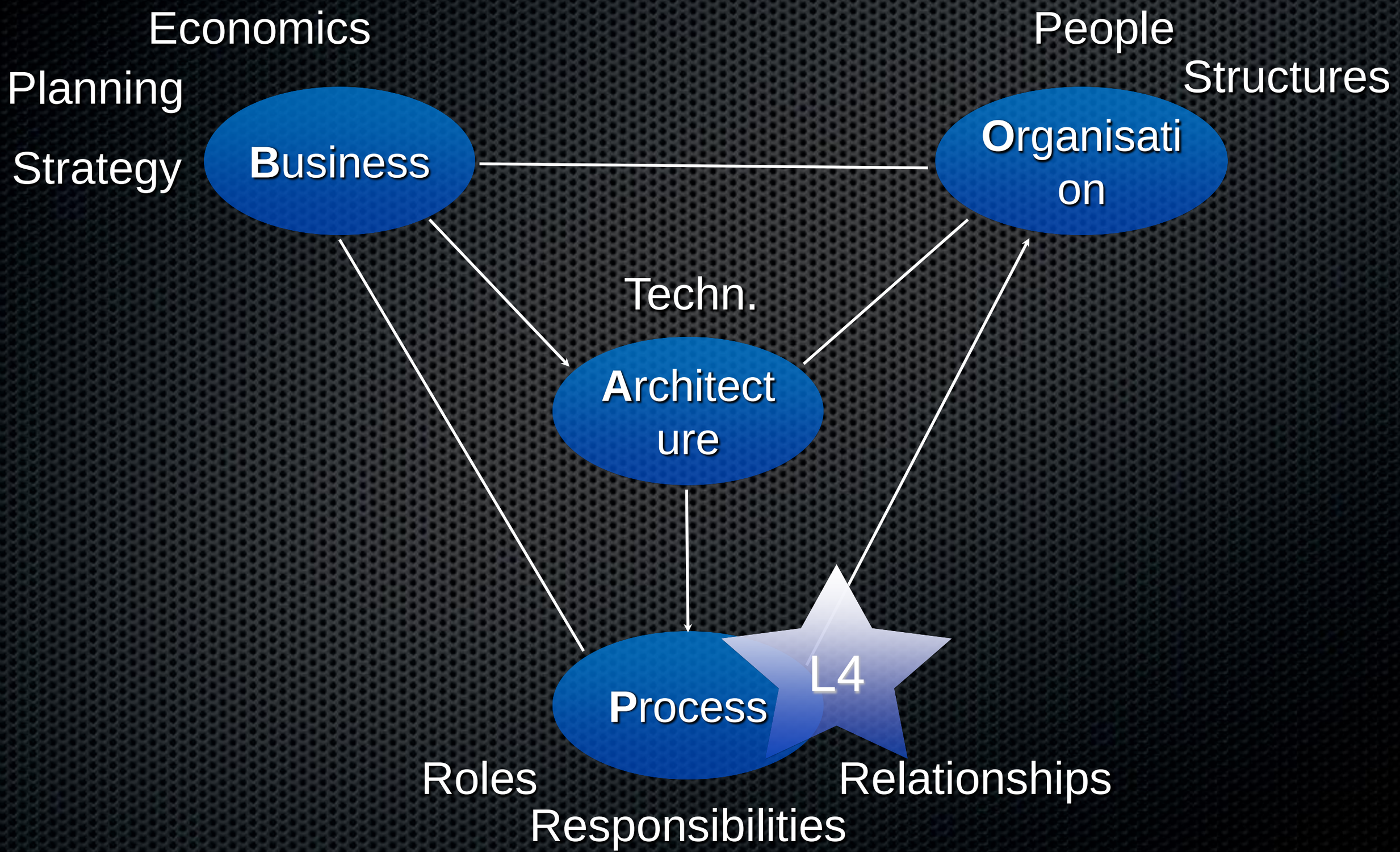
Software Product Line Engineering

L4:Processes and SPL

L4:Processes and SPL



Processes

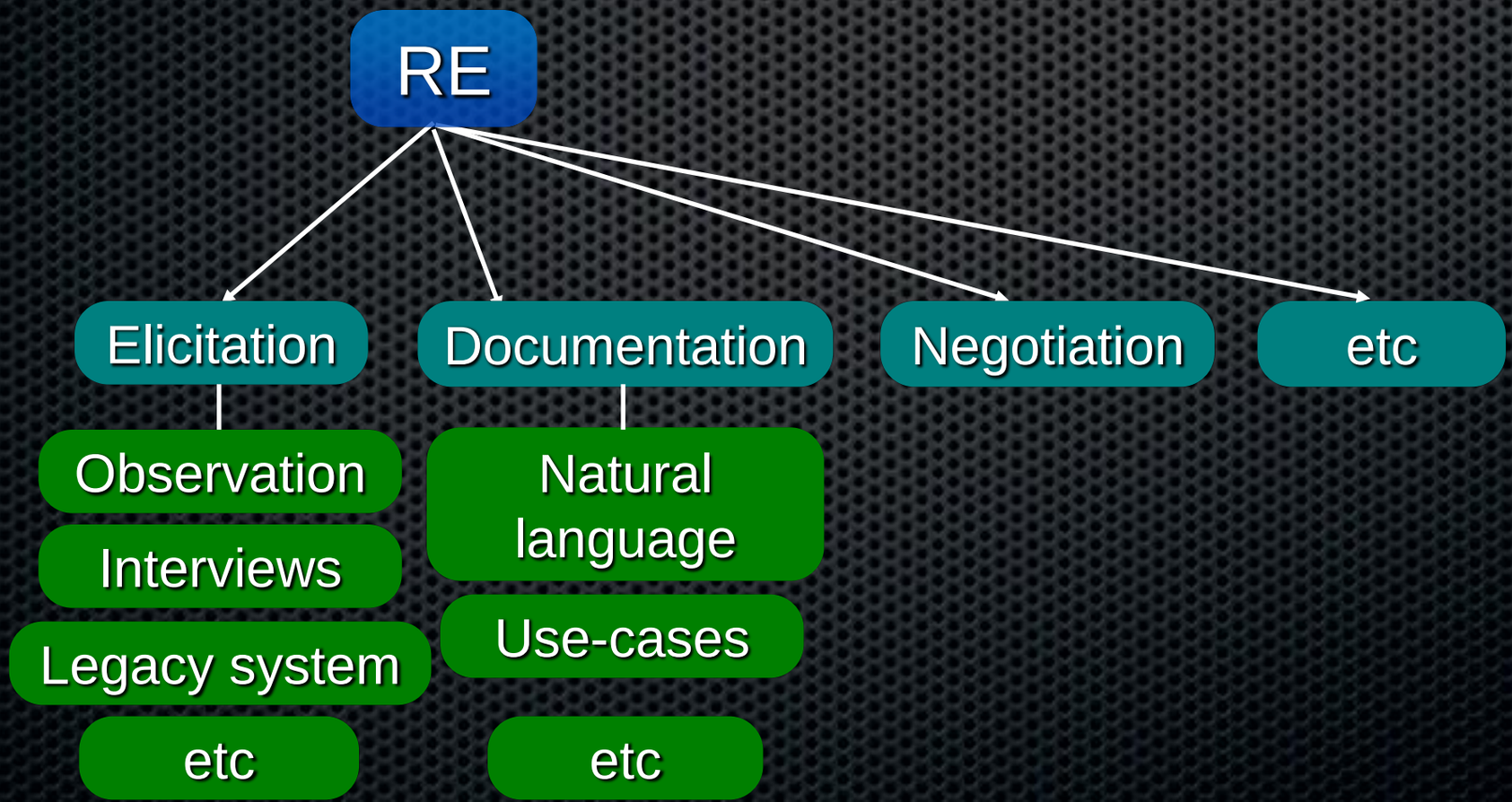
- Software Engineering Process: the total set of software engineering activities needed to transform requirements into software
- Product Development Process: the total set of engineering activities needed to transform requirements into products
  - Software (product) engineering refers to the disciplined application of engineering, scientific, and mathematical principles and methods to the economical production of quality software (products).

Process examples

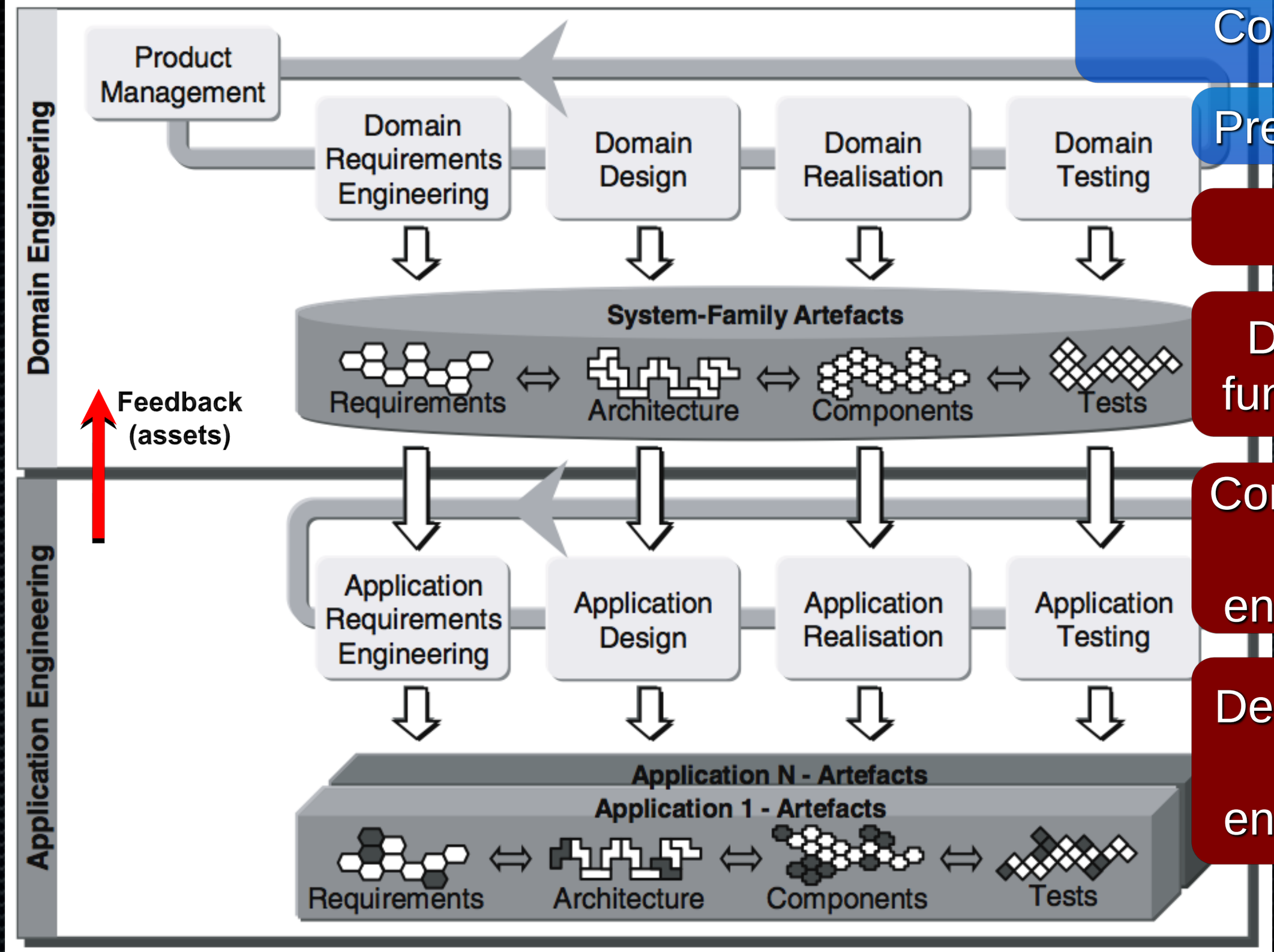
- Requirements Engineering (Main Process Area)
  - Elicitation (Sub-process Area)
    - Task observation (Activity/Action)
- Configuration Management
  - Configuration Item Identification
    - Risk analysis
    - Volatility (change Prone) analysis

Process examples

- Requirements Engineering (Main Process Area)
  - Elicitation (Sub-process Area)
    - Task observation (Activity/Action)
- Configuration Management (MPA)
  - Configuration Item Identification (SPA)
    - Risk analysis (Action), Change Prone analysis (Action)



SPL Process



Coordination and Control

Predictability

Quality

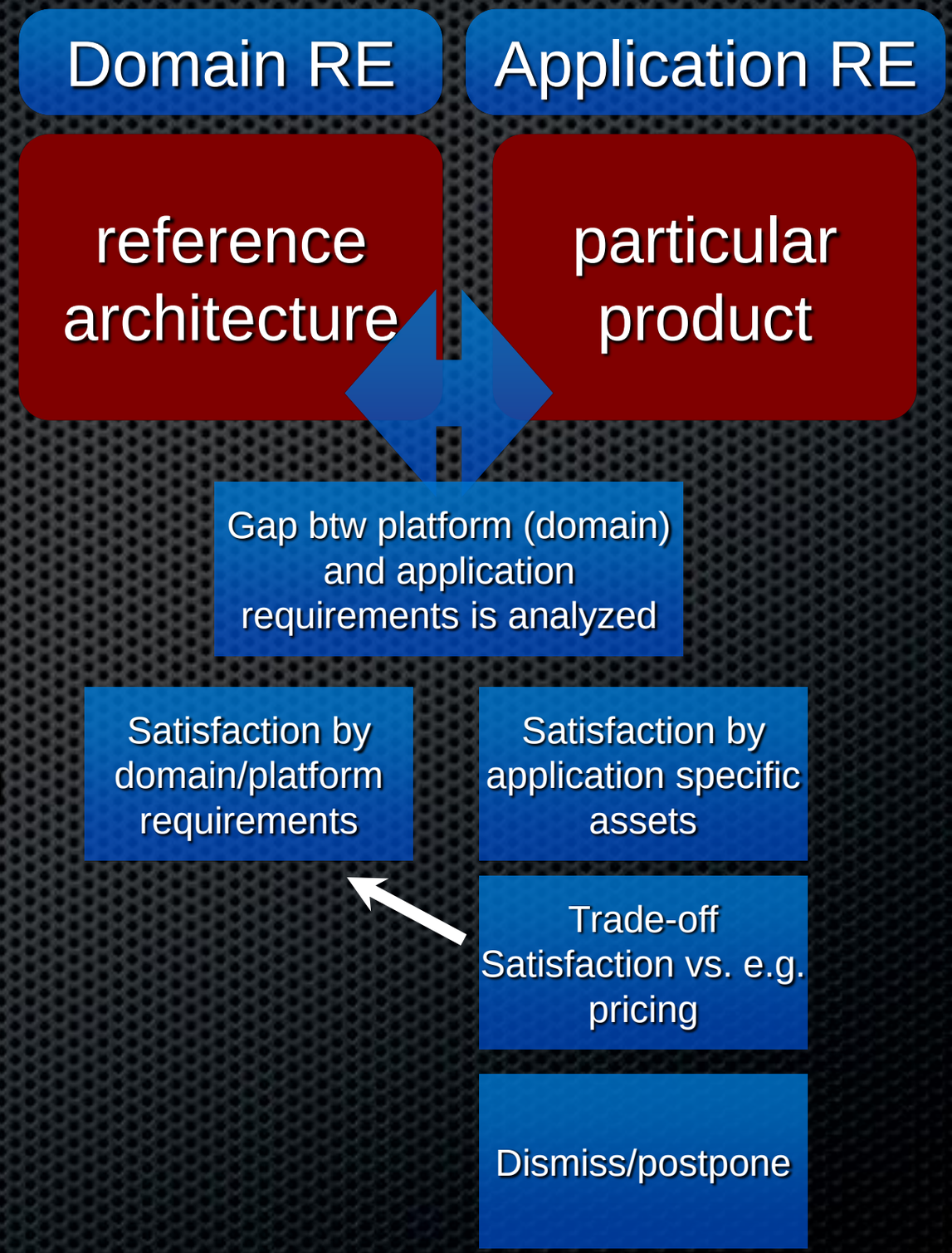
Delivered functionality

Commonality of engineering

Dependency heavy engineering

Requirements Engineering (RE)

- Elicitation
- Documentation
- Analysis and Negotiation
- Validation and Verification
- Management



Elicitation

- **Domain** (Understanding it)
- **Problem (application) domain**
  What's the problem(s) and who can explain it to you
- **History**
  Previous systems / current systems
  Documentation
  Old requirements/design etc.
- **Competitors**
  Have they solved the problem and how?
- **Surrounding environment**
  Other systems, processes which the system should support (and/or processes which the system influences)

Domain | Application

- internal (development org.) stakeholders (e.g. PM, developers, architects, support, STRATEGIES)
- external (customer, domain, environmental, regulatory)

need vs. want
stakeholder weights (politics) and access

- **Stakeholders**
  (management, users, future users, system managers, partners, sub contractors, Law and Policy, customer's customers, domain experts, developers etc)
- Finding them (Stakeholder Identification)
- Getting access to them (Cost, Politics)

PREPARATION

Elicitation techniques

- Interviews
  + Getting to know the present (domain, problems) and ideas for future system
  - Hard to see the goals and critical issues, subjective

- Group interviews
  + Stimulate each other, complete each other
  - Censorship, domination (some people may not get attention)

- Observation (Look at how people actually perform a task (or a combination of tasks) – record and review…)
  + Map current work, practices, processes
  - Critical issues seldom captured (e.g. you have to be observing when something goes wrong), usability issues seldom captured, time consuming

- Task demonstrations (Ask a user to perform a task and observe and study what is done, ask questions during)
  + Clarify what is done and how, current work
  - Your presence and questions may influence the user, critical issues seldom captured, usability problems hard to capture

Elicitation techniques 2

- Questionnaires
  + Gather information from many users (statistical indications, views, opinions)
  - Difficult to construct good questionnaires, questions often interpreted differently, hard to classify answers in open questions and closed questions may be to narrow…

- Use cases and Scenarios (Description of a particular interaction between the (proposed) system and one or more users (or other terminators, e.g. another system). A user is walked through the selected operations and the way in which they would like to interact with the system is recorded)
  + Concentration on the specific (rather than the general) which can give greater accuracy
  - Solution oriented (rather than problem oriented), can result in a premature design of the interface between the problem domain and the solution

- Prototyping
  + Visualization, stimulate ideas, usability centered, (can be combined with e.g. use cases)
  - Solution oriented (premature design), “is it already done?!”

Documentation

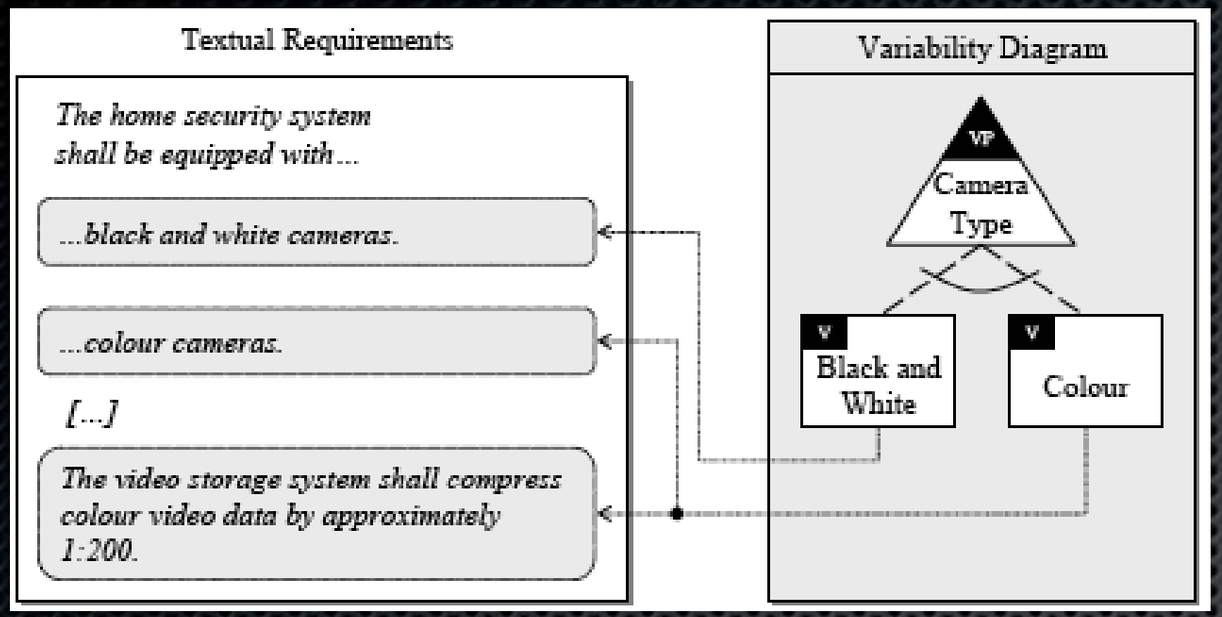
- **Natural Language (NL) Specification**
  (most common in industry)
  + Everyone can do it/understand
  + NL is a powerful notation (if used correctly)
  - Imprecise and Quality may vary
  - Use of attributes can improve accuracy
    ID, Title, Desc, Rationale, Source(s), Conflict, Dependencies, Prio. etc

- **Modeling** (where use-cases most common)
  + Relatively easy to do
  + Structure
  + Reuse of effort (e.g. code generation)
  - Imprecise and Quality may vary
  - Solution oriented, don’t catch non functional aspects (Quality Requirements)
  - Cost/time

Context Diagrams
Event Lists
Screens & Prototypes
Scenarios
Task Descriptions
Standards
Tables & Decision Tables
Textual Process Descriptions
State Diagrams
State Transition Matrices
Activity Diagrams
Class Diagrams
Collaboration Diagrams
Sequence Diagrams

Complete
Correct
Feasible
Necessary
Prioritized
Unambiguous
Verifiable

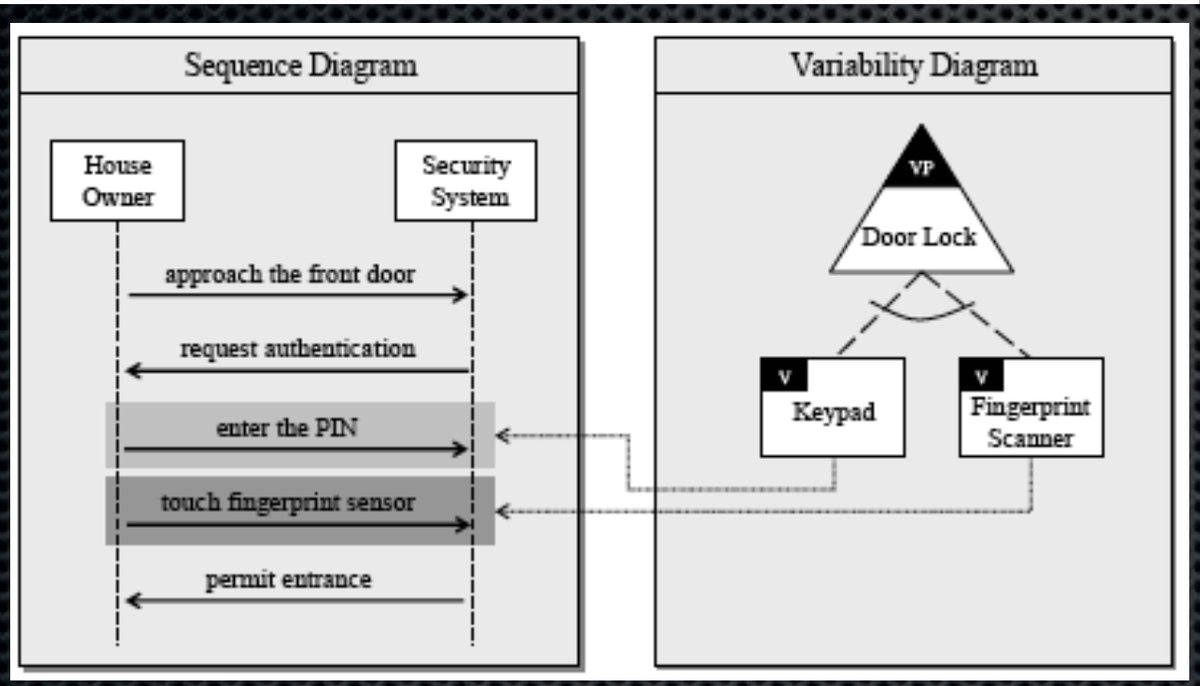
Documentation 2

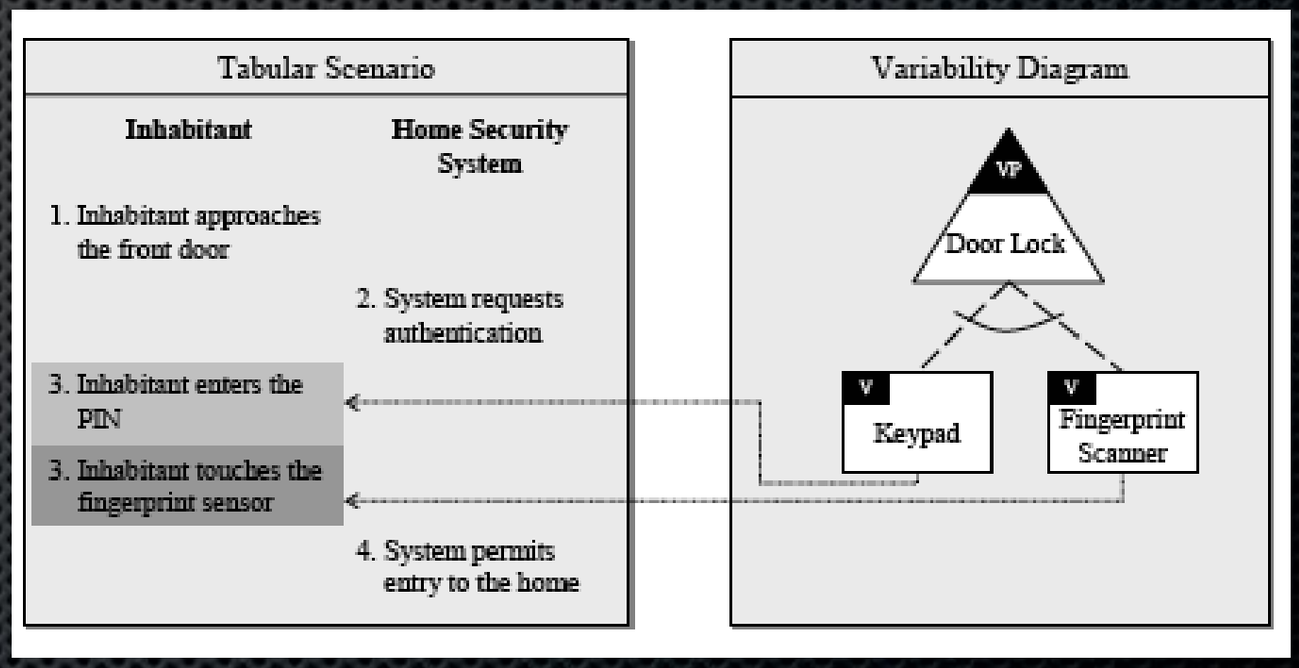


variability has to be mapped to requirements

Decision support: Domain or Application

Influences priority, risk, timeline, cost

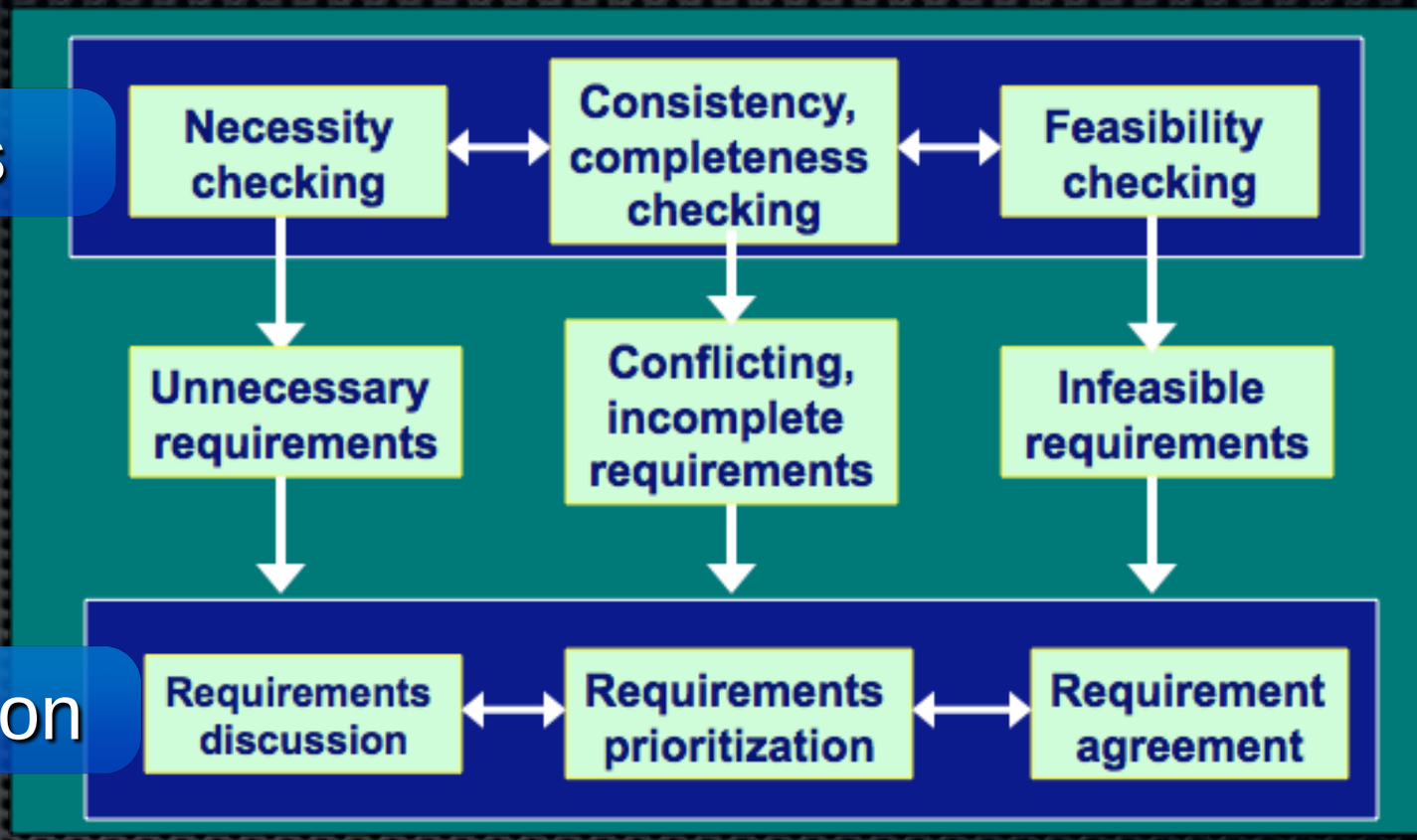




Analysis and Negotiation

Aims to discover problems with requirements and reach agreement that satisfies all stakeholders

Analysis



Negotiation

- Premature design?
- Combined requirements?
- Realistic within Constraints?
- Understandable?
- Conformance with business goals?
- Ambiguous?
- Necessary requirement?

Customer Value

Gold Plating?

- Testable?
- Complete?
- Traceable?
- Consistent Terminology?
- Fit Criteria

Relevant?

Measurable?

- Requirement or Solution?

Techniques

Interaction Matrices
Requirements Classification
Requirements Risk Analysis
Boundary Definition

Verification and Validation (quality assurance)

- Verification is the process of determining that a system, or module, meets its specification
- Validation is the process of determining that a system is appropriate for its purpose

are we building the right system

check if we have elicited and documented the right requirements

Reviews/Inspections

Perspective based reading
Checklist based reading
Test Case Based Inspections
Two Man Inspection

(perspectives and checklist may include product line specific items like variability checks)

Reviews
Inspections
Checklists
Goal-Means Analysis
Req. Classifications
Prototyping
Simulation
Mock-Up
Test-Cases
Draft User Manual

the earlier you find a problem... errors introduced in the RE process are the most resource intensive to fix
(50x more costly to fix defects during test than during the RE)

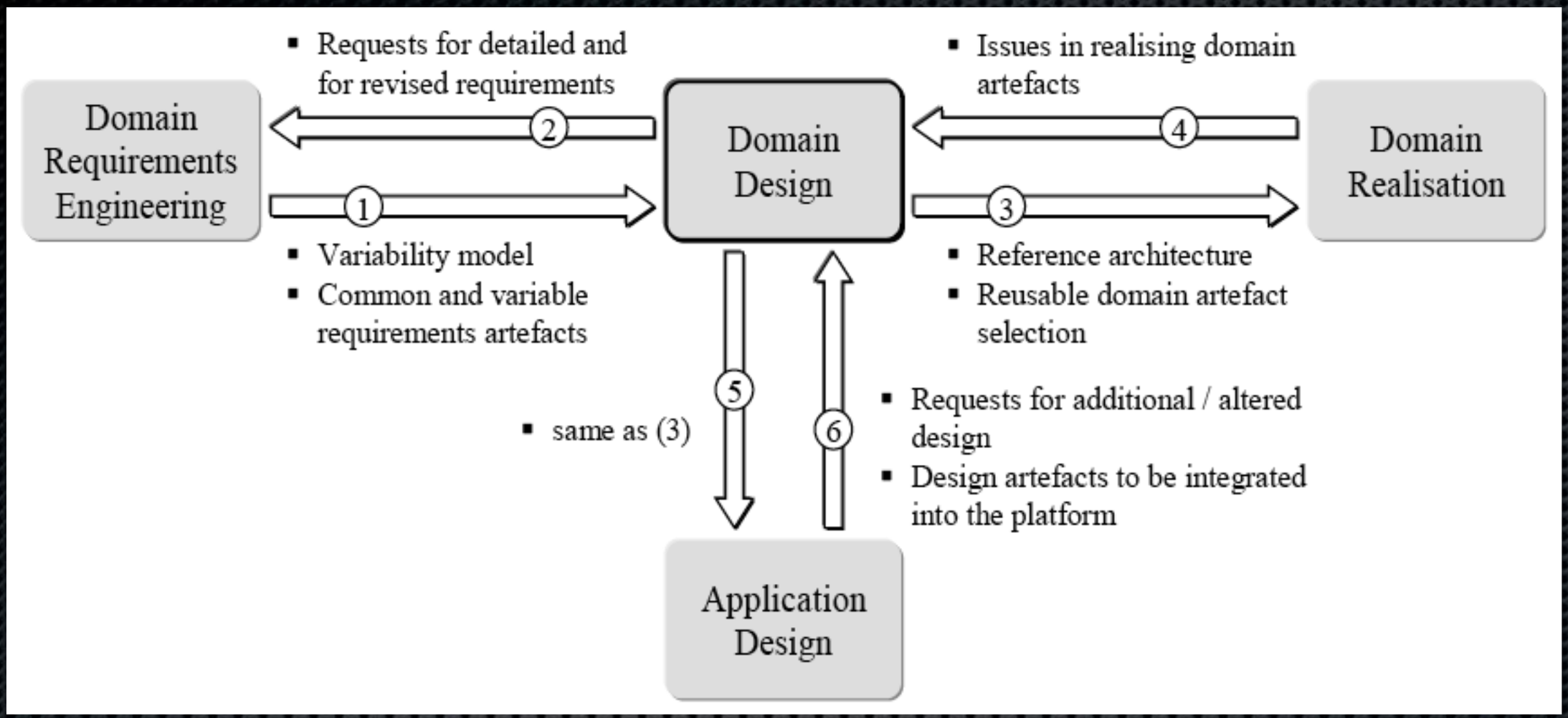
RE Management

- **Definition of the RE process and its interfaces and management of requirements and the requirements process over time**
  - **Configuration Management (!)** what to put under control | change management | version handling
  - **Tool support** tool that supports your process
- **Traceability policies(!)** source, forward, backward (pre-requisite for reuse)

Focal Point, CaliberRM, Serena, Rational Req. Pro

- **Reuse (!)** the artifacts you are creating may be reused = **quality and cost implications**
- **Standards and policies (e.g. documentation)** least common denominator (what is good-enough) for RE you have to see beyond your role/needs
  - **Criteria for when to ignore policies**

Domain Design



- Based on the reference requirements (delivered by PM and RE) create a reference architecture (variability and design covered in different lecture)

Domain Realization

- Make (assets built in-house)

control technical but also from a business perspective - is the asset a competitive (innovative asset)

- Buy (bought off-the-shelf)

often resource intensive assets (e.g. OS, middleware) but also infrastructure like RUP or CMMI

- Mine (reuse)

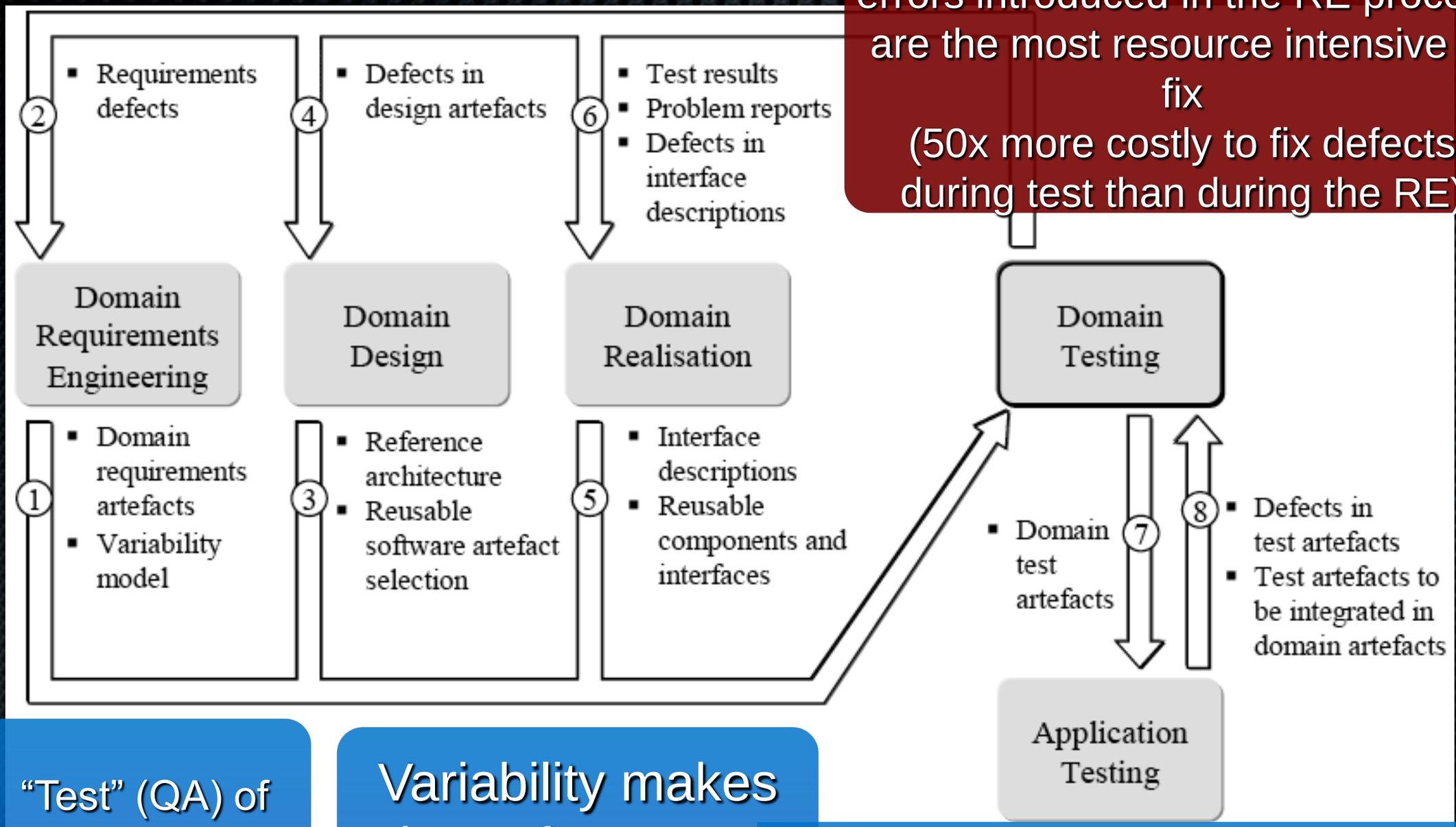
reuse of existing assets (e.g. other products) - often requires a lot of reengineering
BUT application specific assets can be used and turned into a common asset

- Commission (3rd party)

specification in-house as a order to 3rd party (adherence to specification, specification quality, use of e.g. implementation proposals to assure common understanding)

Domain Testing

the earlier you find a problem...
errors introduced in the RE process are the most resource intensive to fix
(50x more costly to fix defects during test than during the RE)



"Test" (QA) of non-executables is !critical!

Variability makes brute force test impossible

Test suitable configurations (selected for best ROI) alt.
Use of e.g. stubs (fill on for absent/future plug-ins)
BUT COST for creating and maintaining tests and e.g. stubs has to be weighed in (not to mention defects in test artifacts themselves)

Testing Strategy

	Time to create	Absent variants	Early validation	Learning effort	Overhead
(BFS)	-	-	+	0	-
(PAS)	0	+	-	+	-
SAS	0	+	+	+	-
CRS	+	+	0	-	+
Combined SAS/CRS	+	+	+	0	0

BFS=Brute Force
PAS=Pure Application Strategy
SAS=Sample Application Strategy
CRS=Commonality and Reuse Strategy