

Vorlesung "Software-Engineering"

Prof. Ralf Möller, TUHH, Arbeitsbereich STS

■ Vorige Vorlesung

■ Software Product Lines

- | Successful software production is more than just composing modules / components

■ Heute

■ Software Reengineering for Product Lines

■ Über das Bewußtwerden und Explizitmachen

- | Process CMM (Capability Maturity Model)
- | People CMM

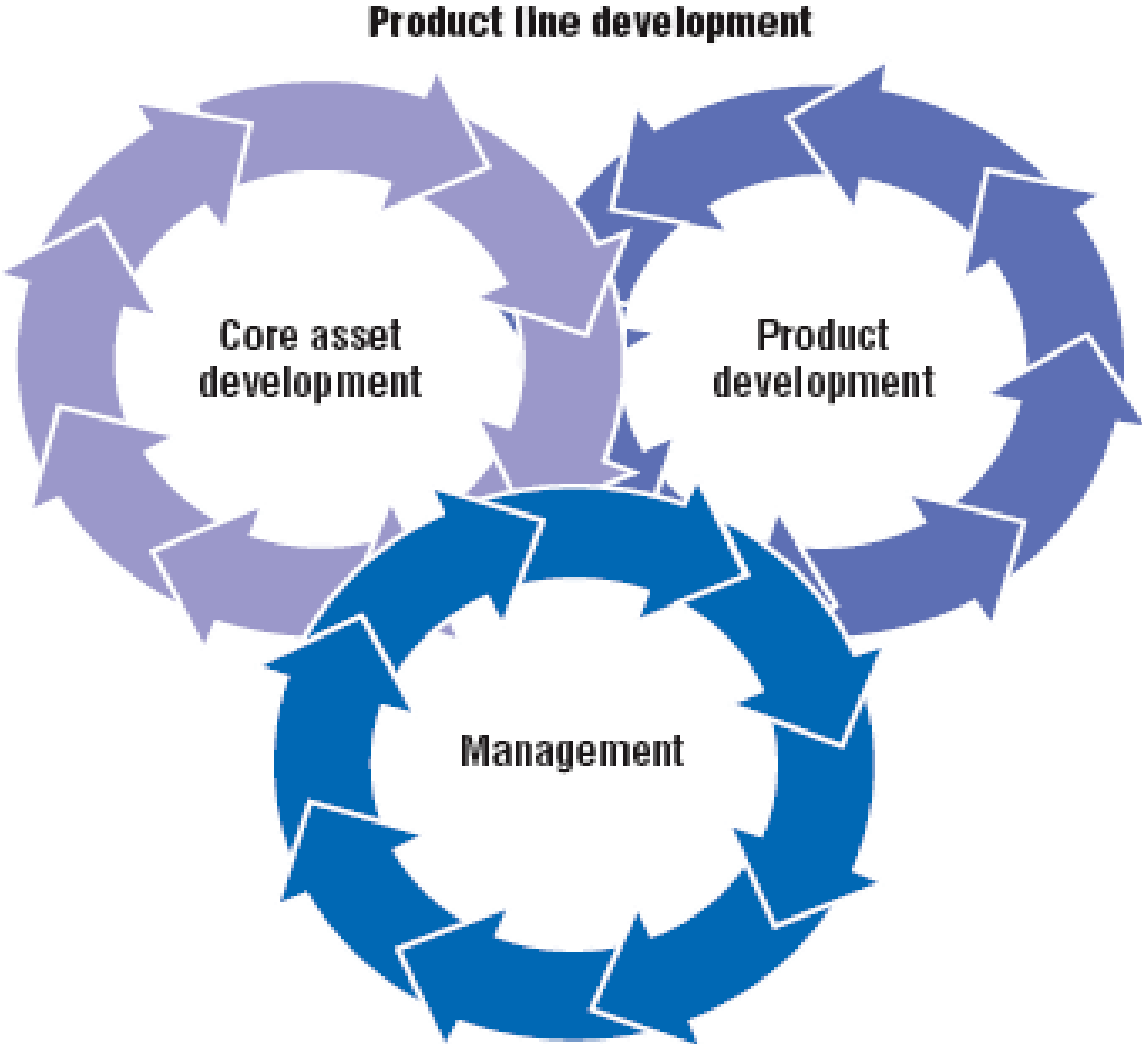
What is a Product Line?

“ A set of software-intensive systems that share a common, managed feature set satisfying a particular market segment’s specific needs or mission and that are developed from a common set of *core assets* in a **prescribed way**”.

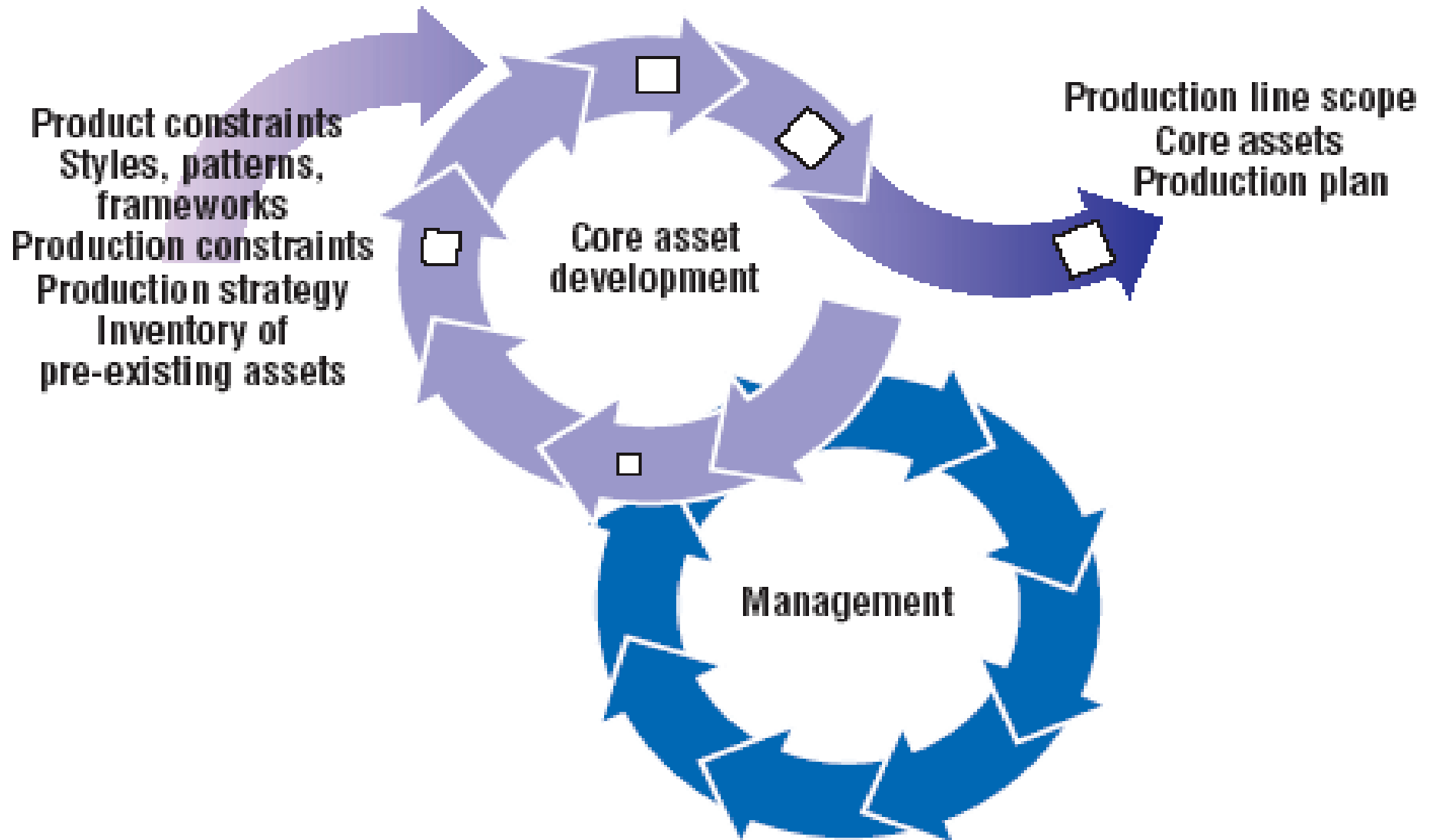
Core Assets

- Basis for the software product line
 - Architecture
 - Reusable Components
 - Domain Models
 - Requirements
 - Schedules
 - Budgets
 - Test plans
 - Process descriptions
 - And more

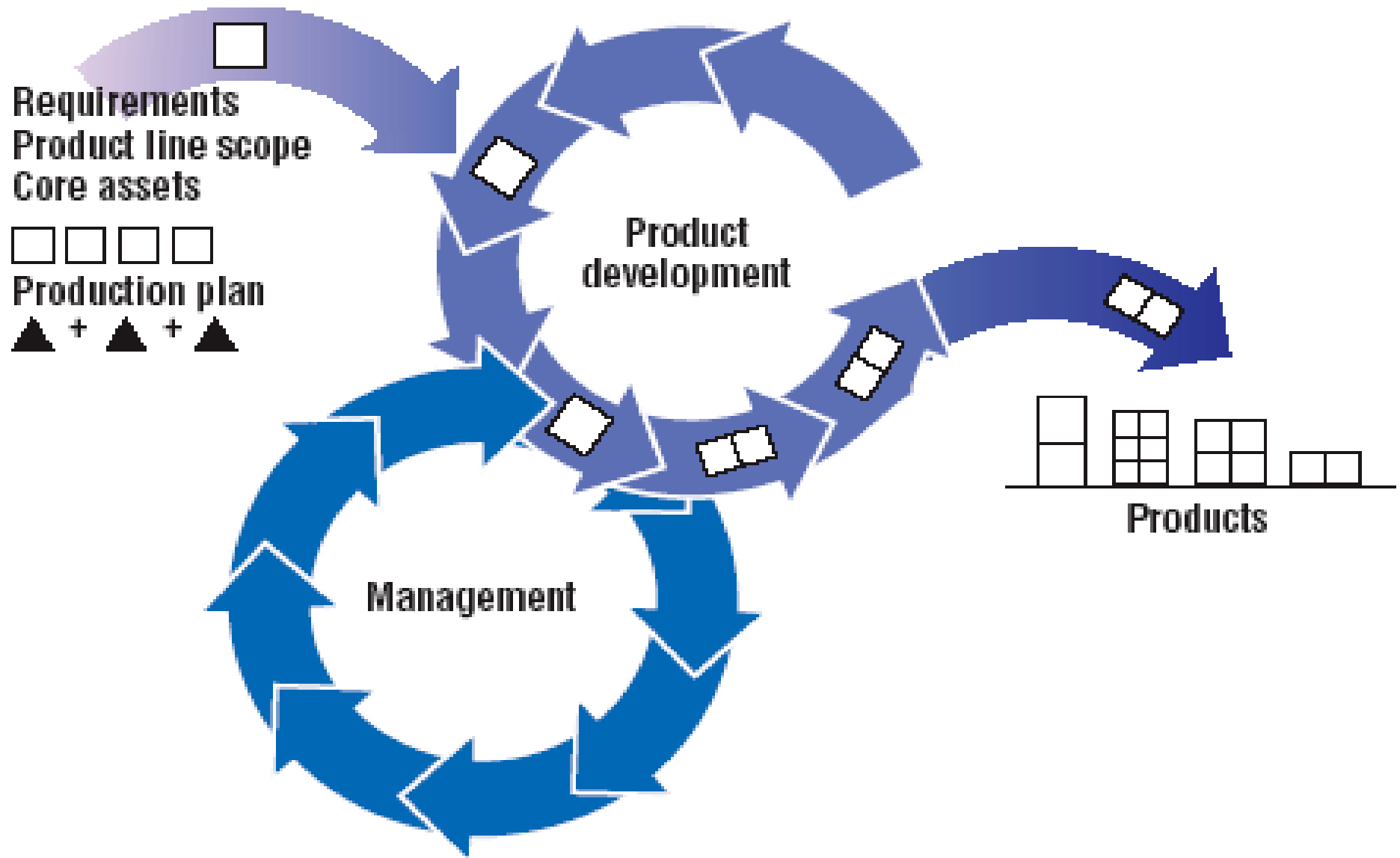
Essential Product line activities



Core assessment development



Product development



Management

- Technical management: core assets development and product development activities
- Organizational management: a funding model that ensures core asset evolution & orchestrates the technical activities and iterations between core asset development and product development.
- Important! **PRODUCT LINE MANAGER**

Example

- A company is producing a range of cell phones:
 - Low-end models support basic phone functionality such as placing and receiving calls,
 - Mid-range models support all the basic features plus additional ones such as call waiting,
 - High-end models include still more features such as email support, games, etc.

Example

- Traditional approach to development:
 - Every phone system developed in isolation - several separate systems,
 - Separate development teams,
 - No team communication and little code reuse,
 - Separate documentation,
 - Separate analysis,
 - Different hardware,
 - Different development tools, etc.

Example

- Traditional approach results in:
 - No common product vision,
 - Reinvention and redevelopment of both hardware and software,
 - Experience not shared among different teams,
 - Slow response to changing market demands,
 - Slow penetration into new markets,
 - Maintenance difficulties, etc.

In General

- Software is becoming a commodity – people want it customized to their particular needs.
- Also, companies want to improve ROI by reusing different assets.
- This led to the product line approach as a form of large scale reuse.

Product Line

- A product line is a set of products sharing common infrastructure.
- This infrastructure is a set of assets which encapsulate commonalities that can be reused among different existing and future products.
- Therefore, ideally, developing a new product using product line approach consists out of reusing a large portion of existing assets and the development of variable new functionality.

PL Software Engineering Areas

- Architecture development,
- Requirements engineering,
- Business domain analysis,
- Component design and development,
- Mining existing assets,
- Software systems integration, etc.

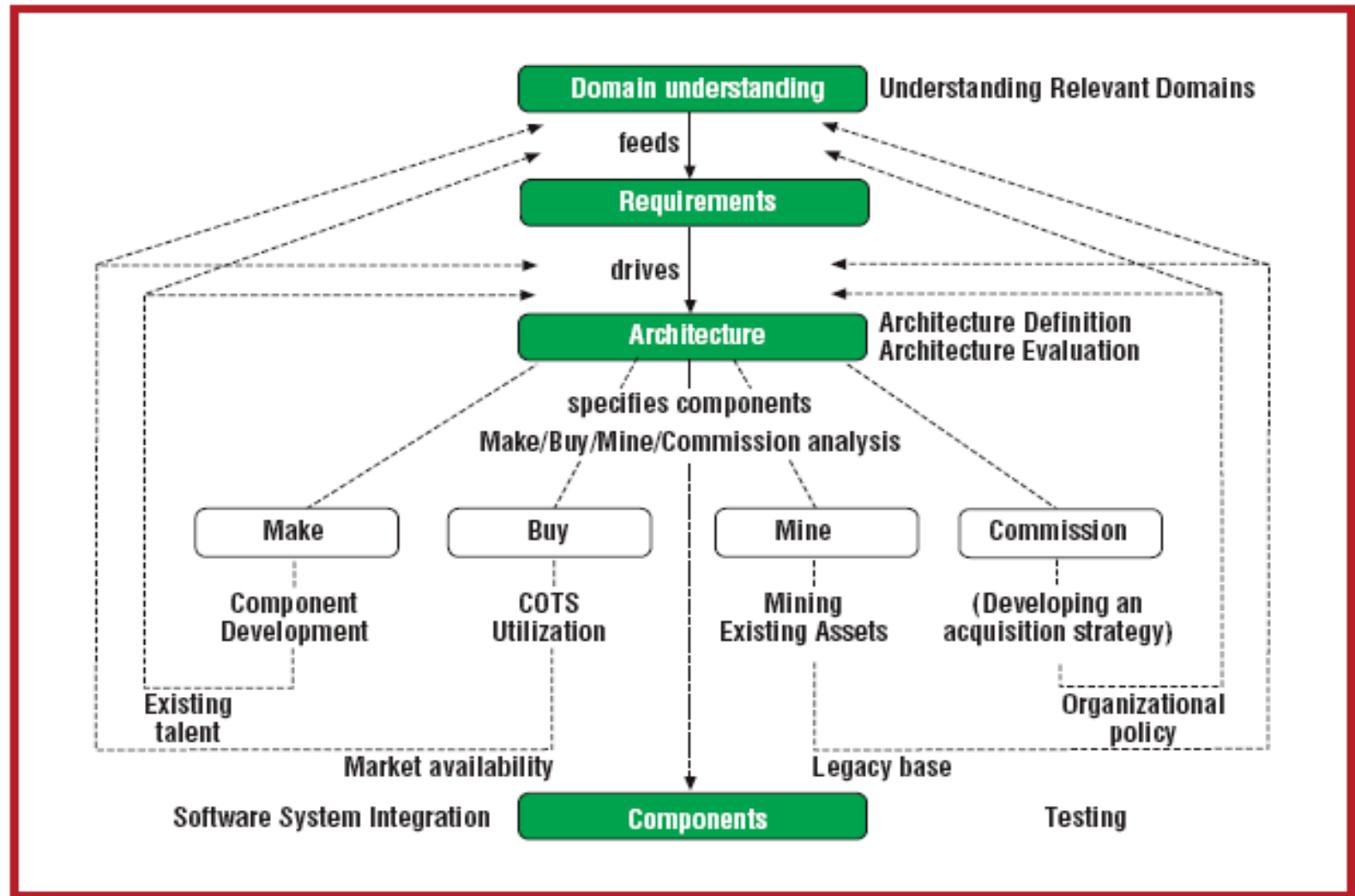
PL vs. Low-Level Reuse

- Traditional forms of reuse: frameworks, libraries, components, etc.
- Intended for a wide variety of products and code based.
- Product lines introduce reuse of other artifacts but code.
- Product line assets are intended for a close family of products, thus more specialized than the traditional ones.
- Main distinction is in the fact that product line assets are developed with a clear vision of which products will be developed in the future.

PL Adoption Schemes

- There are four main situations for adoption of product lines engineering:
 - Company starts a completely new product line,
 - Company restructures already existing products and forms a product line (low amount of reengineering),
 - Company performs major reengineering activity, and major modification of legacy systems to form a product line, and
 - Company sets up a new product line based on an already existing one.
- There can be also situations that combine several of previously mentioned situations.

Figure 4.
Relationship among
software engineering
practice areas.



Mining for Product Lines

- Reuse existing components / apps?
- Hardly possible without modification
- Variability does not come for free
- Consequence: Reengineering required

Software Re-engineering

- Reorganising and modifying existing software systems to make them more maintainable
- "the examination of a subject system to reconstitute it in a new form and the subsequent implementation of the new form."

[ElliotChikofsky and JamesCross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software 7(1):13-17, 1990.]

What is Reengineering?

- Reengineering changes a “bad” design into a good design
- Change an existing design instead of throwing it out
- Often difficult, since changes often affect the whole system
- *Reengineering patterns*: solutions to recurring reengineering problems

Why use Reengineering Patterns?

- New technology
- Unexpected change in requirements
- Repair fault in original design
- Software Integration

A Reengineering Pattern

- Transforms inheritance into composition
- Why?
 - Overuse of inheritance in legacy OO systems
 - Loosen coupling to increase flexibility
 - Allows components to be replaced without affecting all the subclasses

A Reengineering Pattern

- When should it be applied?
 - If you want more flexibility (Bridge pattern)
 - If you want more extensibility (Strategy pattern)
 - If you want to eliminate a large number of conditional statements (State pattern)
- Should not be applied if efficiency is the issue

Software Reengineering Process Model (1)

■ Inventory analysis

- sorting active software applications by business criticality, longevity, current maintainability, and other local criteria
- helps to identify reengineering candidates

■ Document restructuring options

- live with weak documentation
- update poor documents if they are used
- fully rewrite the documentation for critical systems focusing on the "essential minimum"

Software Reengineering Process Model (2)

■ Reverse engineering

- process of design recovery
- analyzing a program in an effort to create a representation of the program at some abstraction level higher than source code

■ Code restructuring

- source code is analyzed and violations of structured programming practices are noted and repaired
- revised code needs to be reviewed and tested

Software Reengineering Process Model (3)

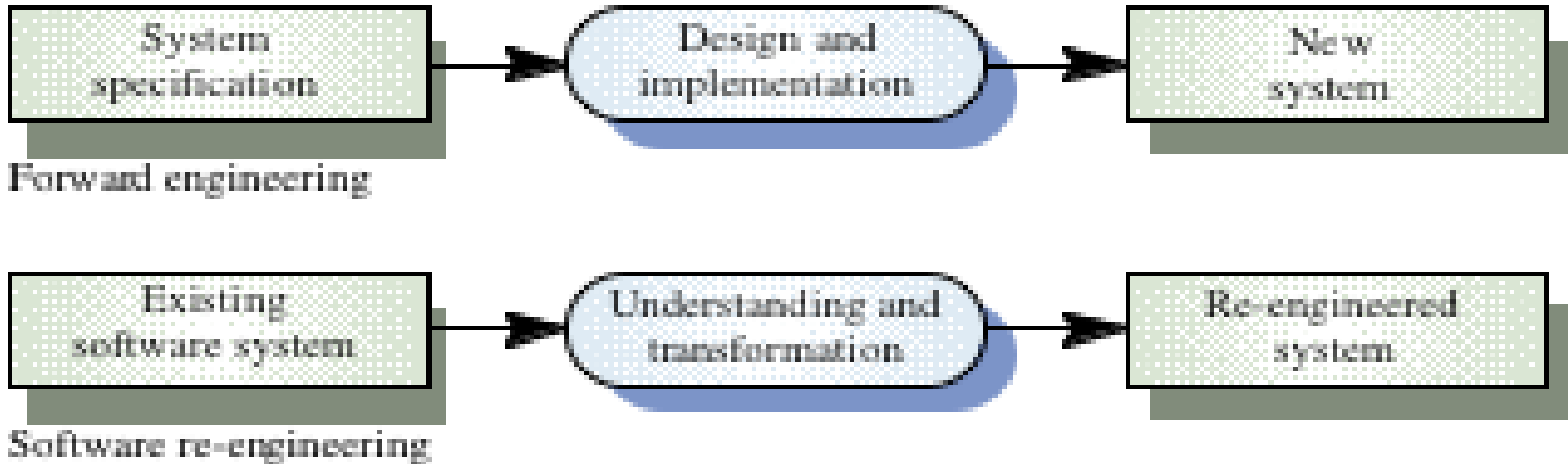
■ Data restructuring

- usually requires full reverse engineering
- current data architecture is dissected
- data models are defined
- existing data structures are reviewed for quality

■ Forward engineering

- sometimes called reclamation or renovation
- recovers design information from existing source code
- uses this design information to reconstitute the existing system to improve its overall quality or performance

Forward Engineering and Reengineering



Reverse Engineering

- Analyzing software with a view to understanding its design and specification
- May be part of the reengineering process
- May be used to specify a system for prior to reimplementation
- Program understanding tools may be useful (browsers, cross-reference generators, etc.)

Reverse Engineering Concepts (1)

■ Abstraction level

- ideally want to be able to derive design information at the highest level possible

■ Completeness

- level of detail provided at a given abstraction level

■ Interactivity

- degree to which humans are integrated with automated reverse engineering tools

Reverse Engineering Concepts (2)

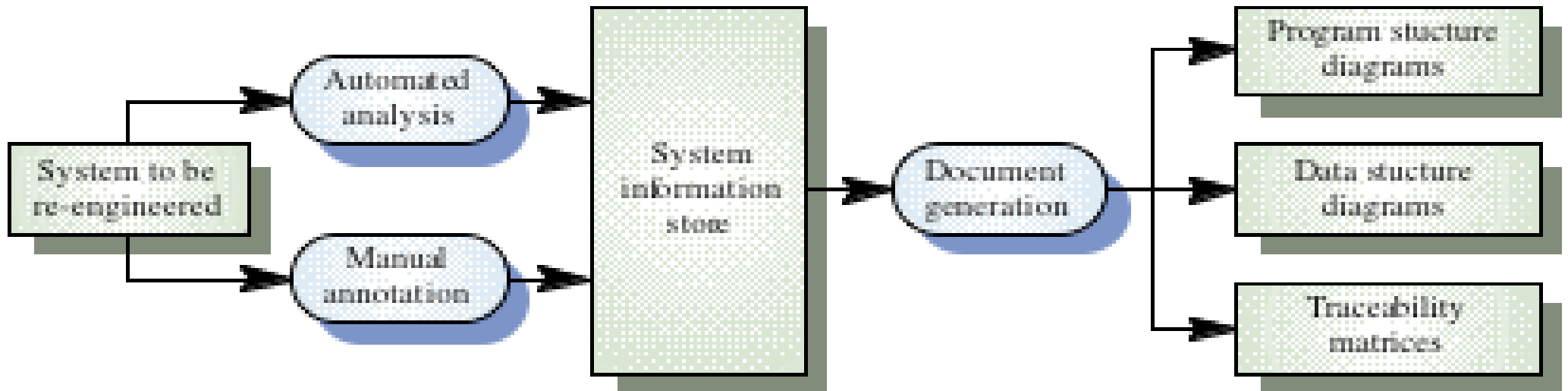
■ Directionality

- one-way means the software engineer doing the maintenance activity is given all information extracted from source code
- two-way means the information is fed to a reengineering tool that attempts to regenerate the old program

■ Extract abstractions

- meaningful specification of processing performed is derived from old source code

Reverse Engineering Process



Reverse Engineering Activities (1)

■ Understanding process

- source code is analyzed to at varying levels of detail
 - | system
 - | program
 - | component
 - | pattern
 - | statement
- to understand procedural abstractions and overall functionality

Reverse Engineering Activities (2)

- Understanding data

- internal data structures
- database structure

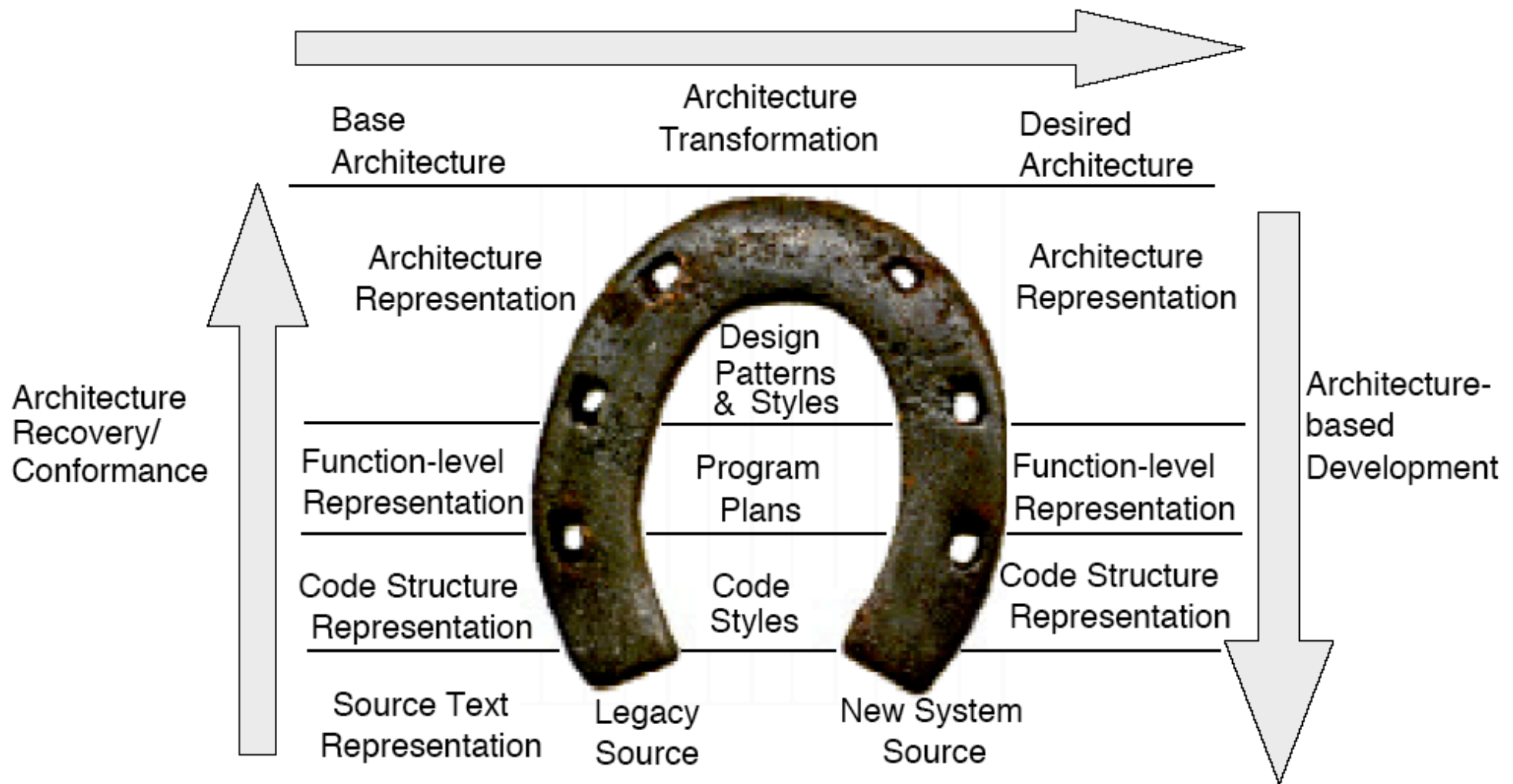
- User interfaces

- what are the basic actions (e.g. key strokes or mouse operations) processed by the interface?
- what is a compact description of the system's behavioral response to these actions?
- what concept of equivalence of interfaces is relevant?

Reverse Engineering Applicability

- Reverse engineering often precedes reengineering
- Sometimes reverse engineering is preferred
 - if the specification and design of a system needs to be defined prior using them as input to the requirements specification process for a replacement systems
 - if the design and specification for a system is needed to support program maintenance activities

Reengineering Vorgehensmodell: Horseshoe Model



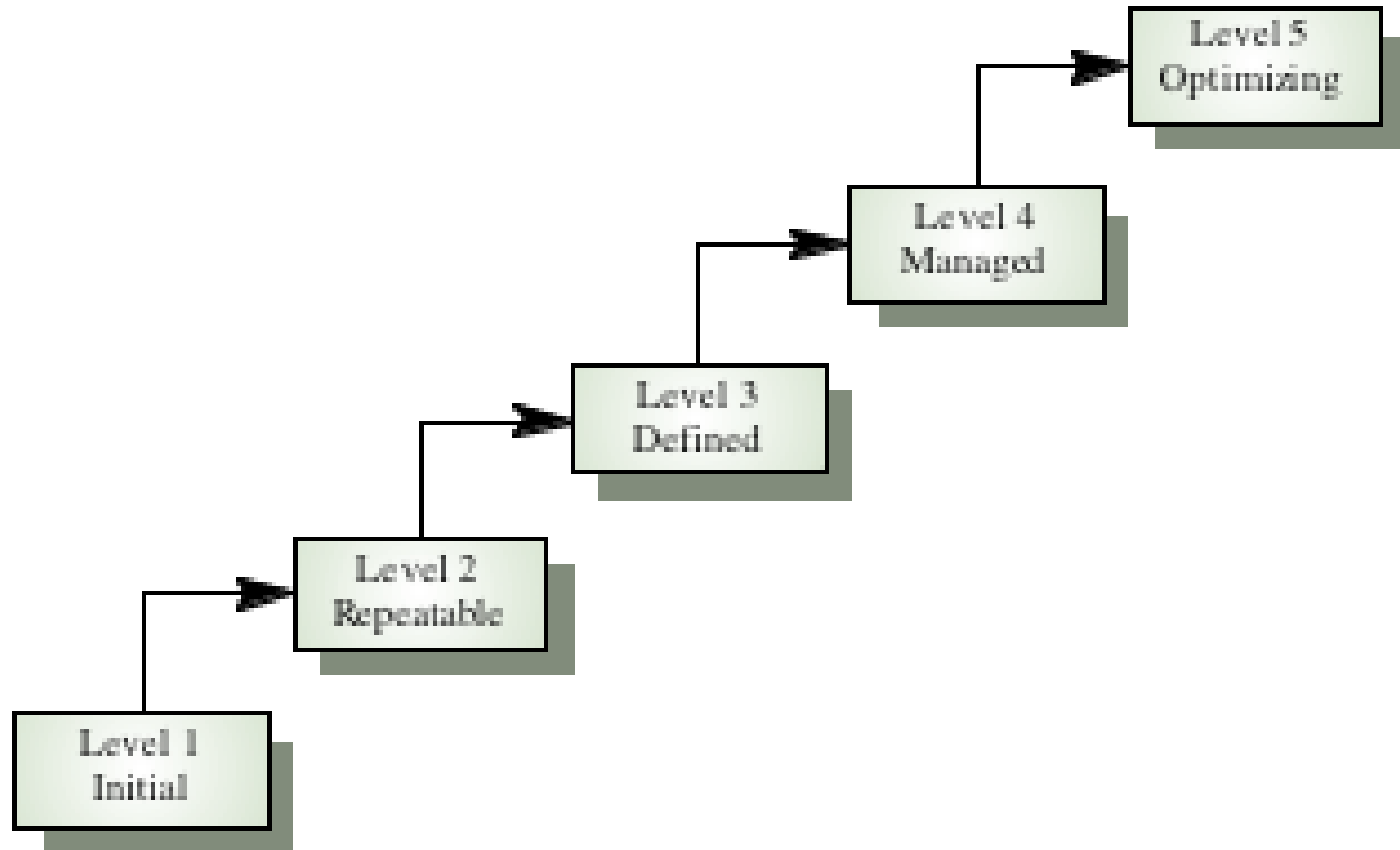
Über das Bewußtwerden und Explizitmachen

- Phasen der Software-Entwicklung
- Organisationsprinzipien
- Projektmanagement
- Vorgehensmodelle
- ...

-> Software Engineering ist mehr als Programmentwicklung

- It's all about processes ...
... and people.

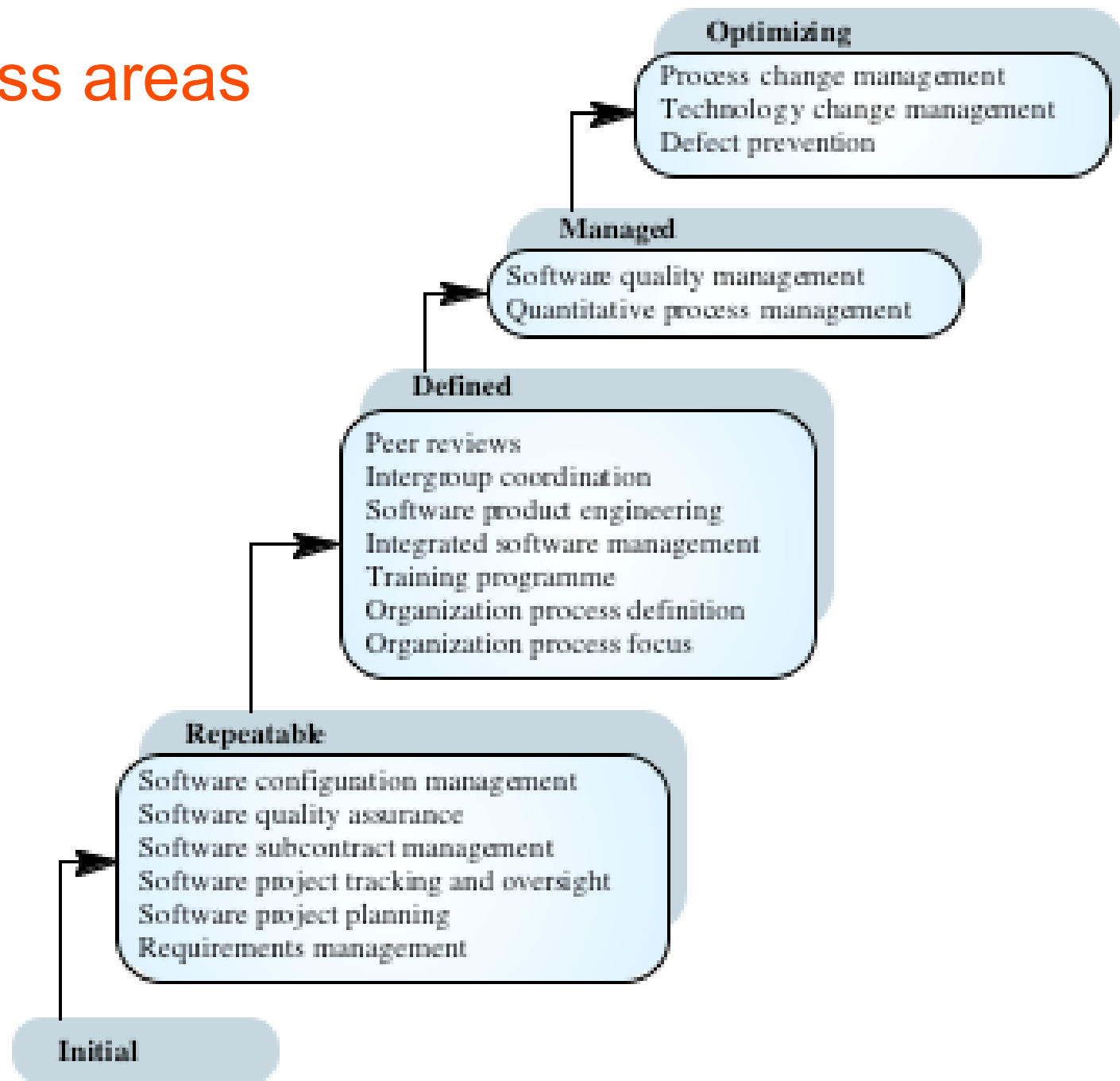
Processes: The SEI process maturity model



Maturity model levels

- Initial
 - Essentially uncontrolled
- Repeatable
 - Product management procedures defined and used
- Defined
 - Process management procedures and strategies defined and used
- Managed
 - Quality management strategies defined and used
- Optimising
 - Process improvement strategies defined and used

Key process areas



SEI model problems

- It focuses on project management rather than product development.
- It ignores the use of technologies such as rapid prototyping.
- It does not incorporate risk analysis as a key process area
- It does not define its domain of applicability

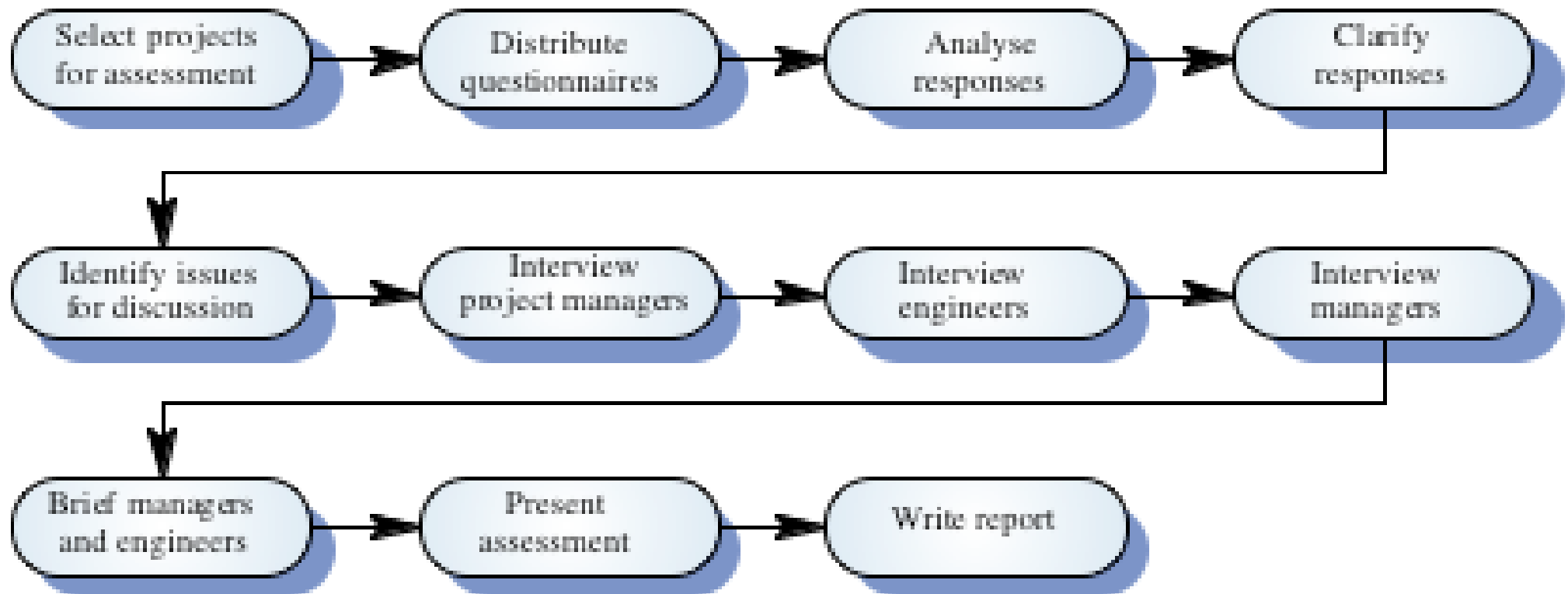
The CMM and ISO 9000

- There is a clear correlation between the key processes in the CMM and the quality management processes in ISO 9000
- The CMM is more detailed and prescriptive and includes a framework for improvement
- Organisations rated as level 2 in the CMM are likely to be ISO 9000 compliant

Capability assessment

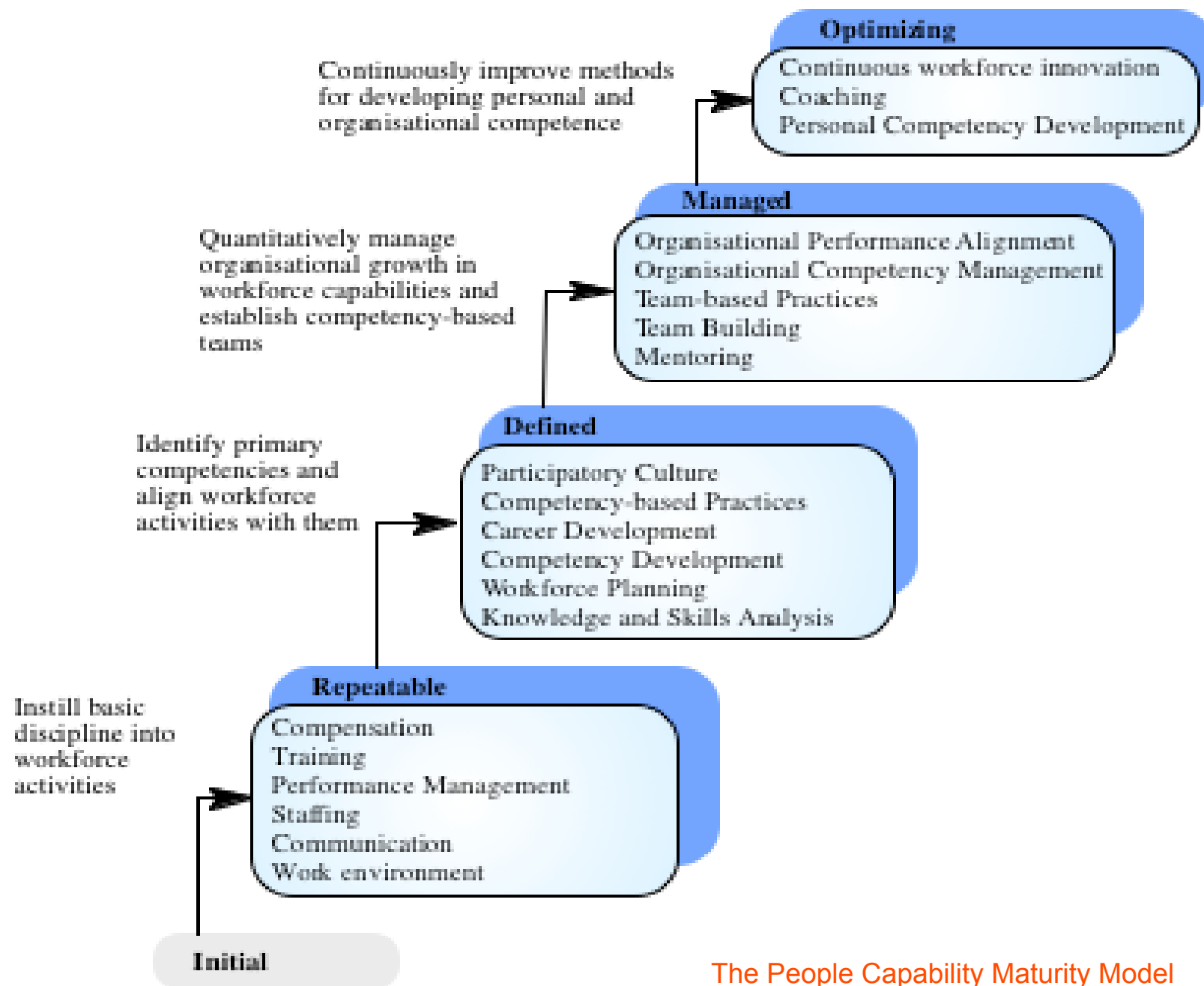
- An important role of the SEI is to use the CMM to assess the capabilities of contractors bidding for US government defence contracts
- The model is intended to represent organisational capability not the practices used in particular projects
- Within the same organisation, there are often wide variations in processes used
- Capability assessment is questionnaire-based

The capability assessment process



The People Capability Maturity Model

- Intended as a framework for managing the development of people involved in software development
- Five stage model
 - Initial. Ad-hoc people management
 - Repeatable. Policies developed for capability improvement
 - Defined. Standardised people management across the organisation
 - Managed. Quantitative goals for people management in place
 - Optimizing. Continuous focus on improving individual competence and workforce motivation



The People Capability Maturity Model

P-CMM Objectives

- To improve organisational capability by improving workforce capability
- To ensure that software development capability is not reliant on a small number of individuals
- To align the motivation of individuals with that of the organisation
- To help retain people with critical knowledge and skills

Software-Engineering 2004

- Das war's.
- Viel Erfolg bei den Prüfungen...