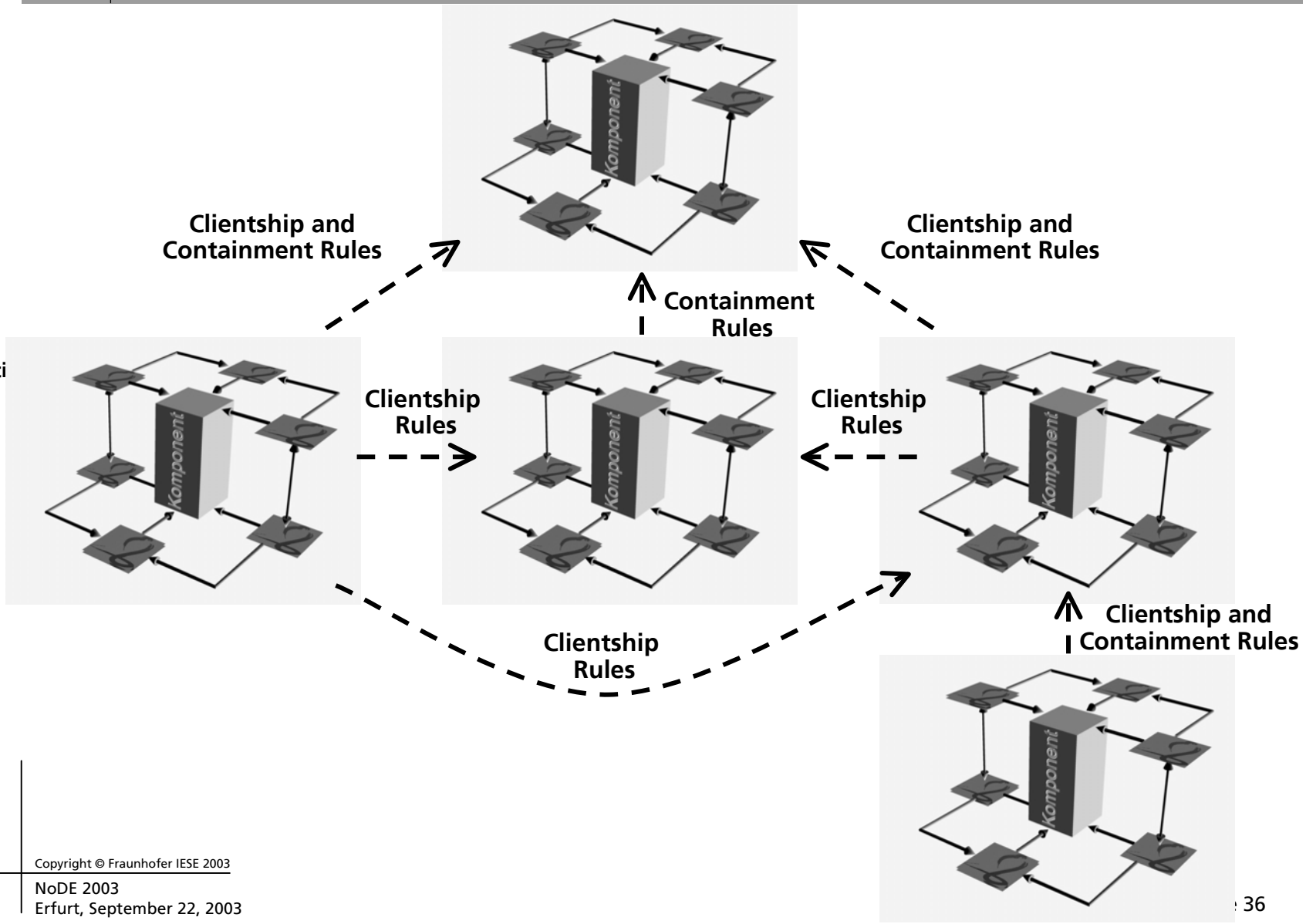
Component Model Suite

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



Component Framework

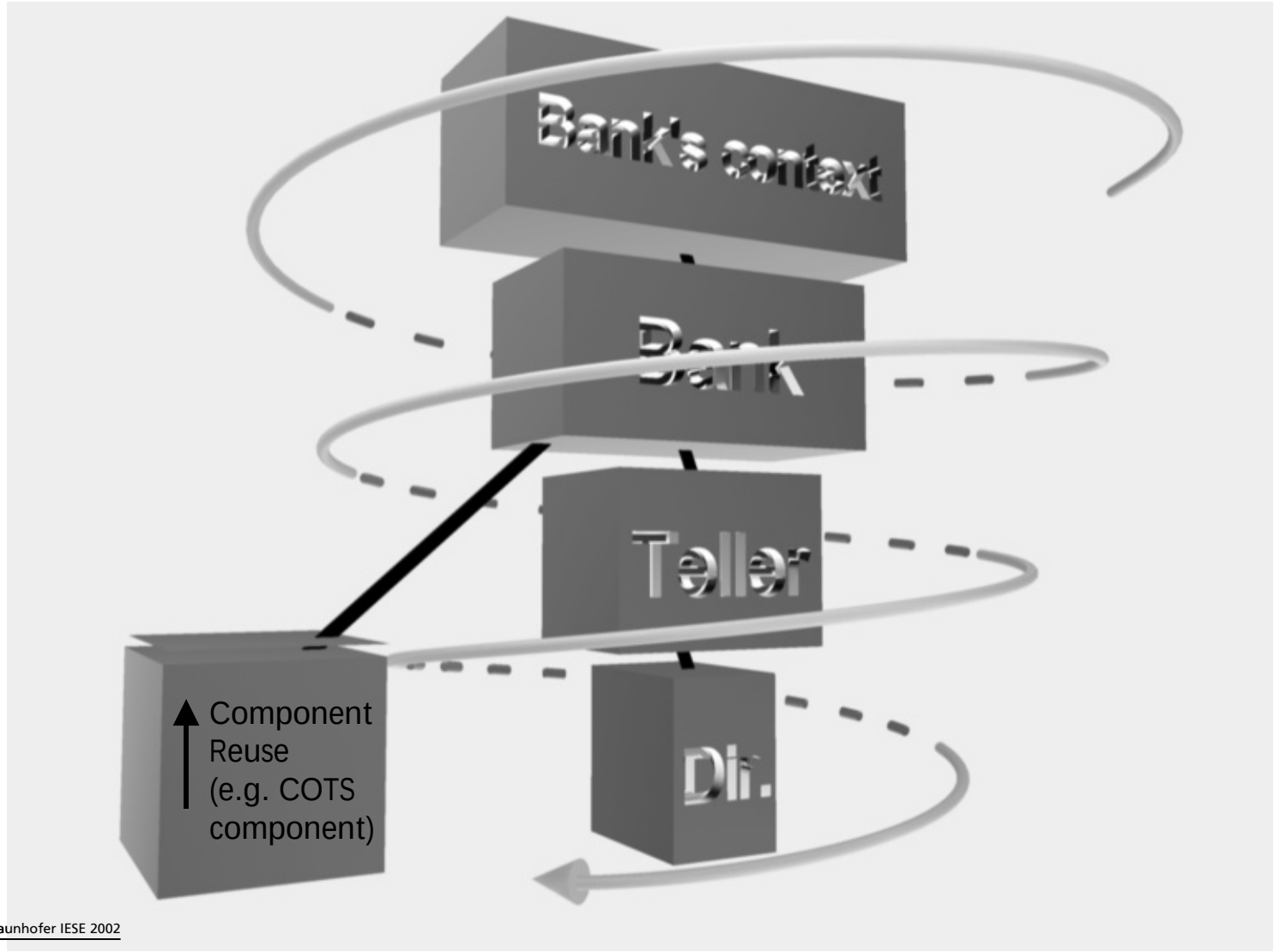
Part I
Introduction
Background
Component Modeling
Component Implementati



Copyright © Fraunhofer IESE 2003

Component Engineering: Modeling

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



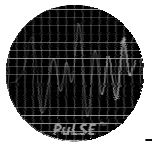
Copyright © Fraunhofer IESE 2002

Product Line Engineering

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

- Most software development organizations develop a line of similar but slightly different products
- KobrA describes a family of similar systems (i.e. components) by modeling
  - variabilities
  - decisions
- A component (class) with (non-empty) decision models (i.e. optional features) is known as a **generic component**
- A tree of generic components represents a generic framework
  - decision models are related hierarchically
  - resolution implies the resolution of all lower decision models

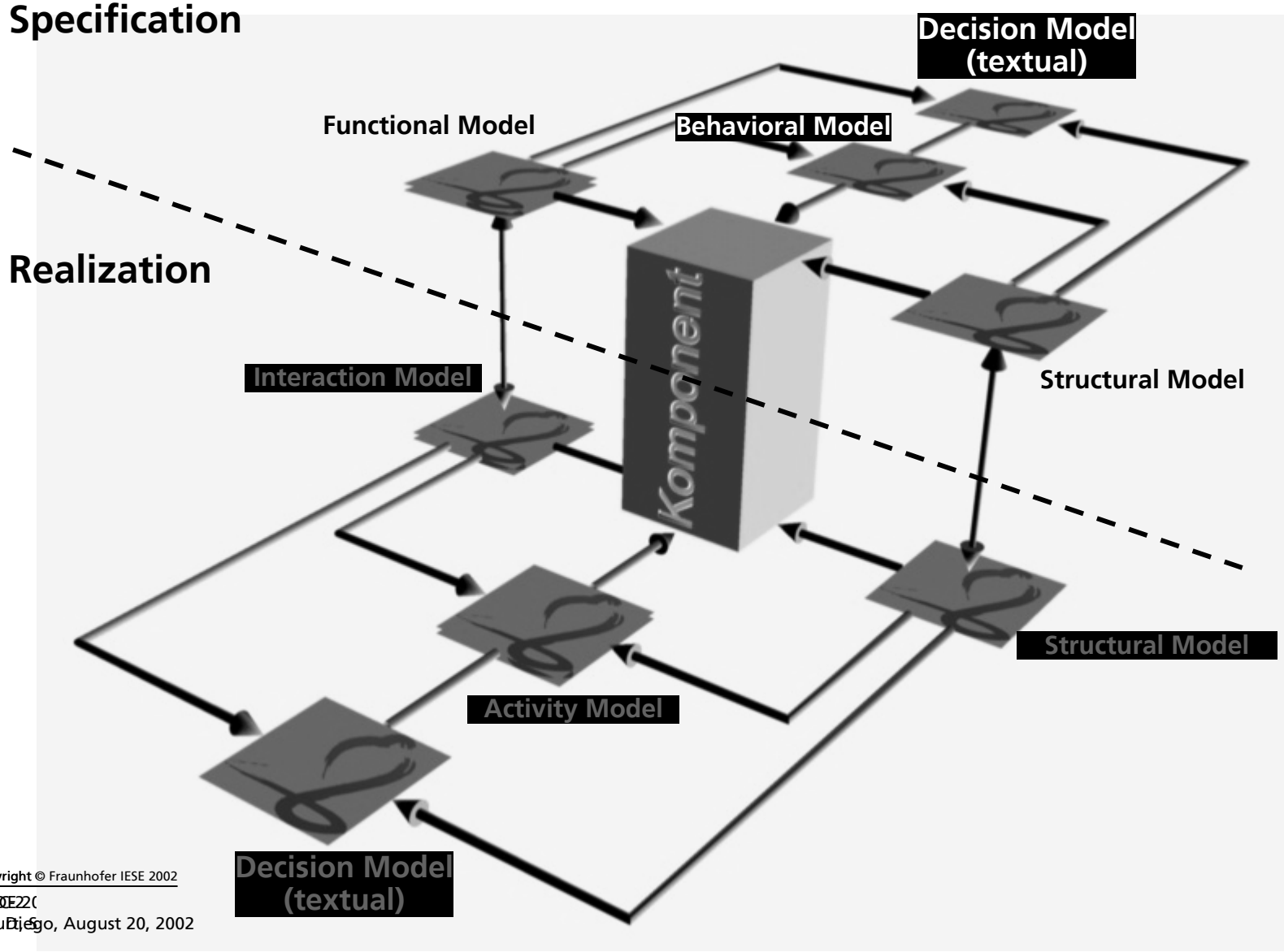
=> Incremental introduction of product lines



Copyright © Fraunhofer IESE 2003

Generic Component Models

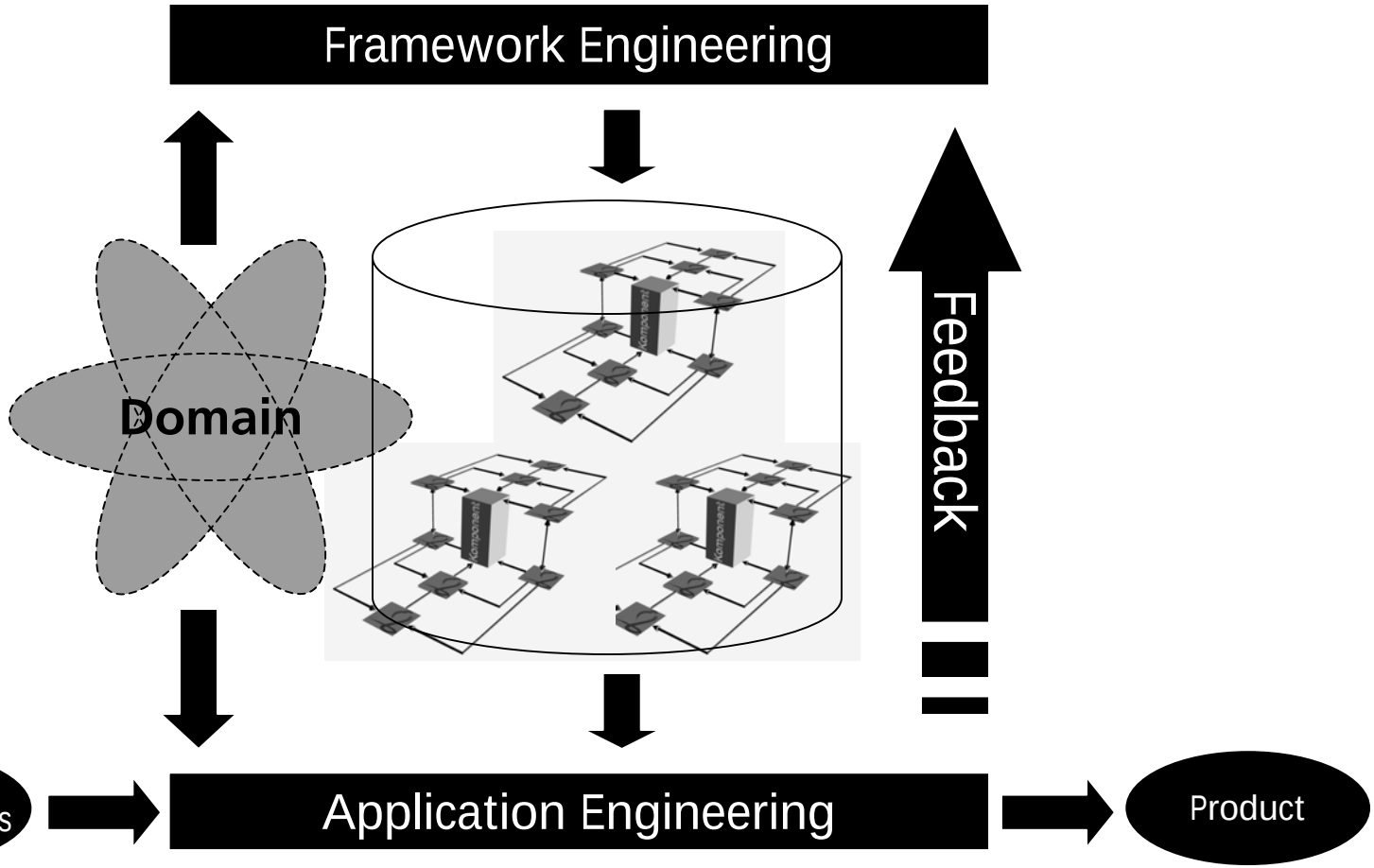
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation



Copyright © Fraunhofer IESE 2002

Product Line Lifecycle

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

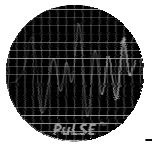


Copyright © Fraunhofer IESE 2003

Schedule

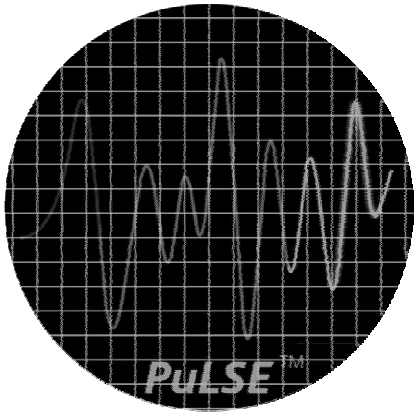
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

- *Part I*
  - Introduction (10 min)
    - → Who we are and why we are here
    - → Who you are and why you are here
  - Background (30 min)
    - → KobrA Method
    - → PoLITe project
  - Component Modeling and Frameworks (25 min)
  - **Model-driven Component Implementation (25 min)**

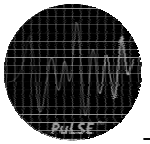


Copyright © Fraunhofer IESE 2003

Part I
Introduction
Background
Component Modeling
Component Implementation



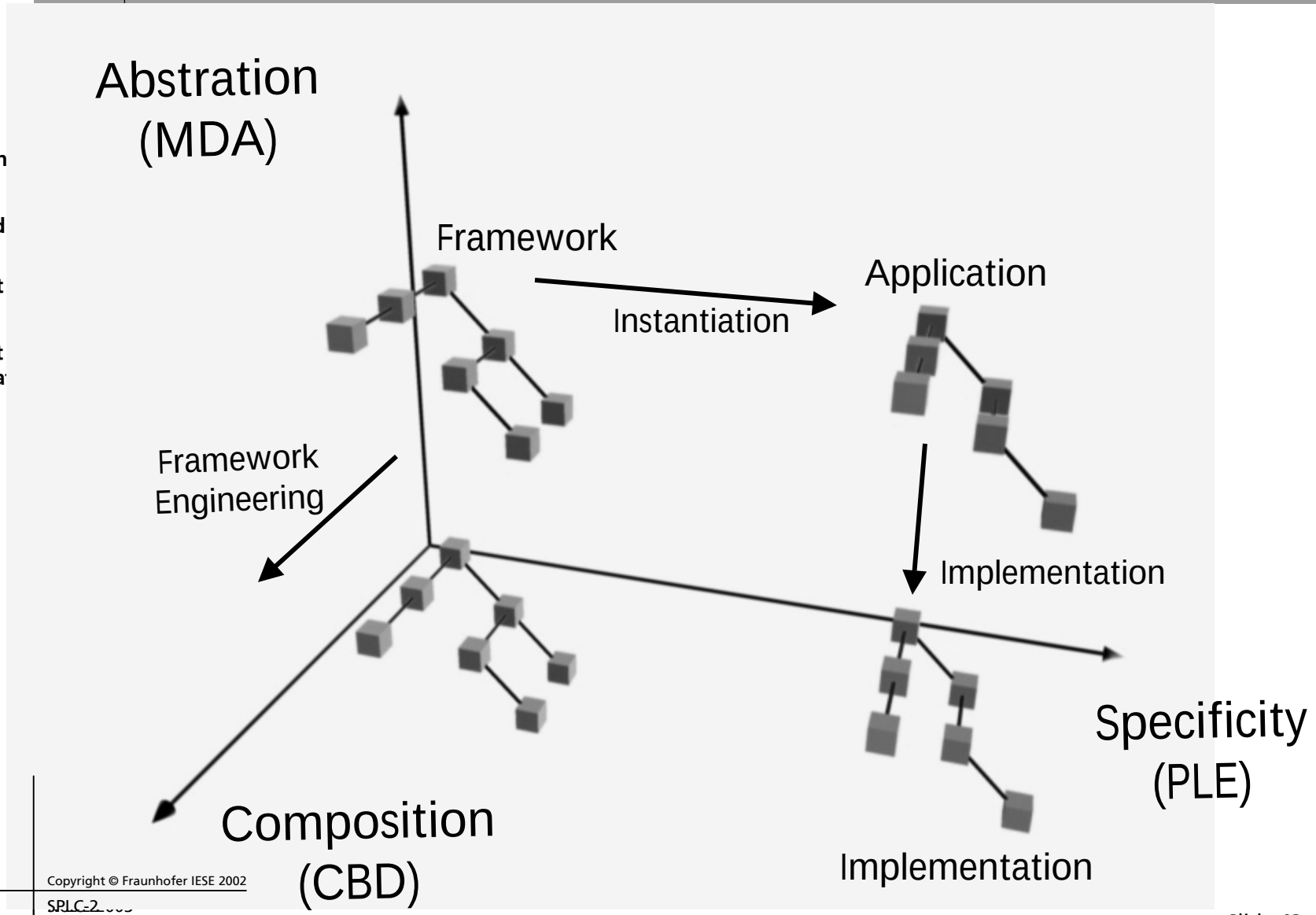
Model driven Component implementation (MDA)



Copyright © Fraunhofer IESE 2003

Separation of Concerns

Part I
Introduction
Background
Component Modeling
Component Implementa



Copyright © Fraunhofer IESE 2002

OBJECT MANAGEMENT GROUP

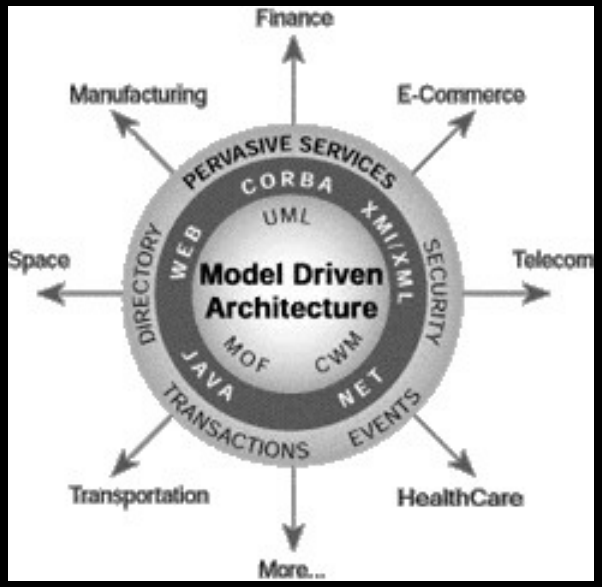
Setting vendor-neutral software standards, and enabling distributed, enterprise-wide interoperability

Members' Area

Become a Member

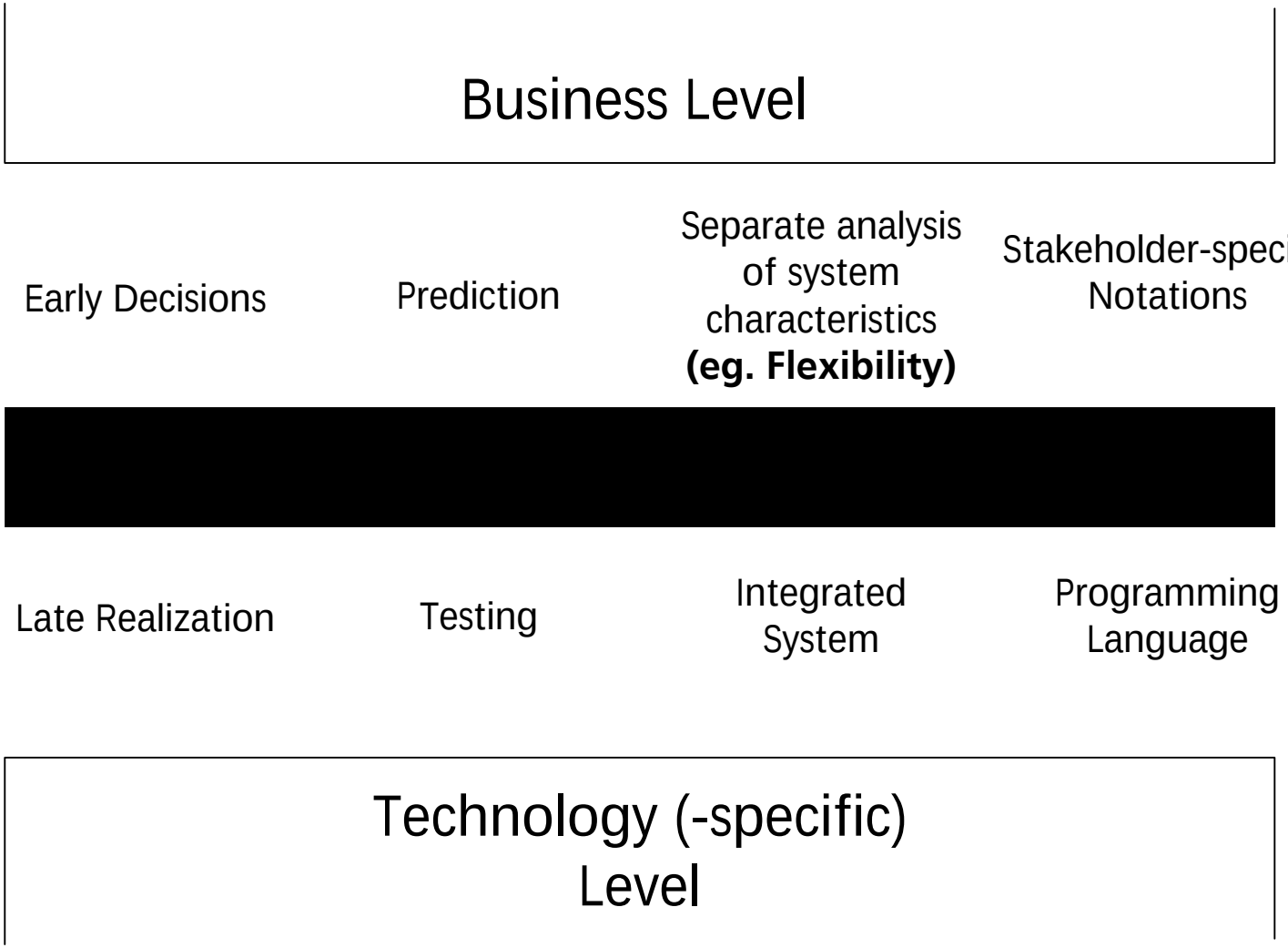
About Us | News & Information | Calendar | Documents & Specifications | Member Shortcuts | Technology Central | Industry Specific Markets

The Architecture of Choice for a Changing World™



Architectural Level

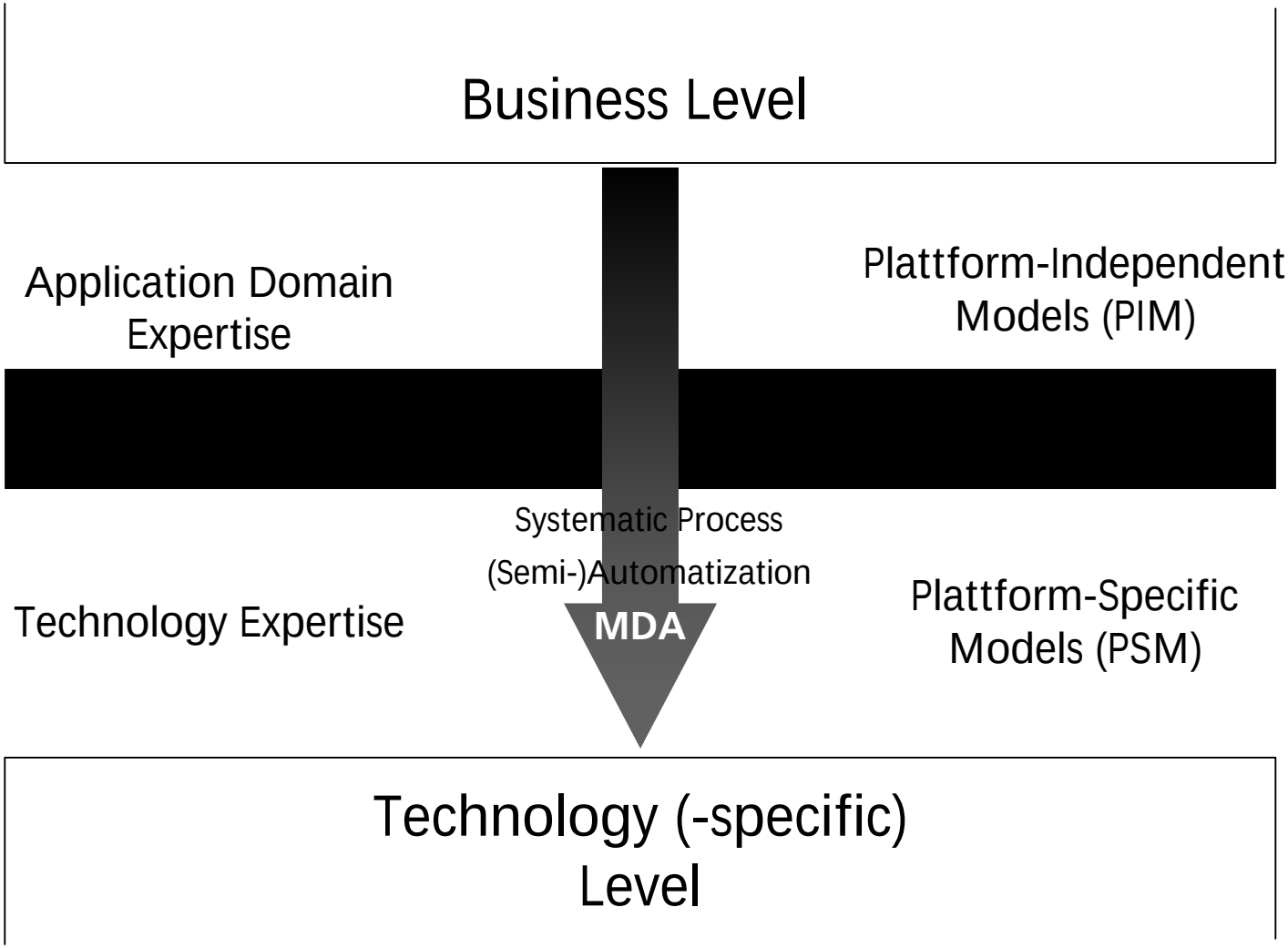
Part I
Introduction
Background
Component Modeling
Component Implementation



Copyright © Fraunhofer IESE 2003

Model-driven Architecture (MDA)

Part I
Introduction
Background
Component Modeling
Component Implementation

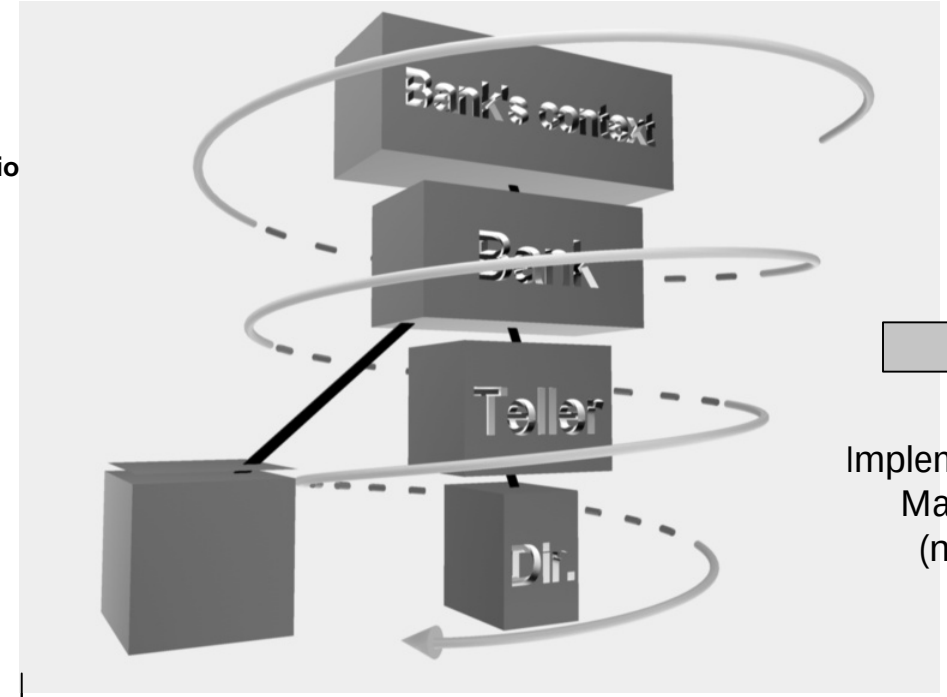


Copyright © Fraunhofer IESE 2003

Component Engineering Process: Implementation

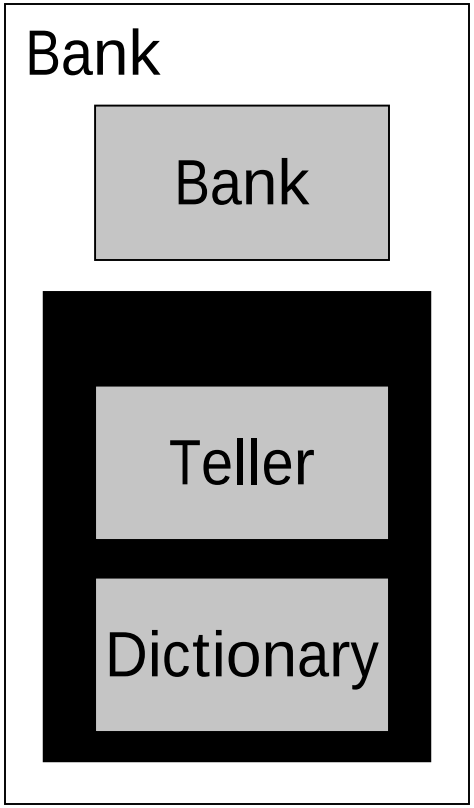
Part I
- Introduction
- Background
- Component Modeling
- Component Implementatio

Plattform-Independent Models (PIM)



Implementation Mapping (n:1:1)

Plattform-Specific Models (PSM)



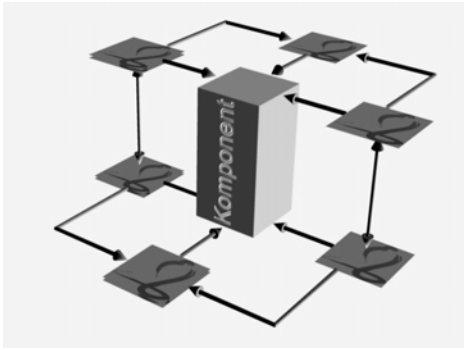
Copyright © Fraunhofer IESE 2003

Implementation of Component-based Systems

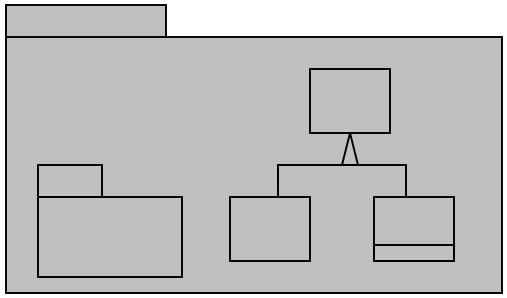
Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

Plattform-Independent Models (PIM)

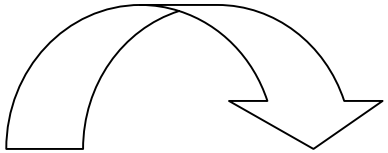
Plattform-Specific Models (PSM)



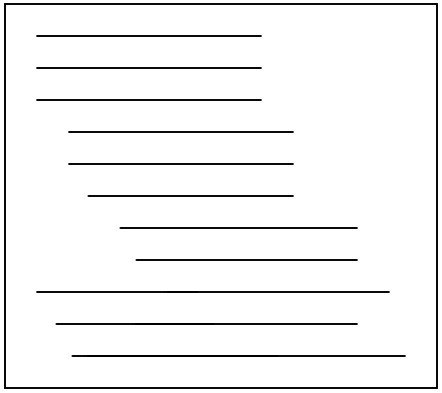
Component Models



Component Implementation



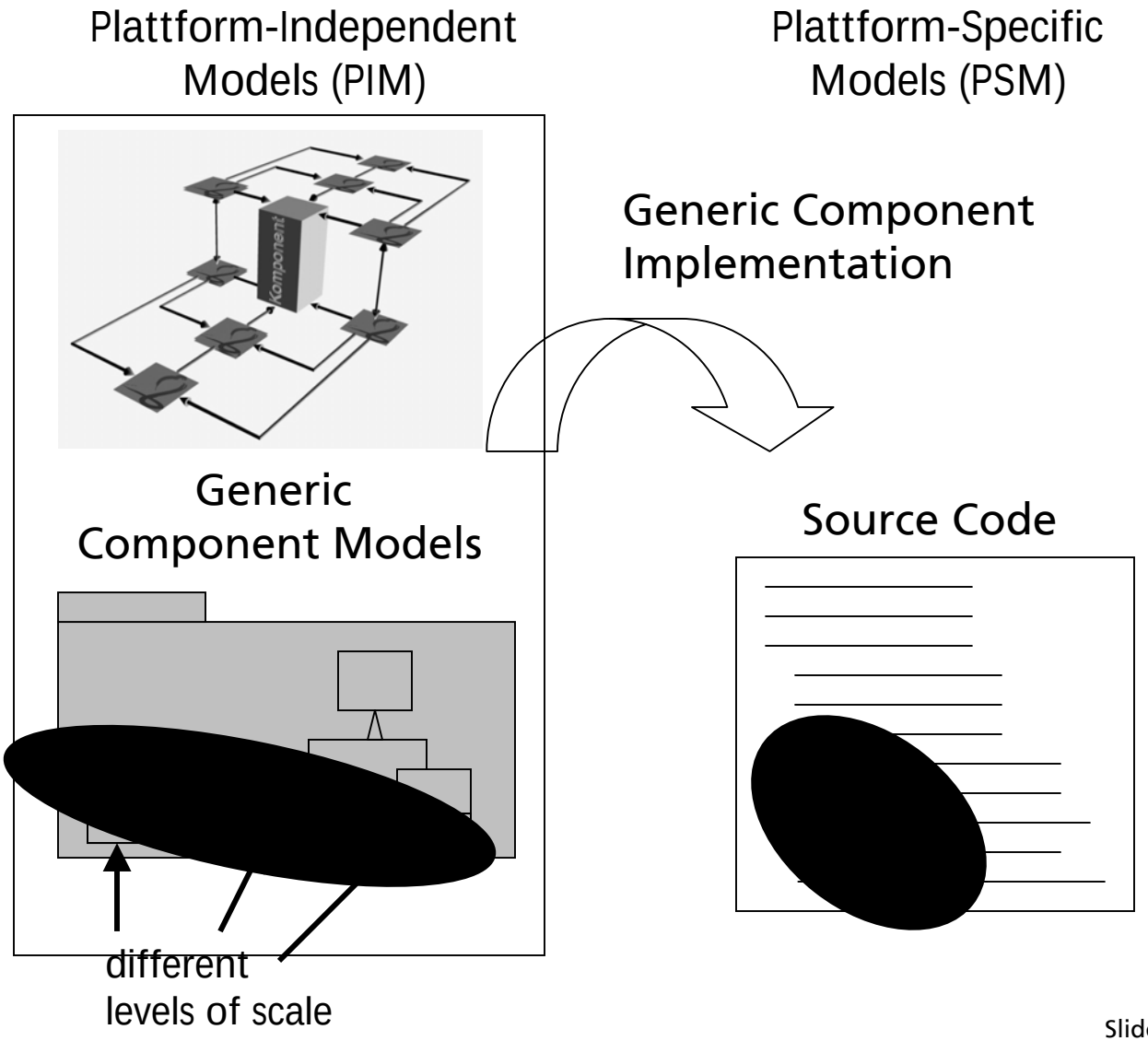
Source Code



Copyright © Fraunhofer IESE 2003

Implementation of Product Lines

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

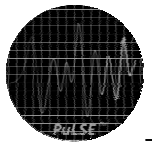


Copyright © Fraunhofer IESE 2003

Variability Management

Part I
Introduction
Background
Component Modeling
Component Implementation

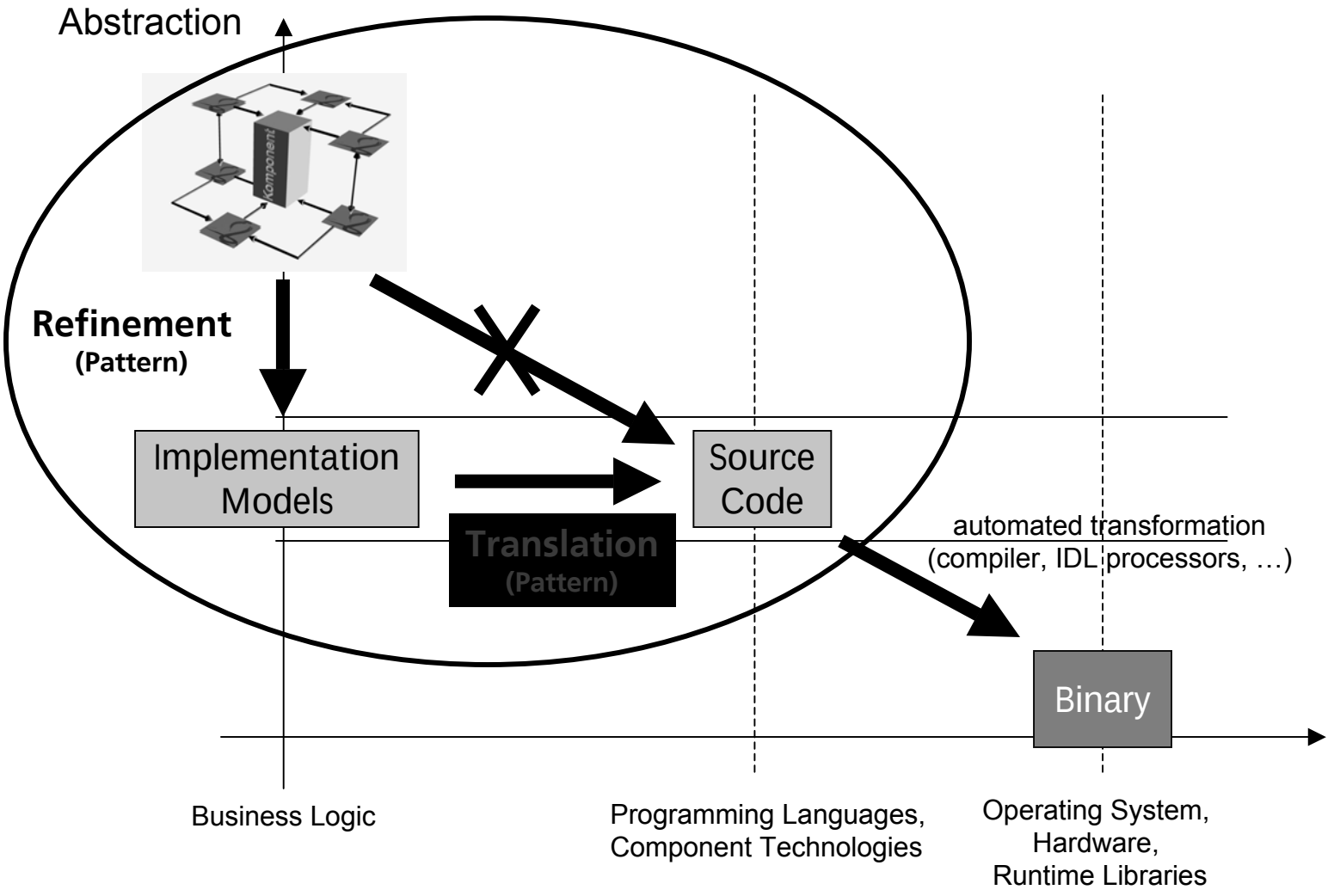
- Technological variations are handled by
  - Technology-specific refinement patterns or pattern sets
  - Alternative refinement or translation patterns
- Variations in application or business logic are handled
  - At the plattform-independent level (component models)
  - Through variability patterns that produce generic implementations
- Products (or variants) are created by
  - Modifications of the plattform-independent component models
  - (Re-)application of implementation patterns



Copyright © Fraunhofer IESE 2003

Implementation Patterns (1/3)

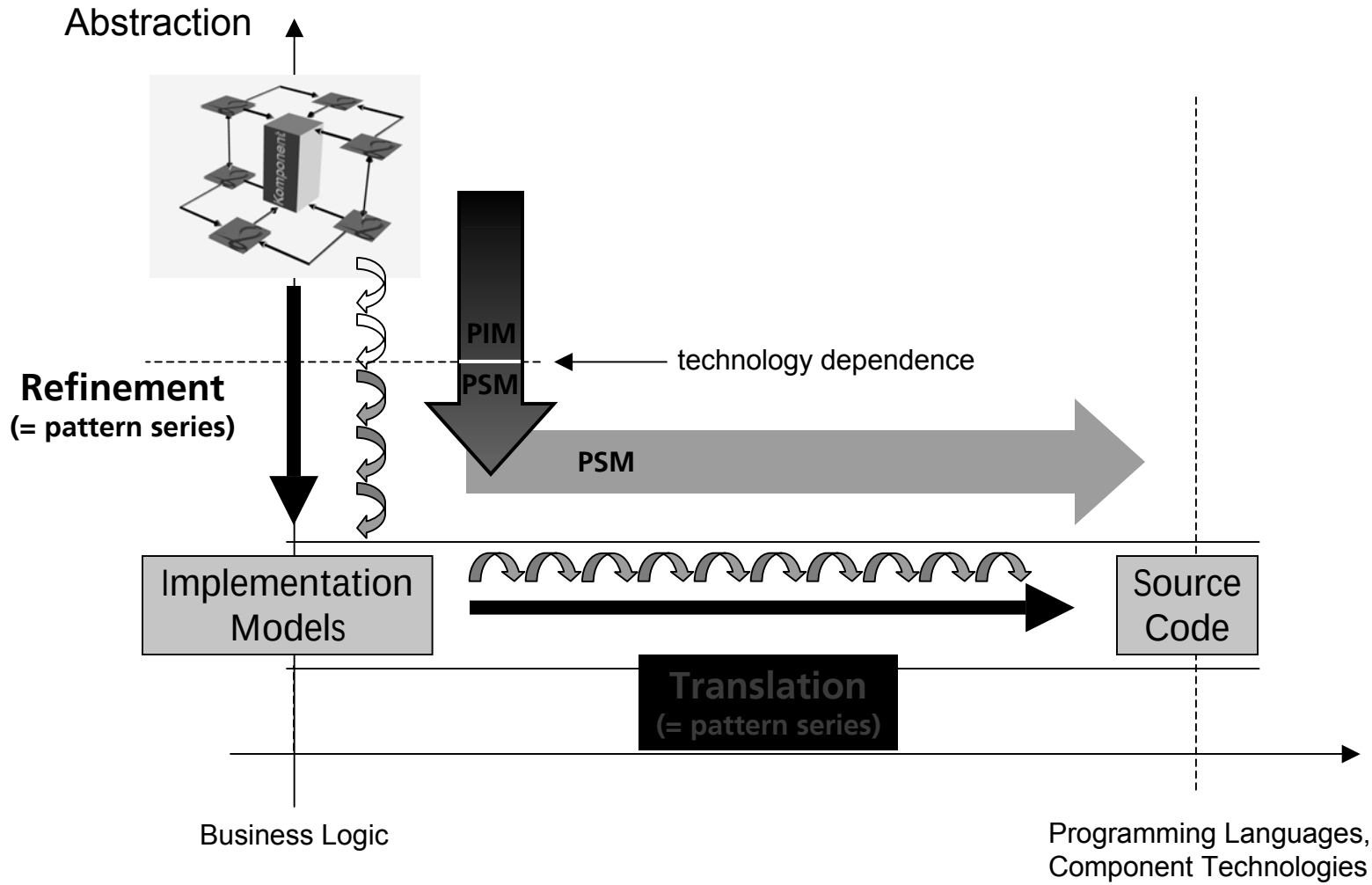
Part I
Introduction
Background
Component Modeling
Component Implementation



Copyright © Fraunhofer IESE 2003

Implementation Patterns (2/3)

Part I
Introduction
Background
Component Modeling
Component Implementation

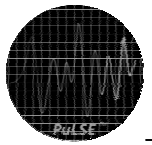


Copyright © Fraunhofer IESE 2003

Implementation Patterns (3/3)

Part I
Introduction
Background
Component Modeling
Component Implementation

- Pattern
  - Describes a typical problem with one of its potential solutions in a reusable form
  - Captures characteristics of the proposed solution explicitly
- Pattern Set
  - Collection of patterns for problems in a particular application domain
  - Common toolbox for implementation activities
  - Captures implementation knowledge in an application domain explicitly
- Pattern Application
  - Systematic execution of implementation activities
    → Small, explicit pattern steps instead of complex, implicit transformations
  - Documentation wrt. performed pattern series

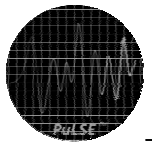


Copyright © Fraunhofer IESE 2003

Outlook: Part II

Part I
- Introduction
- Background
- Component Modeling
- Component Implementation

- *Part II*
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - J2EE (35 min)
  - COM (10 min)
  - .NET (10 min)
  - Variability Management (10 min)
  - Conlusions and Discussion (15 min)



Copyright © Fraunhofer IESE 2003

Model Driven Product Line Architectures – Part II: Mapping Models to J2EE, COM, .NET

NODE Erfurt

September 22, 2003



Fraunhofer Institut
Experimentelles
Software Engineering

Stefan Kettemann
Dirk Muthig
Sauerwiesen 6
D-67661 Kaiserslautern
Germany

Tutorial Organization

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

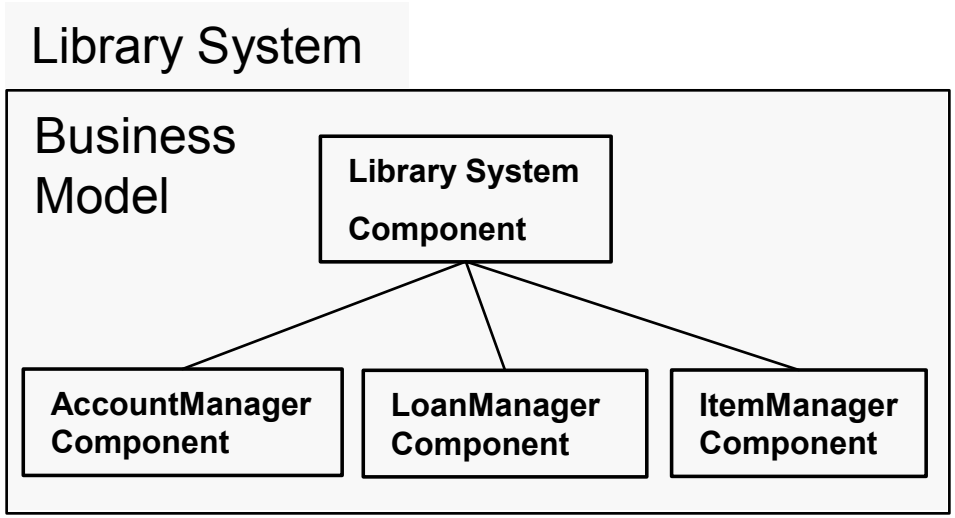
- **Part II**
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - J2EE (30 min)
  - COM (10 min)
  - .NET (10 min)
  - Variability Management (10 min)
  - Conclusions and Discussion (15 min)

Copyright © Fraunhofer IESE 2003

Case Study – Library System

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



PIM (Platform-Independent Model, Business Model)

Users with a valid account can loan Items, e.g., books

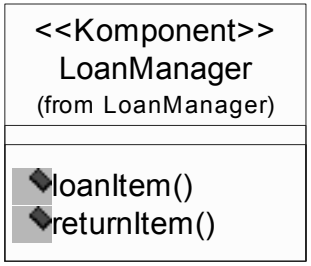
Business Objects: Items, Accounts, Loans, ItemManager, LoanManager, etc.

Copyright © Fraunhofer IESE 2003

MDA – From PIM to PSM

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

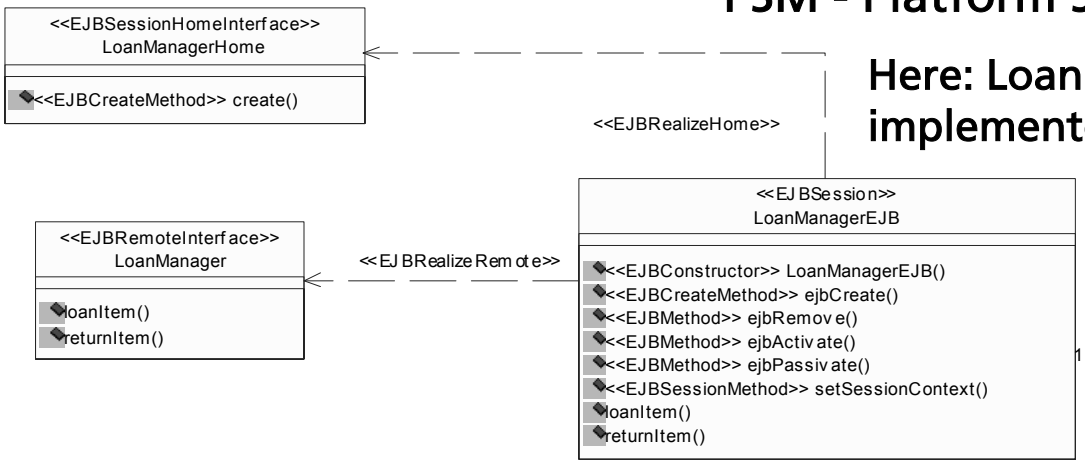


PIM - Platform Independent Model

Transformation

PSM - Platform Specific Model: J2EE

Here: LoanManager is implemented as EJB SessionBean

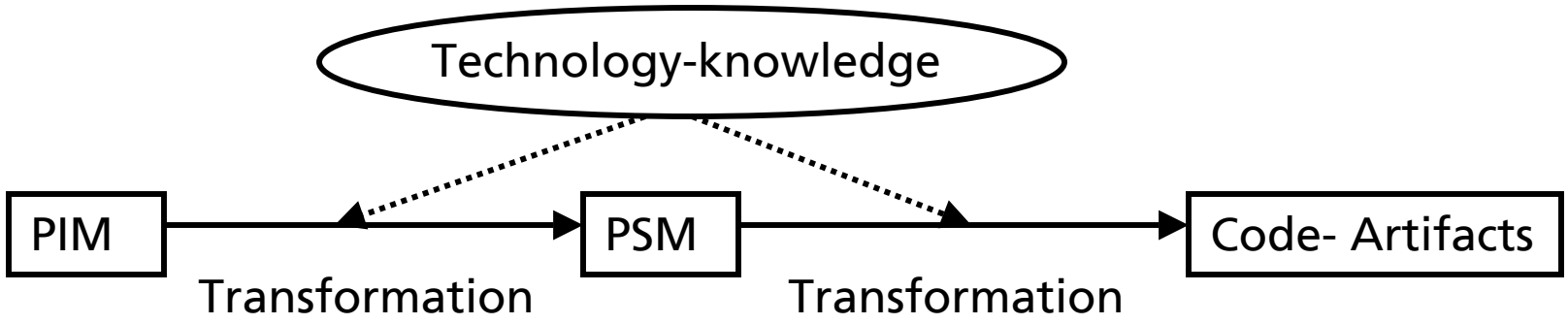


Copyright © Fraunhofer IESE 2003

Approach

What do I need in order to perform the transformation(s) ?

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Approach:

1. UNDERSTAND	How does a component look like in J2EE (COM, etc.) ? What are the technology concepts that are applied ?
2. DEFINE	How technology concepts shall be applied in the mapping process -> MAPPING RULES
3. APPLY	Mapping Rules in transformation

→ We will apply this scheme for all 3 technologies: J2EE, COM, .NET

Copyright © Fraunhofer IESE 2003

Mapping Rules – Why ?

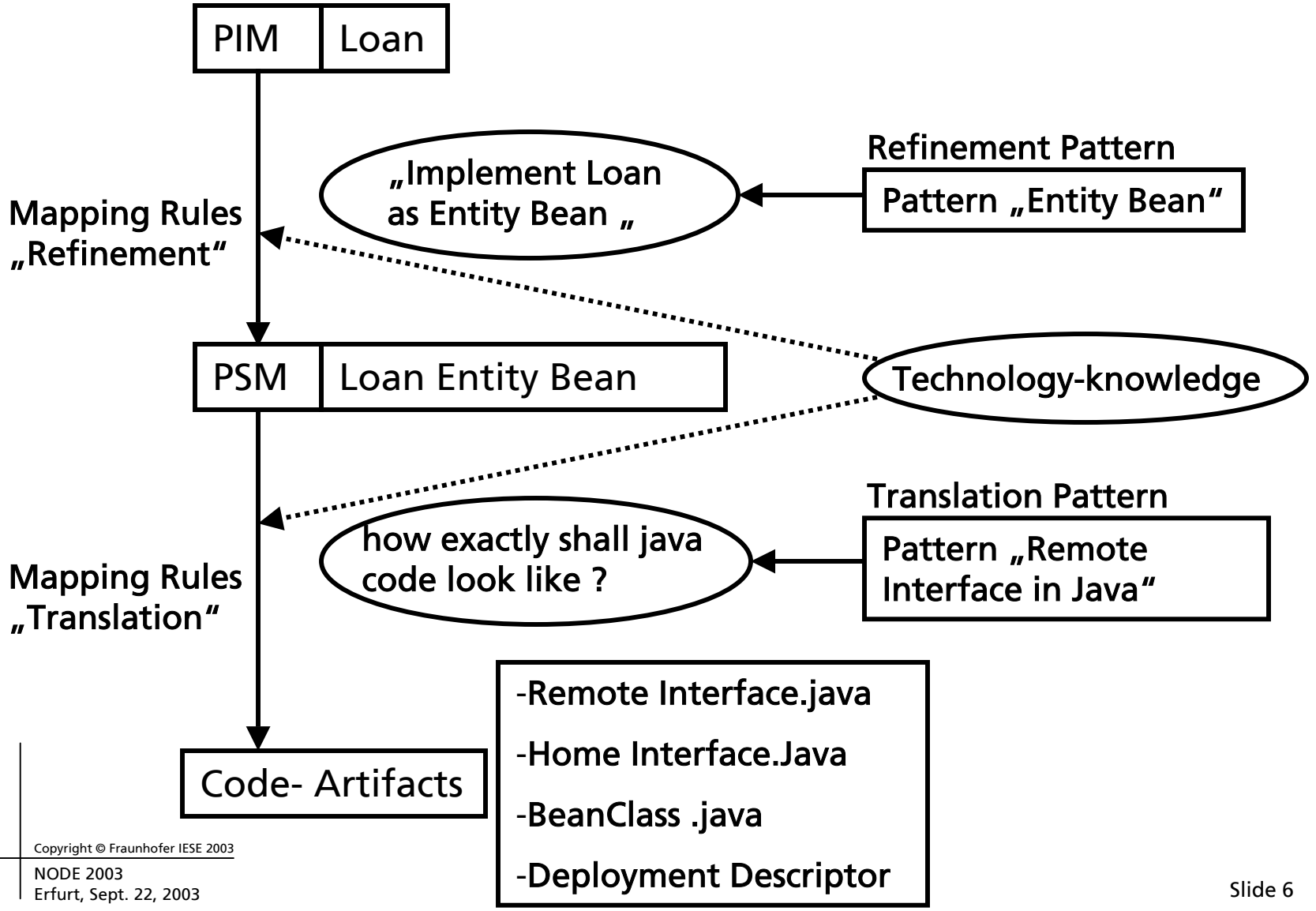
Outline
Introduction
J2EE
COM
.NET
Variability Management
Conclusions & Discussion

- What happens in traditional implementation ?
- Example - PIM: Persistent Object „Loan“
  - Developer A implements Loan as javaclass using a third-party Persistence Framework (e.g. Castor)
  - Developer B implements Loan as Entity Bean
  - Developer C implements Loan as Session Bean using JDO (Java Data Objects)
- Result ?
  - Inhomogeneous, non-systematic solutions
  - Hard to maintain & understand
  - Nearly impossible to accomplish best practices
- Reduce these problems -> Control, Systematization

Copyright © Fraunhofer IESE 2003

Mapping Rules – How ?

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

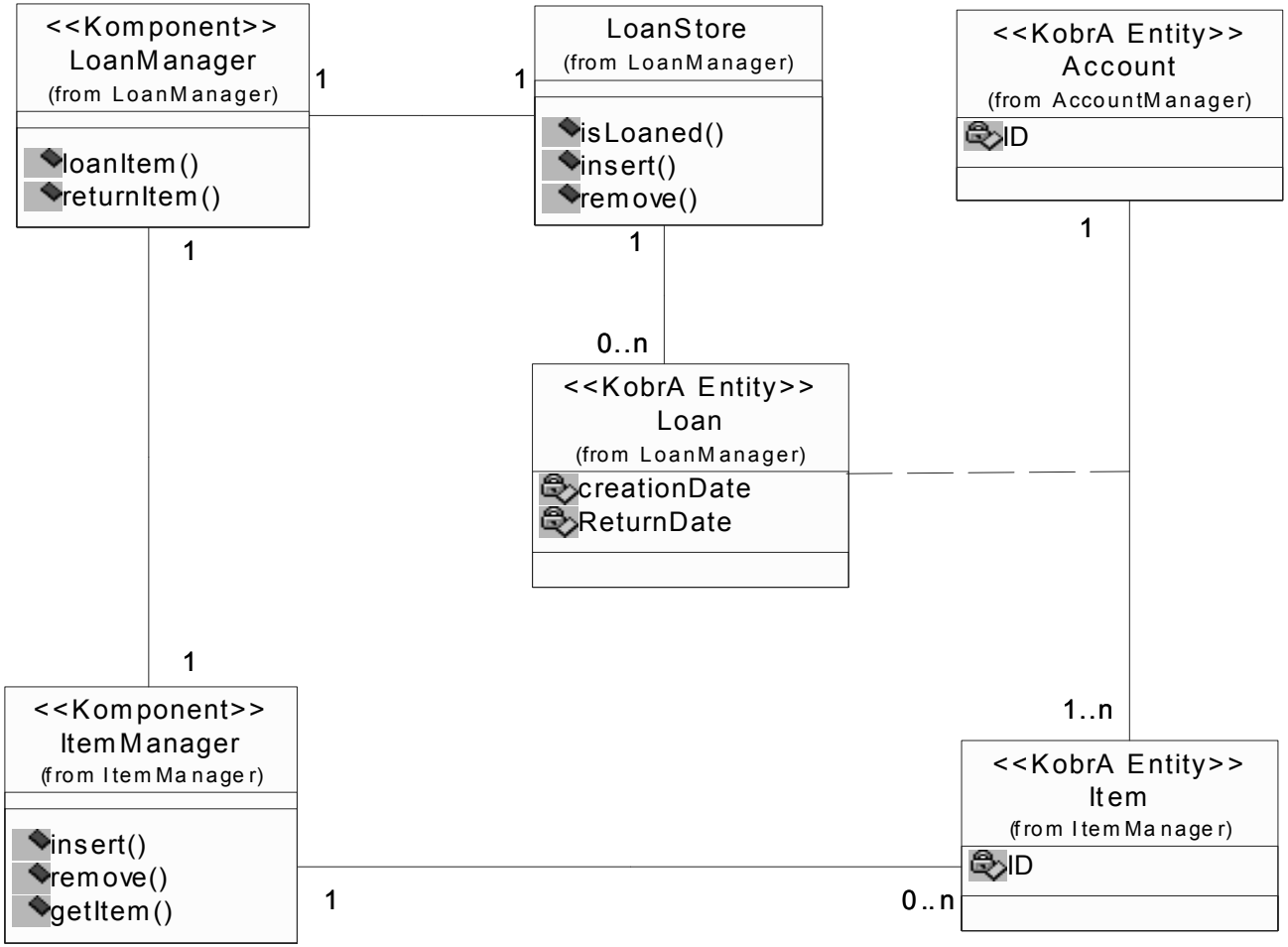


Copyright © Fraunhofer IESE 2003

Casestudy : PIM for LoanManager

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

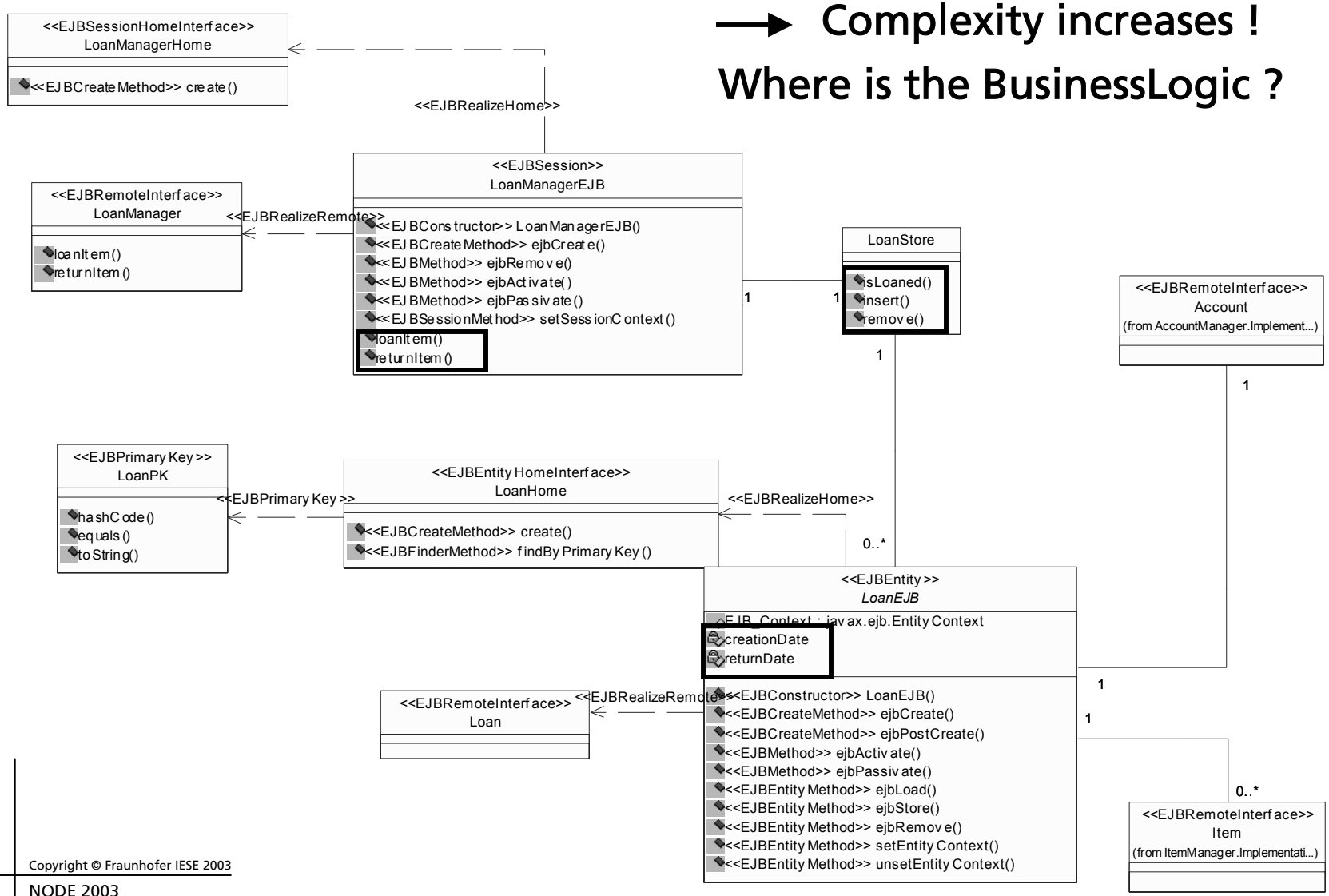
Casestudy: PSM in J2EE for LoanManager

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

→ Complexity increases !

Where is the BusinessLogic ?



Copyright © Fraunhofer IESE 2003

Schedule

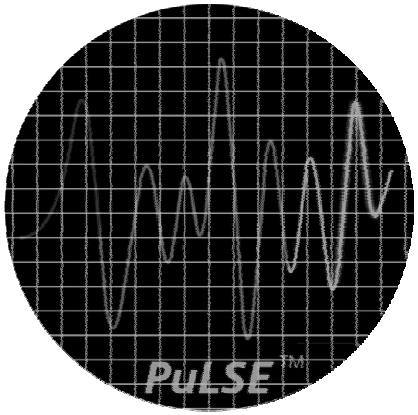
Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- **Part II**
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - **J2EE (30 min)**
  - COM (10 min)
  - .NET (10 min)
  - Variability Management (10 min)
  - Conclusions and Discussion (15 min)

Copyright © Fraunhofer IESE 2003

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

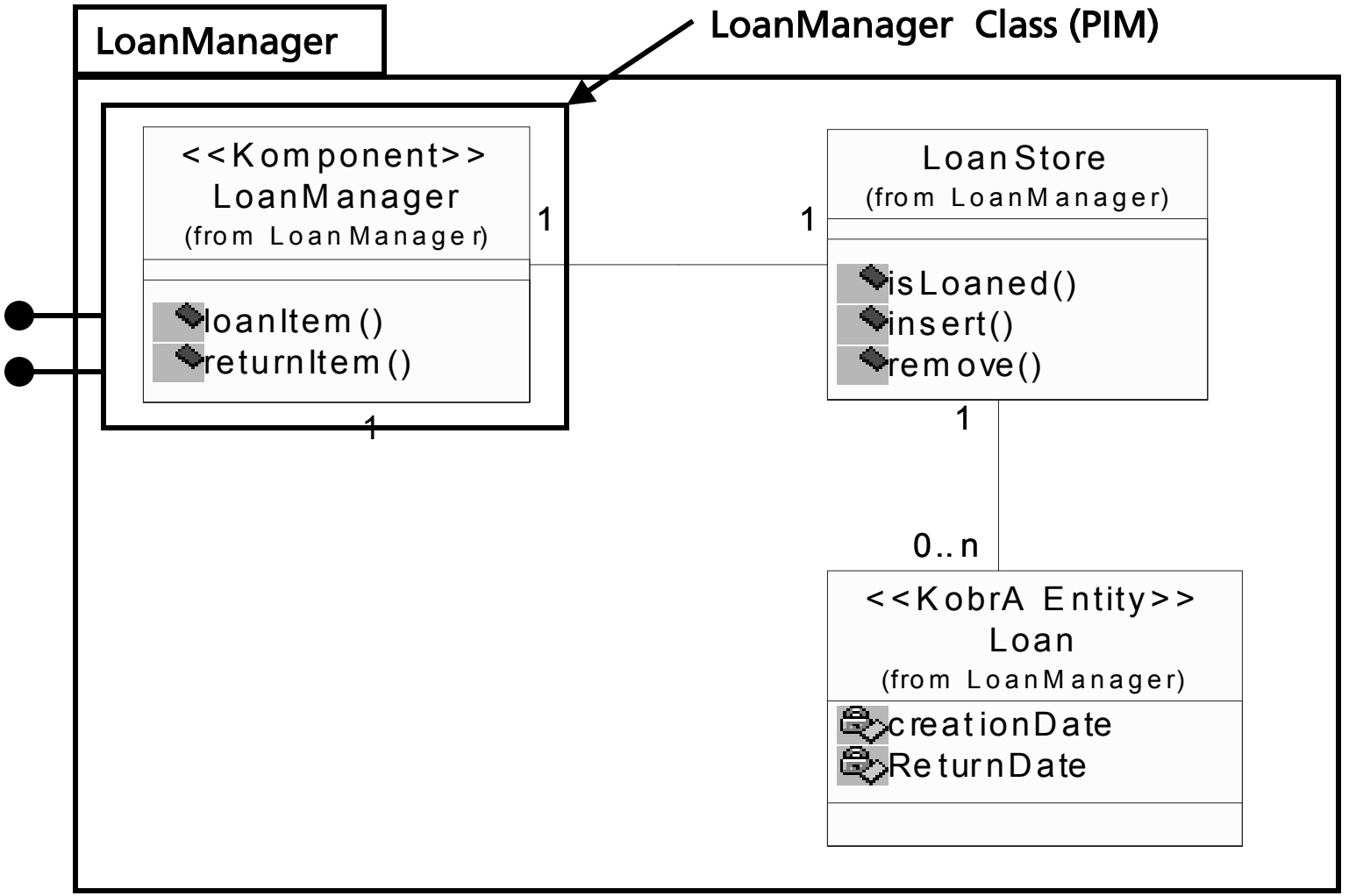


J2EE

Copyright © Fraunhofer IESE 2003

J2EE Mapping: Step by Step

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Mapping: Understand, Define, Apply

PIM: LoanManager Class

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

1. Understand Technology Concepts in J2EE

- SessionBeans are used for non-persistent Components
- EntityBeans are used for persistent components
- MessageDrivenBeans are applied for special message-services
- Java Classes for internal functionality (i.e. not available from outside)

2. Define Mapping Rules (to-be-applied technology concepts)

„A non-persistent component shall be implemented as a SessionBean“

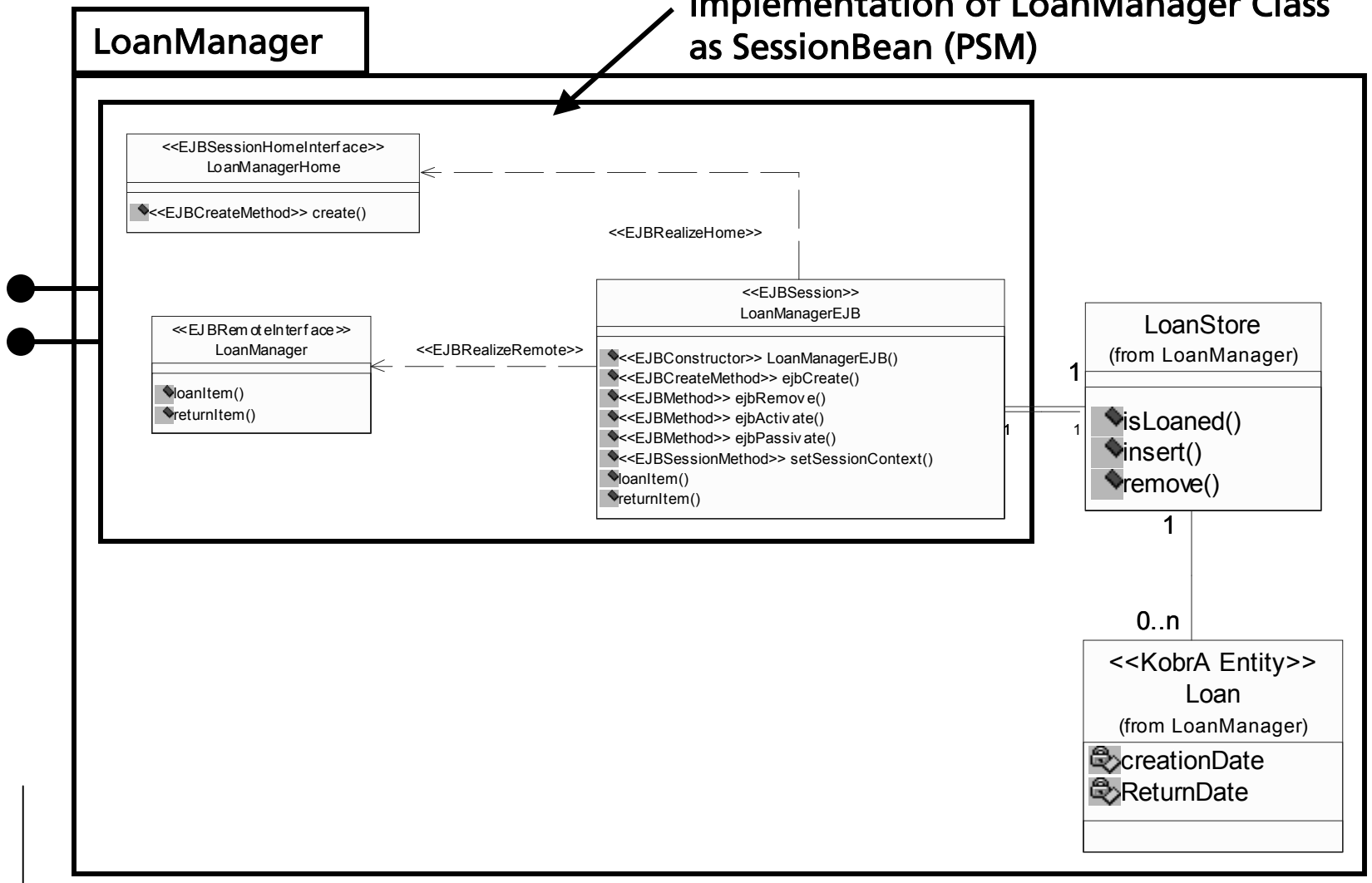
3. Apply Mapping Rules

LoanManager Class itself is not persistent, rather it manages the loaning process and respective loan entities.
-> implement LoanManager Class as SessionBean

Copyright © Fraunhofer IESE 2003

LoanManager: PSM for J2EE

Implementation of LoanManager Class as SessionBean (PSM)



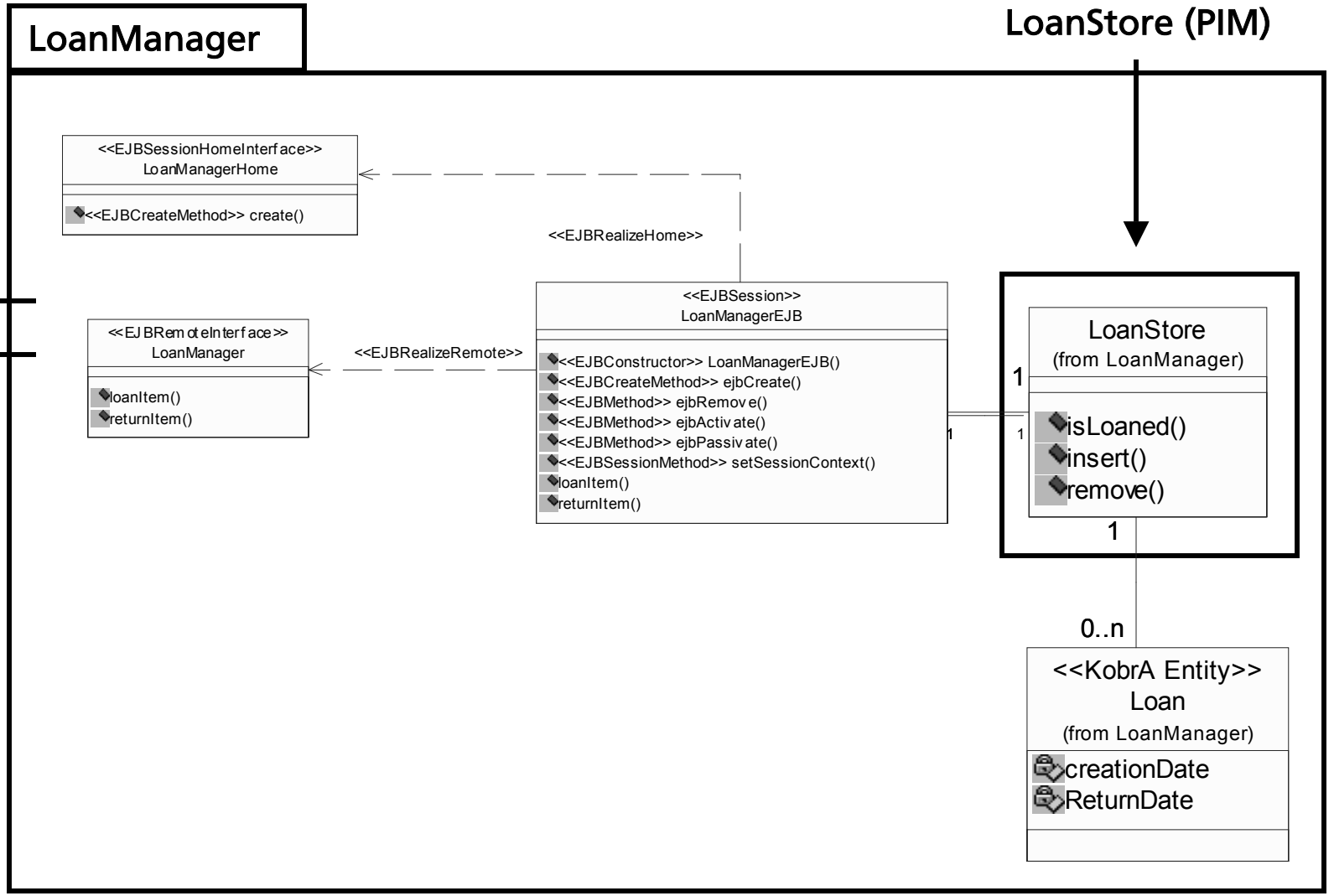
Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

Copyright © Fraunhofer IESE 2003

LoanStore: PIM

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Mapping: Understand, Define, Apply

PIM: LoanStore Class

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

1. Understand Technology Concepts in J2EE

- SessionBeans are used for non-persistent Components
- EntityBeans are used for persistent components
- MessageDrivenBeans are applied for special message-services
- Java Classes for internal functionality (i.e. not available from outside)

2. Define Mapping Rules (to-be-applied technology concepts)

„Internal functionality is implemented using simple Java Classes

3. Apply Mapping Rules

LoanStore implements functionality that is not used outside the component, i.e. used only inside the component

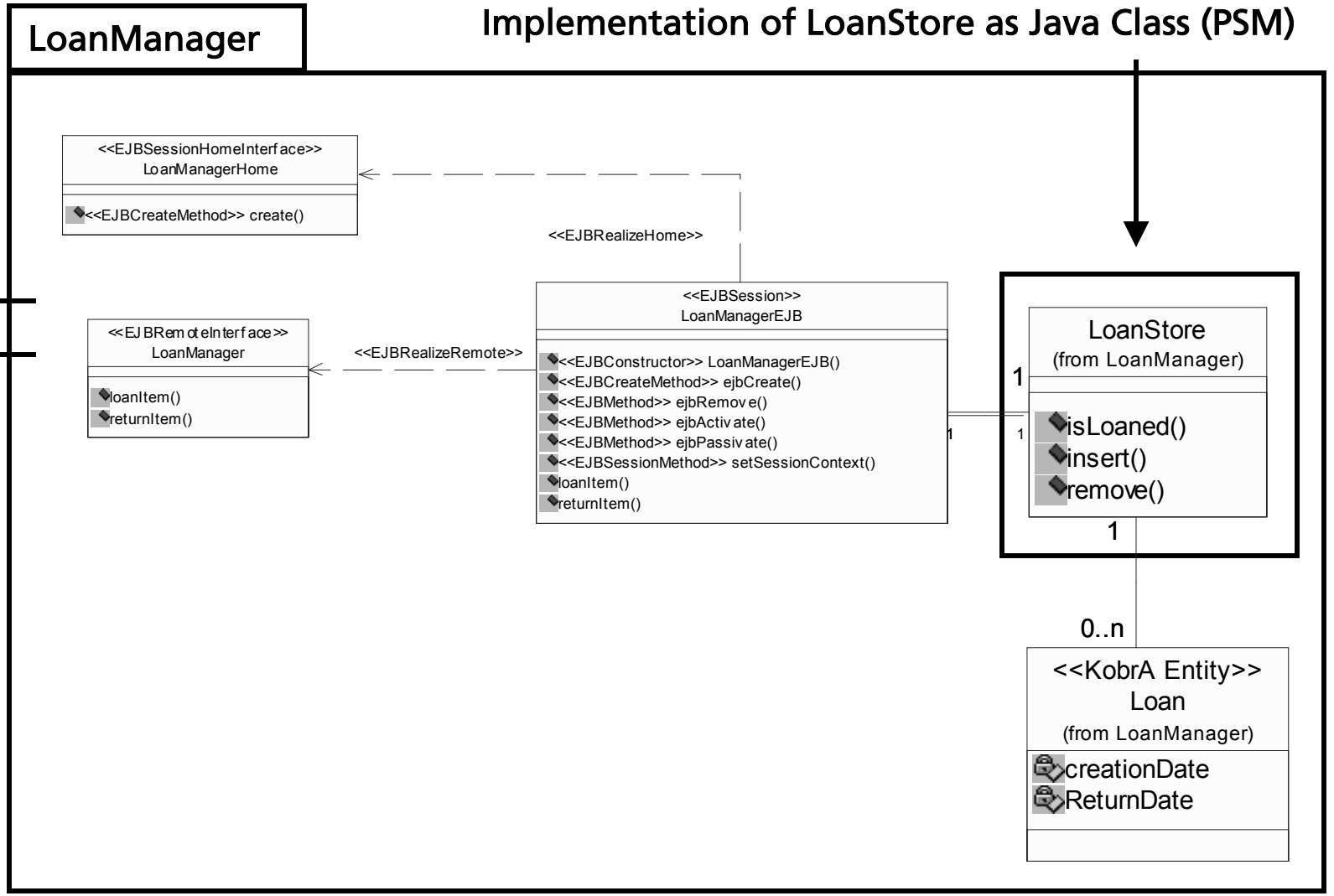
-> internal functionality
-> implement LoanStore as simple java class

Copyright © Fraunhofer IESE 2003

LoanStore: PSM for J2EE

Implementation of LoanStore as Java Class (PSM)

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

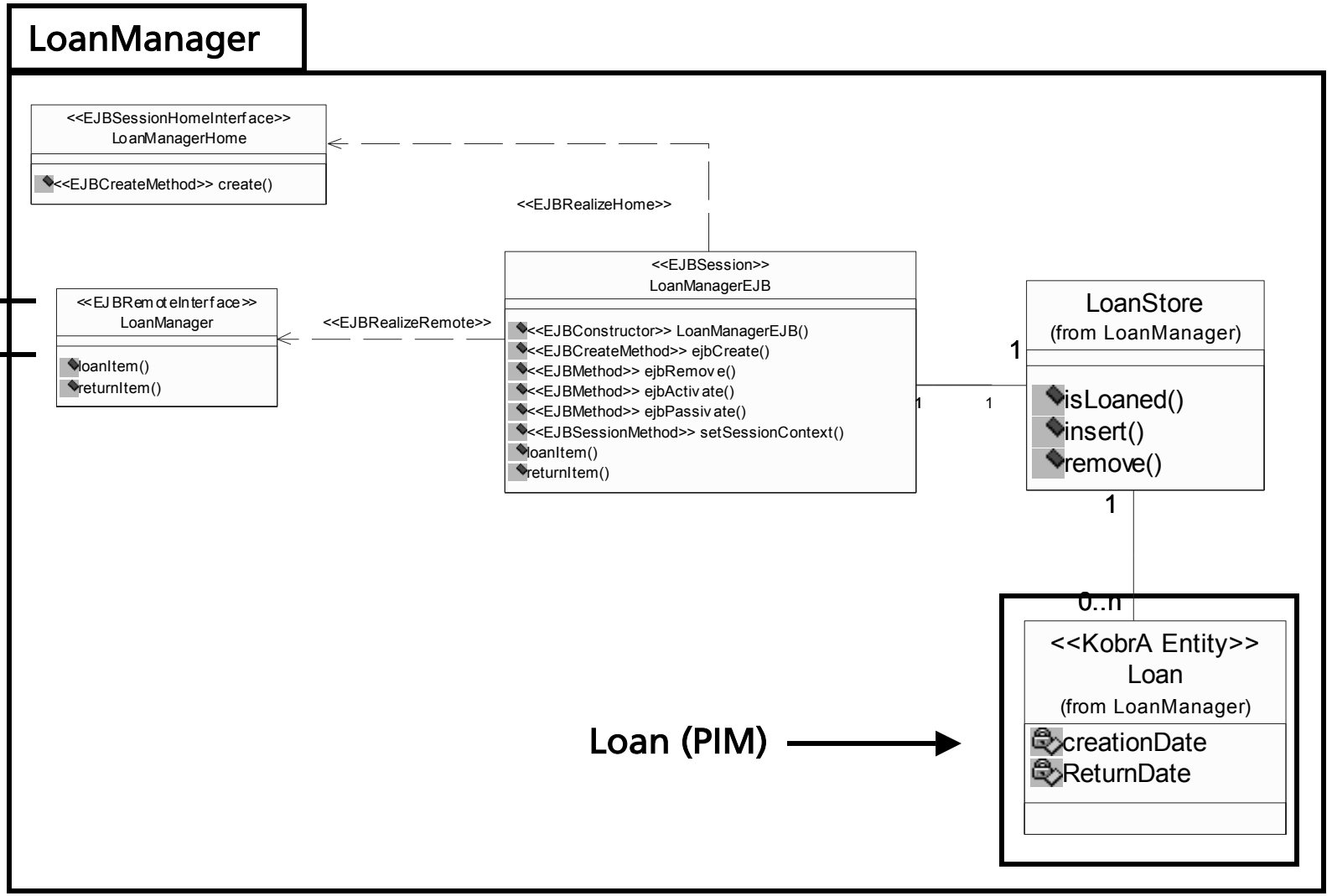


Copyright © Fraunhofer IESE 2003

Loan: PIM

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Mapping: Understand, Define, Apply

PIM: Loan Class

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

1. Understand Technology Concepts in J2EE

- SessionBeans are used for non-persistent Components
- EntityBeans are used for persistent components
- MessageDrivenBeans are applied for special message-services
- Java Classes for internal functionality (i.e. not available from outside)

2. Define Mapping Rules (to-be-applied technology concepts)

„Persistent Entities are implemented as Entity Beans“

3. Apply Mapping Rules

Loan is a persistent entity

-> implement Loan as Entity Bean

Copyright © Fraunhofer IESE 2003

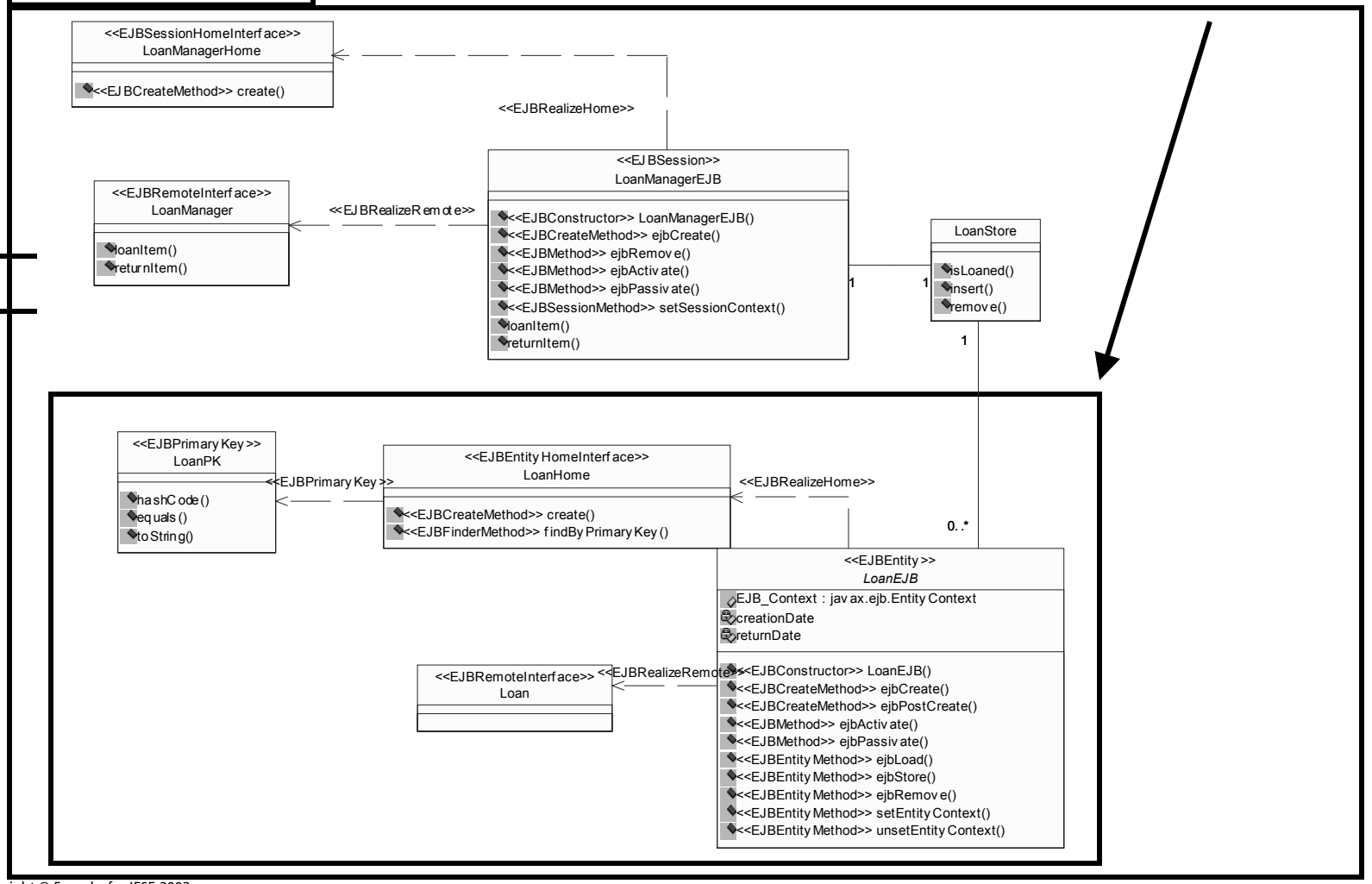
Loan: PSM for J2EE

LoanManager

Implementation of Loan as Entity Bean (PSM)

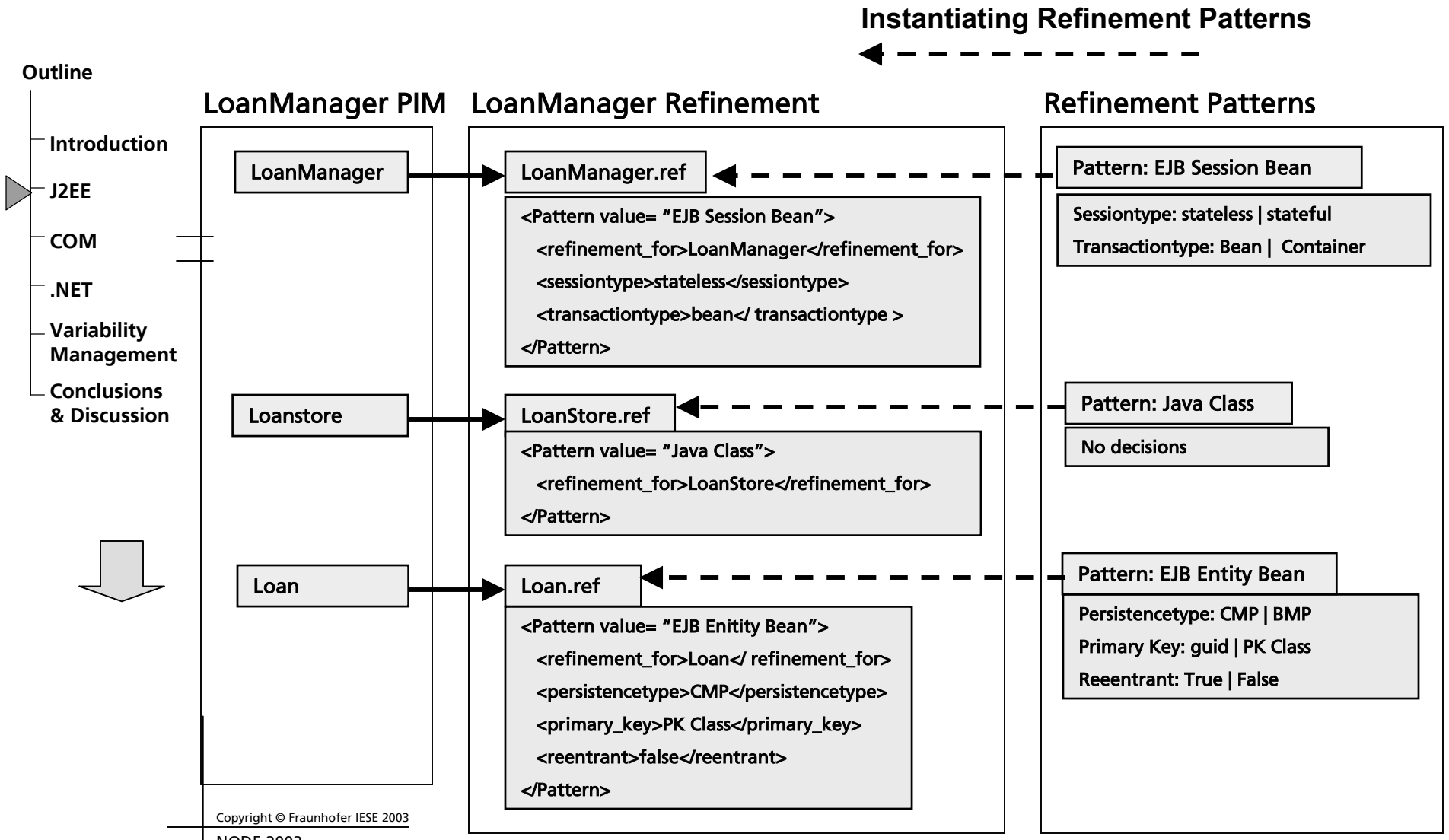
Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Wrap-Up: Pattern-Based Refinement



Copyright © Fraunhofer IESE 2003

Mapping to J2EE - Conclusions

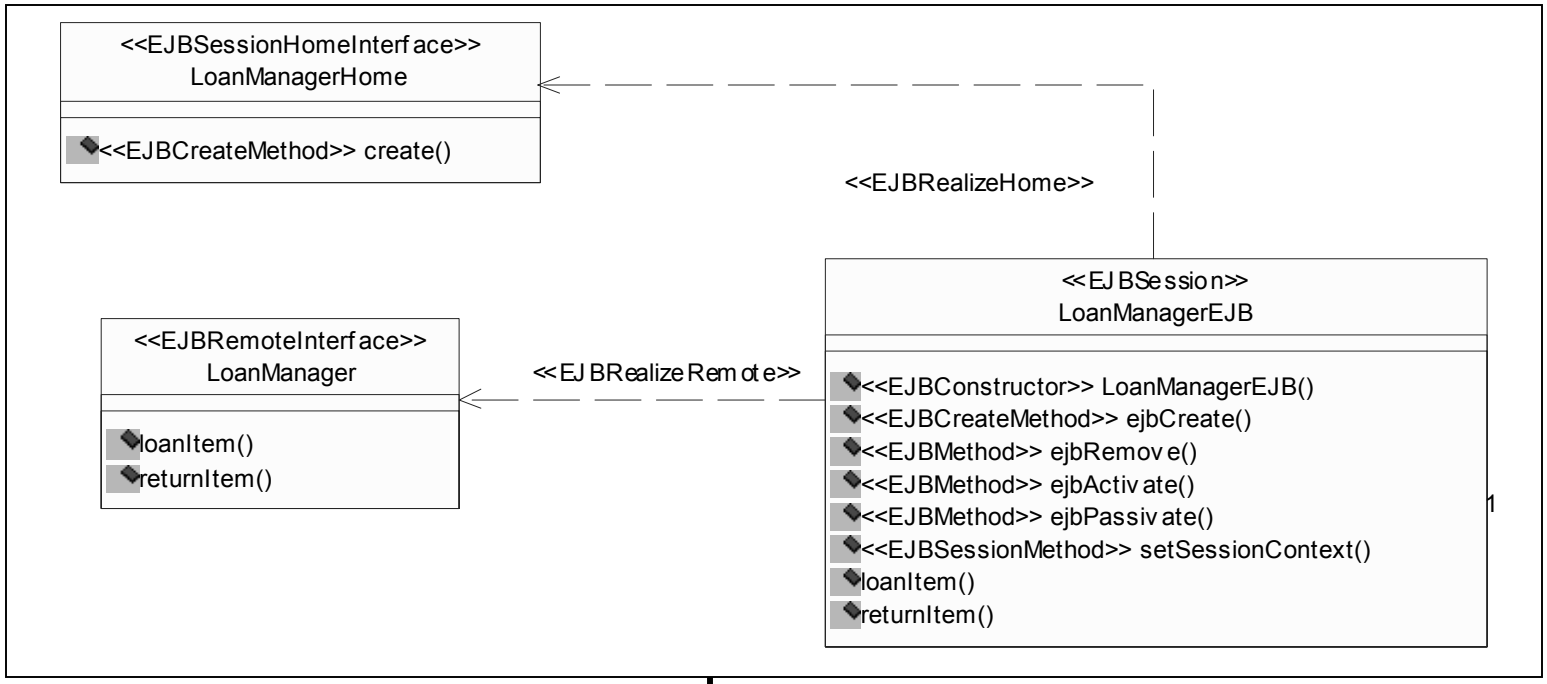
Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- Refinements (Mapping Rules) - Benefits
  - Platform decisions are explicitly tracked and available !
  - Helps to concentrate on the essential engineering information: -> what happened during platform mapping ?
  - Focus on few decisions helps to keep intellectual control
  - Mapping Decisions have an impact on complexity
- Refinements handle technological variability
  - J2EE Refinement Patterns support mapping PIM -> J2EE
  - COM Refinement Patterns support mapping PIM -> COM
- PSM
  - Hard to oversee, mixing of different informations: business information, platform information & platform decisions
  - Contains Platform Overhead (e.g., EJBSessionBean)
  - Really important ?
- J2EE Platform
  - Exposes many technological details (see PSM -> too many ?)
  - A lot of decisions needs to be resolved (sessiontype, etc.)

Copyright © Fraunhofer IESE 2003

Translation: Mapping PSM -> Code

PSM: J2EE Implementation of LoanManager Class



Mapping Rules „Translation“

Code- Artifacts

Remote Interface.java, HomeInterface.Java,

BeanClass .java, Deployment Descriptor

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

Copyright © Fraunhofer IESE 2003

Mapping: Understand, Define, Apply

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

Implementation of Remote Interface in java

```
package LoanManager;
import javax.ejb.EJBObject;
import java.rmi.RemoteException;
public interface LoanManagerRemote extends EJBObject  {
    public String loanItem(String id)  throws RemoteException;
    public String returnItem(String id) throws RemoteException; }
```

1. Understand Language Concepts in J2EE (java)

- Remote Interfaces import RemoteExceptions
- Methods throw Remote Exceptions

2. Define Mapping Rules (to-be-applied technology concepts)

- „Remote Interfaces always import java.rmi.RemoteException“
- apply specific organization styles: naming conventions, code-docu

3. Apply Mapping Rules

Remote Interfaces „all look the same “, no component-specific decisions.
-> For all (!) Remote Interfaces in the PSM the code shall be generated applying the same generic TRANSLATION PATTERN.

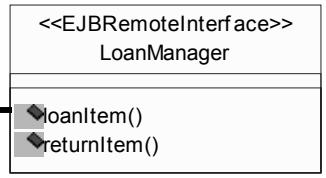
Copyright © Fraunhofer IESE 2003

Translation: Mapping PSM -> Code

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

PSM: LoanManager Remote Interface



Alternatives: e.g. different coding styles

Translation Pattern for Remote Interfaces

```
<?xml version="1.0" encoding="UTF-8"?>
<translation_pattern value="remote_interface_to_java">
<package>component_name</package>
   <imports>
     <import value="javax.ejb.EJBObject"/>
     <import value="javax.rmi.RemoteException"/>
   </imports>
   <interface_definition>
     <interface_scopec>public interface</interface_scopec>
     <interface_name>component_name</interface_name>
     <throws>throws RemoteException</throws>
     <methods>component_methods</methods>
   </interface_definition>
</translation_pattern>
```

For example, the Pattern defines which libraries are imported: javax.rmi.RemoteException

Code: LoanManagerRemote.java

```
package LoanManager;
import javax.ejb.EJBObject;
import javax.rmi.RemoteException;
public interface LoanManagerRemote extends EJBObject  {
   public String loanItem(String id)  throws RemoteException;
   public String returnItem(String id) throws RemoteException; }
```

Copyright © Fraunhofer IESE 2003

Schedule

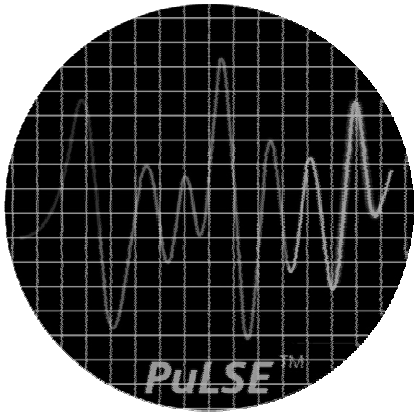
Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- **Part II**
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - J2EE (30 min)
  - **COM (10 min)**
  - .NET (10 min)
  - Variability Management (10 min)
  - Conclusions and Discussion (15 min)

Copyright © Fraunhofer IESE 2003

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



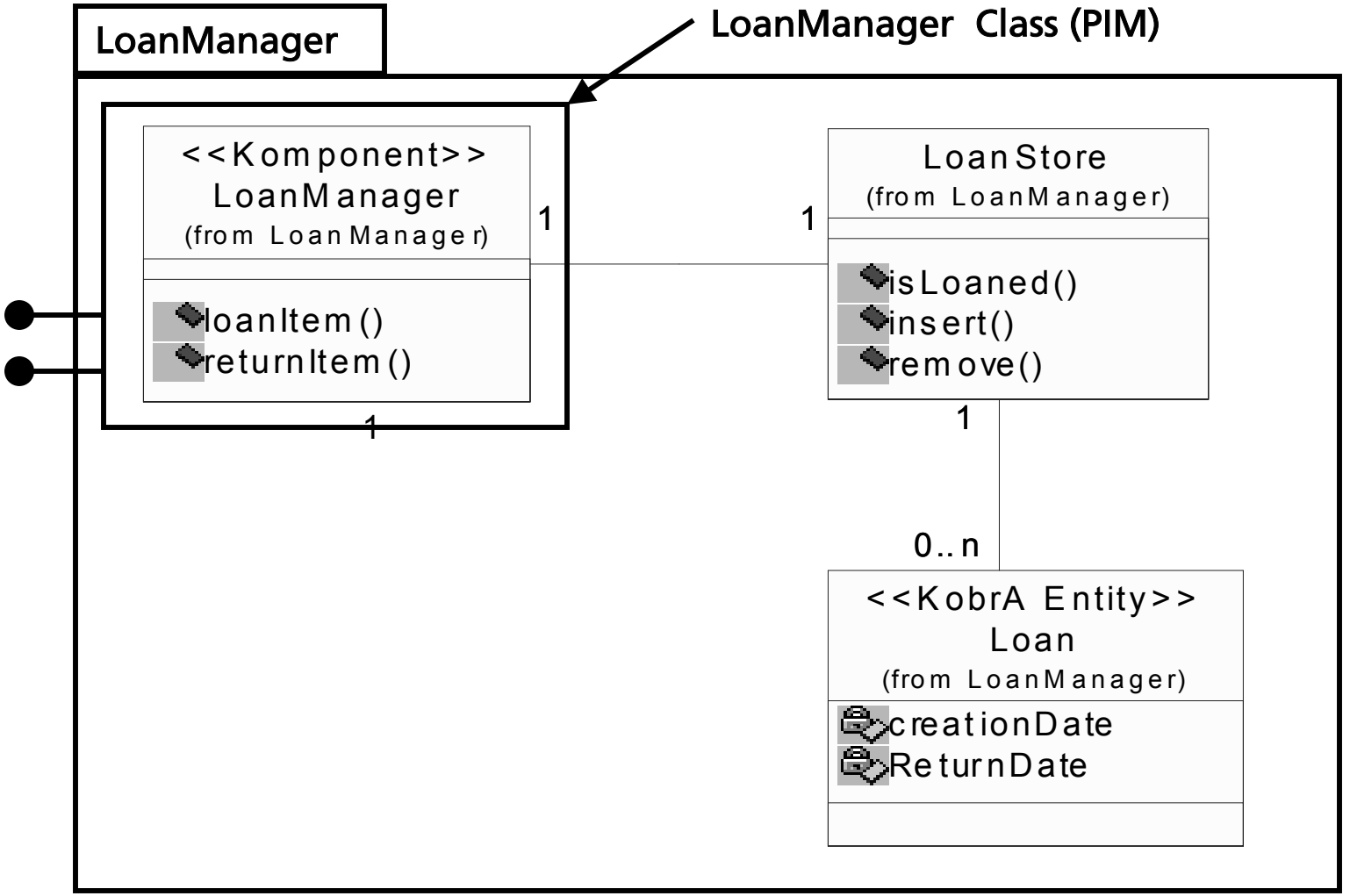
COM

Copyright © Fraunhofer IESE 2003

COM (C++ and ATL) Mapping: Step by Step

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Mapping: Understand, Define, Apply

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

PIM: LoanManager Class

1. Understand Technology Concepts in COM (C++ and ATL)

Classes that shall be available as COM components:

- have to be derived from two generic template-classes: „CCOMCoClass“ & IDispatchImpl
- Implementation („LoanManager“) and Interface („ILoanManager“) have to be separated from each other

2. Define Mapping Rules (to-be-applied technology concepts)

ATL (Active Template Library) predefines the mapping rules and generates code-templates

3. Apply Mapping Rules

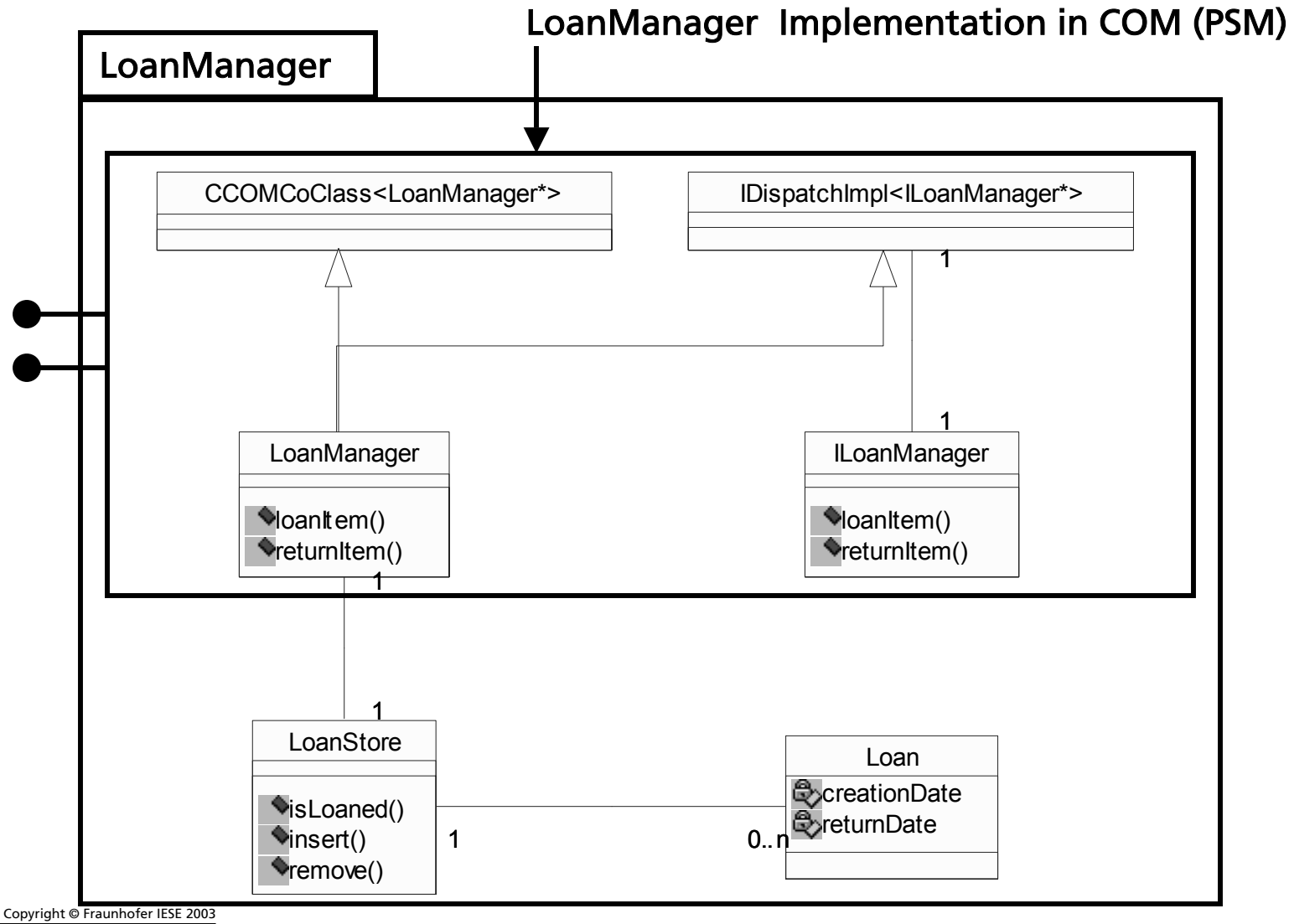
-> ATL-wizards for creating COM-components automatically generate the code-artifacts (i.e. mapping into PSM and code is provided by IDE)

Copyright © Fraunhofer IESE 2003

LoanManager: PSM for COM

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Mapping to COM - Conclusions

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- ATL - Active Template Library
  - ATL provides wrapping around COM (hides complexity, e.g. reference counting and memory management)
  - Wizard generates most classes (stubs)
  - Complex architecture: templates & inheritance
  - Usage is error-prone:
    -> forward generation OK, iterative changes such as removal or renaming of elements problematic (have to be done on generated code, model/wizard-level badly supported).
  - No built in Concept for Object-Persistence
  - No analogous construct to „Entity Beans“ (EJB, J2EE)

Copyright © Fraunhofer IESE 2003

Schedule

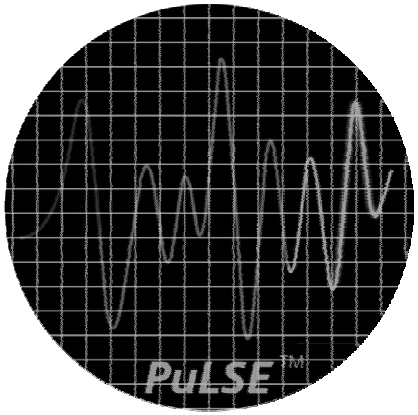
Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- **Part II**
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - J2EE (30 min)
  - COM (10 min)
  - **.NET (10 min)**
  - Variability Management (10 min)
  - Conclusions and Discussion (15 min)

Copyright © Fraunhofer IESE 2003

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



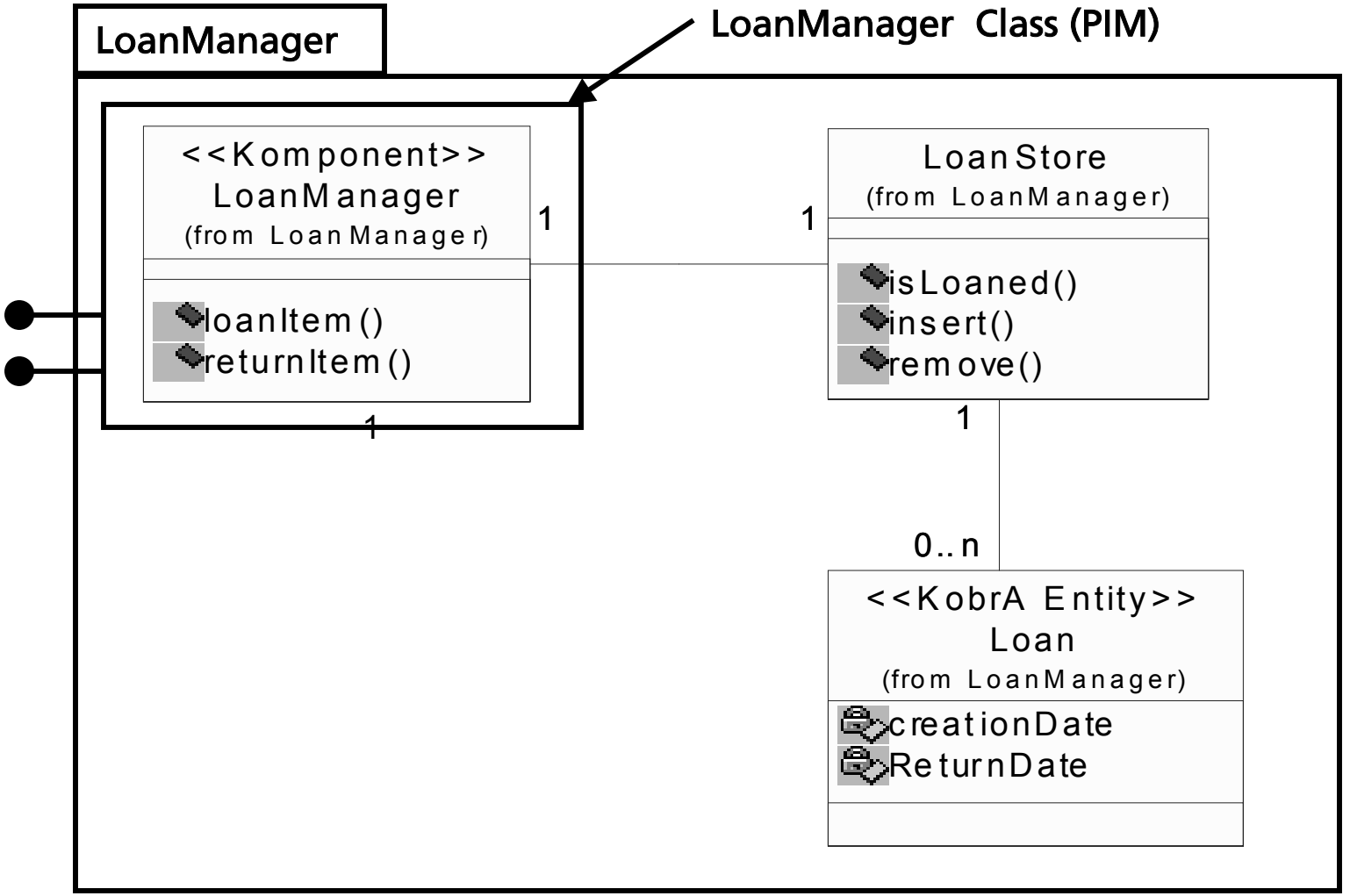
dOtNET

Copyright © Fraunhofer IESE 2003

.NET Mapping: Step by Step

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Mapping: Understand, Define, Apply

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

PIM: LoanManager Class

1. Understand Technology Concepts in .NET

Technology-Options:
- „Serviced Components“ provide rich support for Business Components (e.g. mechanisms for security, transactions, roles, etc.)
- „Remoting Components“ provide minimal support for Business Components (only support for remote access).

2. Define Mapping Rules (to-be-applied technology concepts)

„Business components shall be derived from Serviced Components“

„Separation between Implementation and Internal functionality is implemented using two different Classes“

3. Apply Mapping Rules

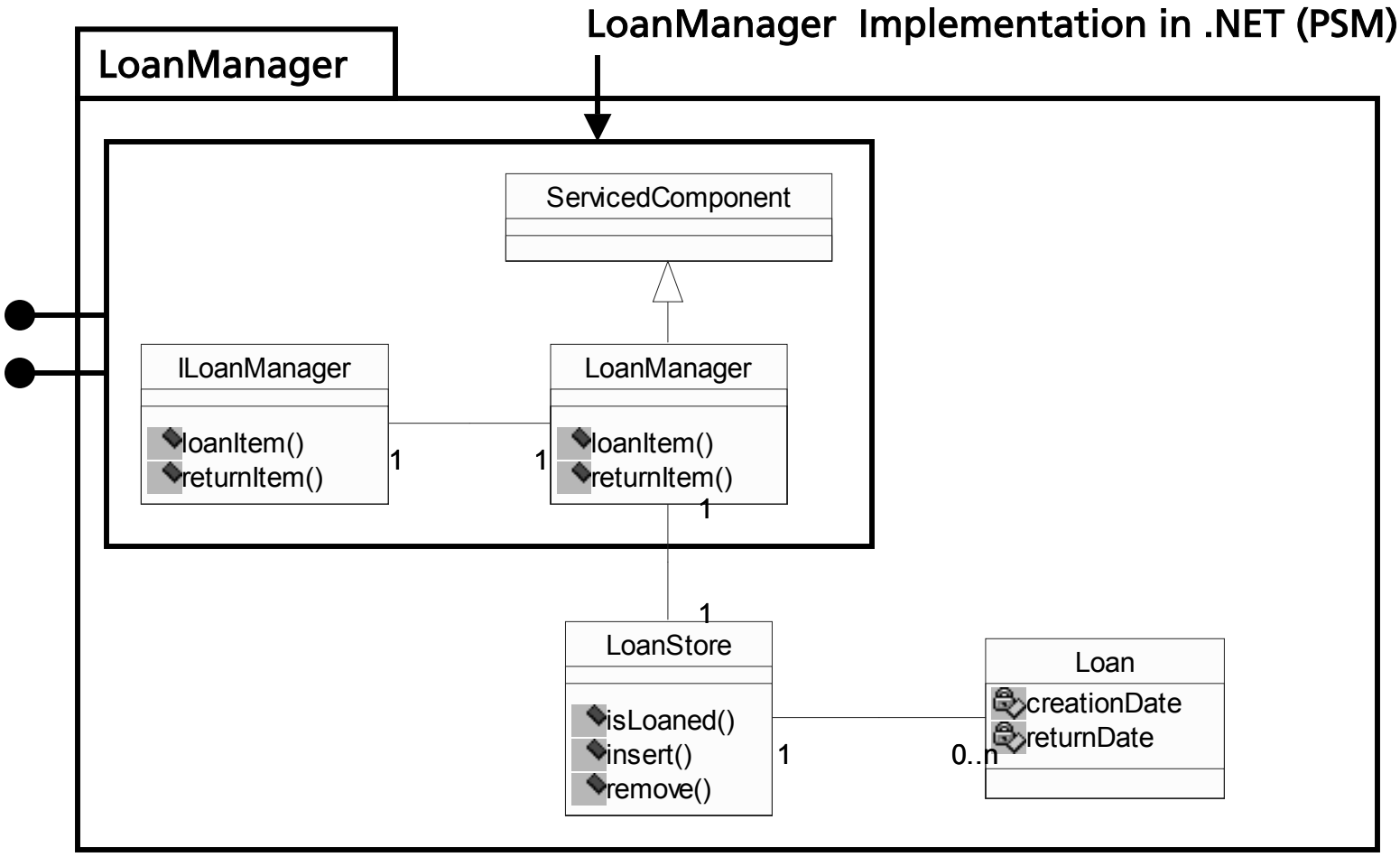
-> LoanManager is derived from Serviced components
-> LoanManager is separated into the interface class „ILoanManager“ and the implementation class „LoanManager“

Copyright © Fraunhofer IESE 2003

LoanManager: PSM for .NET

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Mapping to .NET - Conclusions

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- Comparison to COM+ (ATL & C++) Architecture)
  - Simplified Component Model
  - Simplified Component Development & Deployment
  - COM+ Infrastructure still available: Serviced Components
  - Like COM: no built-in persistence (cf. EJB EntityBeans)
- 3 Component Approaches in .NET (-> Workshop 04)
  - Serviced Components (reusing COM+)
  - Remoting (lightweight)
  - Web Services: internet & platform-independent connectivity

Copyright © Fraunhofer IESE 2003

Schedule

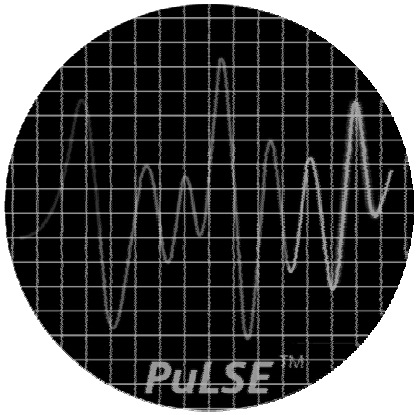
Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- **Part II**
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - J2EE (30 min)
  - COM (10 min)
  - .NET (10 min)
  - **Variability Management (10 min)**
  - Conclusions and Discussion (15 min)

Copyright © Fraunhofer IESE 2003

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



variability Management

Copyright © Fraunhofer IESE 2003

Variation in Technology (Platform)

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- Technological variations are handled by
  - Technology-specific refinement patterns or pattern sets
  - Alternative translation patterns
- Refinements handle technological variation
  - PIM component - > Pattern „EJBSessionBean“ -> J2EE
  - PIM component -> Pattern „.NET Serviced Component -> .NET
- Translations handle technological variation
  - Required for code generation in specific platform
  - Allows to establish organization-specific coding styles, - guidelines and documentation principles.

Copyright © Fraunhofer IESE 2003

Variation in Business Logic

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

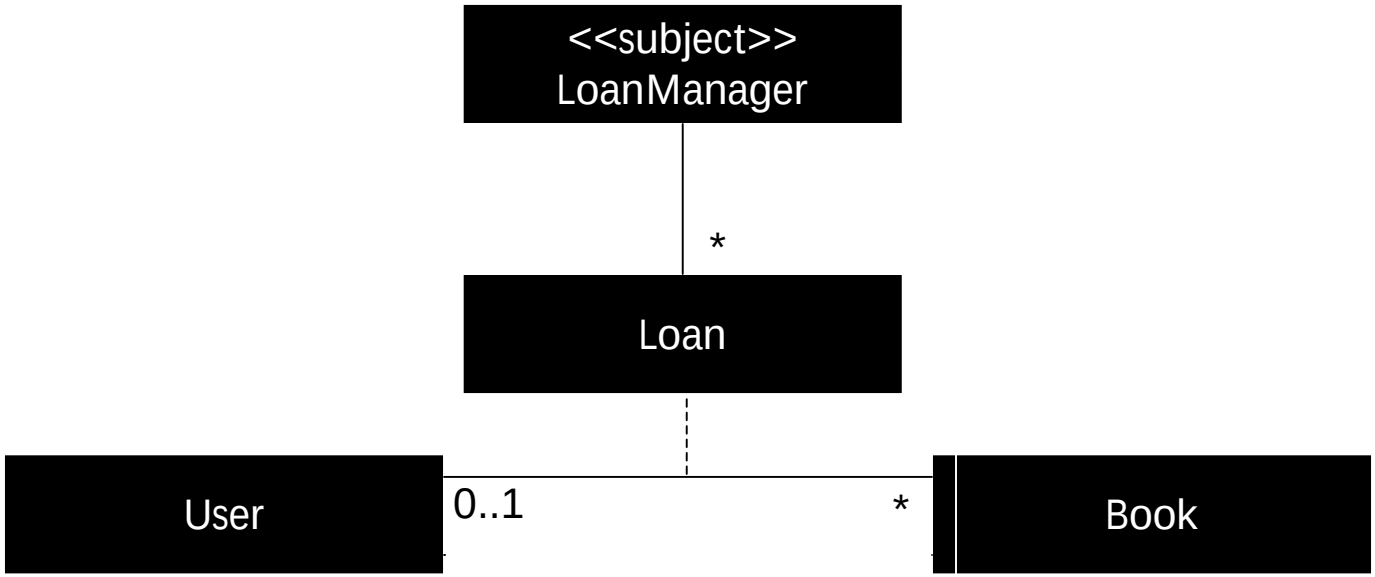
- Variations in application or business logic are handled
  - At the plattform-independent level (PIM, component models)
  - Through variability patterns that produce generic implementations
- Products (or variants) are created by
  - Modifications of the plattform-independent component models
  - (Re-)application of implementation patterns

Copyright © Fraunhofer IESE 2003

Instantiation of a PL UML Class Diagram

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Schedule

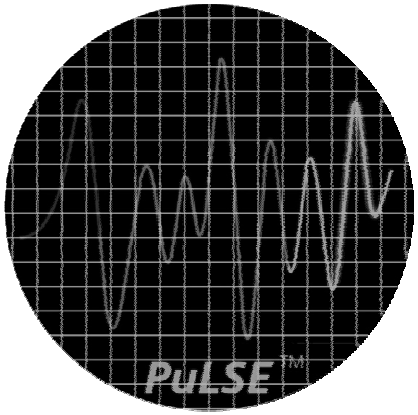
Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- **Part II**
  - Introduction (15 min)
    - → Case Study: Library System
    - → MDA principles: Mapping PIMs to PSMs
  - J2EE (30 min)
  - COM (10 min)
  - .NET (10 min)
  - Variability Management (10 min)
  - **Conclusions and Discussion (15 min)**

Copyright © Fraunhofer IESE 2003

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

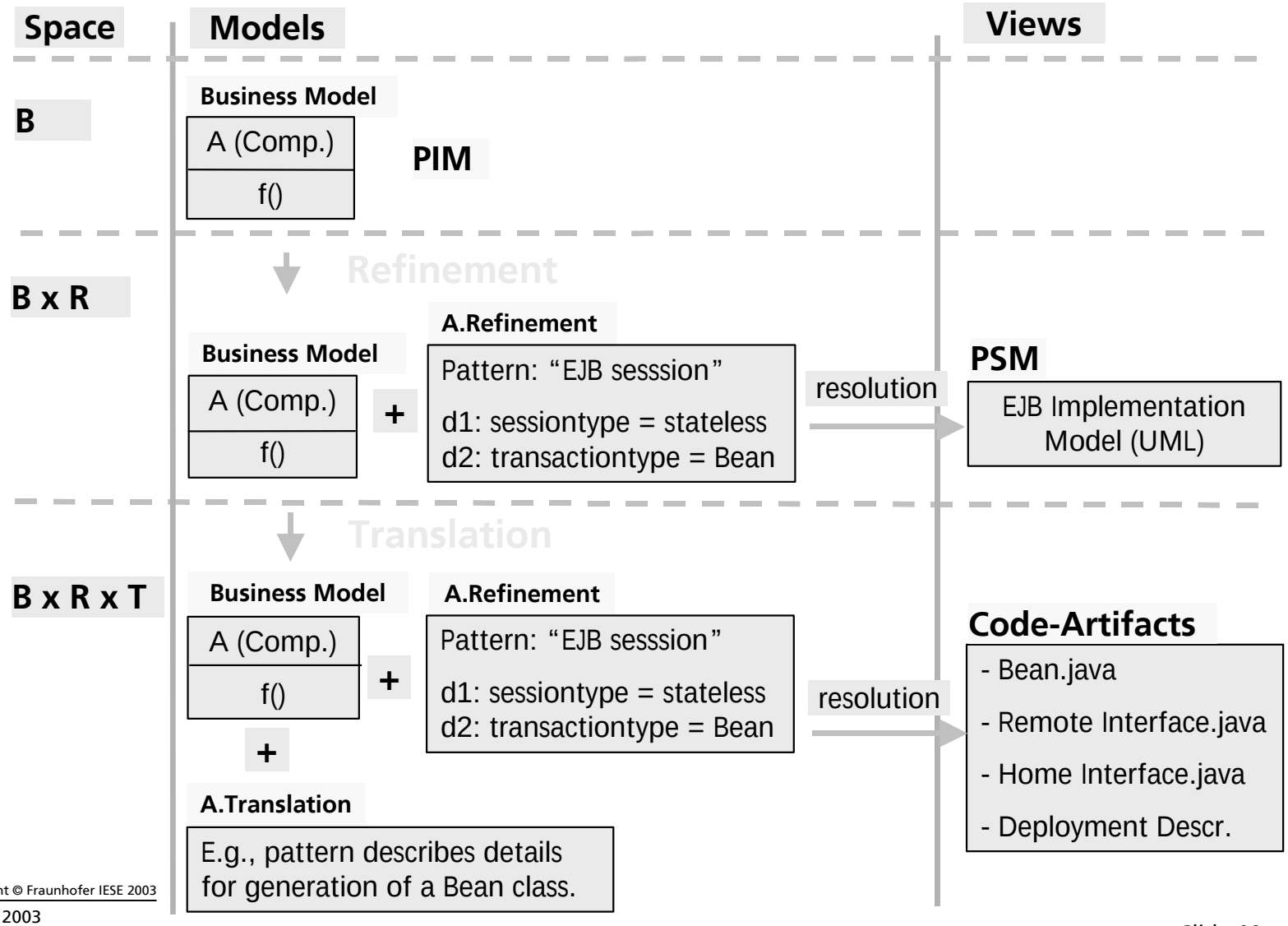


conclusion

Copyright © Fraunhofer IESE 2003

Conclusion I: Concept for MDA Infrastructure

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Conclusion II: Benefits

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- Handles Technological Variability
  - „Refinements“ capture platform specific mapping rules: PIM -> PSM
  - „Translations“ capture platform specific mapping rules: PSM -> Code
  - Both are based on Patterns -> Implementation follows best practices
- Approach – Other Characteristics
  - Highly separated model (PIM, Refinement, Translation)
  - No Redundancy
  - High Potential for Automation (e.g. RemoteInterface to Code) -> acceleration of the development process

Copyright © Fraunhofer IESE 2003

Conclusion II: Issues

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

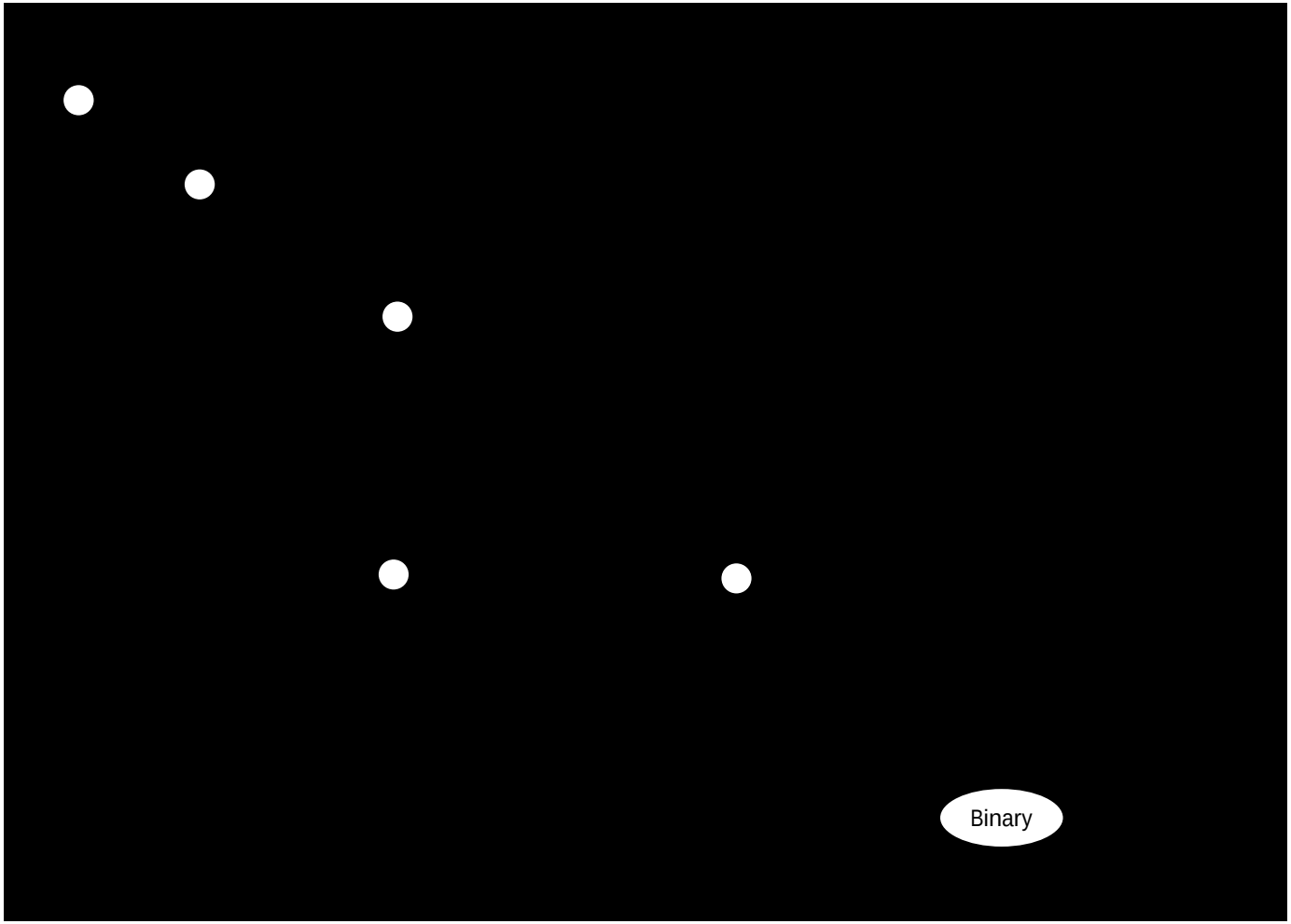
- Issues in „traditional“ MDA Infrastructure
  - Ambiguous descriptions: How exactly are the transformations done ?
  - PSM duplicates business logic of PIM -> Redundancy !
  - Redundancy requires extra Synchronization [Des01] !
  - Is extra Synchronization accelerating development ?
  - Is it possible to track and understand the engineering decisions that were made during platform mapping ?

Copyright © Fraunhofer IESE 2003

Conclusion III: Implementation Process

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Questions and Discussion

Outline

- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion



Copyright © Fraunhofer IESE 2003

Related Work

Outline
- Introduction
- J2EE
- COM
- .NET
- Variability Management
- Conclusions & Discussion

- **IESE Technical reports (www.iese.fhg.de/Publications)**

  [KMA03] S. Kettemann, D. Muthig, M. Anastasopoulos. Product Line Implementation Technologies: Component Technology View. IESE-Report No. 015.03/E, 2003. **-> Refinements & Translations**

  [KM02] S. Kettemann, R. Laqua. **Technology EJB**: Developing and Deploying Components with EJB. IESE-Report No. 050.02/E, 2002.

  [KC03] S. Kettemann, C. Jung. **Technology .NET**: Developing and Deploying Business Components with .NET. IESE-Report No. 020.03/E, 2003.

- **IESE Papers**

  [KLM03] ECOOP ws18 (MDA): S. Kettemann, R. Laqua, D. Muthig: „An Optimized MDA Infrastructure Optimizing Agile Development“.

  [KM03] ECOOP ws04 (.NET): S. Kettemann, D. Muthig: „Building Business Components with .NET“.

- **MDA Literature (www.omg.org/mda/presentations.htm)**

  [ORM01] ORMSC Architecture Board. Model Driven Architecture. 2001.

  [Sol+] R. Soley and OMG Staff Strategy Group. Model Driven Architecture. White paper, 2000.

  [Des01] Ph. Desfray. When a major software industry trend meets our toolset. 2001.

Copyright © Fraunhofer IESE 2003