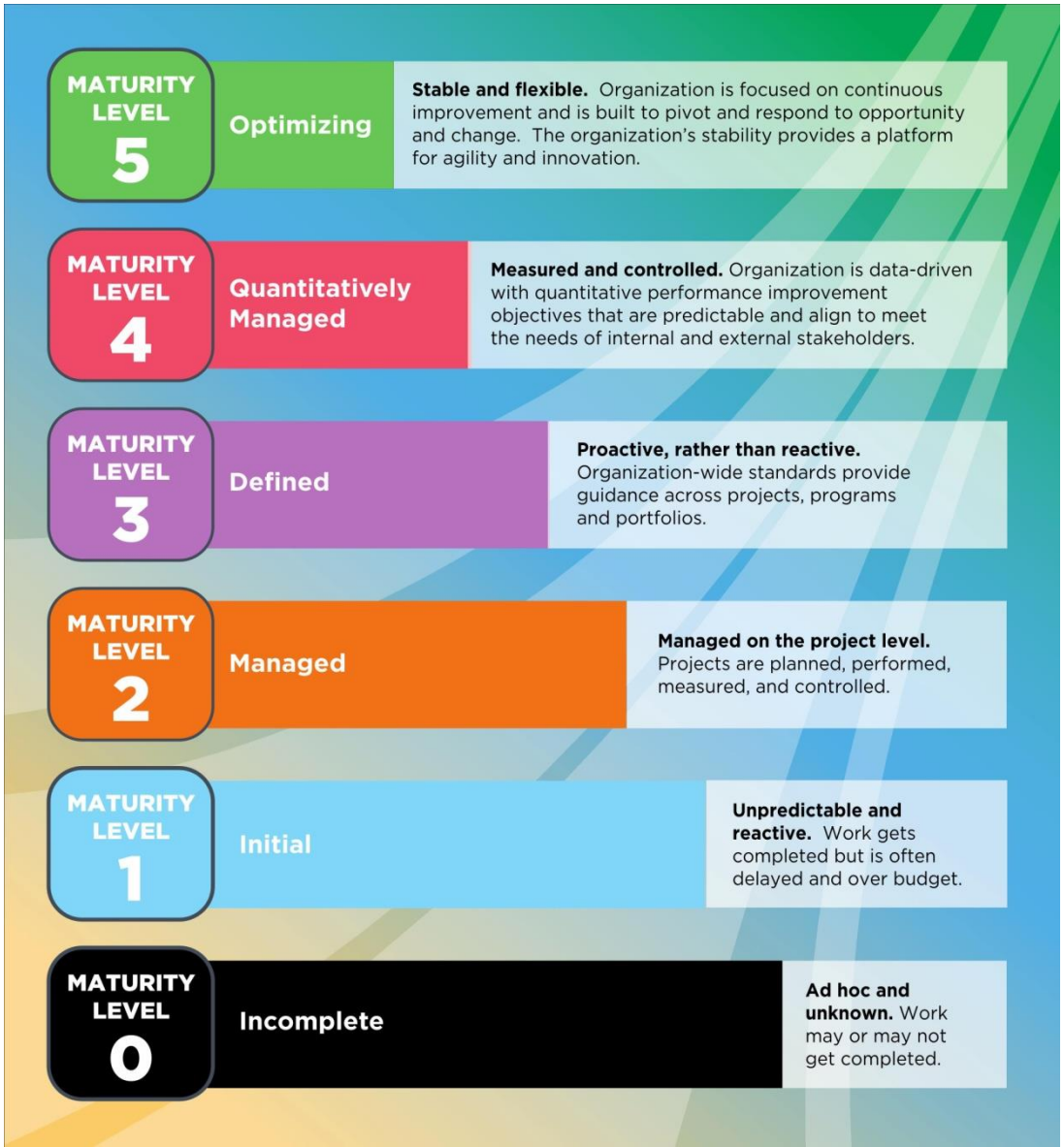
Produkto reikalavimų inžinerijos procesas

Dr. Asta Slotkienė

CMMI 2.0 Maturity levels



CMMI for Development (CMMI-DEV**)**

- **CMMI for Development is a framework** for any organization that **builds products and/or services.**
  - Organizations that are involved with developing may include **hardware and software companies**

- CMMI-DEV applies to all organizations that must consider **design and engineering during the development of a product or service.**

New and Changed Practices

- Requirement Development and Management!!!

Level	Focus	Process Areas
5 Optimizing	Continuous Process Improvement	Causal Analysis and Resolution (CAR) Organizational Performance Management (OPM)
4 Quantitatively Managed	Quantitative Management	Organizational Process Performance (OPP) Quantitative Project Management (QPM)
3 Defined	Process Standardization	Integrated Project Management (IPM) Risk Management (RSKM) Decision Analysis and Resolution (DAR) Requirements Development (RD) Technical Solution (TS) Product Integration (PI) Verification (VER) Validation (VAL) Organizational Process Focus (OPF) Organizational Process Definition (OPD) Organizational Training (OT)
2 Managed	Basic Project Management	Configuration Management (CM) Measurement and Analysis (MA) Project Monitoring and Control (PMC) Project Planning (PP) Process and Product Quality Assurance (PPQA) Requirements Management (REQM) Supplier Agreement Management (SAM)
1 Initial		

Quality Productivity

Risk Rework

CMMI Development V1.3

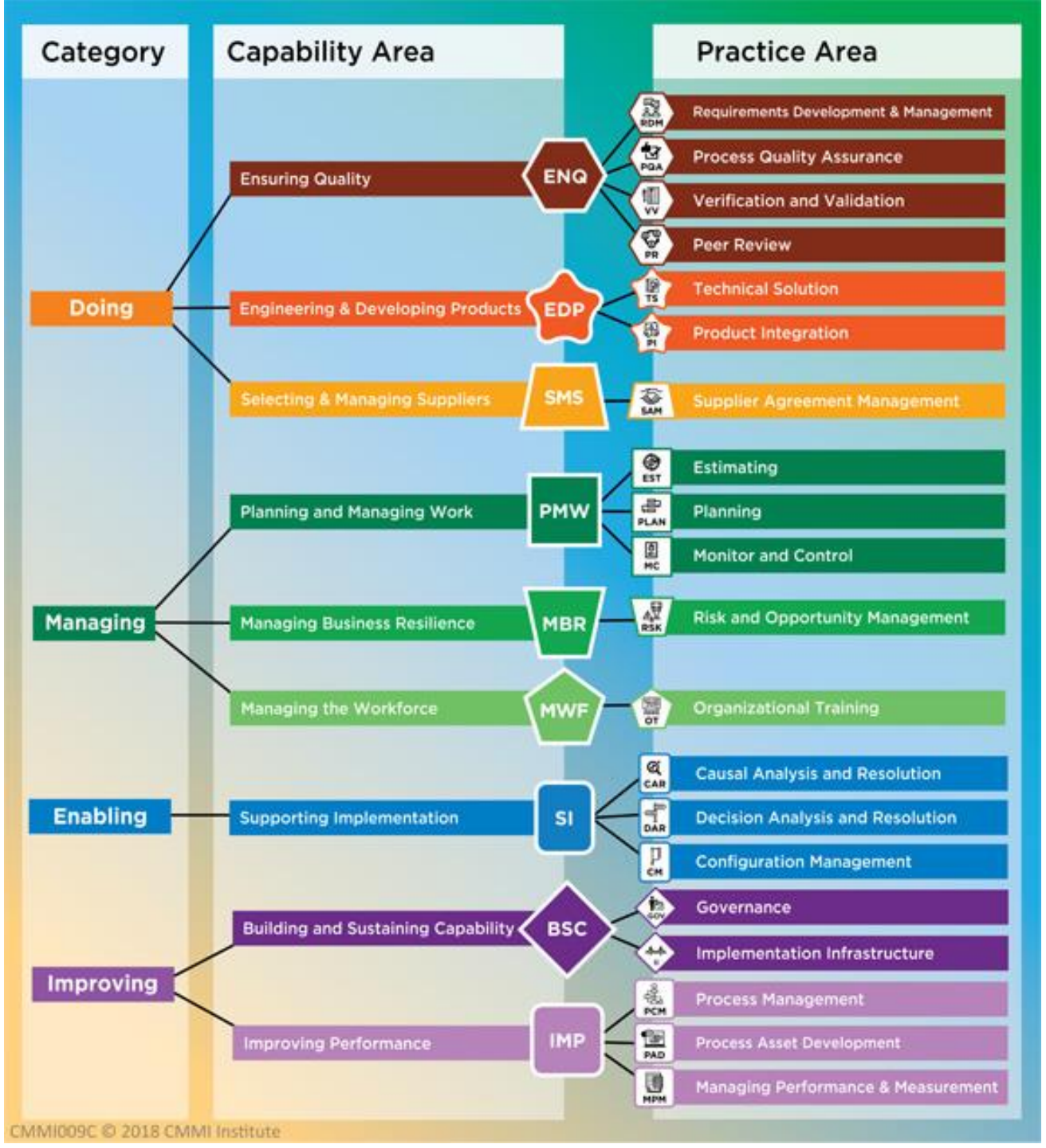
Level	✓ ← PA also exists at higher level	Practice Areas
5 Optimizing		CAR, MPM (level 5 practices)
4 Quantitatively Managed		CAR, PCM, SAM, MPM, PLAN, GOV (level 4 practices)
3 Defined	✓	Causal Analysis and Resolution (CAR)
		Decision Analysis and Resolution (DAR)
		Risk and Opportunity Management (RSK)
		Organizational Training (OT)
	✓	Process Management (PCM)
		Process Asset Development (PAD)
		Peer Reviews (PR)
		Verification and Validation (VV)
		Technical Solution (TS)
		Product Integration (PI)
2 Managed	✓	Supplier Agreement Management (SAM)
	✓	Managing Performance and Measurement (MPM)
	✓	Process Quality Assurance (PQA)
		Configuration Management (CM)
	✓	Monitor and Control (MC)
	✓	Planning (PLAN)
	✓	Estimation (EST)
	✓	Requirements Development & Management (RDM)
	✓	Governance (GOV)
	✓	Implementation Infrastructure (II)
1 Initial / 0 Incomplete		

Performance Capability

Risk Rework

CMMI Development V2.0 – Simple View

CMMI v2.0 Capability Areas



CMMI v2.0 Capability Areas: Doing

- Includes Capability Areas for **producing, buying, and delivering quality solutions.**

ENSURING QUALITY -helps to improve product and service quality

- **REQUIREMENTS DEVELOPMENT AND MANAGEMENT (RDM)**
  - RDM helps to develop and update a joint understanding of the necessities and expectations of the solution.
  - Maturity Level 1=ML1, ML2, ML3
- **PROCESS QUALITY ASSURANCE (PQA)**
  - PQA helps to identify and prevent adverse effects, while ensuring the recurrence of positive results. Addressing the root problems quickly reduces the need for subsequent rework and improves productivity.
  - ML1, ML2, ML3
- **VERIFICATION AND VALIDATION (VV)**
  - VV ensures that requirements are fulfilled and the solution works as planned in the target environment.
  - ML1, ML2, ML3
- **PEER REVIEWS (PR)**
  - It helps to identify defects and issues in the solution.
  - ML1, ML2, ML3

Ensuring Quality-> RDM

- **Requirements Development and Management** enables developing and keeping updated a common ***understanding of needs and expectations for the solution***.

  Intent:

  – Elicit requirements,

  – ensure common **understanding by stakeholders**,

  – align requirements, plans, and work products.

  Value: Ensures that **customer's needs and expectations are satisfied**

  https://cmmiinstitute.com/cmmi

Practices of Requirements Engineering I

- **Maturity Level 1 - Initial**
  - RDM 1.1 Record requirements.

https://cmmiinstitute.com/cmmi

CMMI-DEV 2.0:

Developing the Customer Requirements

- Techniques to elicit the requirements:
  - Interview sessions
  - **Prototypes**
  - Brainstorming
  - Market surveys
  - **Use cases**
  - **User Story**
  - **….**

Practices of Requirements Engineering II

- **Maturity Level 2 - Managed**
  - RDM 2.1 Elicit stakeholder needs, expectations, constraints, and interfaces or connections.
  - RDM 2.2 **Transform stakeholder needs, expectations, constraints, and interfaces or connections into prioritized customer requirements**.
  - RDM 2.3 **Develop an understanding with the requirements providers on the meaning of the requirements.**

https://cmmiinstitute.com/cmmi

CMMI-DEV 2.0:

Transform the Stakeholder Needs into Customer Requirements

1. Check whether any **information is missing**, and update the missing part
2. Determine whether any **conflicts** exist in the requirements

Practices of Requirements Engineering III

- **Maturity Level 2 - Managed**
  - RDM 2.4 Obtain commitment from project participants that they can implement the requirements.
  - RDM 2.5 **Develop, record, and maintain bidirectional traceability among requirements** and activities or work products.
  - RDM 2.6 Ensure that plans and activities or work products remain consistent with requirements.

https://cmmiinstitute.com/cmmi

Practices of Requirements Engineering IV

- **Maturity Level 3 - Defined**
- RDM 3.1 **Develop and keep requirements updated for the solution and its components**.
  RDM 3.2 **Develop** operational concepts and **scenarios**.
  RDM 3.3 Allocate the requirements to be implemented.

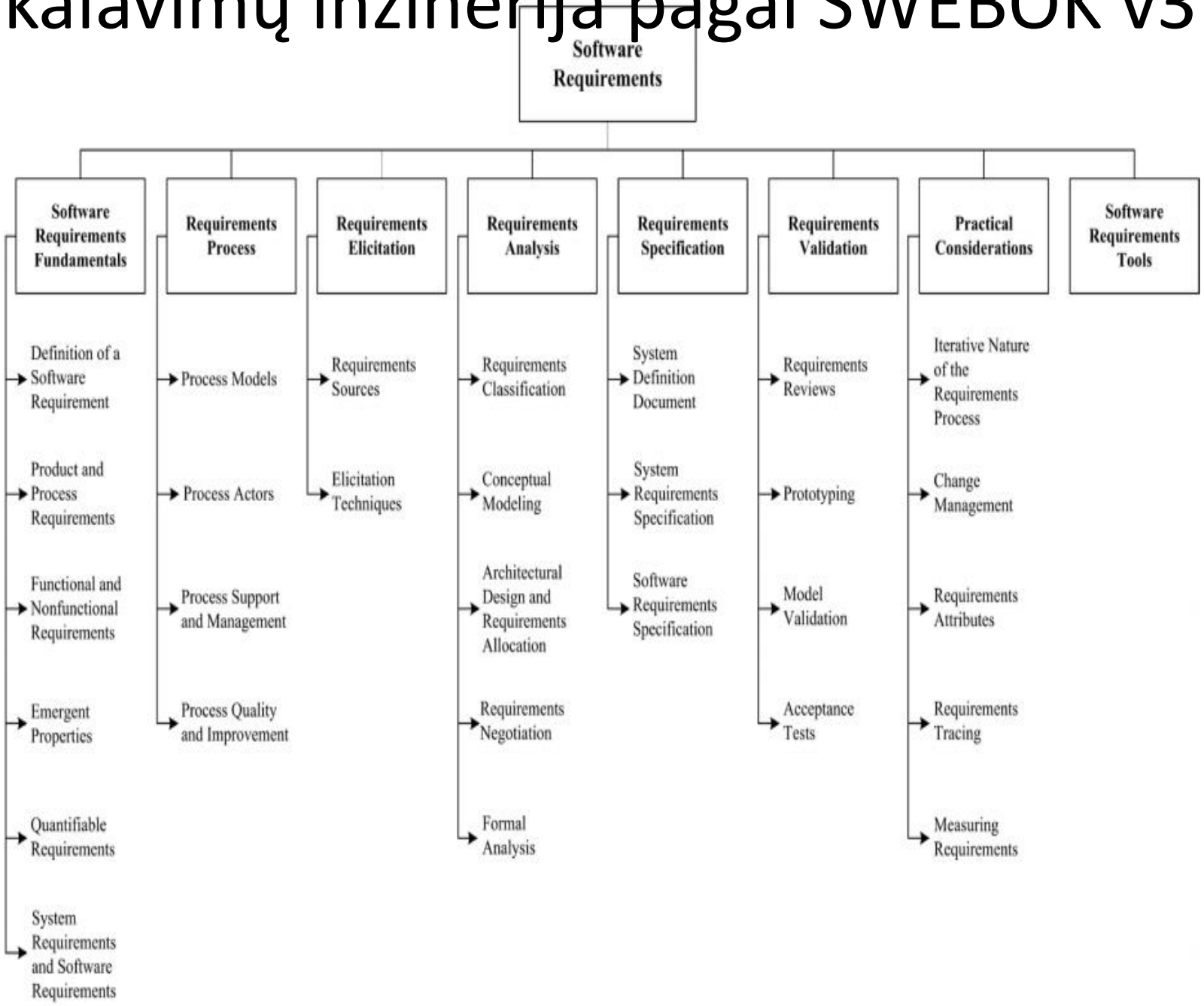
https://cmmiinstitute.com/cmmi

Practices of Requirements Engineering V

- **Maturity Level 3 - Defined**
  - RDM 3.4 Identify, develop, and keep updated interface or connection requirements.
  - RDM 3.5 Ensure that requirements are necessary and sufficient.
  - RDM 3.6 Balance stakeholder needs and constraints.
  - RDM 3.7 Validate requirements to ensure the resulting solution will perform as intended in the target environment.

https://cmmiinstitute.com/cmmi

Reikalavimų inžinerija pagal SWEBOK v3



What is requirement?

- Software requirements express the **needs and constraints** placed on a software product that contribute to the solution of some **real-world problem**

Definition: Requirement (IEEE)

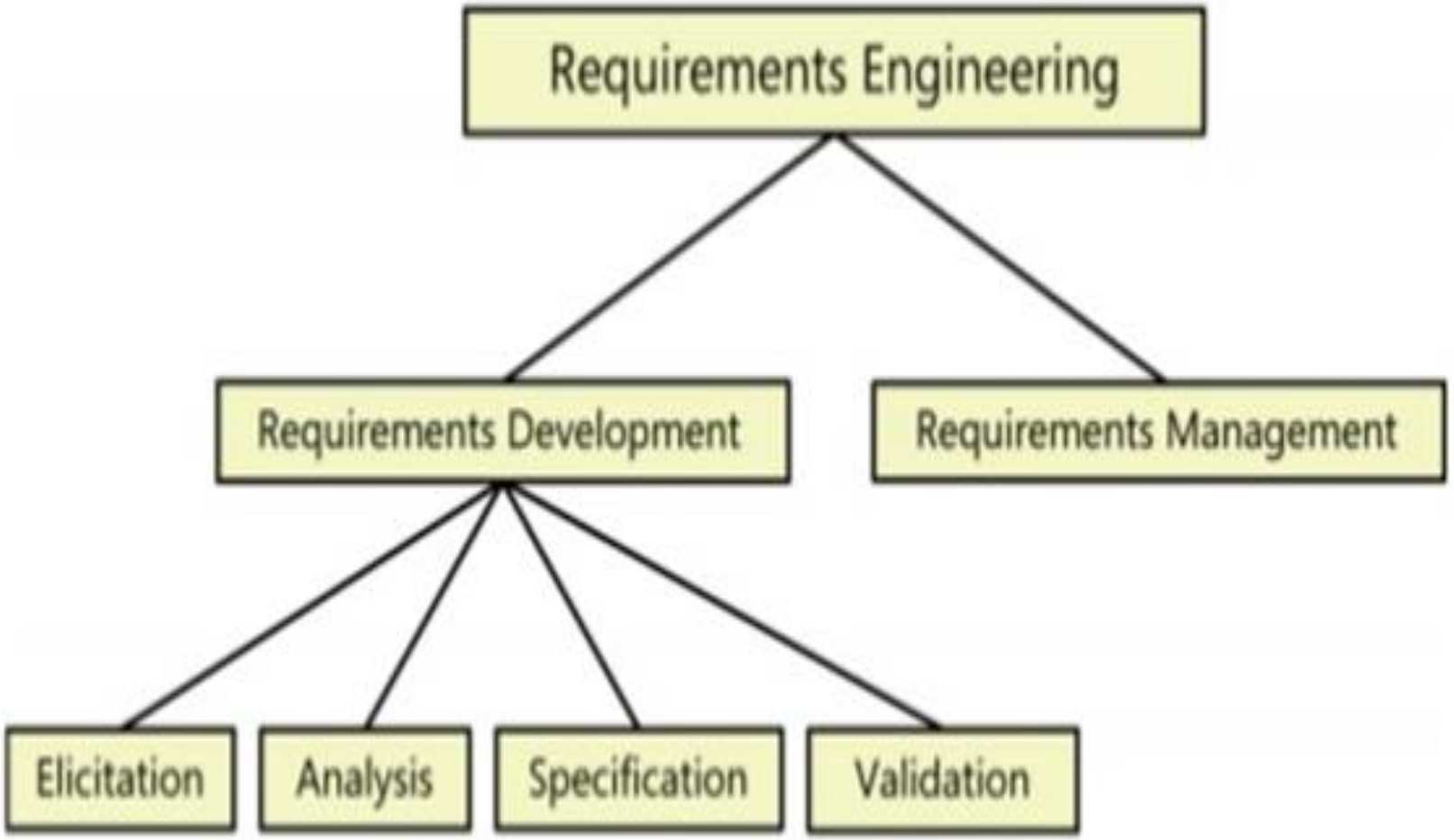
The IEEE Standard Glossary of Software Engineering Terminology defines a requirement as [IEEE 610.12-1990]:

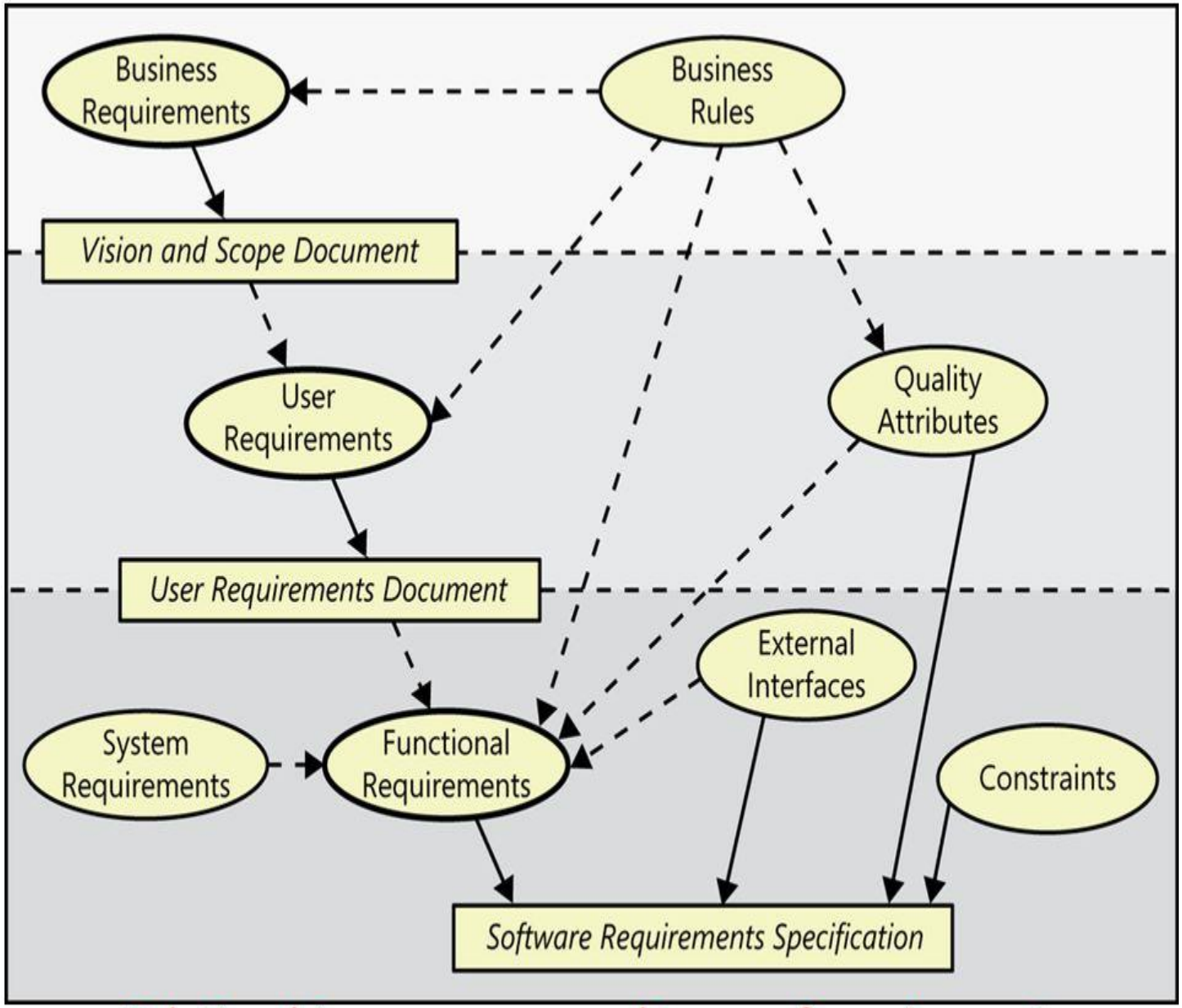
(1) A condition or capability needed by a user **to solve a problem** or **achieve an objective**.

(2) A condition or capability that must be **met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents**.

(3) **A documented representation of a condition** or capability as in (1) or (2).

Subset of Requirement Engineering





Relationships among several types of requirements information.

Why are requirements so difficult to specify?

- Customer's **can't tell** you what they want
  - they don't know
  - they don't clearly articulate their needs

- Customer's have conflicting needs
- Customer's needs change or there understanding of their needs

Why are Requirements Important?

- Causes of failed software projects:

  Incomplete requirements

  Lack of user involvement

  Lack of resources

  Unrealistic expectations

  Lack of executive support

  Changing requirements & specification

  Lack of planning

  System no longer needed

Requirement Level Clasification

Sommerville (2005) suggests organizing them into three levels of abstraction:

- User requirements
- System requirements
- Design specifications

CMMI-DEV 2.0

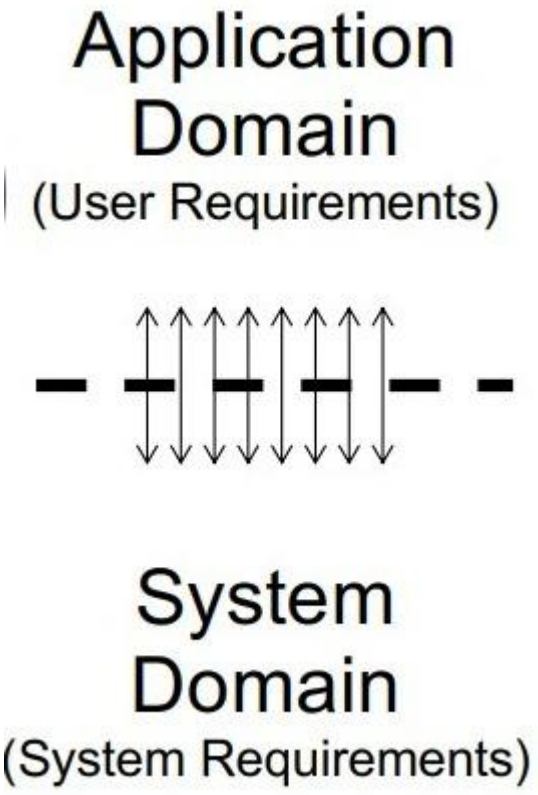
- Three different types:
  - Customer Requirements
  - Product Requirements
  - Product Component Requirements

Definition of UR

- **User requirements** describe goals or tasks the **users must be able to perform** with the product that will provide value to someone

User Requirement

- User requirements are **abstract statements written in natural language** with accompanying informal diagrams.
- They specify what services (user functionality) the system is expected to provide and any constraints.
- In many situations user stories can play the role of user requirements.

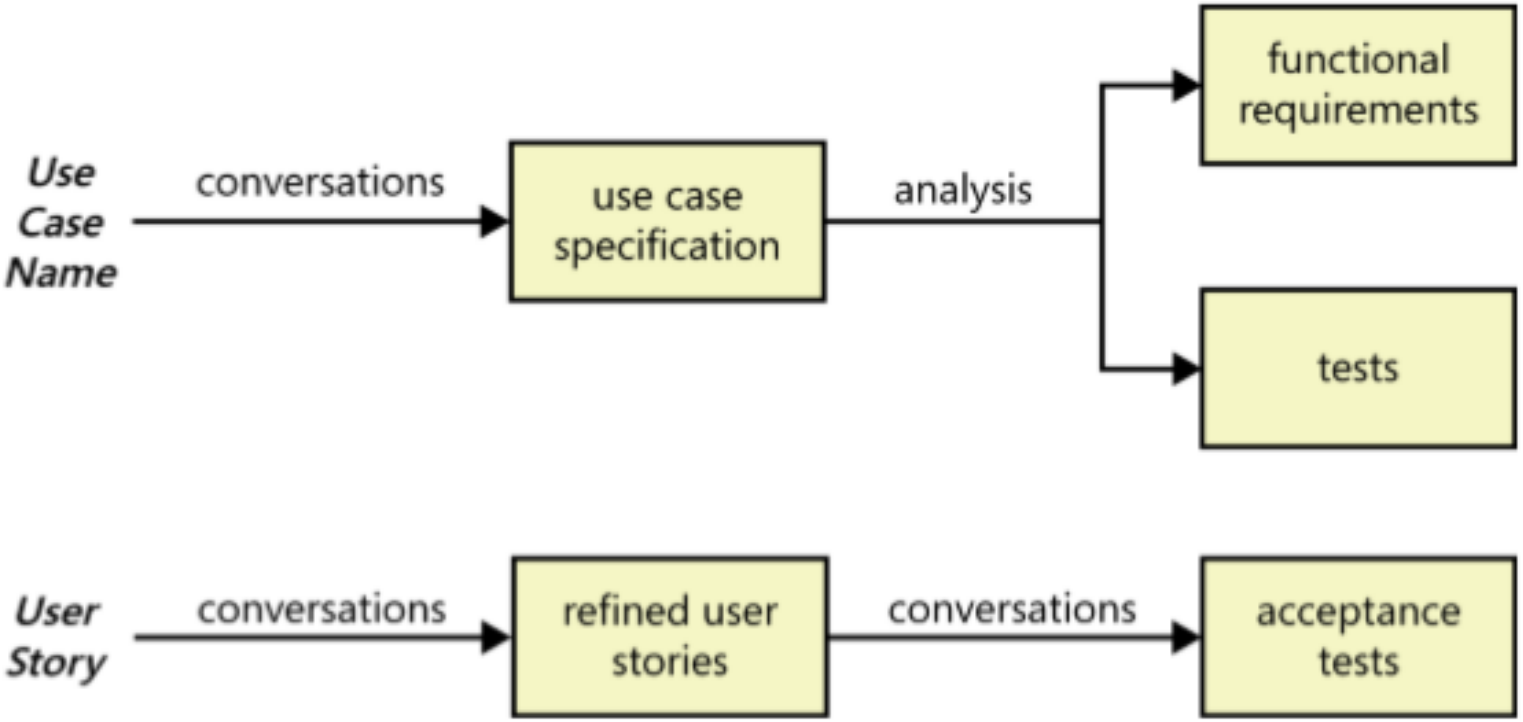


User Requirement

- The usage-centered perspective was formalized:
  - into the **use case approach** to requirements modeling (Jacobson et al. 1992; Cockburn 2001; Kulak and Guiney 2004).
  - More recently, proponents of agile development introduced the concept of a "**user story**," a concise statement that articulates a user need and serves as a starting point for conversations to flesh out the details (Cohn 2004).

User Requirement:

User Story: Use Case



Example of use cases and corresponding user stories

- Airport check-in
  - UC: Check in for a Flight
  - US: As a traveler, I want to check in for a flight so that I can fly to my destination
- Online bookstore:
  - UC: Update Customer Profile
  - US: As a customer, I want to update my customer profile so that future purchases are billed to a new credit card number.

Question of UR

- Both use cases and user stories shift from the **product-centric perspective of requirements** elicitation to discussing:

What users need to accomplish,

*in contrast to asking users**,***

hat they want the system to do?

Product Requirements

- A product requirement is a need or constraint on the software to be developed
- Functional requirement:
  - Functional requirements describe the functions that the software is to execute
  - These requirements describe **the business capabilities that the system must provide, as well as its behavior at run-time.**
  - Related to the functionality of the system.
  - Describe system services or actions.
  - **Can be used to distinguish correct output from incorrect.**

Product Requirements

- Non-functional requirements:
  - These requirements describe **the "Quality Attributes" that the system must meet in delivering functional requirements.**
  - These requirements are qualifications of the functional requirements or **of the overall product**
  - Related to how functionality is performed, not whether the end result is correct.

FR vs. NFR

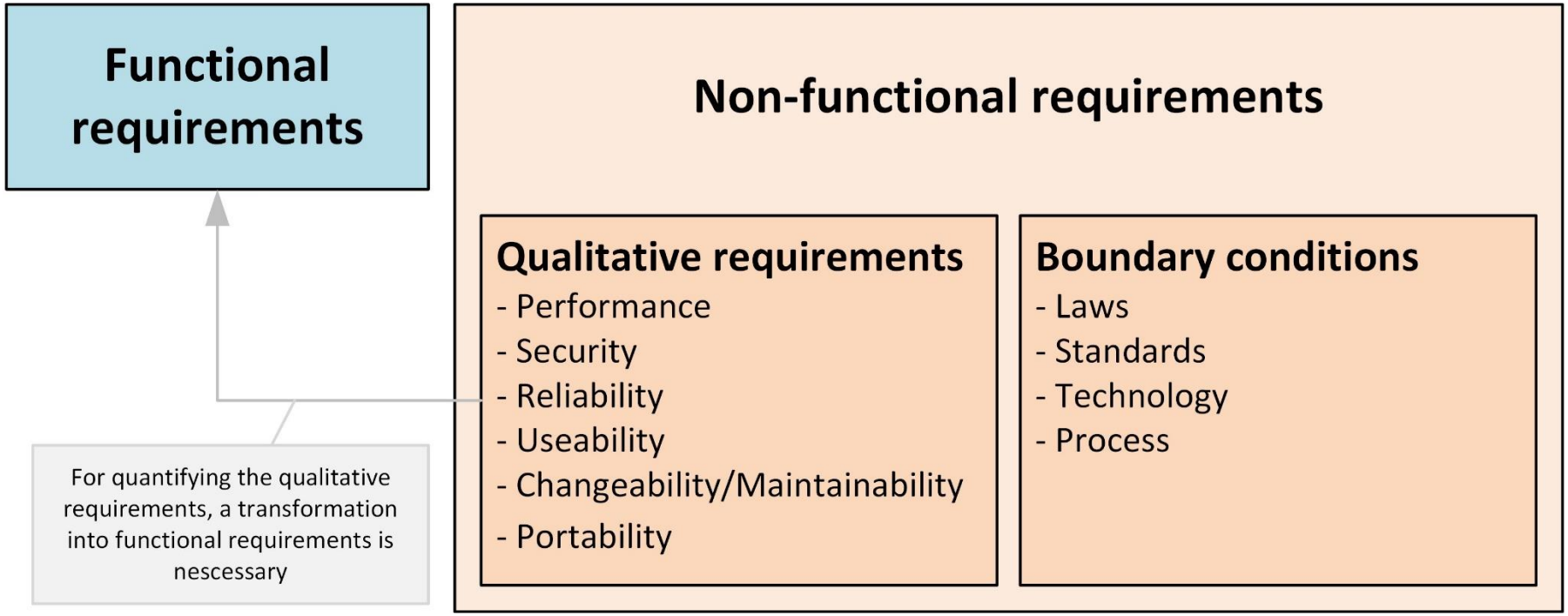
Functional requirements are usually well documented and carefully reviewed by the business stakeholders,

but ***Quality Attributes are documented in a much more succinct manner***

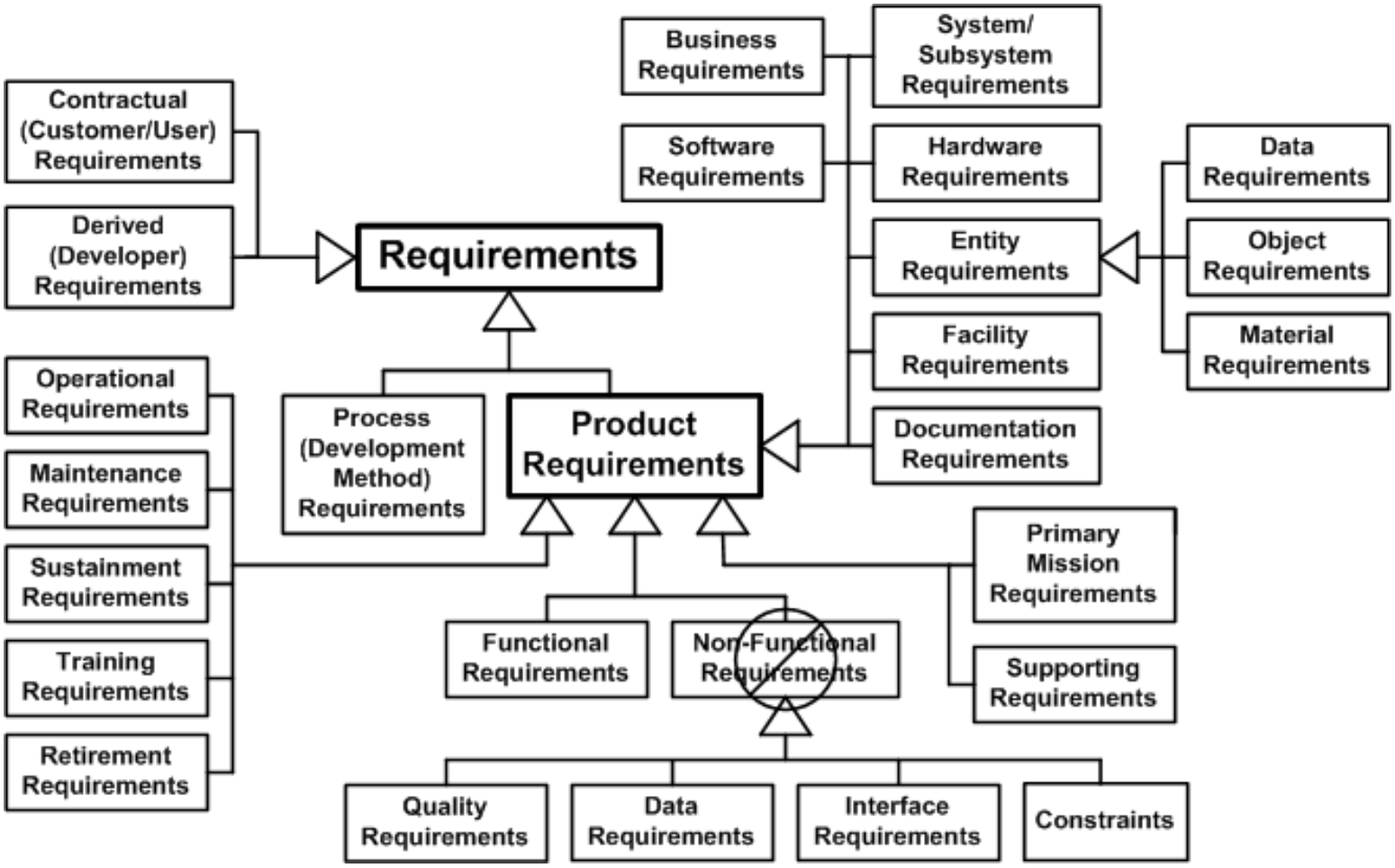
Why?

Quality Attributes drive the architecture design

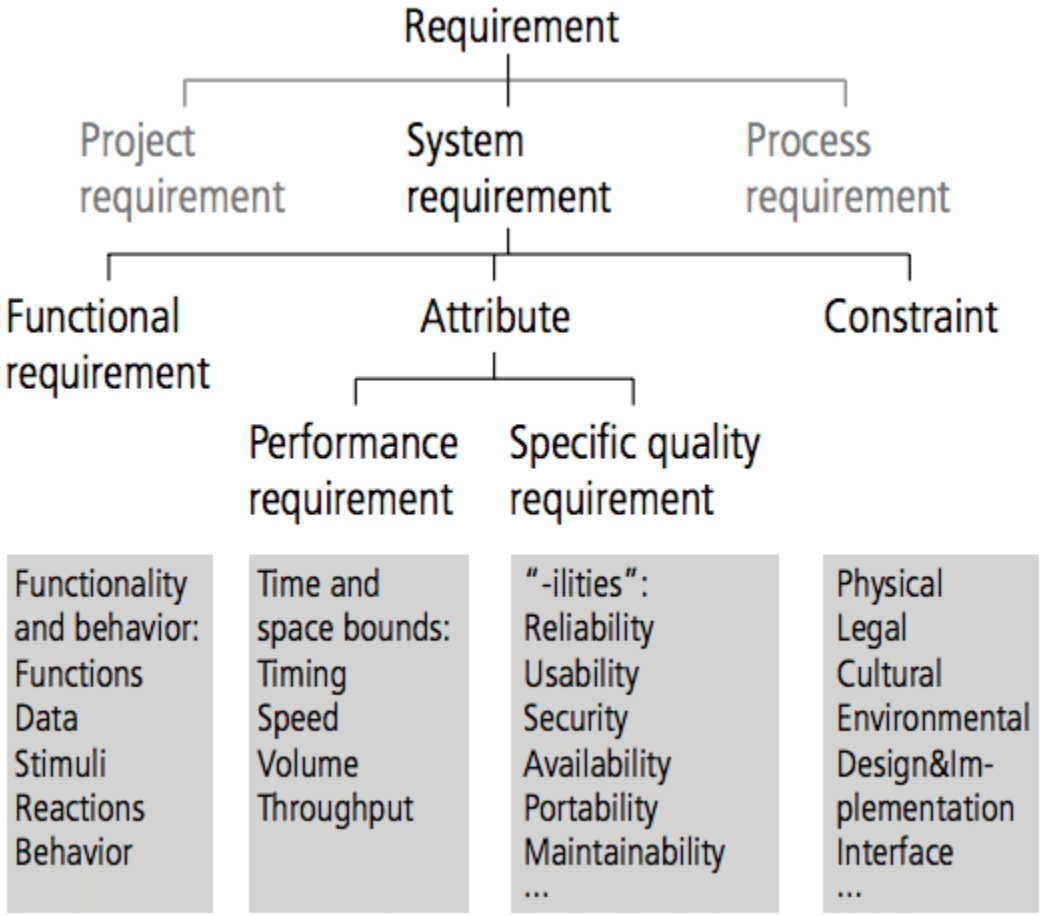
FR vs. NFR


Functional requirements
Non-functional requirements
Qualitative requirements
- Performance
- Security
- Reliability
- Useability
- Changeability/Maintainability
- Portability
Boundary conditions
- Laws
- Standards
- Technology
- Process
For quantifying the qualitative requirements, a transformation into functional requirements is nescessary

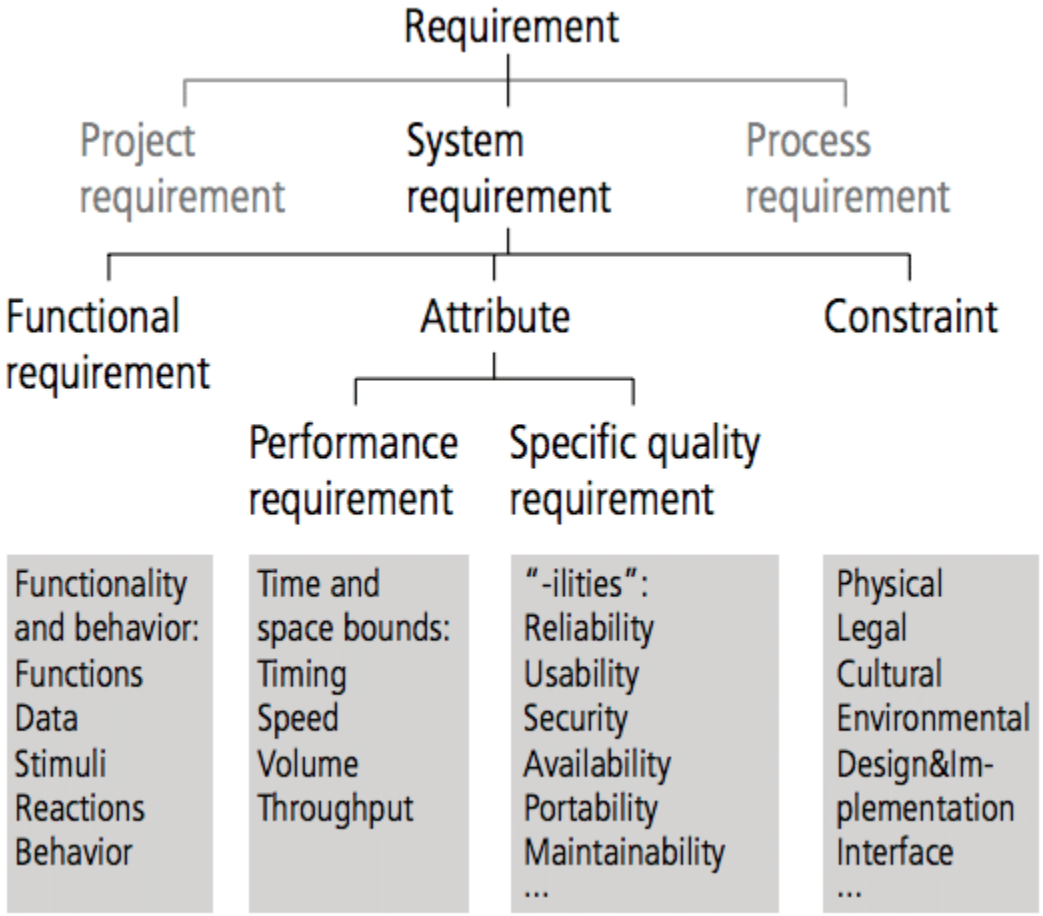
Requirements taxonomy



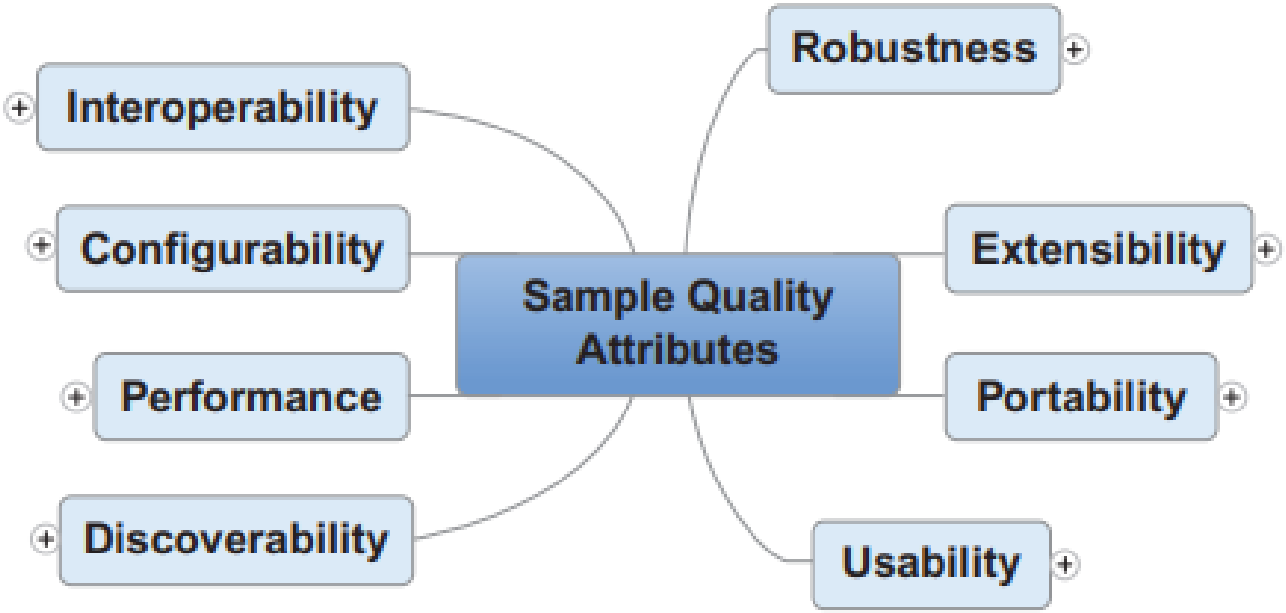
Requirements taxonomy



Requirements taxonomy

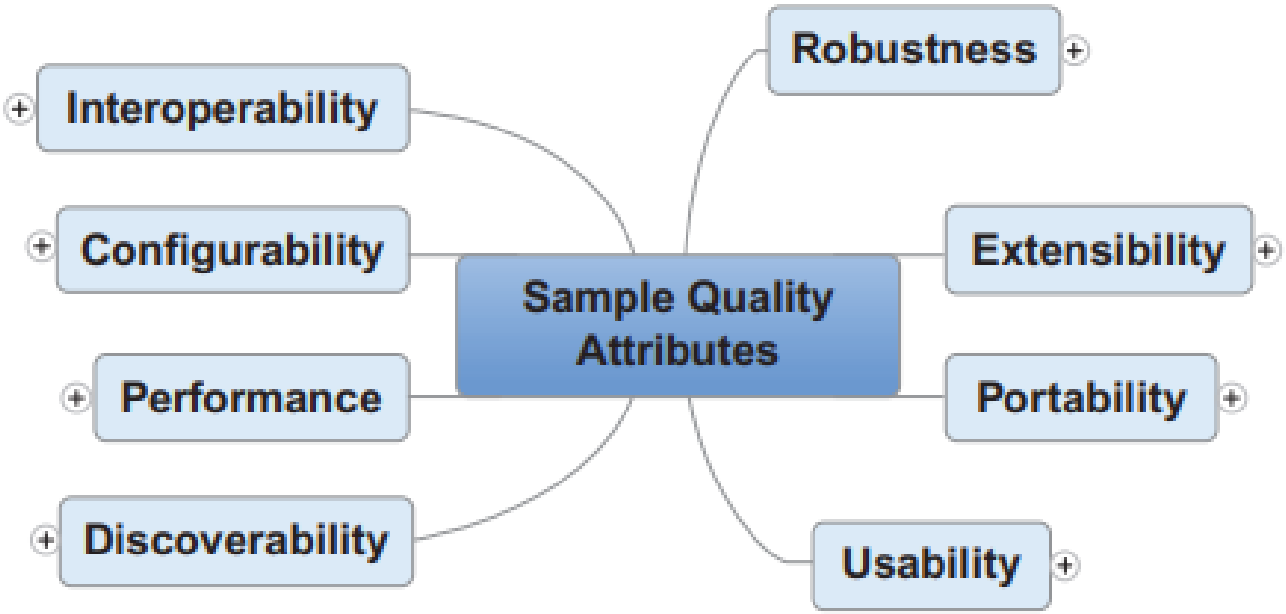


Quality Attributes



The system must be extremely user friendly ???
The system must be very fast ???

Quality Attributes



Functional Requirements define what the system must do

Quality Attributes define how it does it

Clarifying Quality Attributes is important

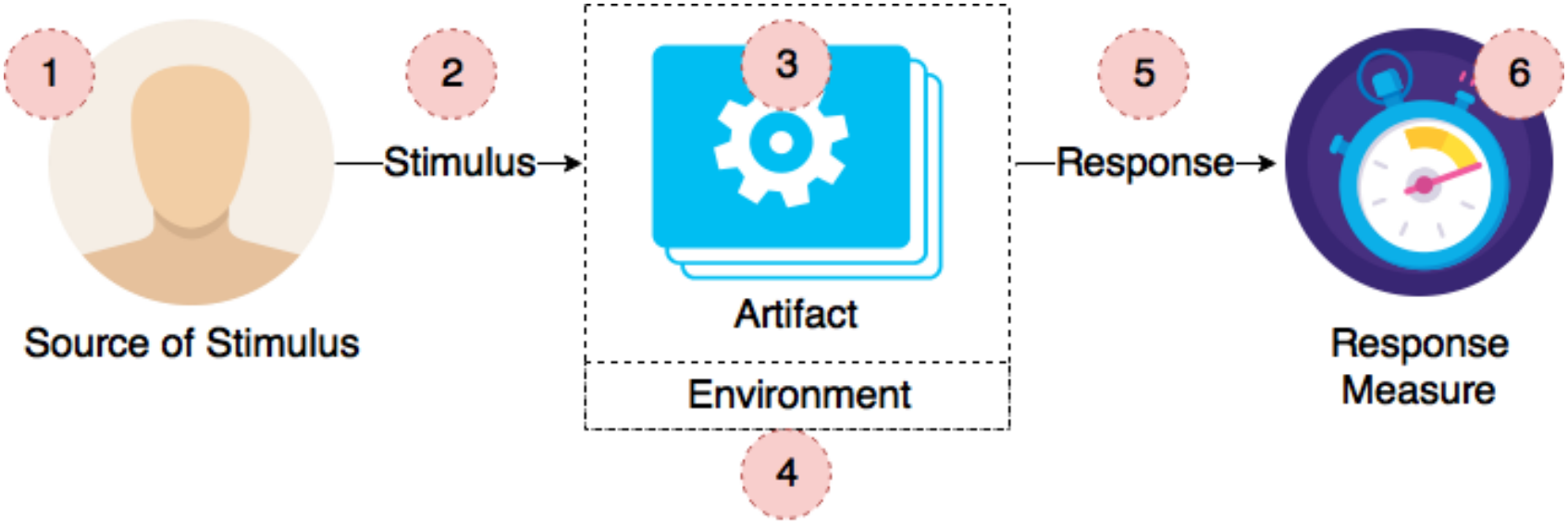
Quality Attributes

- Operational categories:
  - **Availability**
  - **Interoperability**
  - **Usability**
  - **Performance**
  - **Security**
  - Reliability
  - Deployability
  - Scalability
  - Monitorability
  - Mobility
  - Compatibility
  - Safety

Quality Attributes

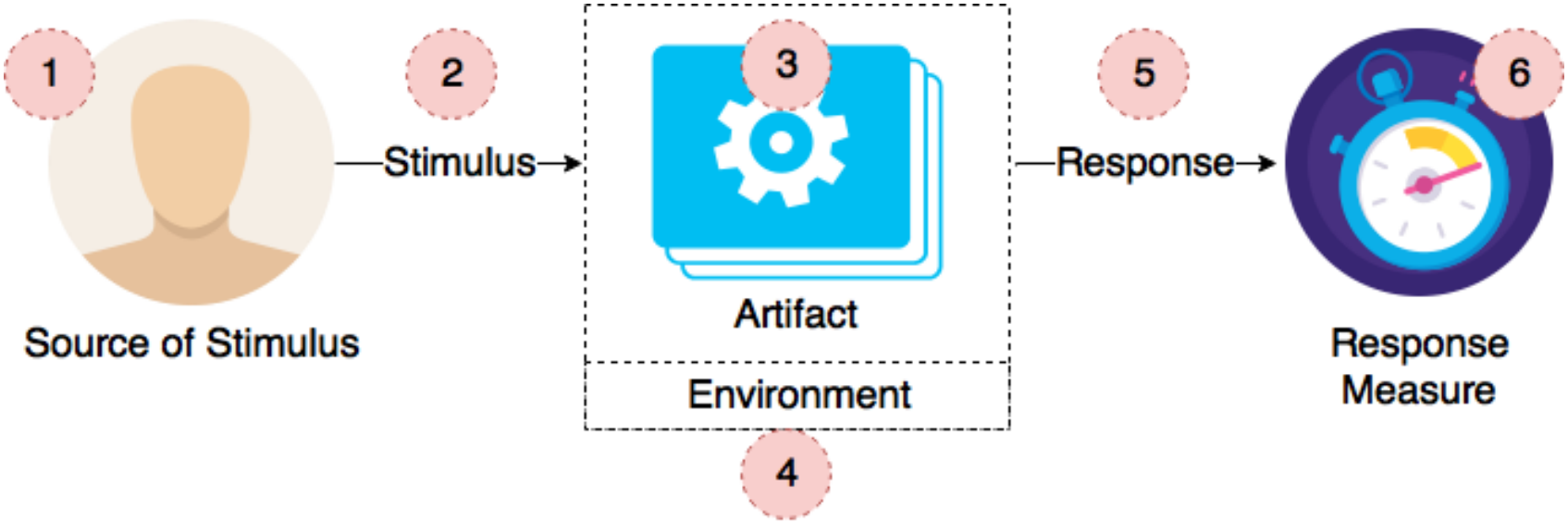
- Development categories:
  - **Modifiability**
  - Variability
  - Supportability
  - **Testability**
  - Maintainability
  - Portability
  - Localizability
  - Development distributability
  - Buildability

Quality Attribute Scenarios



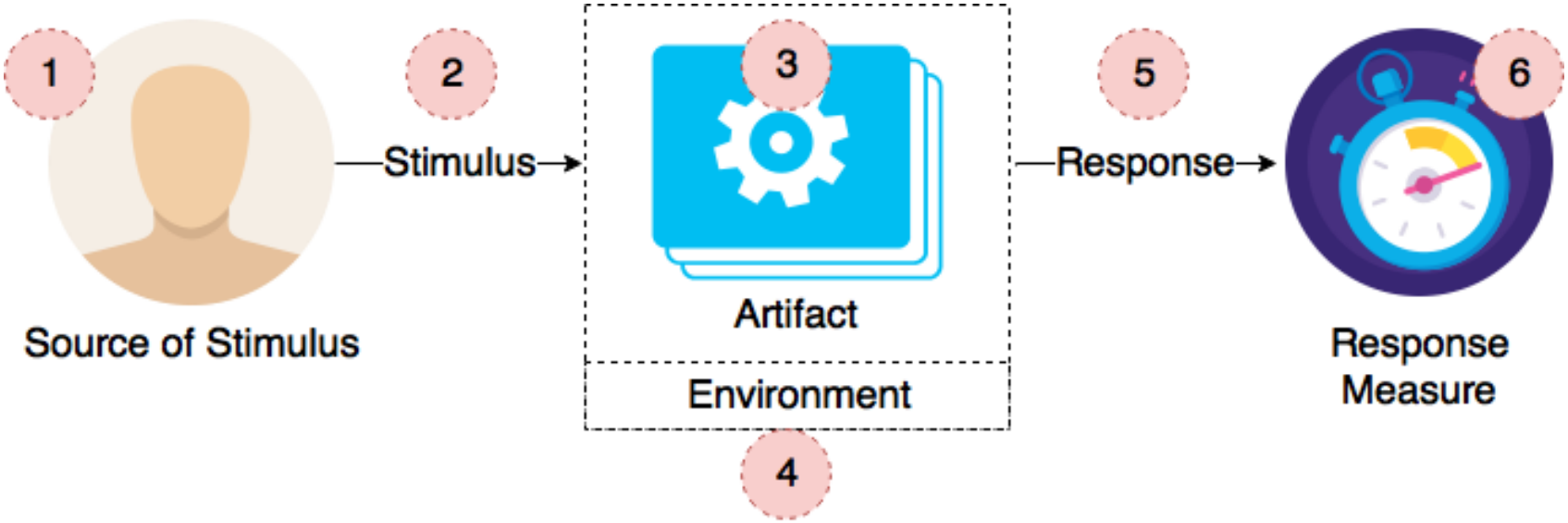
- Source of stimulus
  - An entity capable of creating stimulus (internal or external people, a computer system, etc)

Quality Attribute Scenarios



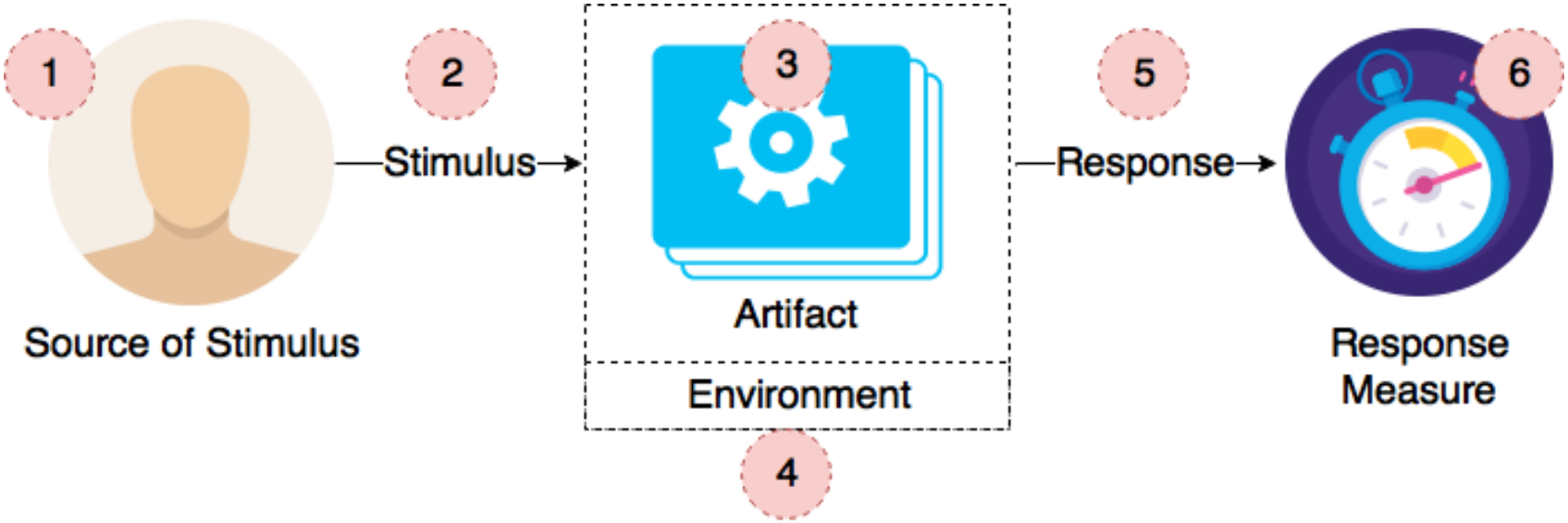
- **Stimulus:**
  - The stimulus is a condition that requires a response when it arrives at a system.

Quality Attribute Scenarios



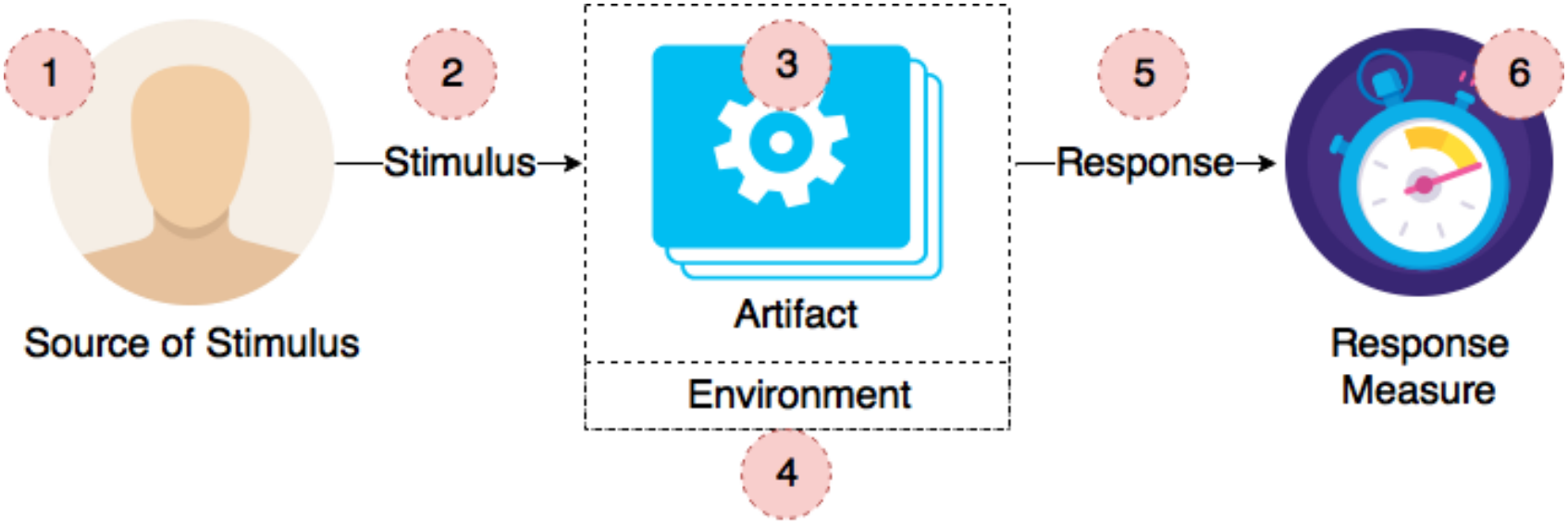
- **Environment:**
  - The environment where the stimulus occurs.
  - For instance, the system may be running in normal conditions, under heavy traffic, or with a high latency or any relevant state.

Quality Attribute Scenarios



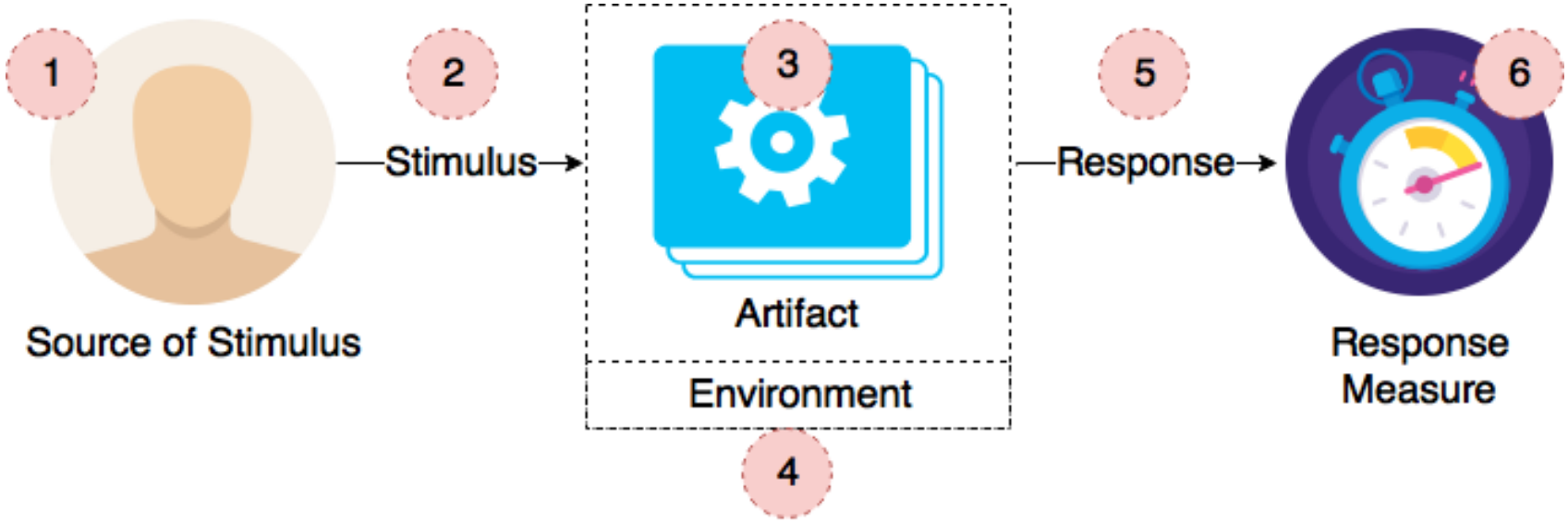
- **Artifact:**
  - The artifact that receives the stimulus.
  - This can be a component of the system, the whole system, or several systems.

Quality Attribute Scenarios



- **Response:**
  - The is the response of the artifact according to the received stimulus.

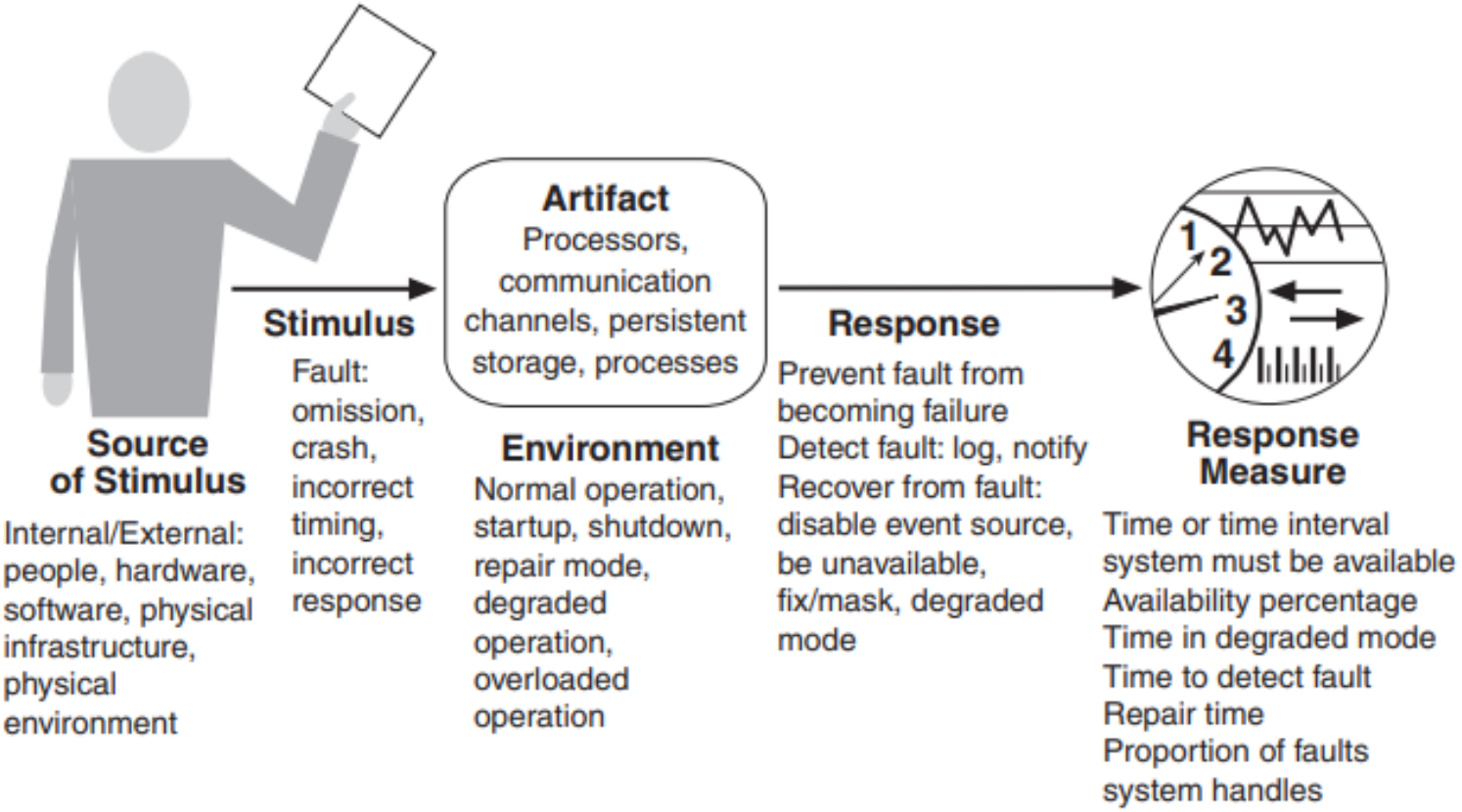
Quality Attribute Scenarios



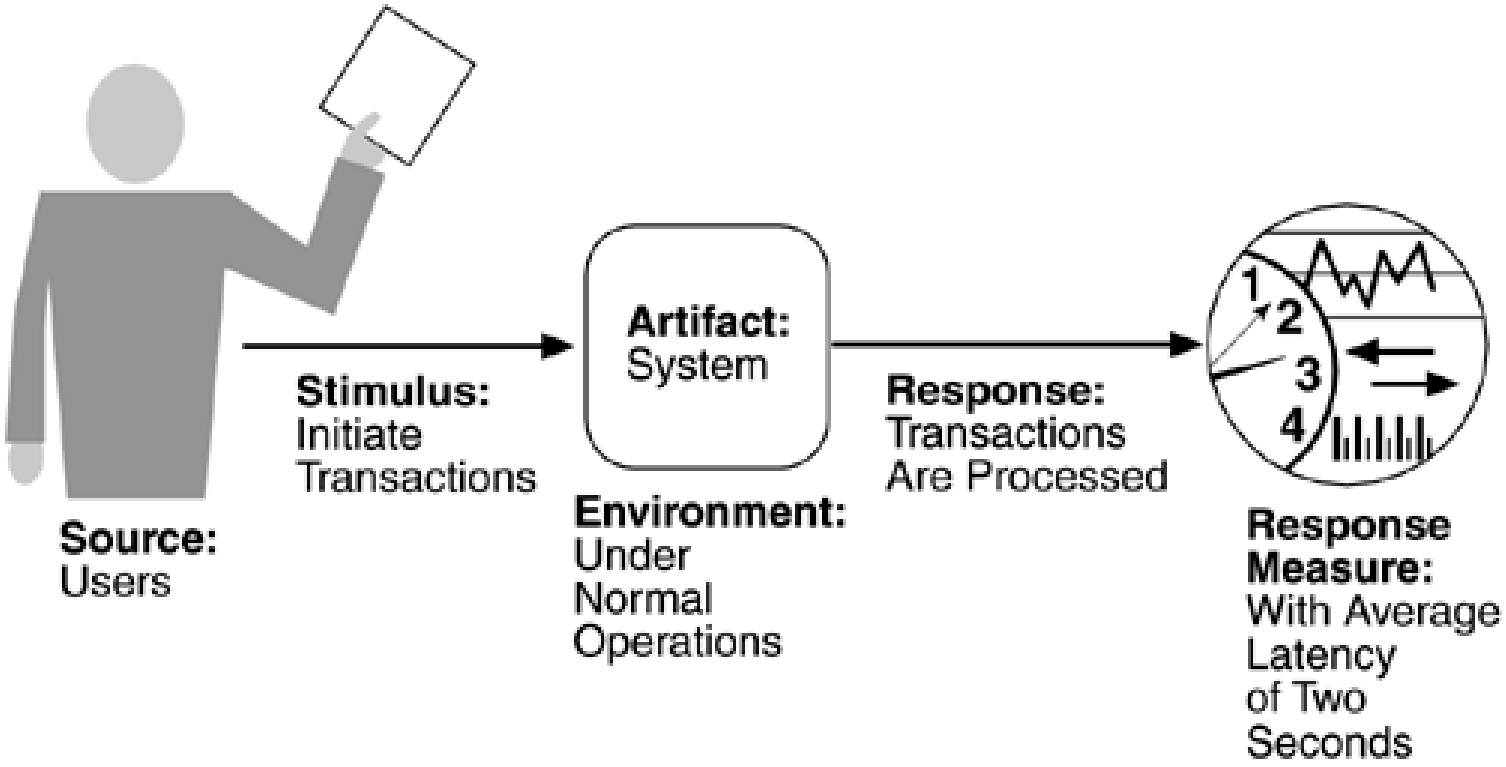
- **Response Measure:**
  - This is the measure that should be tested for the response to test if the requirement is well implemented.

A general scenario for availability

- **Software Architecture in Practice** Third Edition

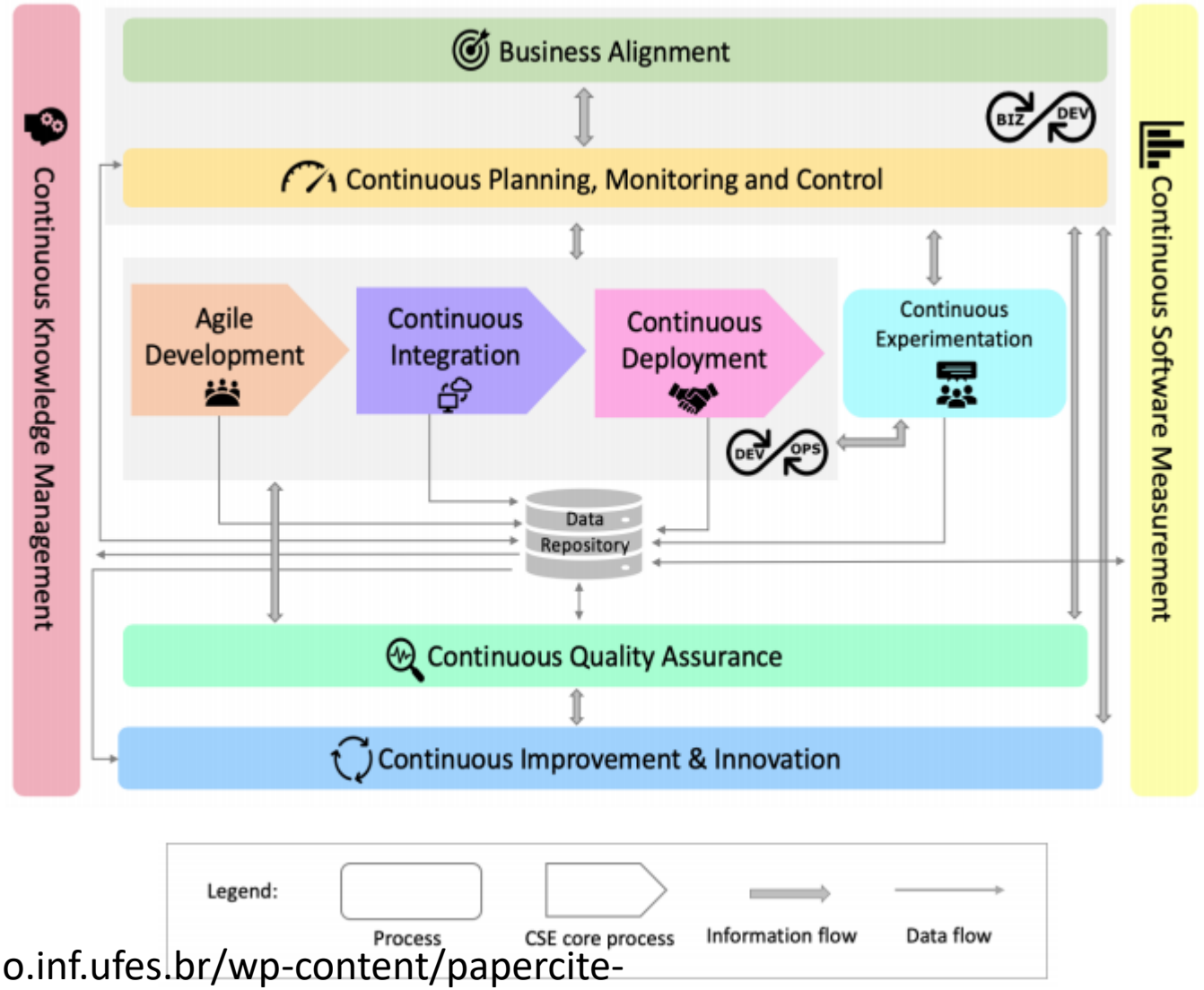


Quality Attribute Scenarios: performance



http://www.ece.ubc.ca/~matei/EECE417/BASS/ch04lev1sec4.html
https://mperry.github.io/2012/02/15/quality-attributes.html
https://www.cs.unb.ca/~wdu/cs6075w10/sa2.htm

Framework for Continuous Software Engineering



https://nemo.inf.ufes.br/wp-content/papercite-data/pdf/towards_a_framework_for_%20continuous_software_engineering_2020.pdf

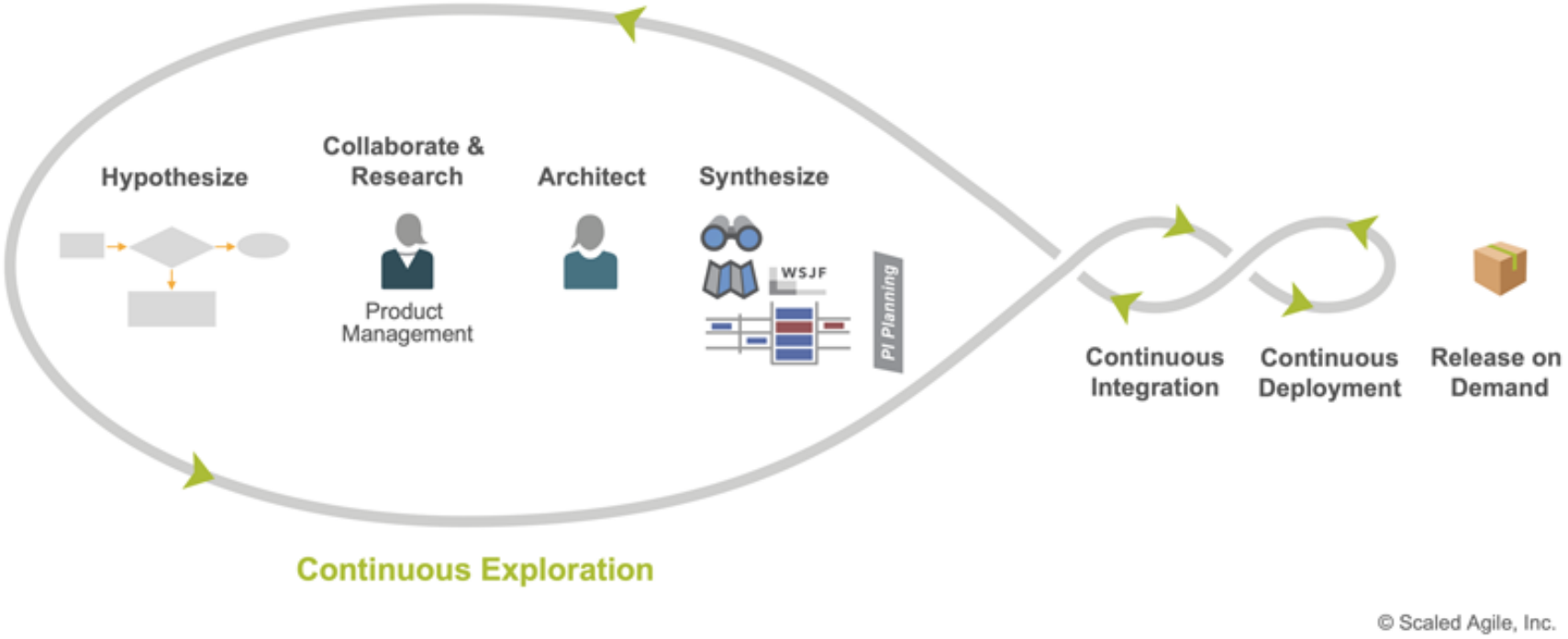
Framework for Continuous Software Engineering

- CSE involves practices and tools that aims at establishing an end-to-end flow between customer demand and the fast delivery of a product or service
- **Continuous Software Engineering** (CSE) consists of a set of practices and tools that support a holistic view of software development with the purpose of making it **faster, iterative, integrated, continuous and aligned with business**.
- It understands that the software development process is **not a sequence of discrete activities**, performed by distinct and disconnected teams. It aims to establish a continuous flow between software-related activities, taking into consideration the entire software life cycle.

BizDevOps

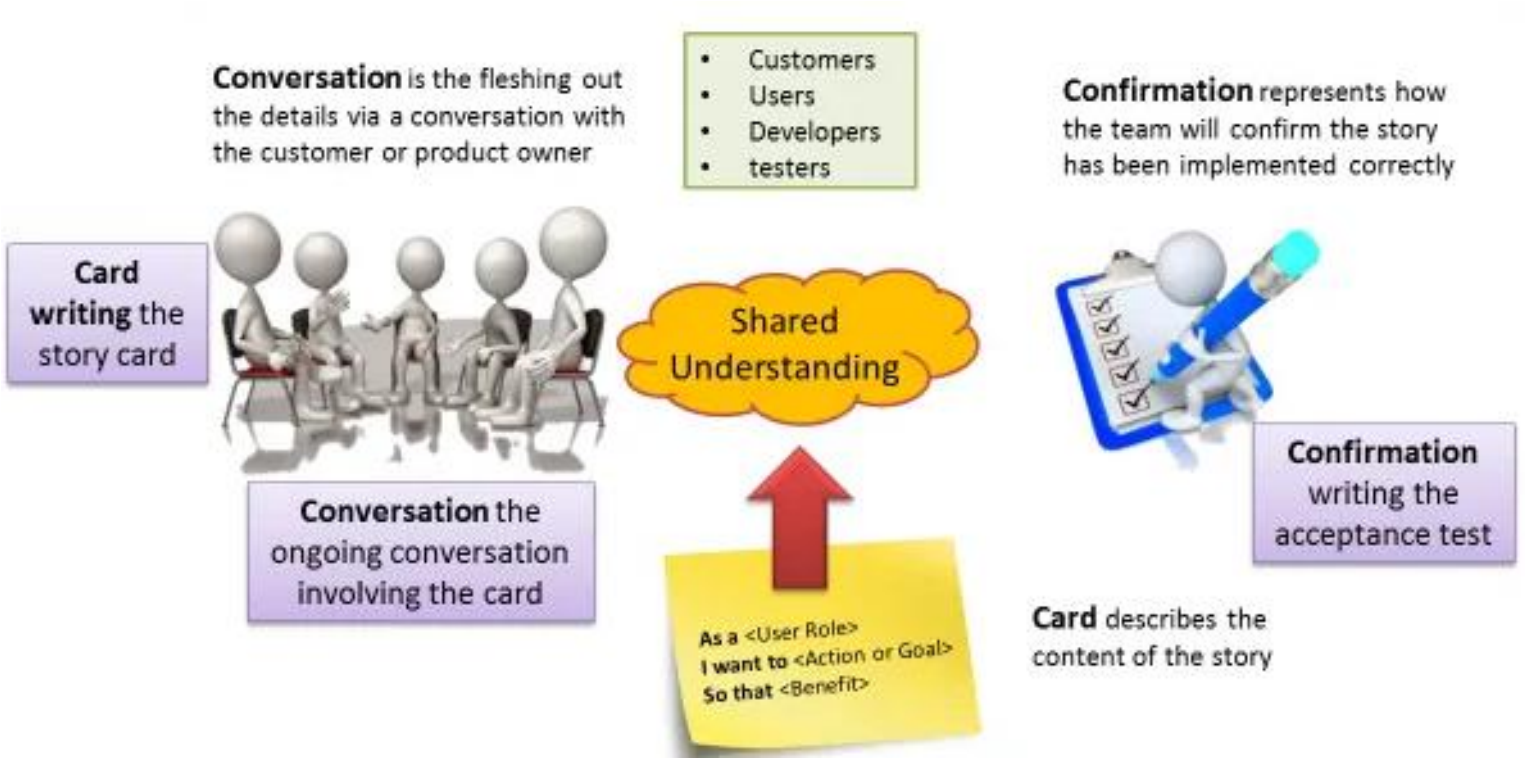

DEFINE
BIZ
ALIGN
PLAN
APPROVE
CREATE
ADAPT
DEV
CONF.
VERIFY
PACKAGE
OPS
MONITOR

Continues Exploration

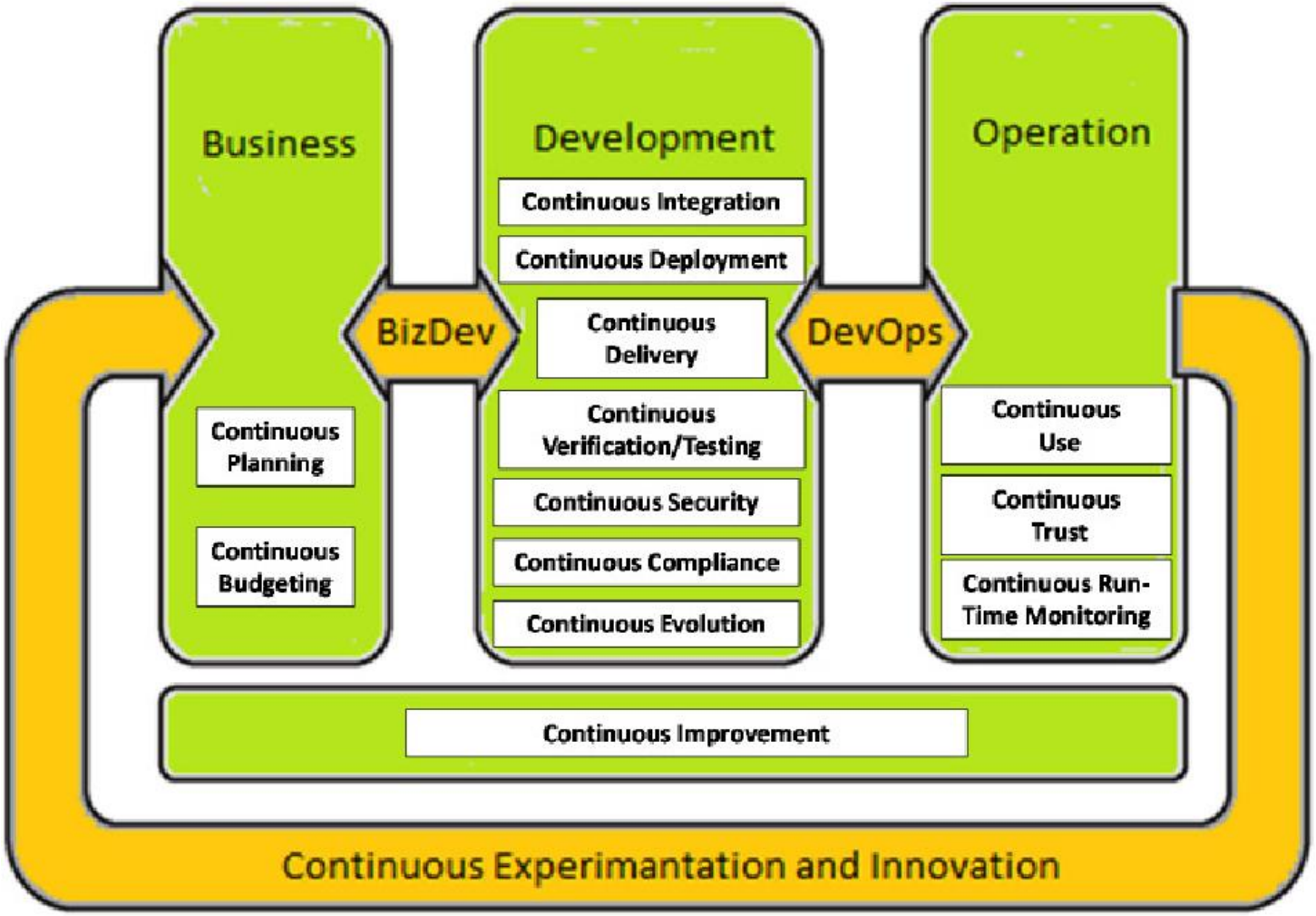


https://www.scaledagileframework.com/continuous-exploration/

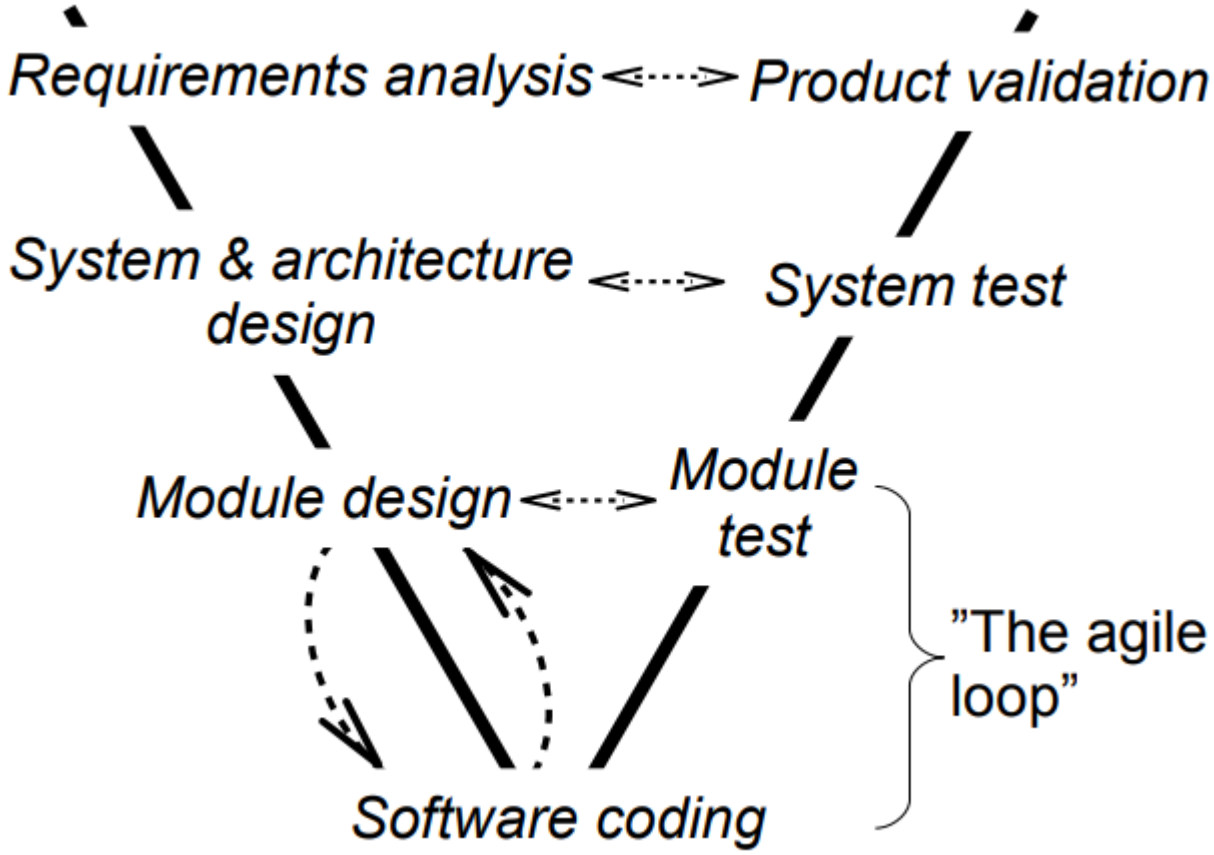
Continues Requirements Engineering



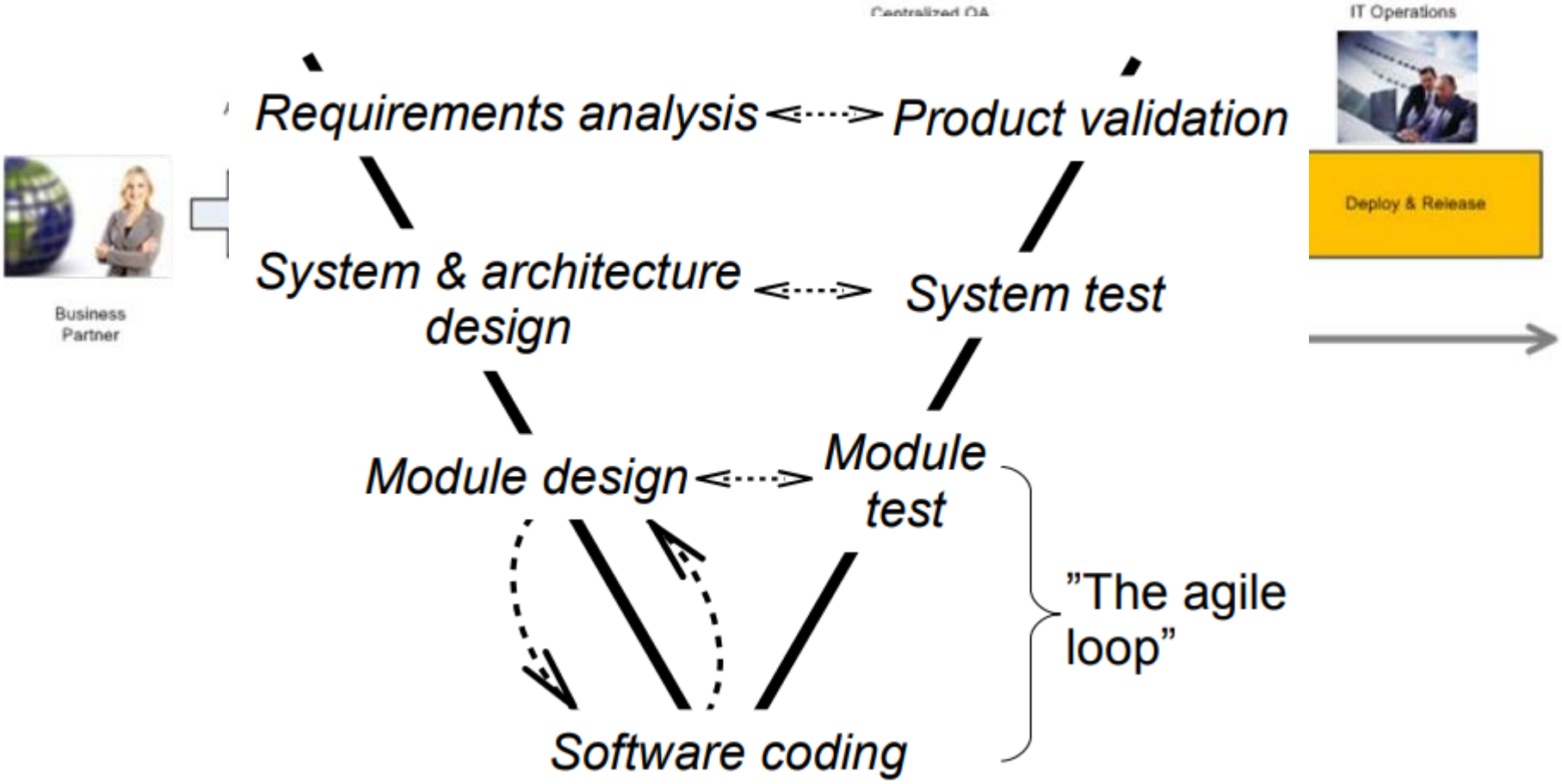
Continuous Software Engineering



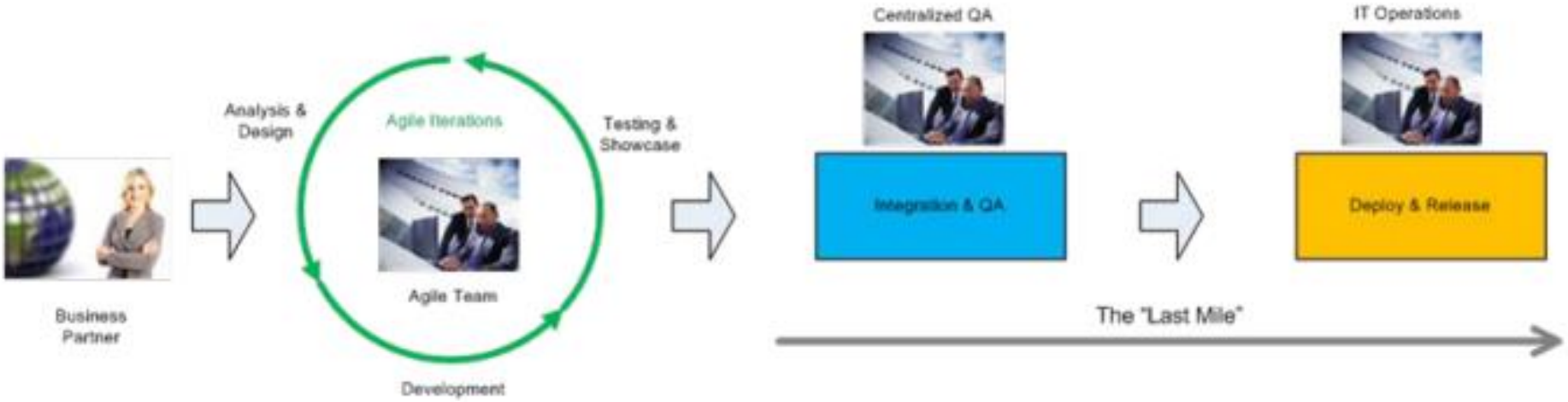
Software development in practice…



From this...


Centralized QA
IT Operations
Requirements analysis
Product validation
Deploy & Release
System & architecture design
System test
Business Partner
Module design
Module test
"The agile loop"
Software coding

From this


Business Partner
Analysis & Design
Agile Iterations
Agile Team
Development
Testing & Showcase
Centralized QA
Integration & QA
IT Operations
Deploy & Release
The "Last Mile"

To this


Business Partner
Integrated Team
Agile Iterations
Integration & QA
Deploy & Release
Continuous Delivery into Production