Software Architecture 2021
Lecture 1, part 2/3

Software engineering process and the role of requirements

Dimitri Van Landuyt

Wouter Joosen

KU LEUVEN DistriNet

Outline

1. **Software engineering processes**

Initial phases of the software development life cycle

2. **Requirements engineering**

First acquaintance with ASRs – architecturally-significant requirements

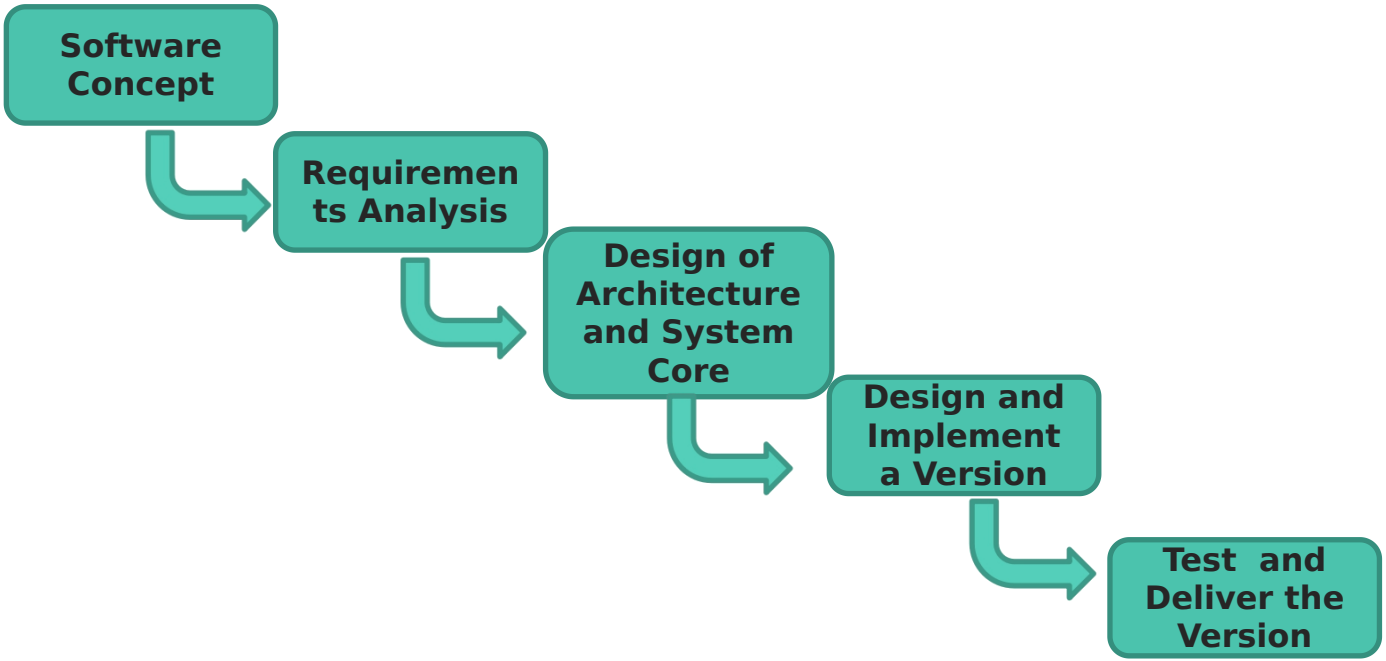
(1) Software Engineering Processes

How to get from *A* to *Z*?

Software Concept

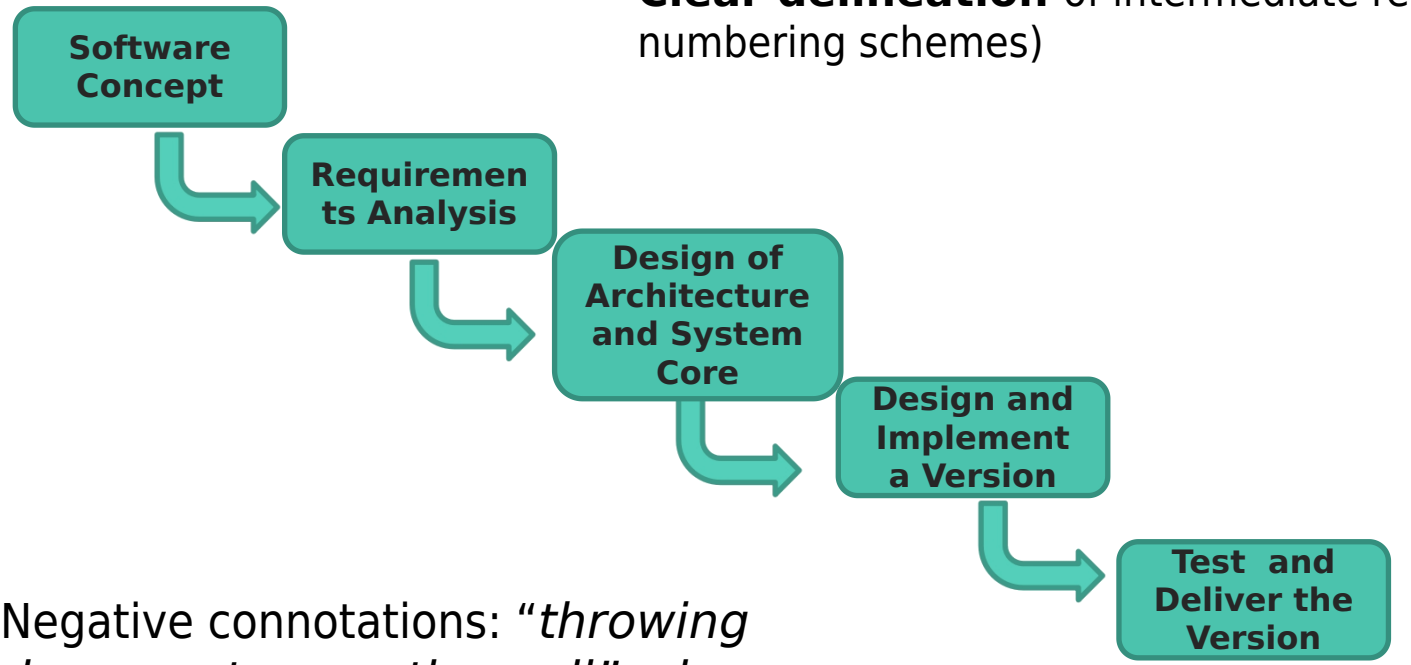
Delivery

Waterfall Model for Software Development



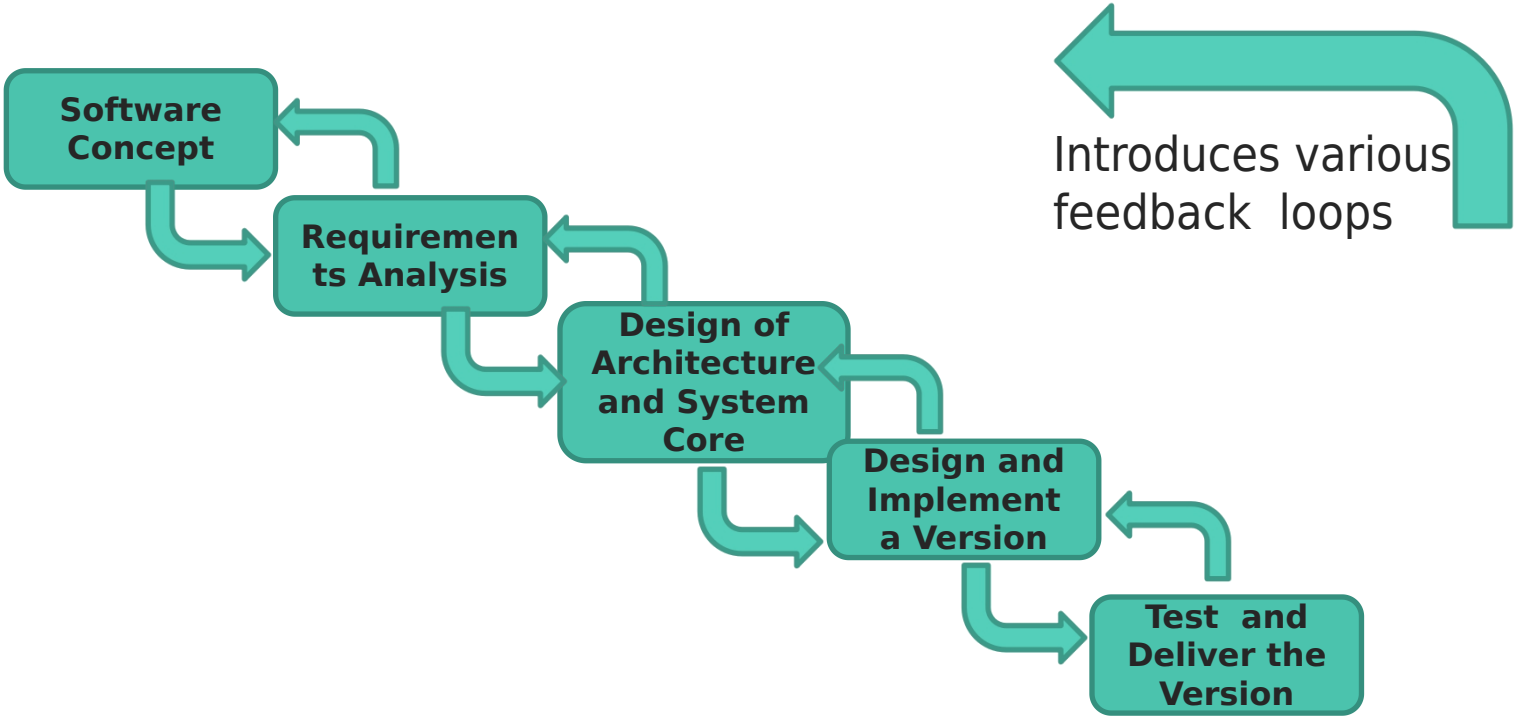
Waterfall Model for Software Development

- Reliance on complete and standalone **specifications**
- **Clear delineation** of intermediate results (e.g. version numbering schemes)

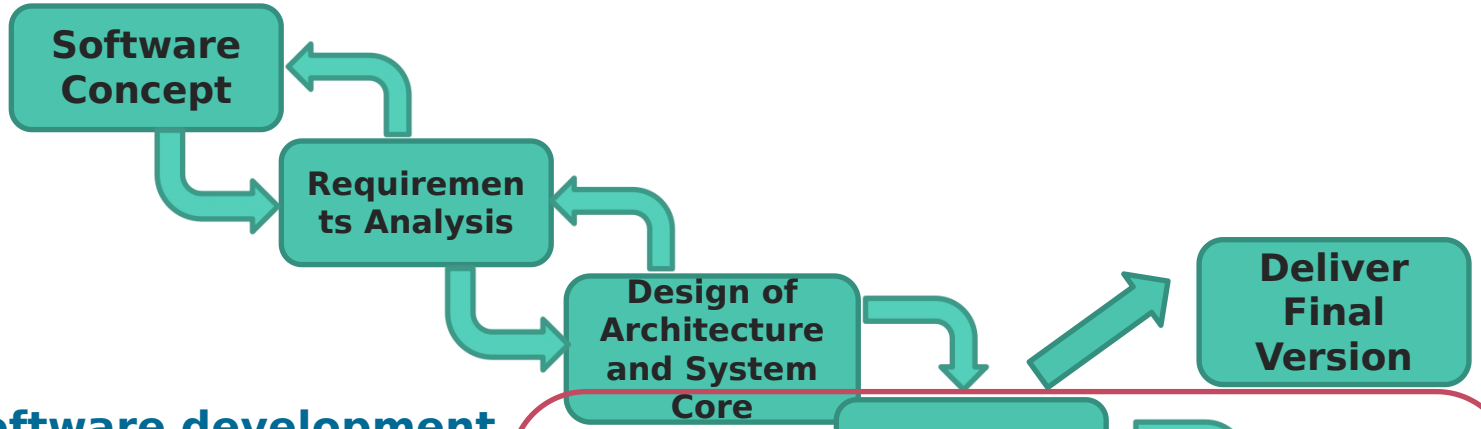


Negative connotations: “*throwing documents over the wall”; slow, heavyweight*

Iterative Software Development (at the basis of the Unified Process)



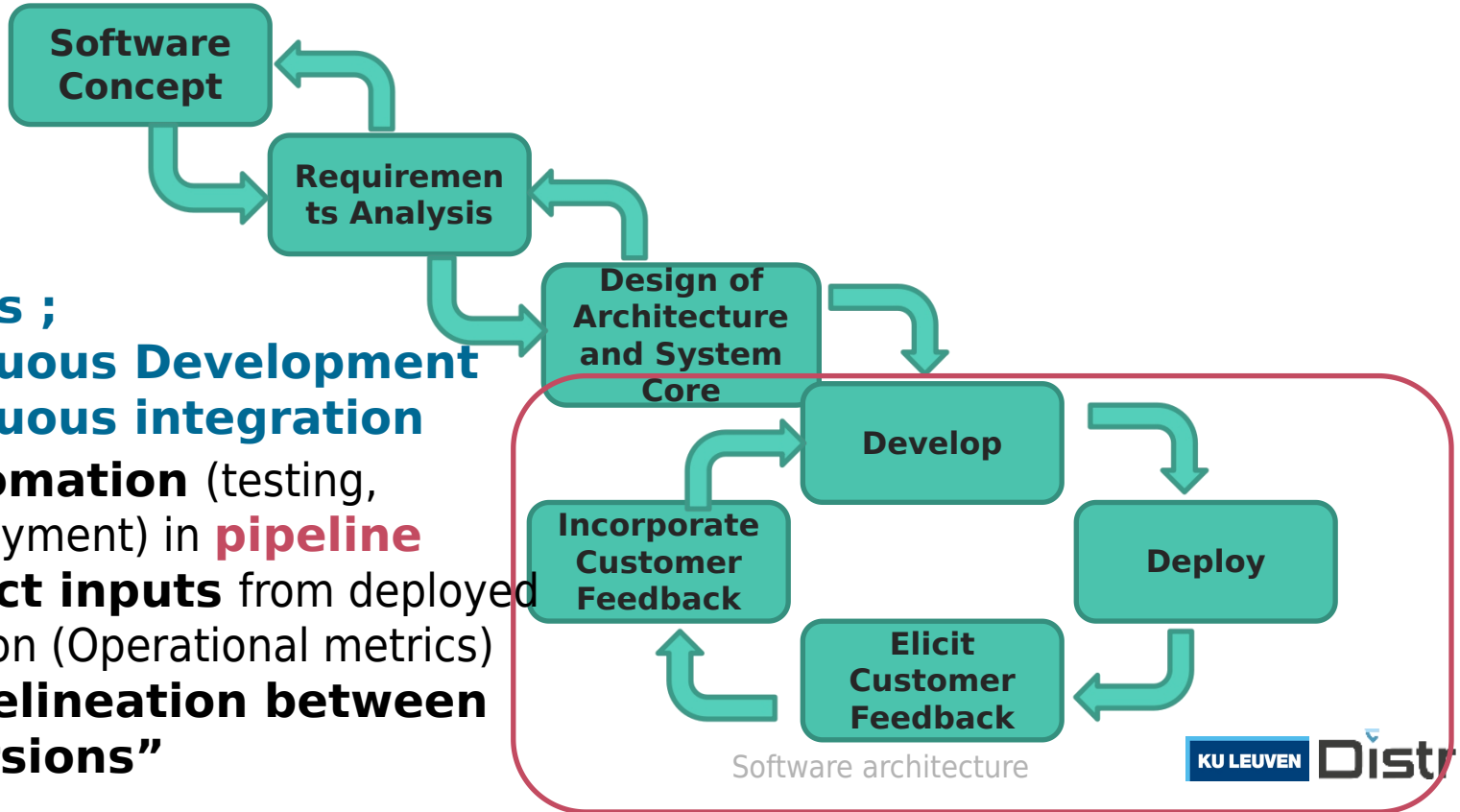
Other development processes



Agile software development

- **Increase iteration frequency** (sprints)
- More degrees of freedom
- More **direct feedback** from customers

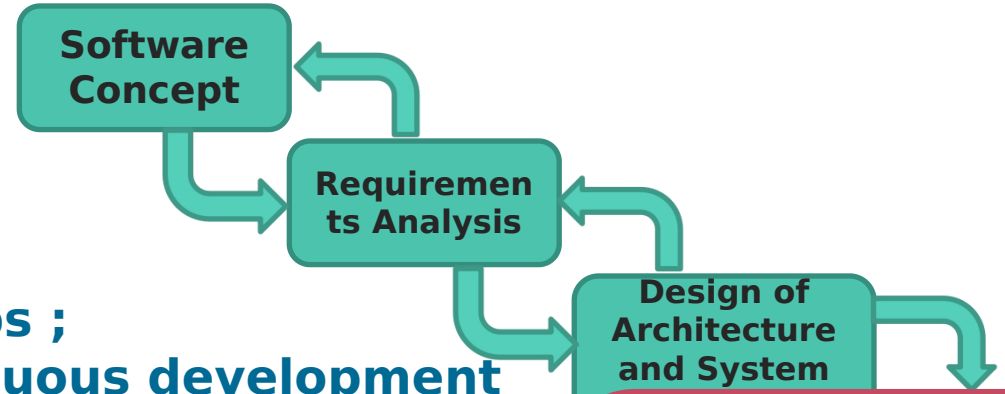
Other development processes



DevOps ;
Continuous Development
Continuous integration

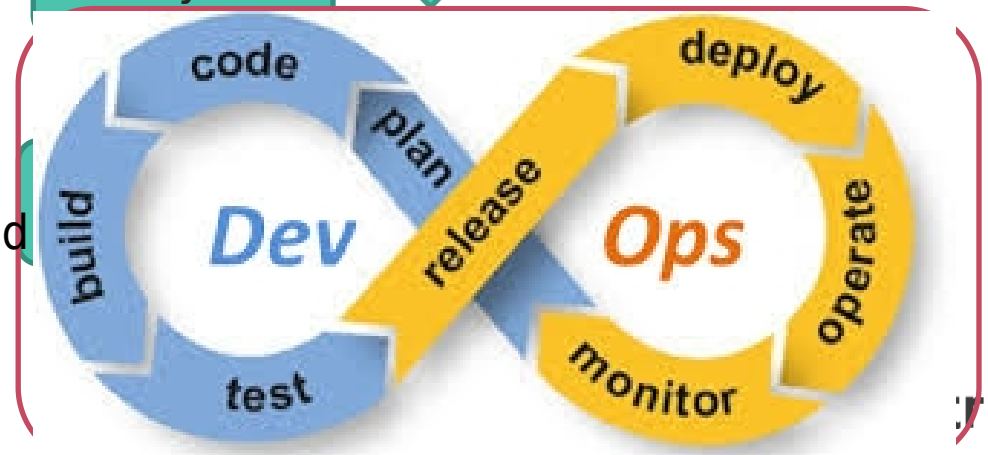
- **Automation** (testing, deployment) in **pipeline**
- **Direct inputs** from deployed version (Operational metrics)
- No **delineation between “versions”**

Other development processes



DevOps ;
Continuous development
Continuous integration

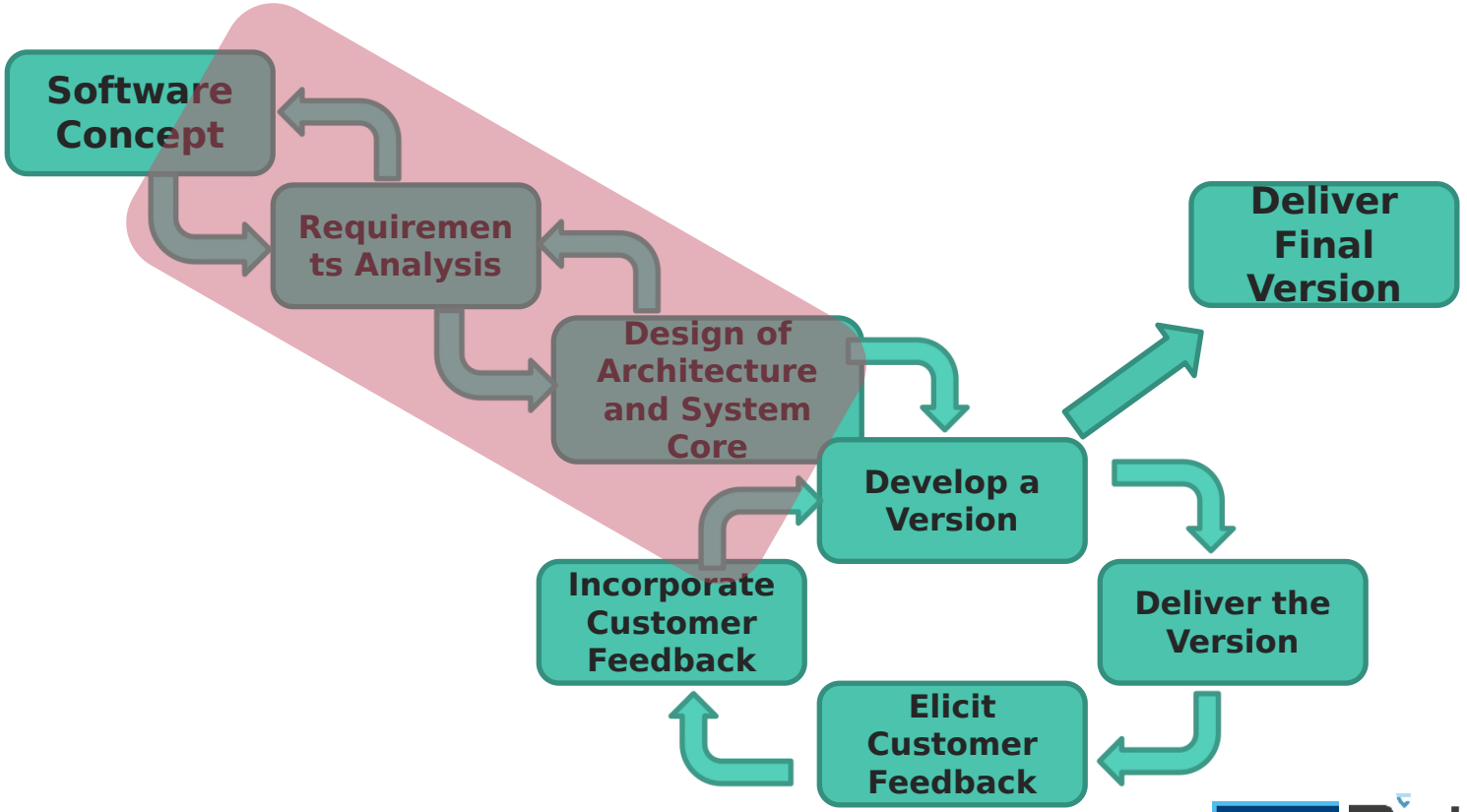
- **Automation** (testing, deployment) in **pipeline**
- **Direct inputs** from deployed version (Operational metrics)
- No **delineation between “versions”**



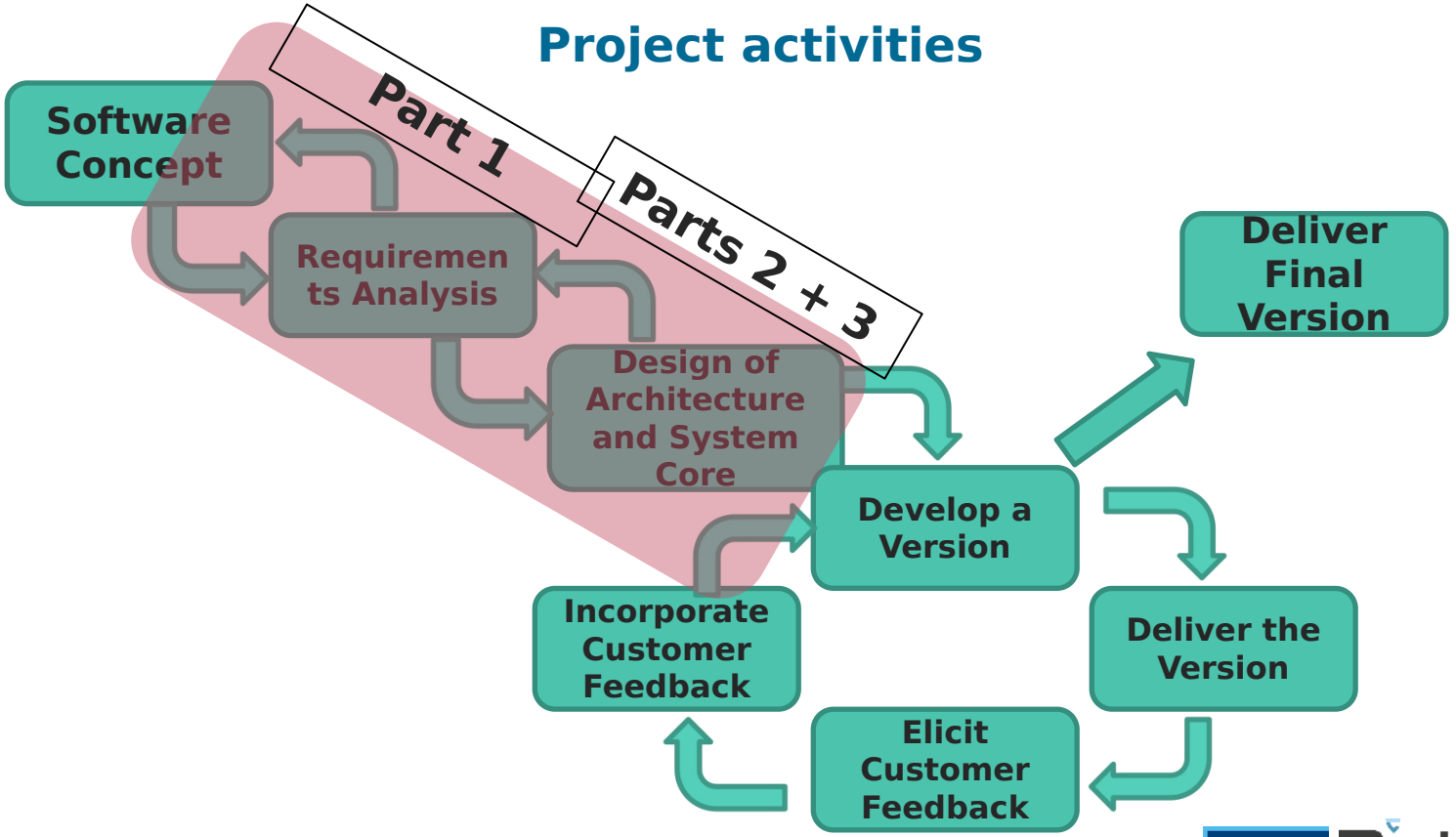
Focus in this course is on the **initial stages of development**

Regardless of your choice of implementation process/ technology/ “philosophy”

Architecture is bridge between requirements and design



Architecture is bridge between requirements and design

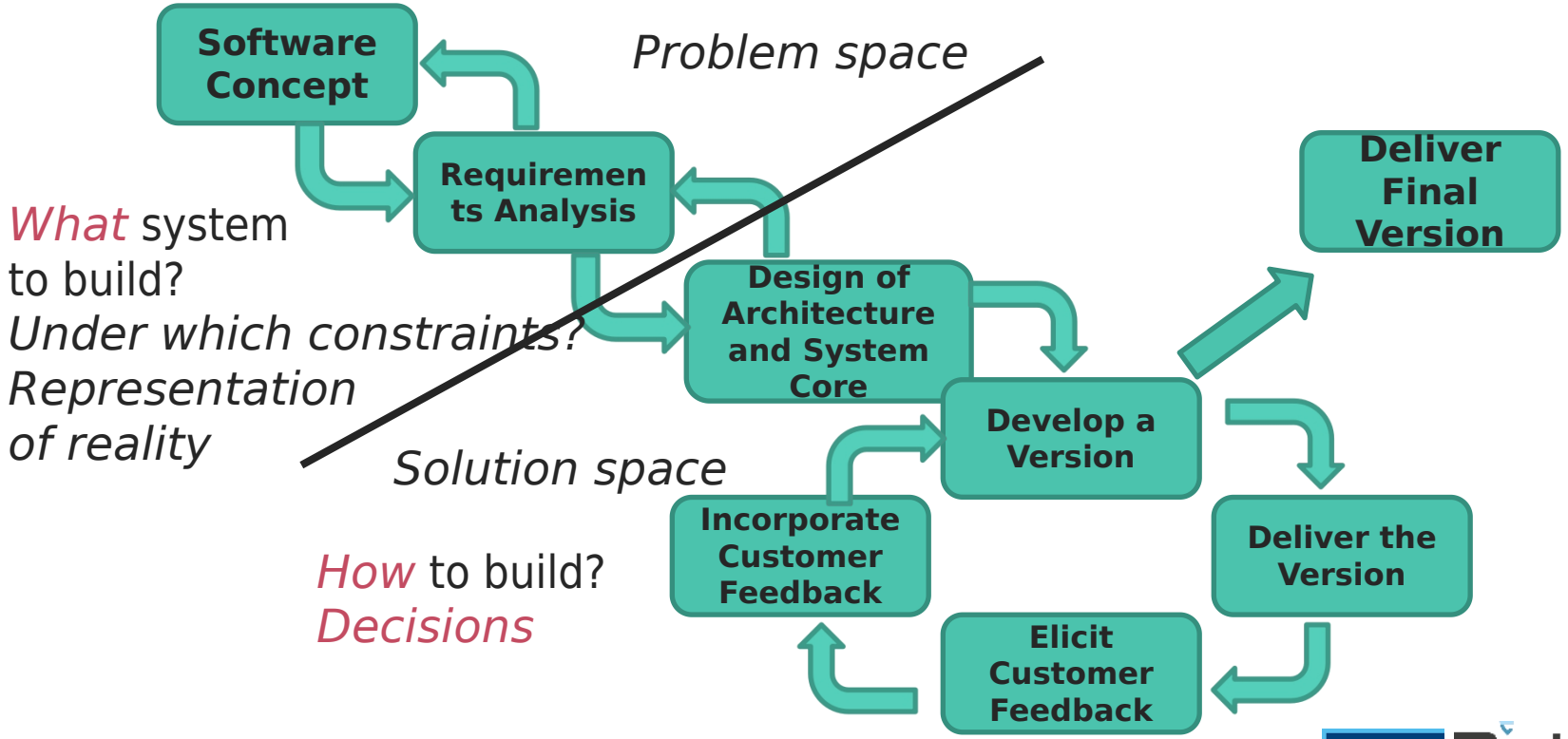


Architecture is the “bridge” between requirements and design

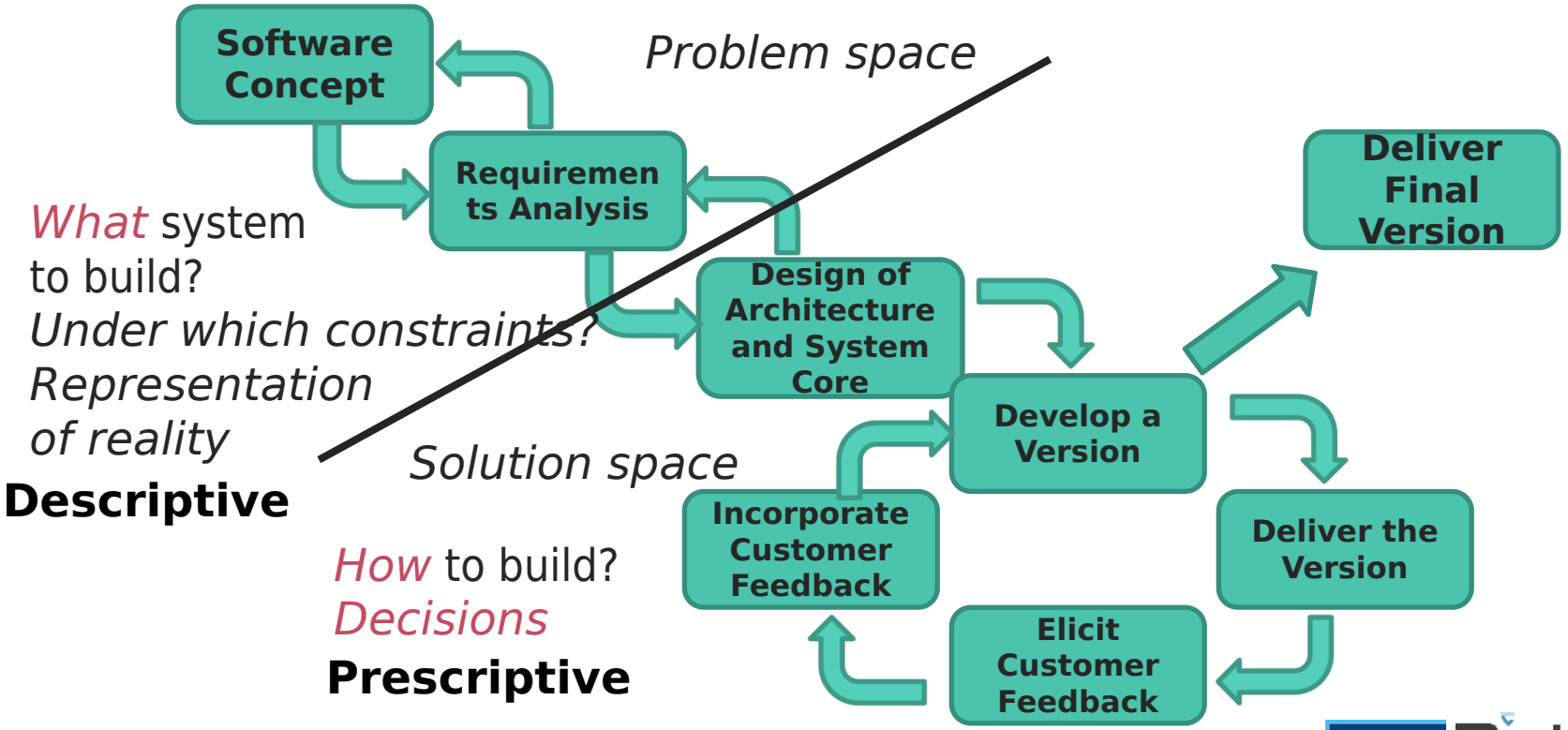
between

Problem space and **Solution space**

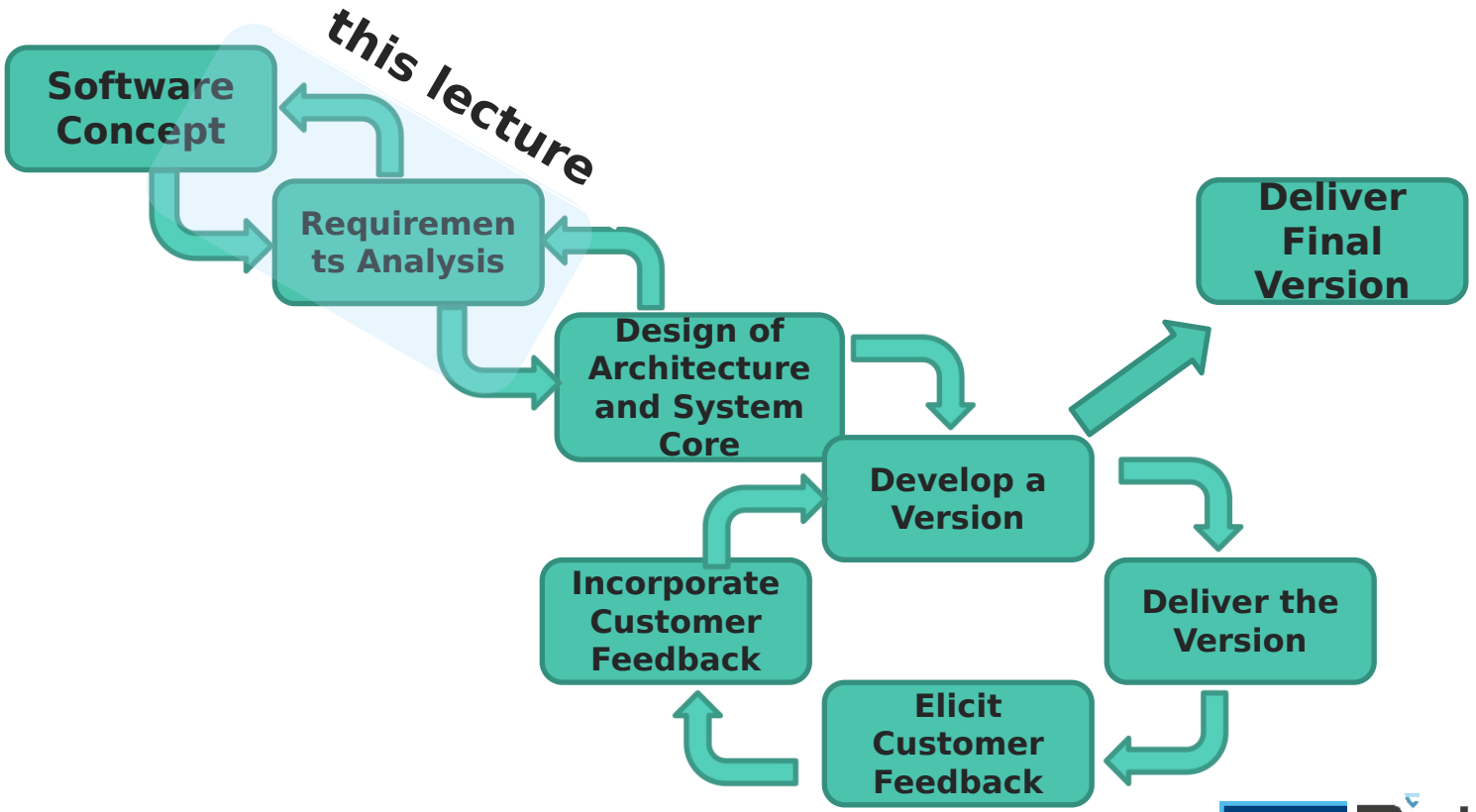
Architecture is bridge between requirements and design



Architecture is bridge between requirements and design



Architecture is bridge between requirements and design



(2) Requirements Engineering

THE PROJECT CONSTRUCTION CYCLE – THE TREE SWING

HOW THE CLIENT DESCRIBED IT

HOW THE ARCHITECT ENVISIONED IT

HOW THE ENGINEER DESIGNED IT

WHAT THE BUDGET ALLOWED

HOW THE LIABILITY INSURANCE AGENT DESCRIBED IT

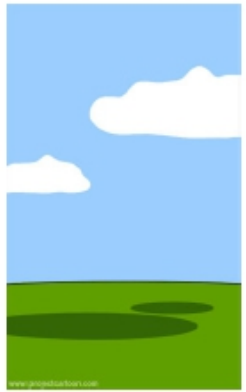
HOW THE ESTIMATOR BID IT

HOW THE MANUFACTURER MADE IT

WHAT THE BUILDING INSPECTOR EXPECTED

HOW THE CONTRACTOR INSTALLED IT

WHAT THE CUSTOMER REALLY WANTED



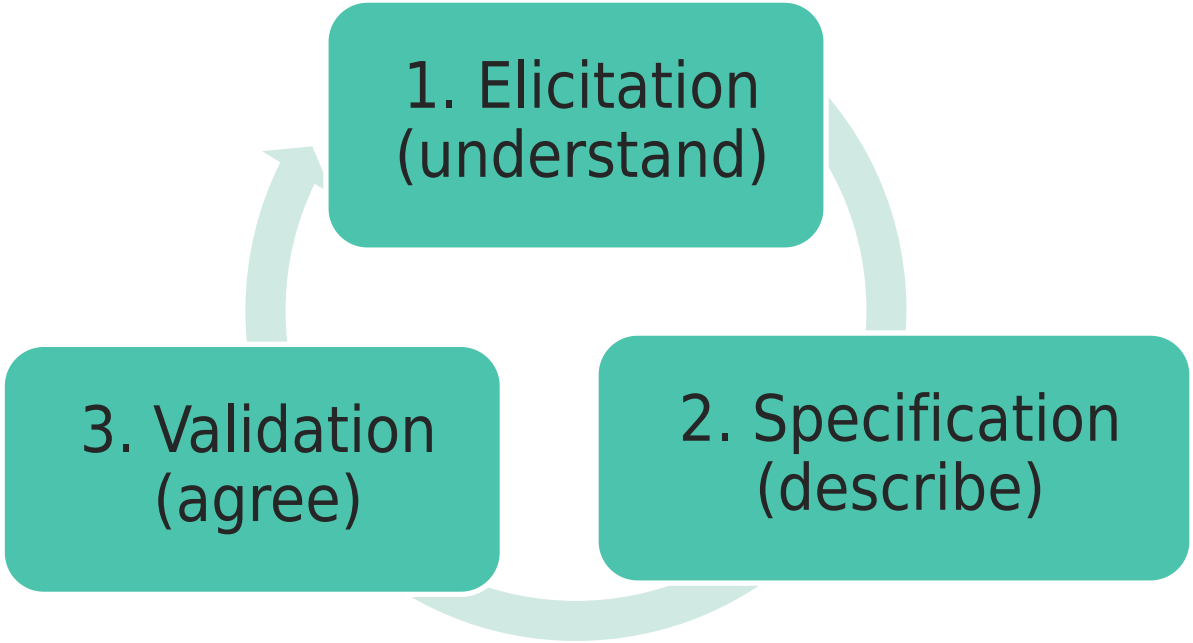
HOW THE PROJECT WAS DOCUMENTED

HOW THE CUSTOMER WAS BILLED

Definition

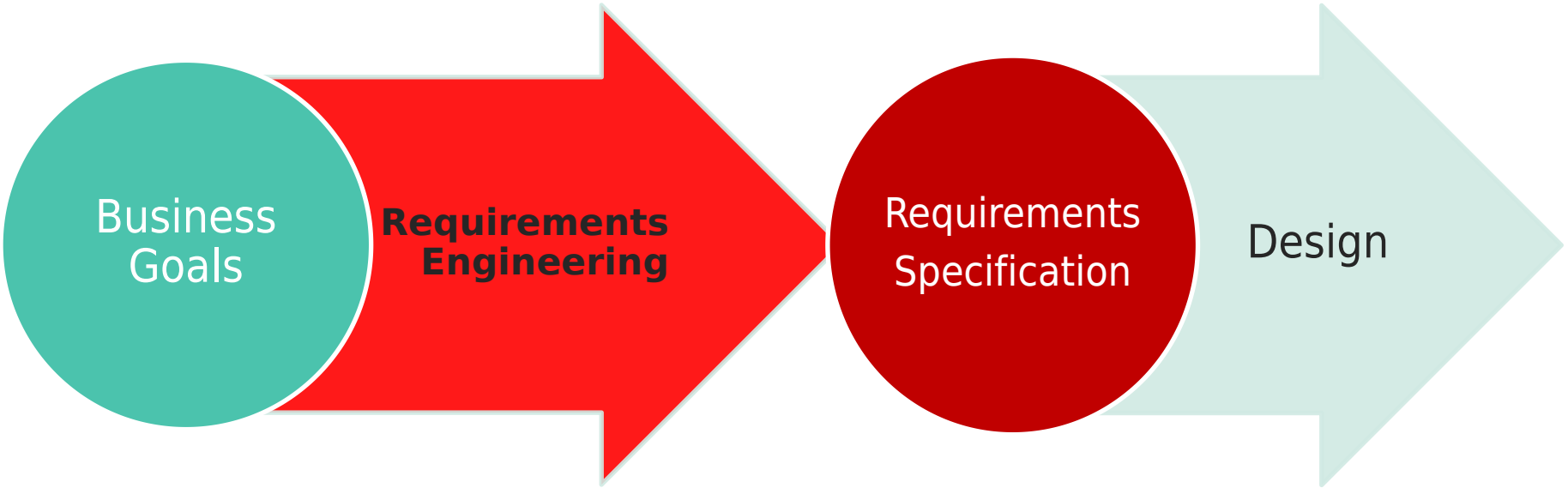
- Requirements engineering is the branch of software engineering concerned with the characterization and analysis of
  - the **real-world goals** for functions of,
  - **constraints on** software systems (“domain analysis”/ “context analysis”)
- It is also concerned with the relationship of these factors
  - to **precise specifications** of software behavior and
  - to their **evolution over time** and across software families

Requirements Engineering Process



Requirements Engineering

- First step in finding a solution



Functional vs. Non-Functional Requirements

1. **Functional requirements**
   - Functionality (“System services”/visible behavior)
2. **Non-functional requirements** or ***quality*** *requirements*
   - Constraints under which the system must operate
   - **Quality** the system must exert
   - *ilities

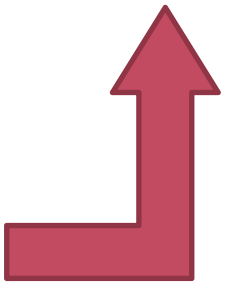
In this course, we rely on **scenario-based** requirements elicitated from a **black box perspective**

Functional vs. Non-Functional Requirements

1. **Functional requirements**
   - Functionality (“System services”/visible behavior)

USE CASES

2. **Non-functional requirements** or ***quality*** *requirements*
   - Constraints under which the system must operate
   - **Quality** the system must exert
   - *ilities



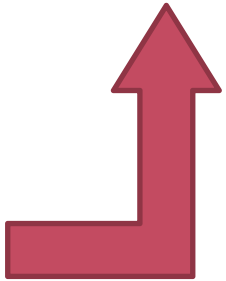
In this course, we rely on **scenario-based** requirements elicitated from a **black box perspective**

Functional vs. Non-Functional Requirements

1. **Functional requirements**
   - Functionality (“System services”/visible behavior)
2. **Non-functional requirements** or ***quality*** *requirements*
   - Constraints under which the system must operate
   - **Quality** the system must exert
   - *ilities

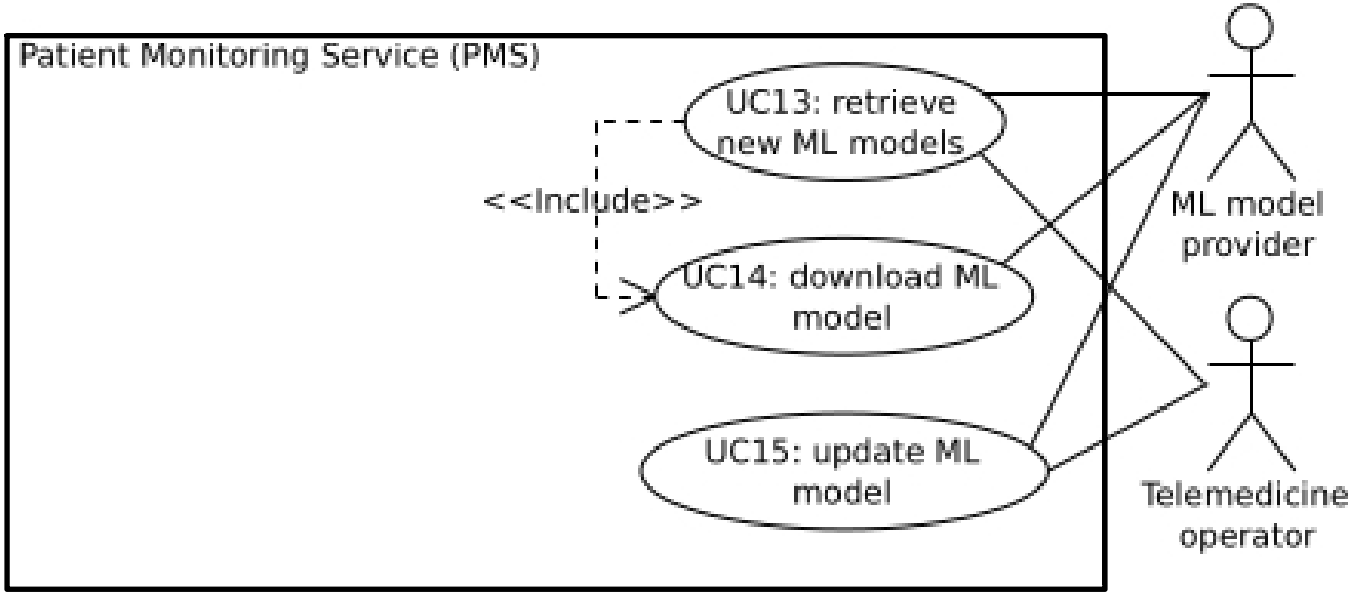
USE CASES

QUALITY ATTRIBUTE SCENARIOS



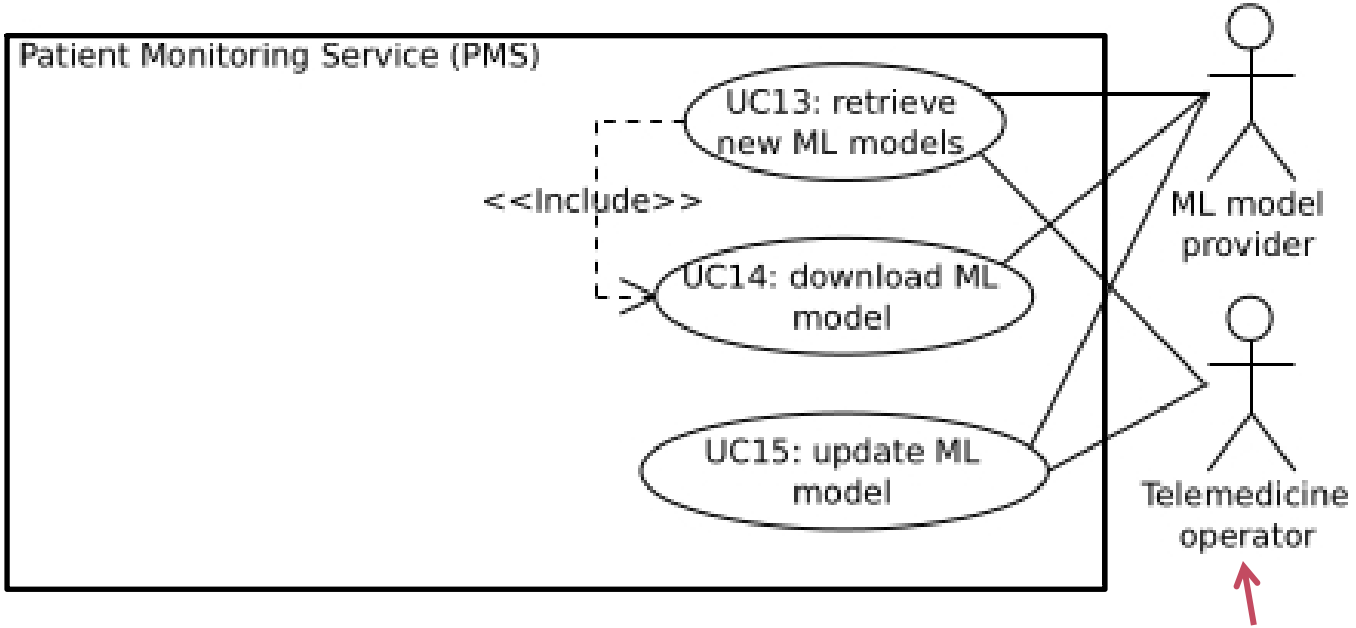
In this course, we rely on **scenario-based** requirements elicitated from a **black box perspective**

Use case model



Functional requirements

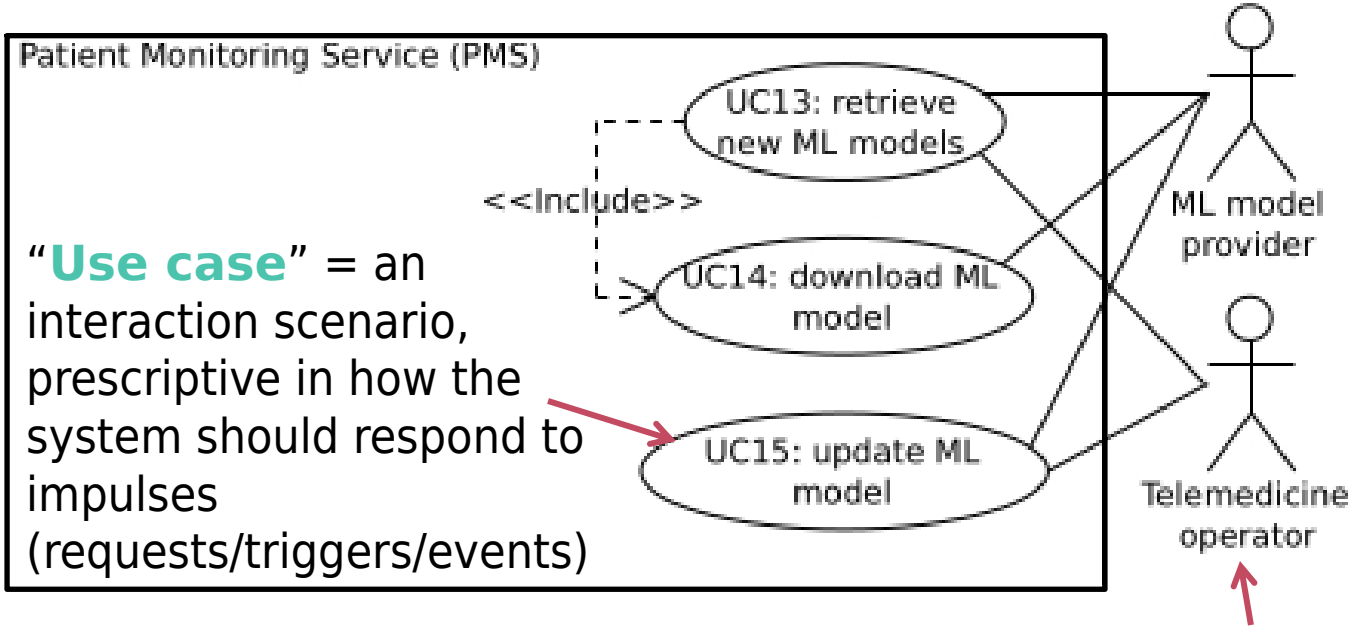
Use case model



"**Actor**" = user or external system

Functional requirements

Use case model



"**Use case**" = an interaction scenario, prescriptive in how the system should respond to impulses (requests/triggers/events)

"**Actor**" = user or external system

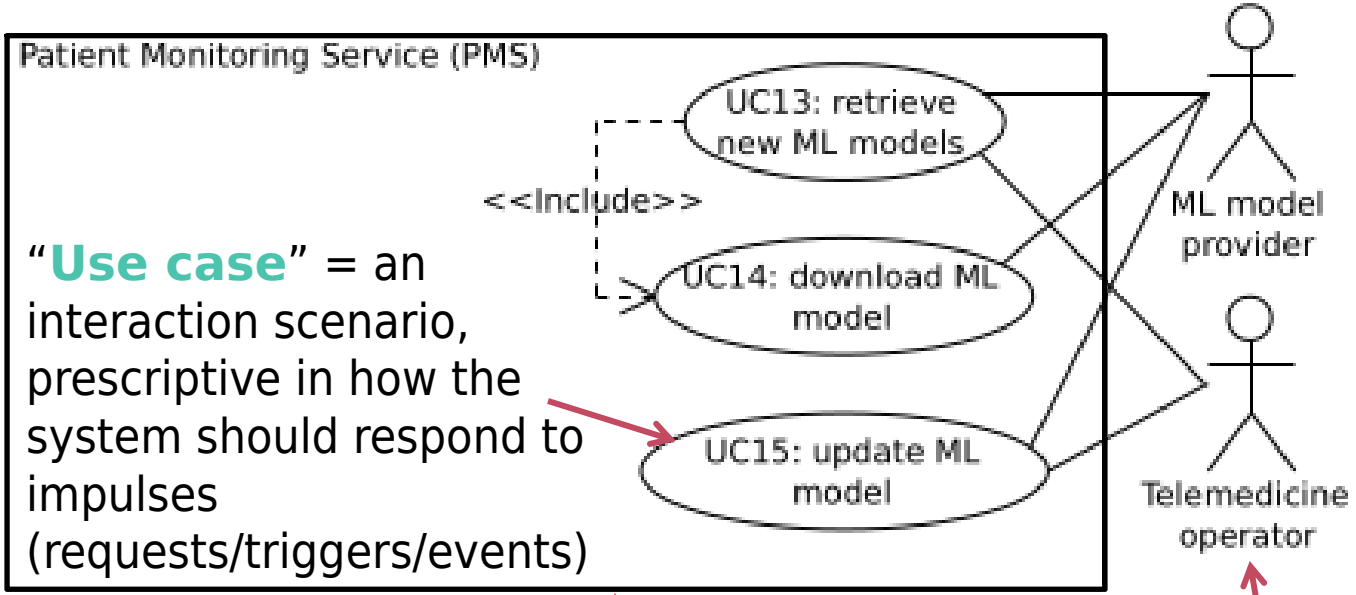
Functional requirements

Use case: retrieve new ML models

1. The **Telemedicine operator** indicates to the system it check for newly available ML models.
2. The **PMS** queries the **ML model provider** for a list of available ML models.
3. The **ML model provider** replies with a list of all available ML models, including metadata such as version number.
4. The **PMS** determines that new ML models are available and for each such model retrieves it from the model provider (Include: UC14 : download ML model)
5. The **PMS** informs the Telemedicine operator which new ML models were added.

Functional requirements **scenario-based** **black-box perspective**

Use case model



"**Use case**" = an interaction scenario, prescriptive in how the system should respond to impulses (requests/triggers/events)

System boundary

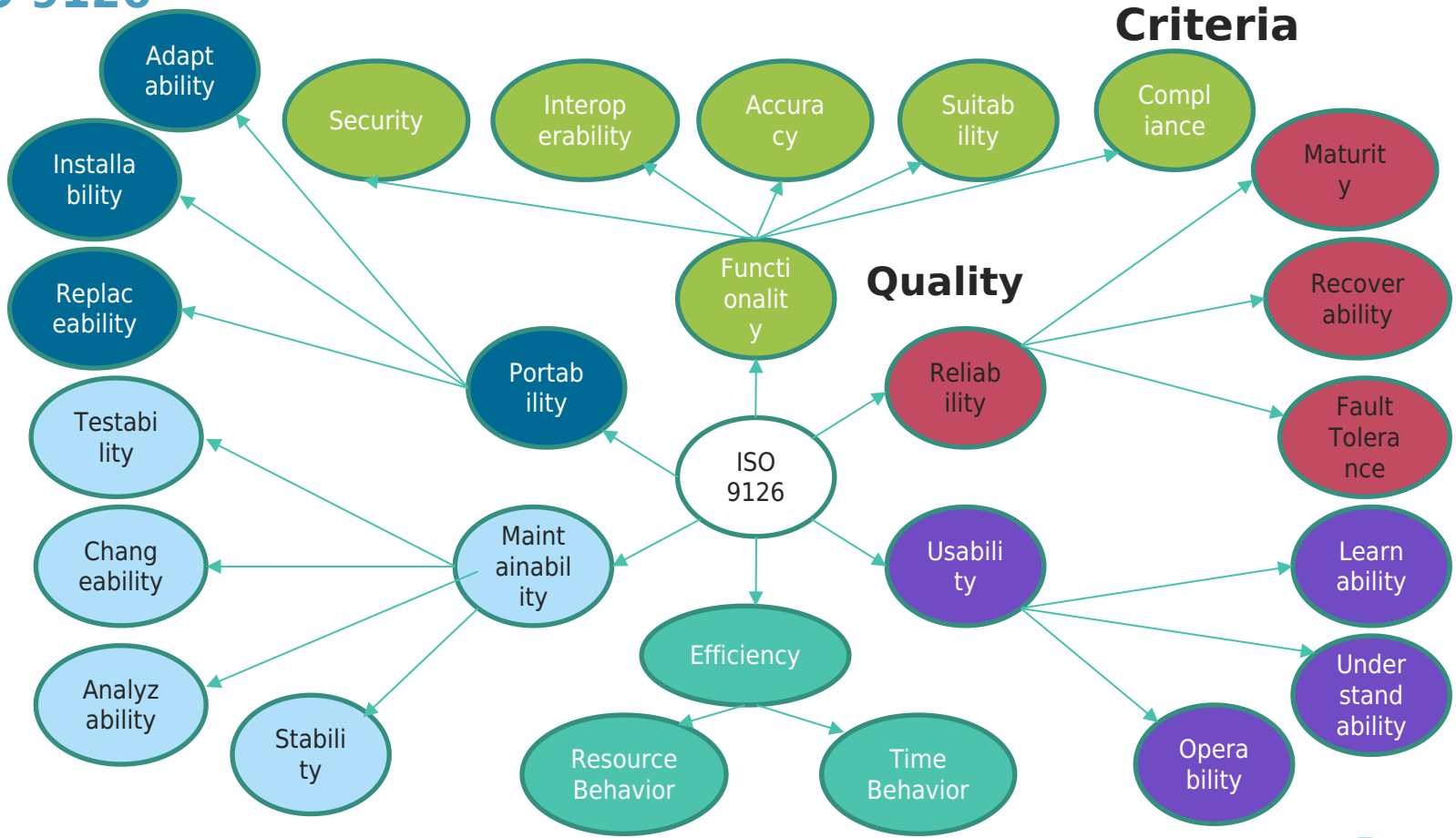
"**Actor**" = user or external system

Functional requirements

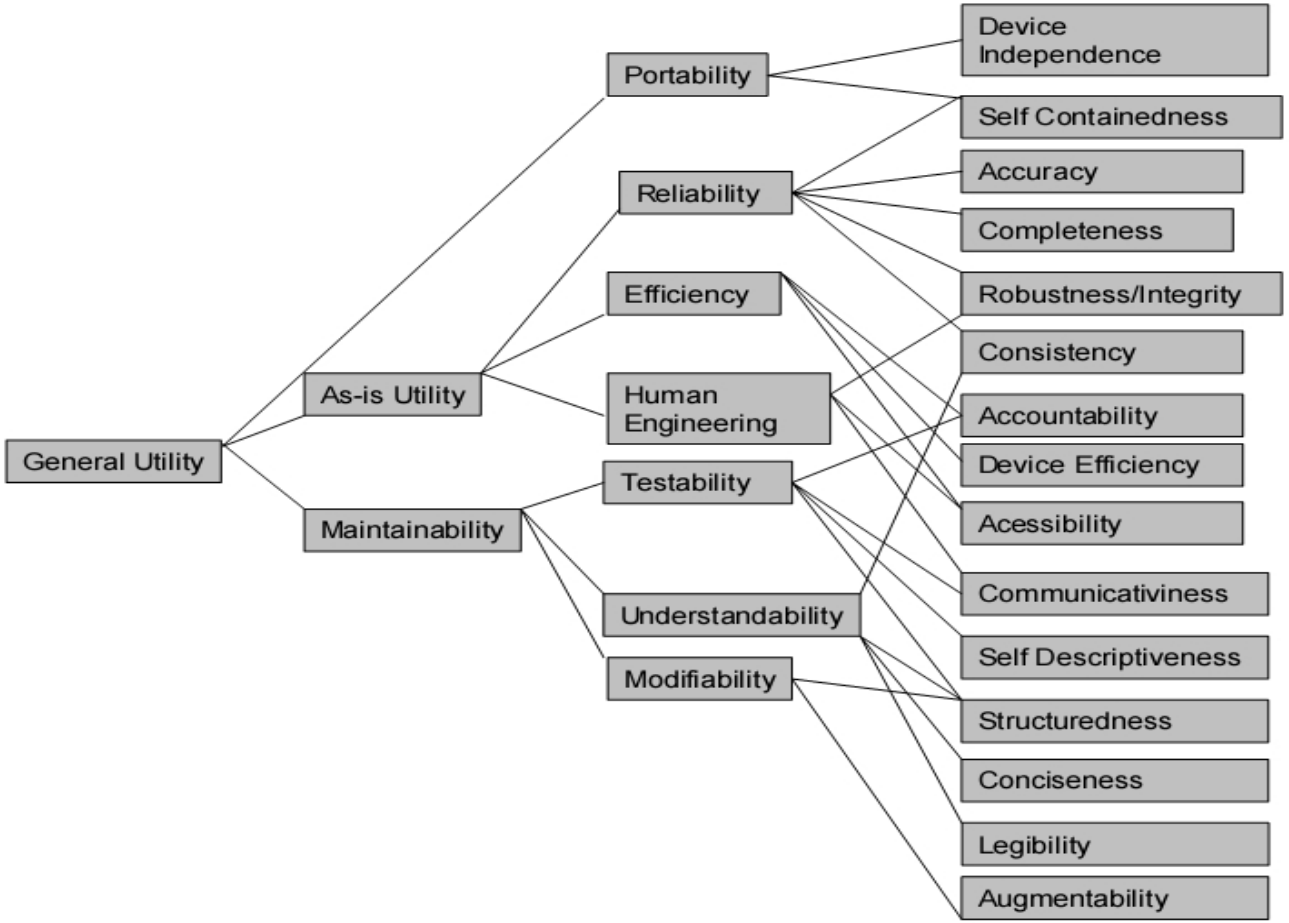
Software Quality Models

- **Lots of ambiguity**: e.g. *performance of an AI system (precision/recall)* vs *performance of a system (throughput/latency)*?
- Aim for a **level of concretization**:
  - allow verifying that quality is achieved: **measures**
- Quality Model
  - Typically tree structures
  - High level goals are refined in low level criteria
  - Metrics are attached to criteria
- Several models: ISO, McCall, Boehm, …

ISO 9126


Criteria
Adapt ability
Security
Interop erability
Accura cy
Suitab ility
Compl iance
Installa bility
Maturit y
Replac eability
Functi onalit y
Quality
Recover ability
Portab ility
Reliab ility
Fault Tolera nce
Testabi lity
ISO 9126
Chang eability
Maint ainabil ity
Usabili ty
Learn ability
Efficiency
Under stand ability
Analyz ability
Stabili ty
Resource Behavior
Time Behavior
Opera bility

Quality Models: Boehm



Criteria

System Quality Attributes

- Each have their own community (taxonomy, vocabulary, etc…)

- Definitions are not operational
- Fuzzy borders: is *system failure* an aspect of security, availability, or usability?

- **Quality attribute scenarios** (spec of NFR)

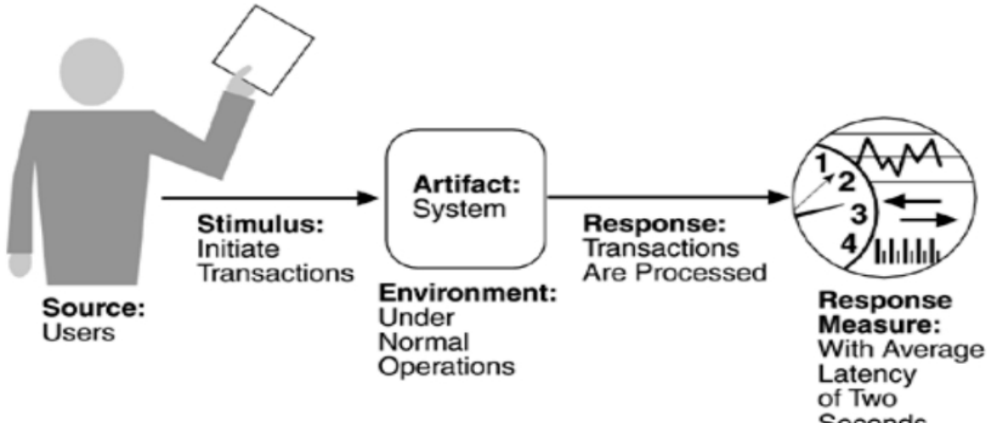
  scenario-based + blackbox perspective

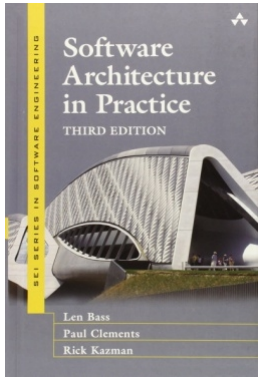
Quality attribute scenario elicitation

Handbook provides checklists

- Primary: *Availability*, *Interoperability*, *Modifiability*, *Performance*, *Security*, *Testability*, *Usability*
- Secondary: *Variability*, *Portability*, *Development distributability*, *Scalability/elasticity*, *Deployability*, *Mobility*, and *Monitorability*

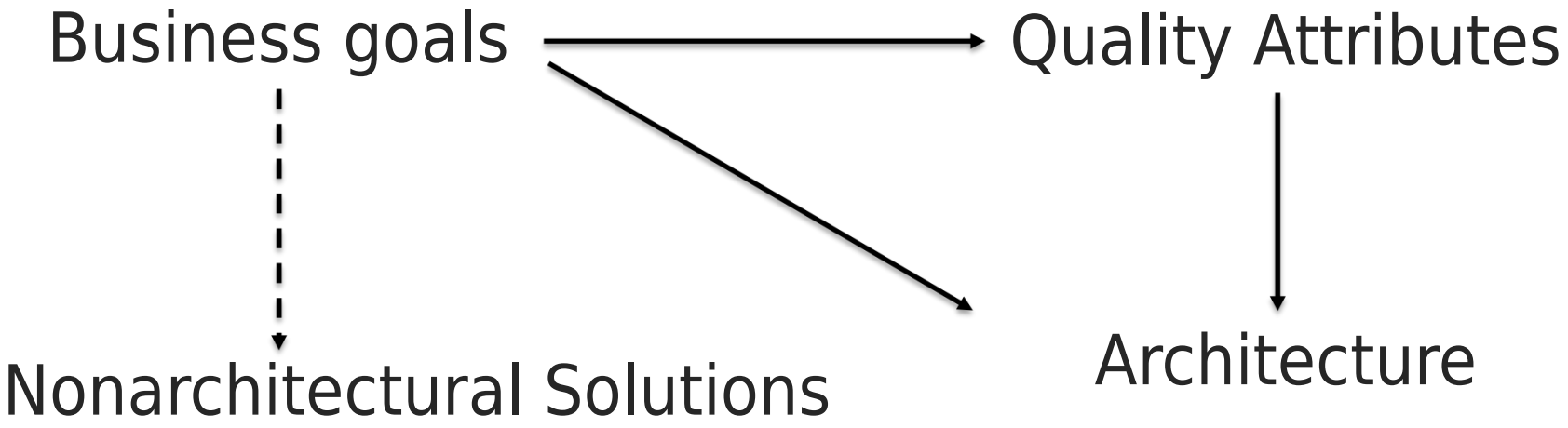
Quality-attribute-specific tables to support creating system-specific scenarios





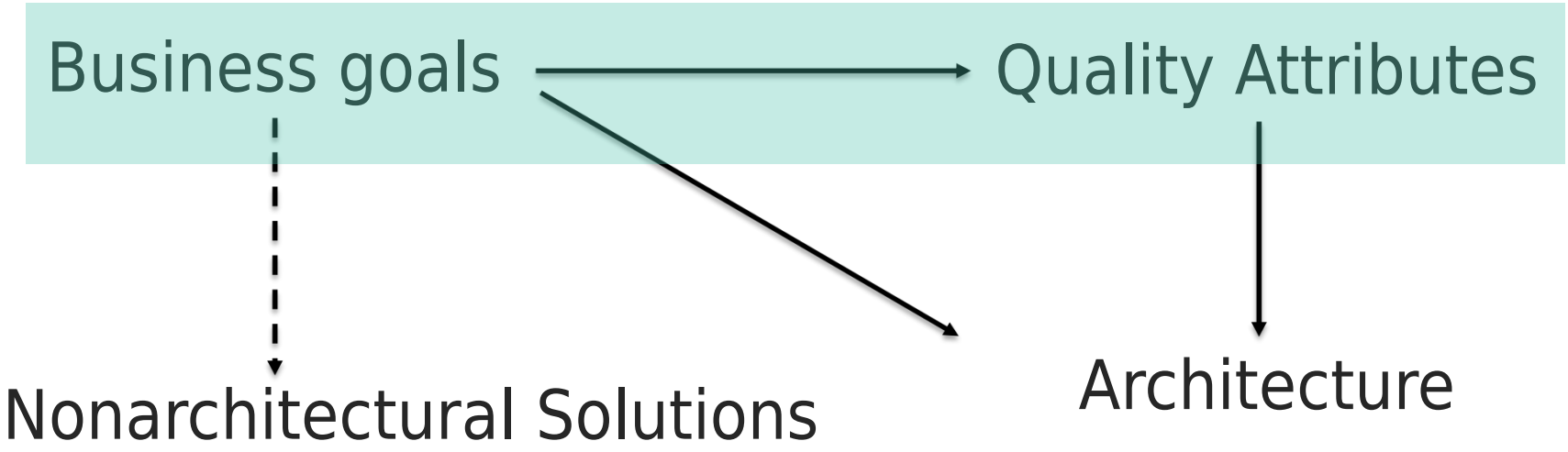
What makes requirements *architecturally significant*?

Business goals



Software Architecture in Practice, ed 3; Chapter 3, Figure 3.2

Business goals



Software Architecture in Practice, ed 3; Chapter 3, Figure 3.2

Business goals

Gathering ASRs by understanding the business goals

- **Categories**
  - Contributing to the growth and continuity of the organization
  - Meeting financial objectives
  - Meeting personal objectives
  - Meeting responsibility to employees
  - Meeting responsibility to society
  - Meeting responsibility to state
  - Meeting responsibility to shareholders
  - Managing market position
  - Improving business processes
  - Managing the quality and reputation of products
  - Managing change in environmental factors

Software Architecture in Practice, ed 3; Chapter 16, Table 16.2

- ***Template*** (PALM):
  - Goal-source
  - Goal-subject
  - Goal-object
  - Environment
  - Goal
  - Goal-measure
  - Pedigree and value

Software Architecture in Practice, ed 3; Chapter 16

In this course:
good understanding of business goals is essential but they are documented informally

Architecturally significant requirements (ASRs)

- Criteria:
  1. **Architectural impact**: including the ASR will result in a very different architecture than if it were not included
  2. **Business or mission value:** importance to stakeholders
- Using a single list can help to evaluate each potential ASR against these criteria **and prioritize**

Architecturally significant requirements (ASRs)

What do you think?

- Color of the *add to shopping basket* button?

Architecturally significant requirements (ASRs)

What do you think?

- Color of the *add to shopping basket* button?
- Database performance?

Architecturally significant requirements (ASRs)

What do you think?

- Color of the *add to shopping basket* button?
- Database performance?
- Android system-level dark mode?

Architecturally significant requirements (ASRs)

What do you think?

- Color of the *add to shopping basket* button?
- Database performance?
- Android system-level dark mode?
- Authentication mechanism?

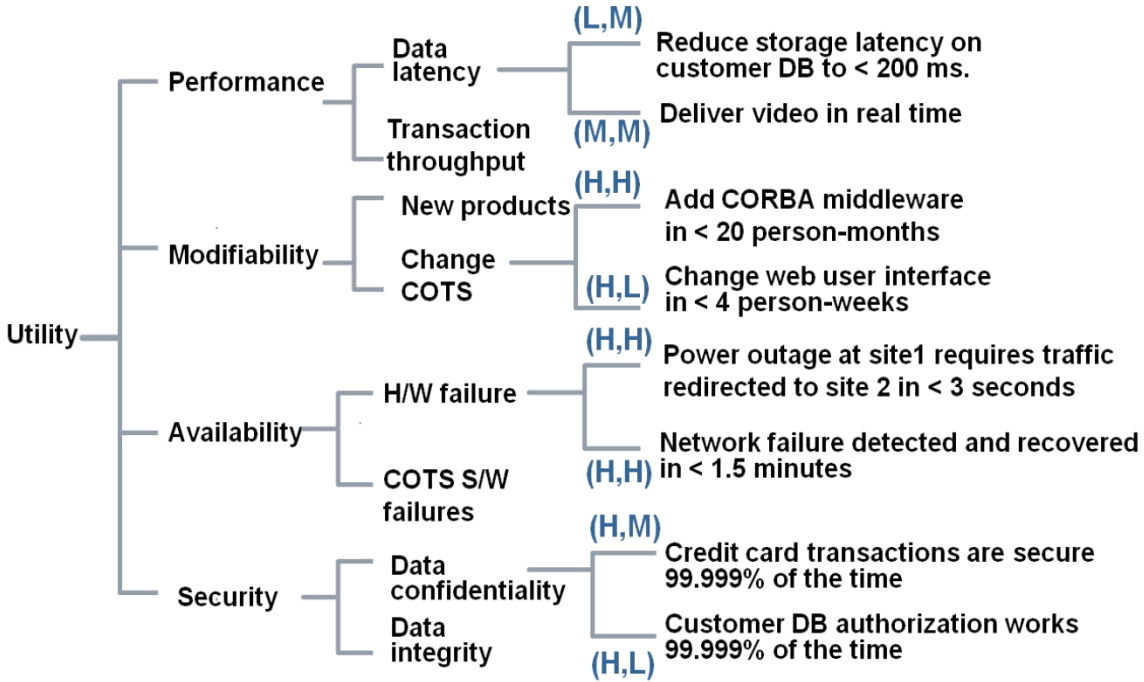
Architecturally significant requirements (ASRs)

What do you think?

- Color of the *add to shopping basket* button?
- Database performance?
- Android system-level dark mode?
- Authentication mechanism?
- Remote updating of car firmware?

Utility tree

- Four levels:
  1. Utility: expression of the overall *goodness* of a system
  2. Quality attribute
  3. Quality attribute refinement
  4. ASR incl. prio at the basis of <impact, value>



Recap

1. **Requirements engineering** is a separate discipline
2. **ASRs refer to software quality and are determined** by
   1. **business value,** and
   2. **architectural impact**

   Overview in **a Utility Tree**

   - Explicit motivation/rationale
3. **Documentation in Quality Attribute Scenarios**

   - Concrete+detailed+Measurable

Part 1 of the assignment

Takeaways

Take away #1

First positioning of
‘**software architecture**’:

the **bridge between requirements and design**

Take away #2

We do not perform requirements analysis for the sake of requirements analysis

we pragmatically explore the key requirements of the system

Breadth first/brainstorm – ASRs
In-depth refinement of the “ones that matter most” - QASs

Take away #2

We do not perform requirements analysis for the sake of requirements analysis

we pragmatically explore the architecturally significant requirements of the system

Breadth first/brainstorm – ASRs
In-depth refinement of the highest ranked NF requirements - QASs

Business value
Architectural impact