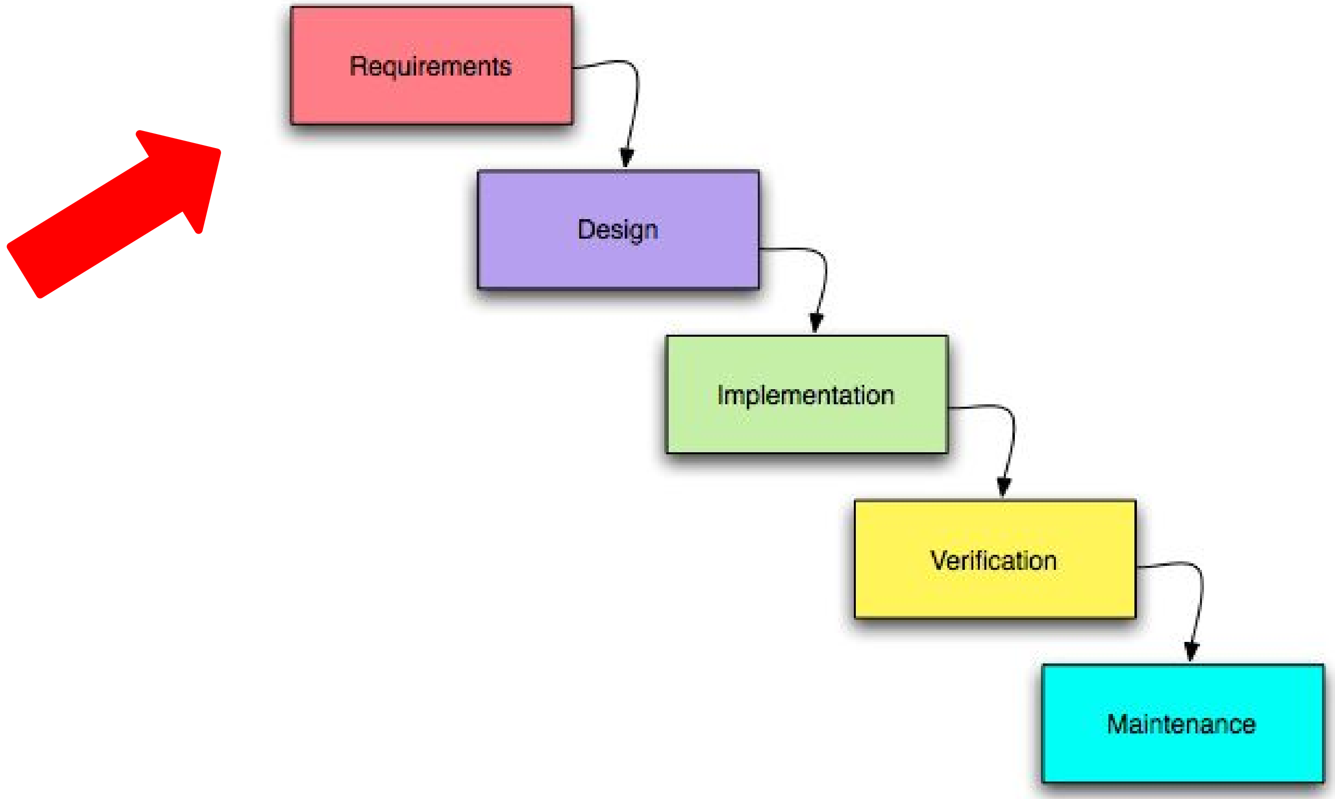
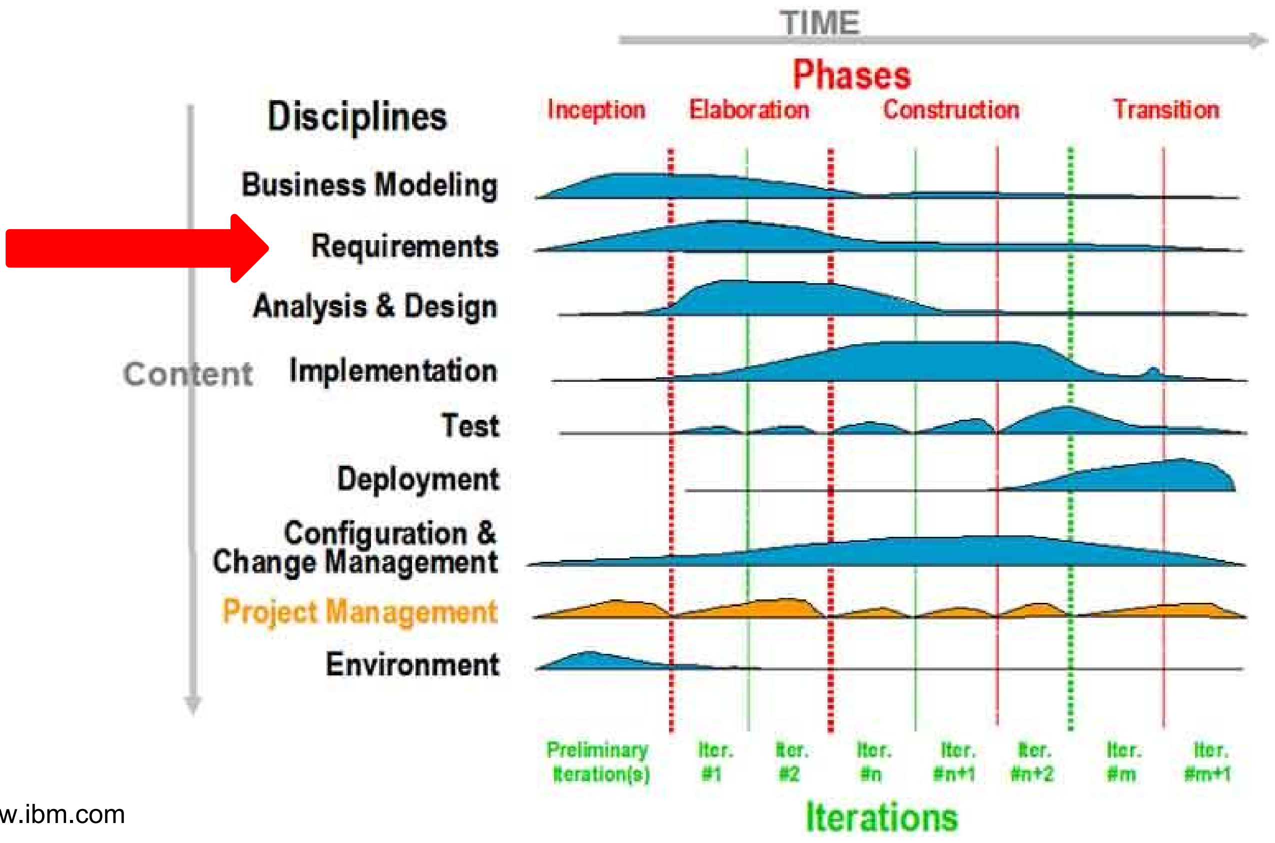
Requirements

Jana Kostičová, 29.2.2016

SDLC - Waterfall



SDLC - Rational Unified Process



Zdroj: http://www.ibm.com

Requirements engineering

Requirement

- A service, constraint or other property that the system must provide to fill the needs of the system's intended users

Engineering

- Implies that a systematic and repeatable techniques should be used

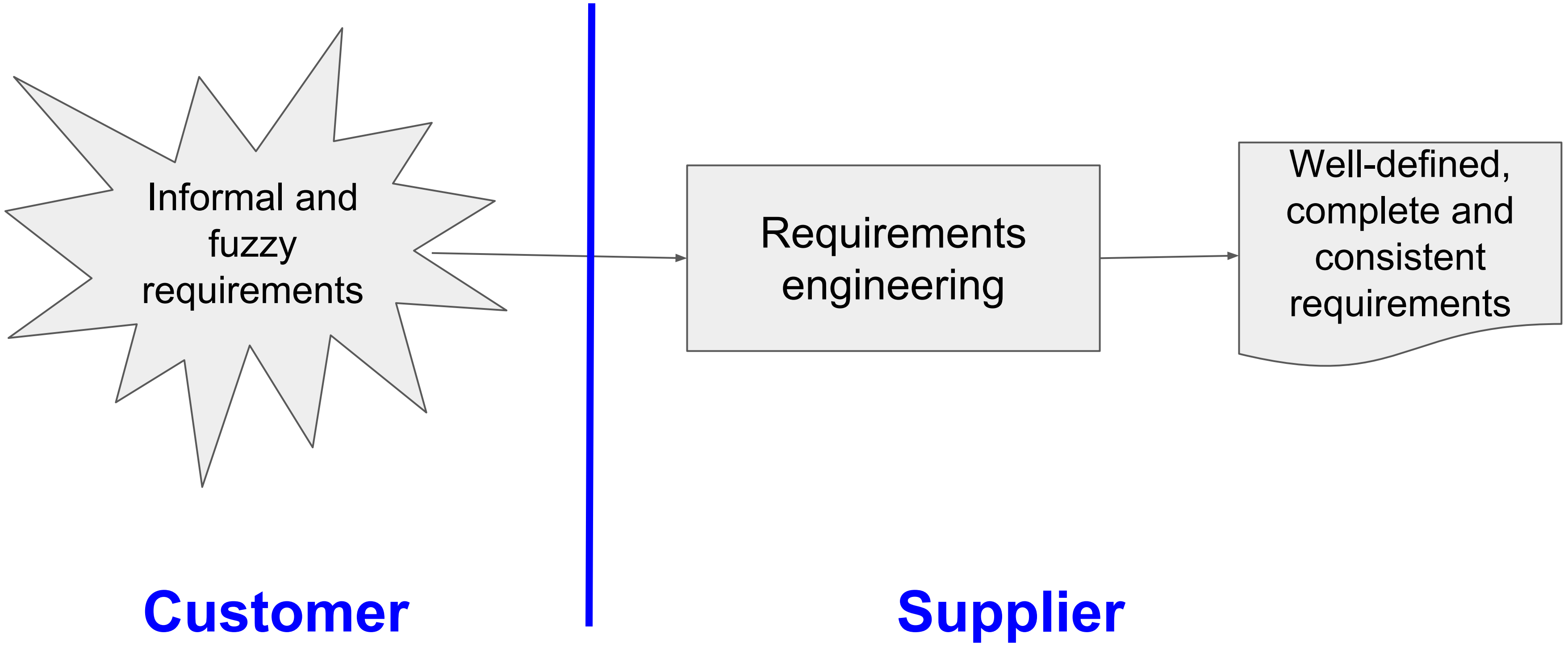
Requirements engineering

- The systematic process which covers all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system

Requirements define **WHAT** the system should do (not **HOW** it should do it)

- In practice, requirements and design overlap, e.g.,
    - GUI design as a part of requirements specification
    - Using system architecture to structure the requirements

Requirements engineering


Informal and fuzzy requirements
Requirements engineering
Well-defined, complete and consistent requirements
Customer
Supplier

Why are requirements important?

“Analysts report that as many as **71 percent of software projects that fail do so because of poor requirements management**, making it the single biggest reason for project failure - bigger than bad technology, missed deadlines or change management fiascoes”

- CIO Magazine, November 2005

Requirements form

Software requirements serve **many purposes**:
= basis for a bid for a contract
= part of the contract (scope definition - what will be delivered)
= basis for realistic estimates of time and schedules
= input for design and implementation
= basis for validation and verification
= basis for the system documentation

Software requirements are intended to a **diverse audience**:

- Customers and users for validation, contract, ...
- Systems (requirements) analysts
- Developers, programmers to implement the system
- Testers to check that the requirements have been met
- Project Managers to measure and control the project

⇒ there is **much variation** in how they are written and presented:

A software requirement may take the form of anything from a high-level, abstract statement of a service or constraint to a detailed, formal specification.

Types of requirements

Business vs. Requirements Analysis
- These two activities overlap

Business analysis:
- Identifies changes within the organisation which are necessary to achieve strategic goals of the organisation
- Changes in strategy, organisation structure, policies, processes, IT, ...

Business requirements

- Describe high-level objectives of the organization itself
- Written for management

User (stakeholder) requirements

- Describe user/stakeholder needs
- Statements in natural language plus diagrams
- Written for **stakeholders** (= any party having an interest in the system developed)

System requirements

- Describe system's functions, services and operational constraints in detail
- Technical language, diagrams, models
- Basis for designing the system (**Software Requirements Specification - SRS**)
- May be incorporated into contract

Example

Business

BR1: Reduce incorrectly processed orders by 50% by the end of next quarter
BR2: increase repeat orders from customer by 10% within six months after deployment

User/Stakeholder

UR1: Create new user account.
UR2: View order history.
UR3: Check order status.
UR4: Create new order.

System

FR1: Create new user account with the following attributes: e-mail address, first name, last name, address line 1, address line 2, city, postal code, phone number,password, timestamp.
FR2: Log in into an existing account using an e-mail address and a password.
...
NFR1: Require passwords of at least 8 characters in length containing a minimum of one non-alphabet character.
NFR2: Must run on all Java platforms including 64-bit versions
…

Types of requirements

Functional requirements

- Describes **services (functions)** the system should provide, how the system should react to particular inputs and how the system should behave in particular situations

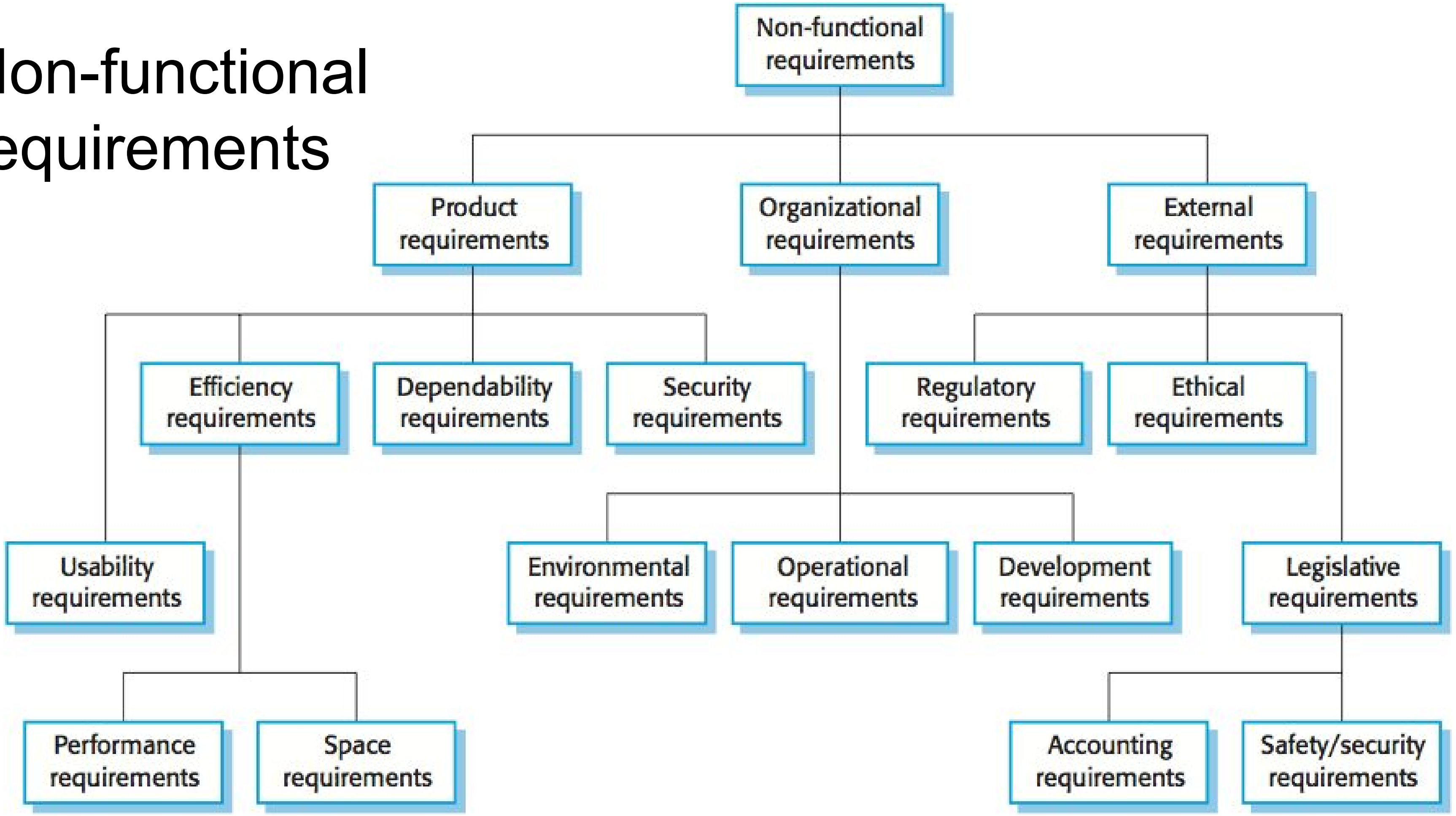
Non-functional requirements

- Describes **constraints** put on the services (functions) offered by the system
- E.g., interface requirements, GUI requirements, localization requirements

Domain requirements

- Requirements that come from the application domain of the system and that reflect characteristics of that domain

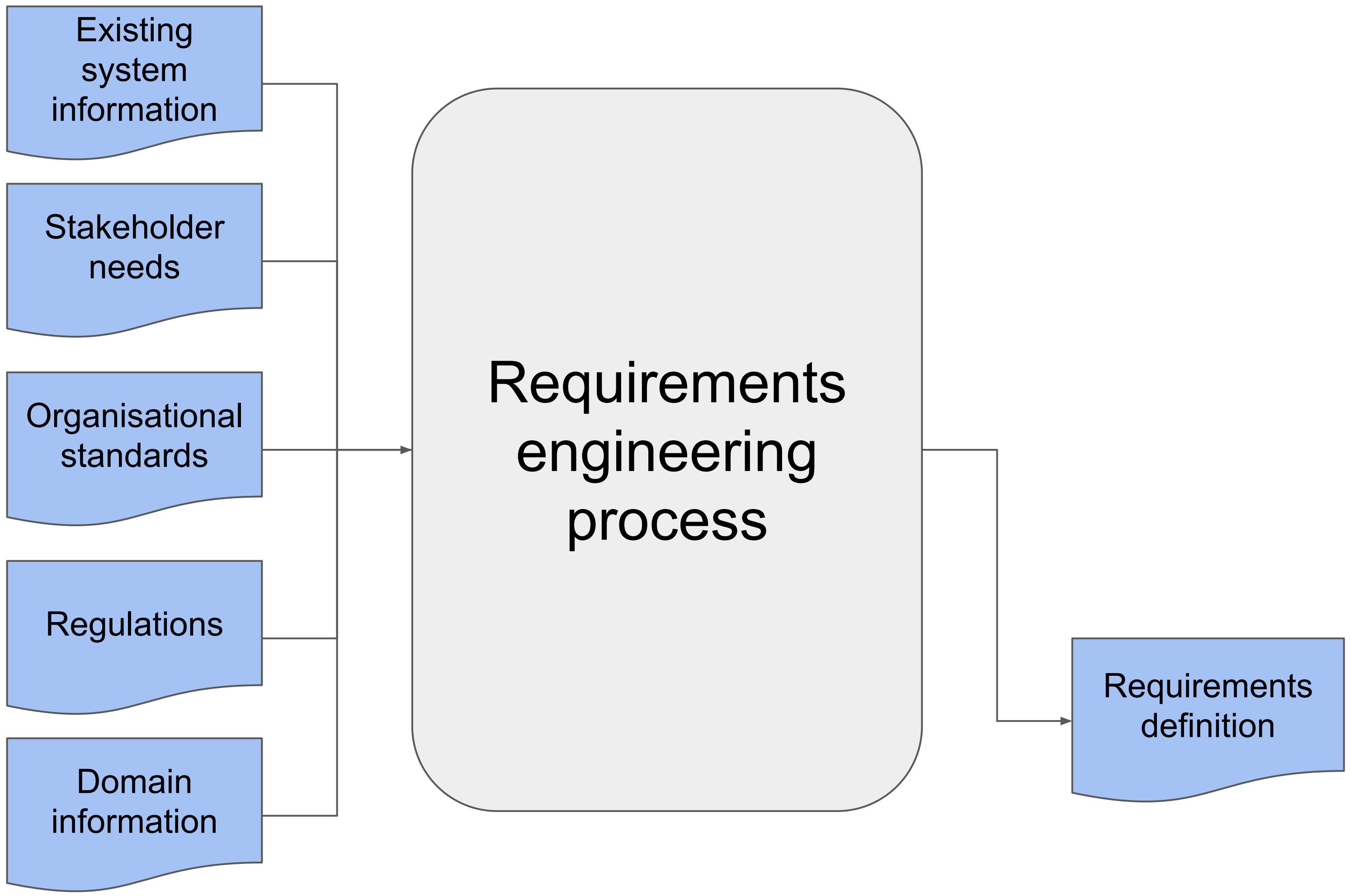
Non-functional requirements



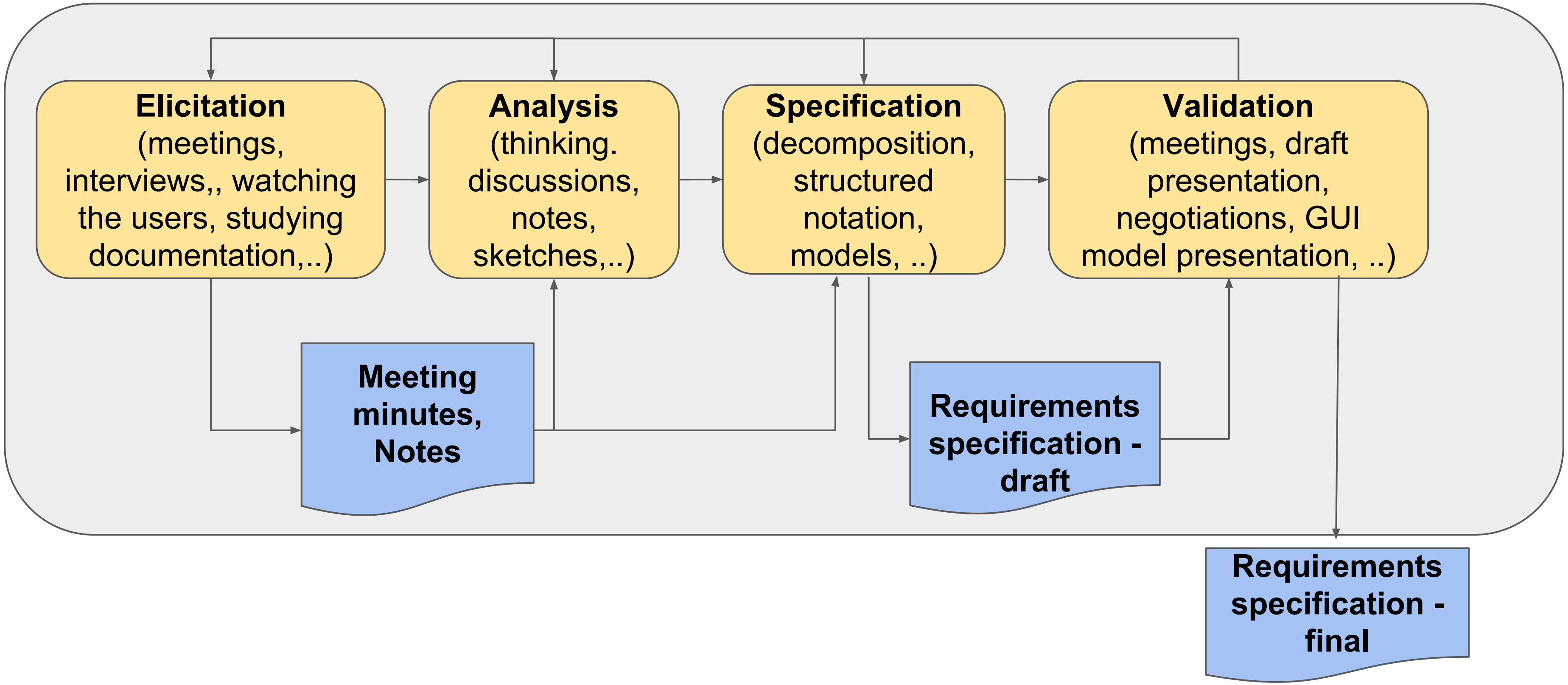
More examples

- Non-functional requirements:
    - PRODUCT REQUIREMENT: The user interface should be implemented as simple HTML without frames or Java applets.
    - ORGANIZATIONAL REQUIREMENT: The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-14.
    - EXTERNAL REQUIREMENT: The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

- Domain requirement:
    - The deceleration of the train shall be computed as: D (train) = D (control) + D (gradient), where D (gradient) is 9.81ms2 * compensated gradient/alpha and the values of 9.81ms2 /alpha are known for different types of train.


Existing system information
Stakeholder needs
Organisational standards
Regulations
Domain information
Requirements engineering process
Requirements definition

Detailed view


Elicitation
(meetings, interviews,, watching the users, studying documentation,..)
Analysis
(thinking. discussions, notes, sketches,..)
Specification
(decomposition, structured notation, models, ..)
Validation
(meetings, draft presentation, negotiations, GUI model presentation, ..)
Meeting minutes, Notes
Requirements specification - draft
Requirements specification - final

Software Requirements Specification

Typically structured text supported by figures/models/diagrams

- Use Case model and Conceptual model (UML)
- GUI model (mock-ups)

Other approaches

- "Victorian novel"
  - Massive narrative sequential description, seldom used today
- Flat catalogue of requirements
  - Often used, not optimal
- UML only
  - Insufficient - UML does not provide means to define non-functional requirements
  - Customer may have poor knowledge of
- CASE tool (e.g., Enterprise Architect)
  - More difficult to create and maintain the specification
  - Provides complete system description
- Collaborative Software, Wiki (e.g., Atlassian Confluence)

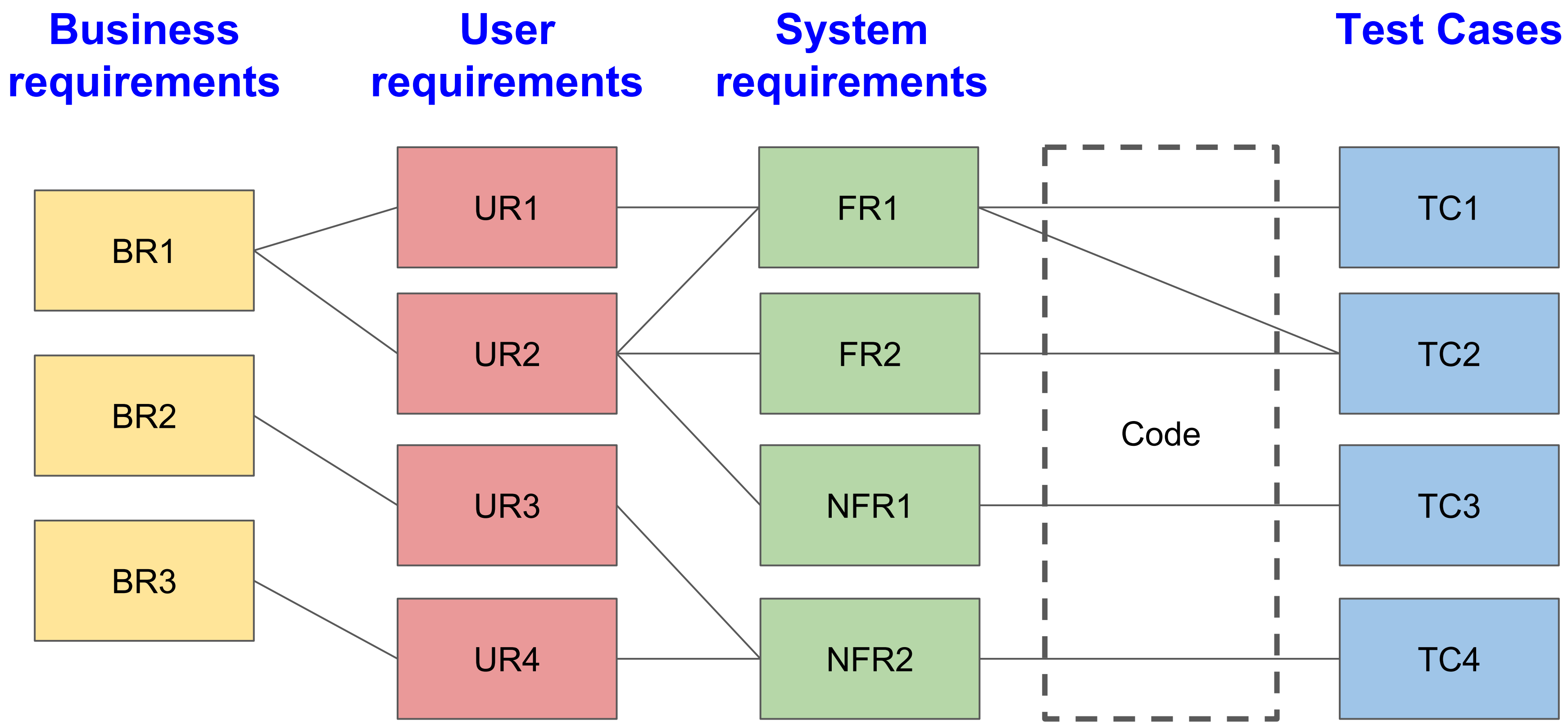
IEEE 830 standard - SRS form

- Title
- Table of Contents
- Introduction
  - Purpose, Scope, Definitions, Acronyms & Abbreviations, References
- Overall description
  - Product Perspective, Product Functions, User Characteristics, Constraints, Assumptions and Dependencies
- Specific Requirements
  - External Interfaces, Functions, Performance Requirements, Logical Database Requirements, Design Constraints, Software System Quality Attributes, Object Oriented Models
- Appendices
- Index

IEEE 830 standard - “good” SRS

- Correct
  - Correctly describes the system’s behavior
- Unambiguous
  - Every requirement has only one interpretation
- Complete
  - Completely describes the system’s expected behavior and feature set
- Consistent
  - Requirements do not contradict each other
- Ranked
  - Each requirement has an identifier to indicate either its importance or stability
- Verifiable
  - Requirements are testable
- Modifiable
  - Requirements are easy to modify or change
- Traceable
  - The origin of each requirement is known and documented

Requirements traceability



Requirements problems

- Users don't understand what they want
- All requirements are critical, no priority is defined
- Business requirements are not clearly defined
- Users won't commit to a set of written requirements
- Users change requirements after the cost and schedule have been fixed
- Communication with users is slow
- Users often do not participate in reviews
- Users don't understand the development process
- Technical personnel and end users may have different vocabularies
- Engineers and developers may try to make the requirements fit an existing system or model, rather than develop a system specific to the needs of the client
- Analysis may be often carried out by engineers or programmers, rather than personnel with the people skills and the domain knowledge to understand a client's needs properly

Best practises

- Take into account all types of requirements
    - Use sweng books or other sources to get a comprehensive survey of requirement types
    - Especially non-functional requirements might be difficult to discover
- Gather requirements from all stakeholders
- Avoid grey zones
    - Customer wants everything what is not explicitly “NO”
    - Supplier delivers only things that are explicitly “YES”
    - Document key non-requirements!
- Document requirements accurately and thoroughly
- Avoid ambiguities
    - Natural language is ambiguous, employ independent (internal) reviewers
- Validate requirements with all stakeholders
- Identify risks
    - Project manager is in charge of risk management

Further reading

- Ian Sommerville: Software Engineering (10th edition)
- IEEE STANDARD 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
- Karl Weigers: Software Requirements (2nd Edition)