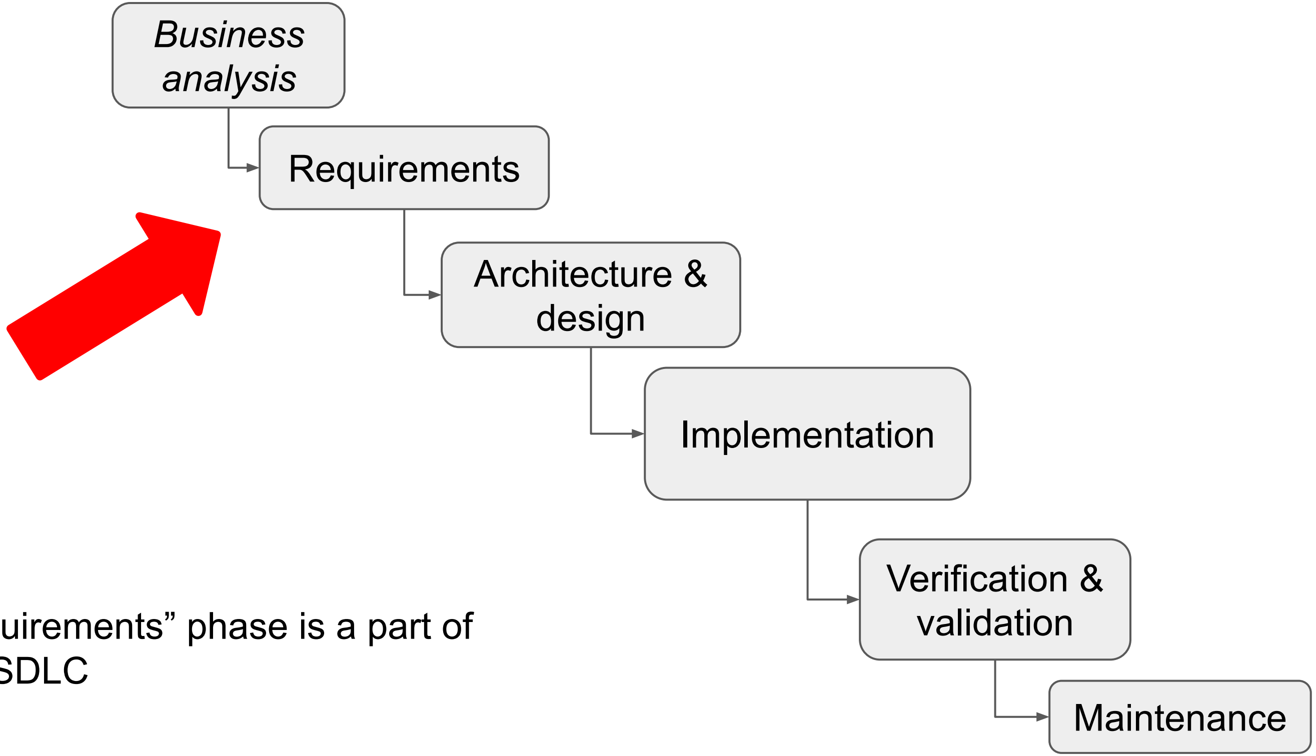
Requirements

Princípy tvorby softvéru 2, FMFI UK

Jana Kostičová, 02.10.2023, 09.10.2023

SDLC - Waterfall



- “Requirements” phase is a part of any SDLC

Terminology

- “Requirements”
- “Requirements analysis”
- “Requirements engineering”
- “Analysis” only (if the context is clear)

Informally also

- “IT business analysis” (often overlapping with “business analysis”)

Why requirements engineering?

Requirement

- A function, constraint or other property that the system must provide to fill the stakeholder needs

Engineering

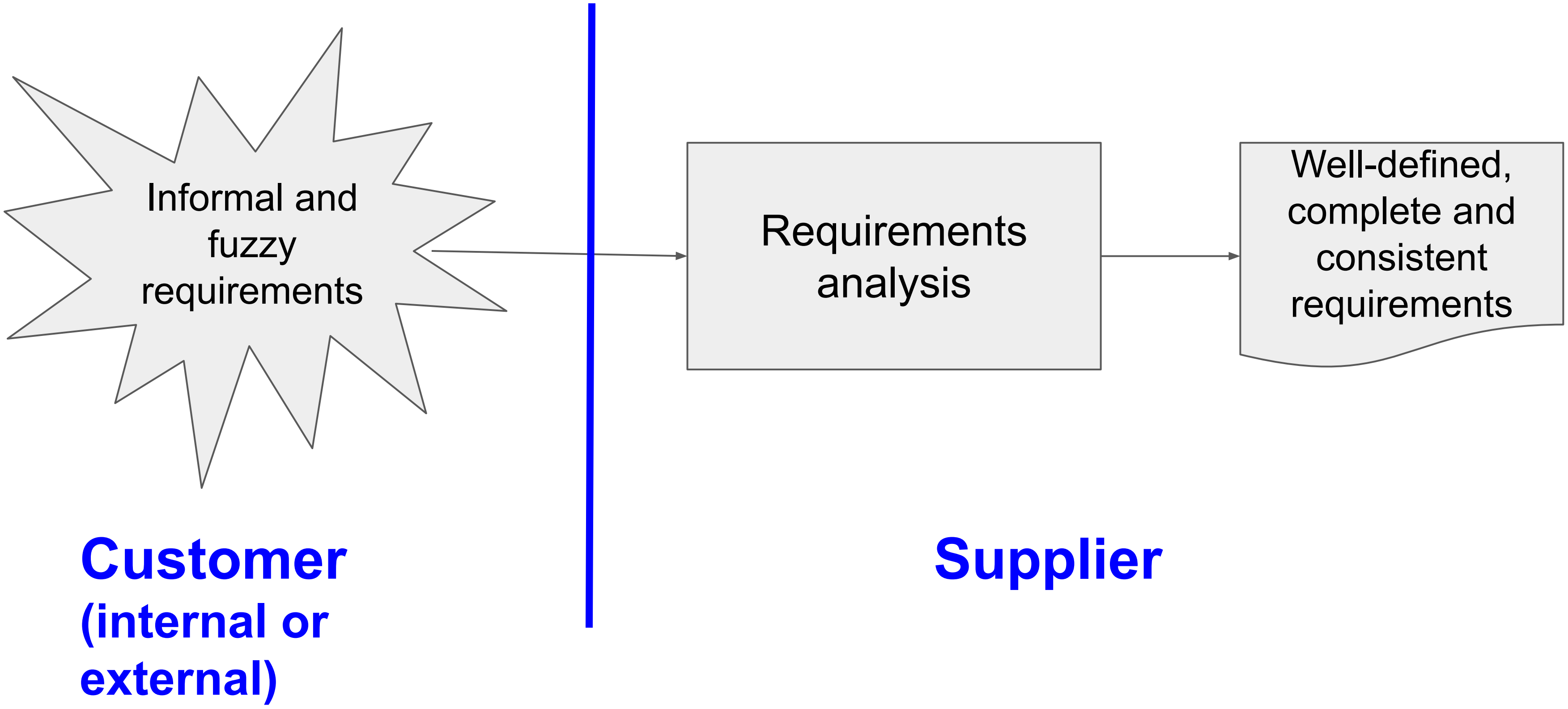
- Implies that a systematic and repeatable techniques should be used

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

○

Big picture



Informal and fuzzy requirements

Requirements analysis

Well-defined, complete and consistent requirements

Customer (internal or external)

Supplier

Why are requirements important?

75% of all IT projects fail due to errors in the set-up phase. According to the study, the most common reasons for the failure of IT projects are **unclear or inadequate requirements**, incorrect time and budget planning, and inadequate communication between project participants.

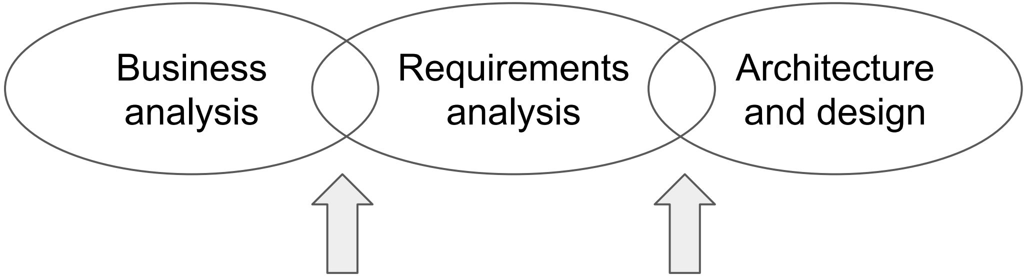
BITKOM e.V. (Germany digital association), 2021

Requirements vs other phases

Requirements define **WHAT** the system should do

- not **WHY** it should be developed
- not **HOW** it should do it

In practice, “requirements” phase **overlaps** with neighboring phases:

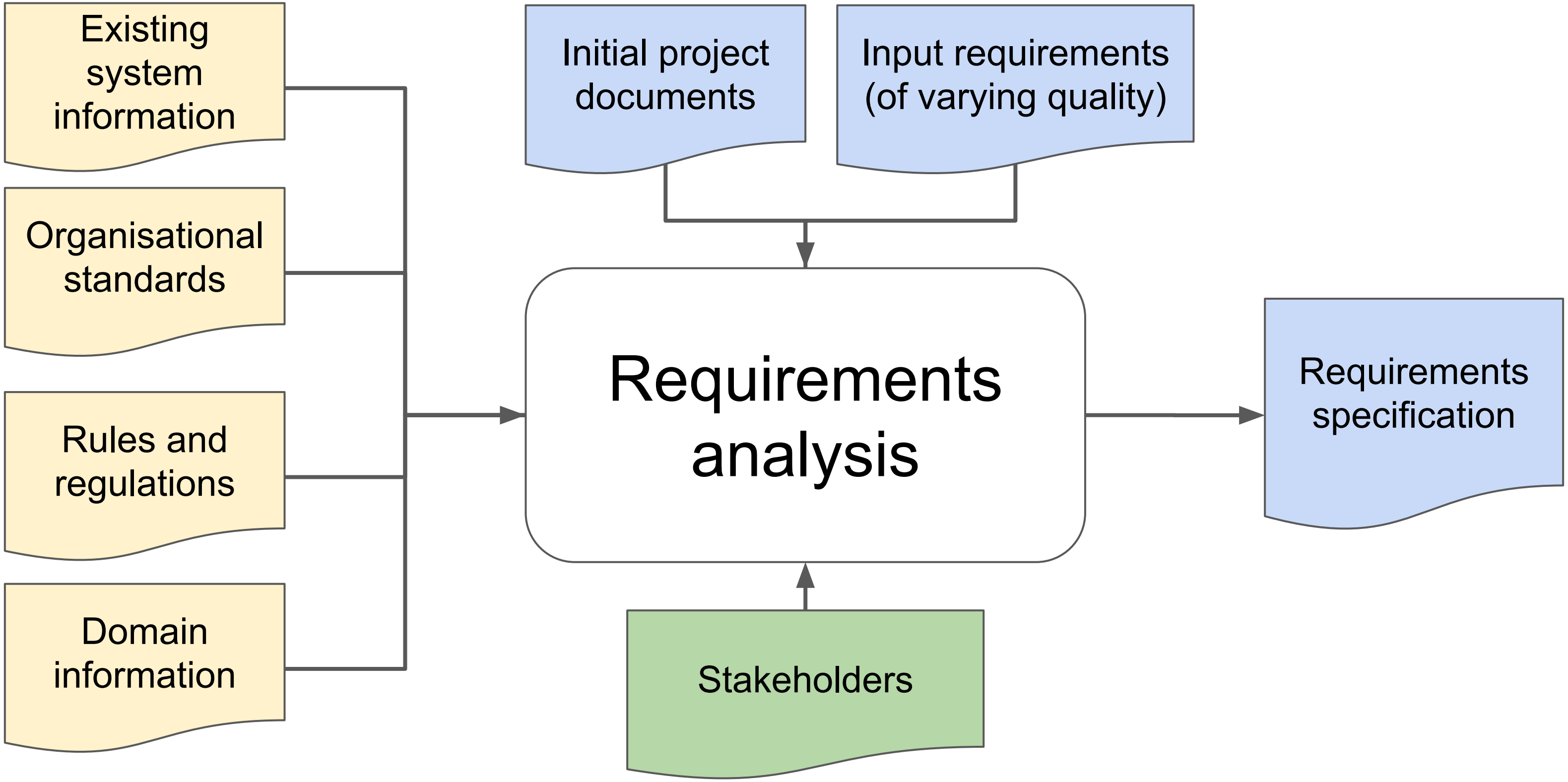


Business requirements

Examples:

- GUI design as a part of requirements specification
- Using system architecture to structure requirements

Requirements analysis - inputs and outputs


Existing system information
Organisational standards
Rules and regulations
Domain information
Initial project documents
Input requirements (of varying quality)
Requirements analysis
Stakeholders
Requirements specification

Initial project documents

- Project Initiation Document (PID) / Project charter
- Business Case (see Example)
- Feasibility studies
- …other

Stakeholder identification

Stakeholder: an individual, group or organization who may affect or be affected by the result of the project

Principles

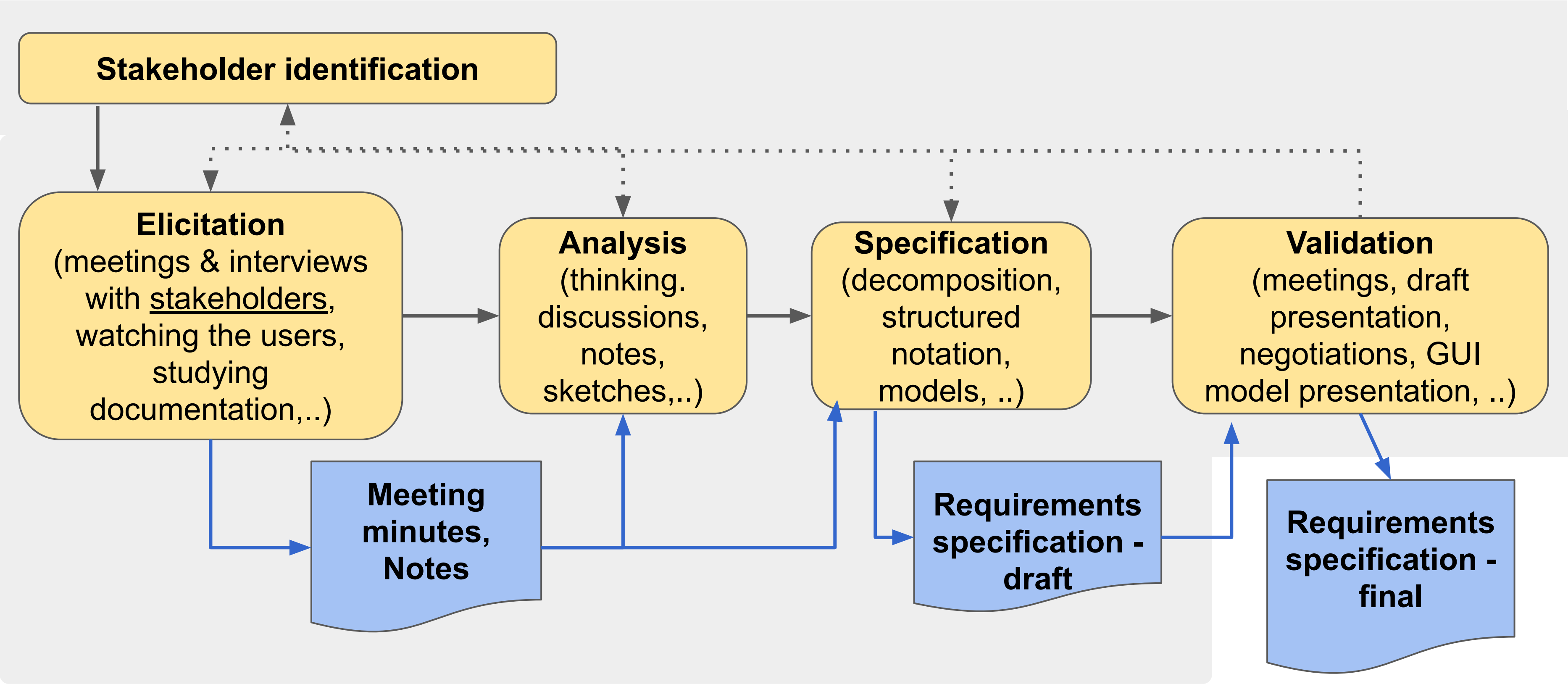
- To identify stakeholders as soon as possible
  - Various stakeholders in various domains / companies / environments
  - Checklists exist for IT projects
  - Include project team members
- To identify specific representatives
  - We need to communicate with real people
- Some stakeholders are discovered later
  - We do our best to make “later” as soon as possible
  - Rework may be needed
- Business stakeholders vs technological (IT) stakeholders

Stakeholder examples

Project owner: bears business responsibility for successful project implementation. Typically head of the business unit receiving the product.

- Company management / various levels
- Project team members
  - Project manager, Analysts, UX designers, Architects, Developers, Testers, Document writers, …
- Project customer
- Product users (may be represented by product manager)
- Other teams
  - Sales representative, Marketing representative, Legal dept. representative, IT support, IT operations, ….

Requirements analysis - how it works inside



Stakeholder identification

Elicitation
(meetings & interviews with stakeholders, watching the users, studying documentation,..)

Analysis
(thinking. discussions, notes, sketches,..)

Specification
(decomposition, structured notation, models, ..)

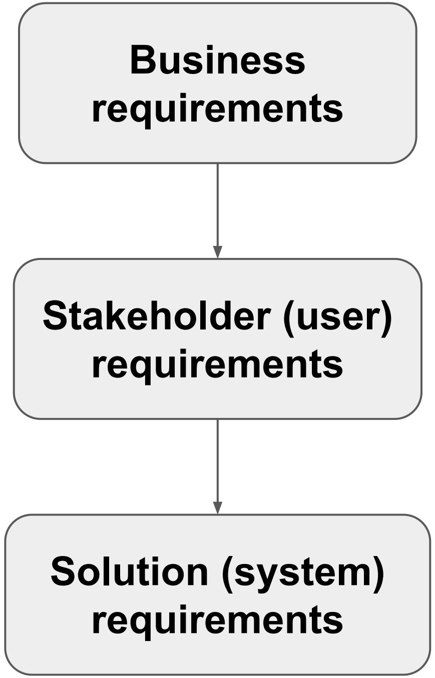
Validation
(meetings, draft presentation, negotiations, GUI model presentation, ..)

Meeting minutes, Notes

Requirements specification - draft

Requirements specification - final

Three levels of requirements



Business requirements

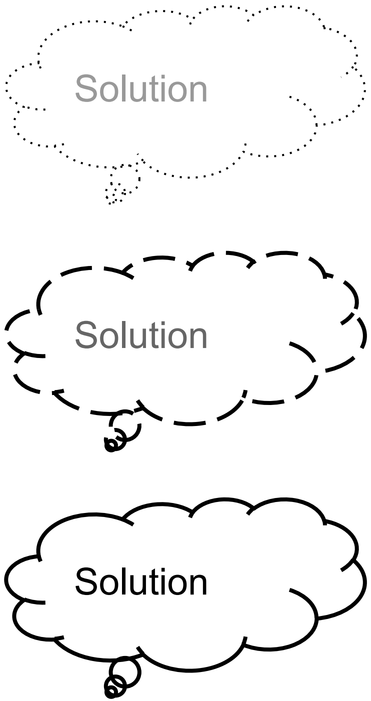
Stakeholder (user) requirements

Solution (system) requirements

These do not describe the solution sufficiently

Even these do not describe the solution sufficiently

These describe the solution quite well (we still abstract from the architecture & design)



Three levels of requirements

Business requirements

- Describe high-level objectives of the organization itself
- Written for management (but also basis for next phases)

Stakeholder (user) requirements

- Describe stakeholder/user needs
- Statements in natural language plus diagrams
- Written for stakeholders (but also input for next steps)

Solution (system) requirements

- Describe system’s functions, services and operational constraints in detail
- Technical language, diagrams, models
- Basis for designing the system
- May be incorporated into contract

Business Analysis vs. Requirements Analysis

- These two activities overlap especially at the level of business requirements
- Ideally, business requirements should be provided as a business analysis output.

Example

Business

BR1: Increase online sales by 20% within the next year.
BR2: increase repeat orders from customer by 10% within six months after deployment

Stakeholder (user)

SR1: Create new user account.
SR2: View order history.
SR3: Check order status.
SR4: Create new order.

Solution (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 solution (system) 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

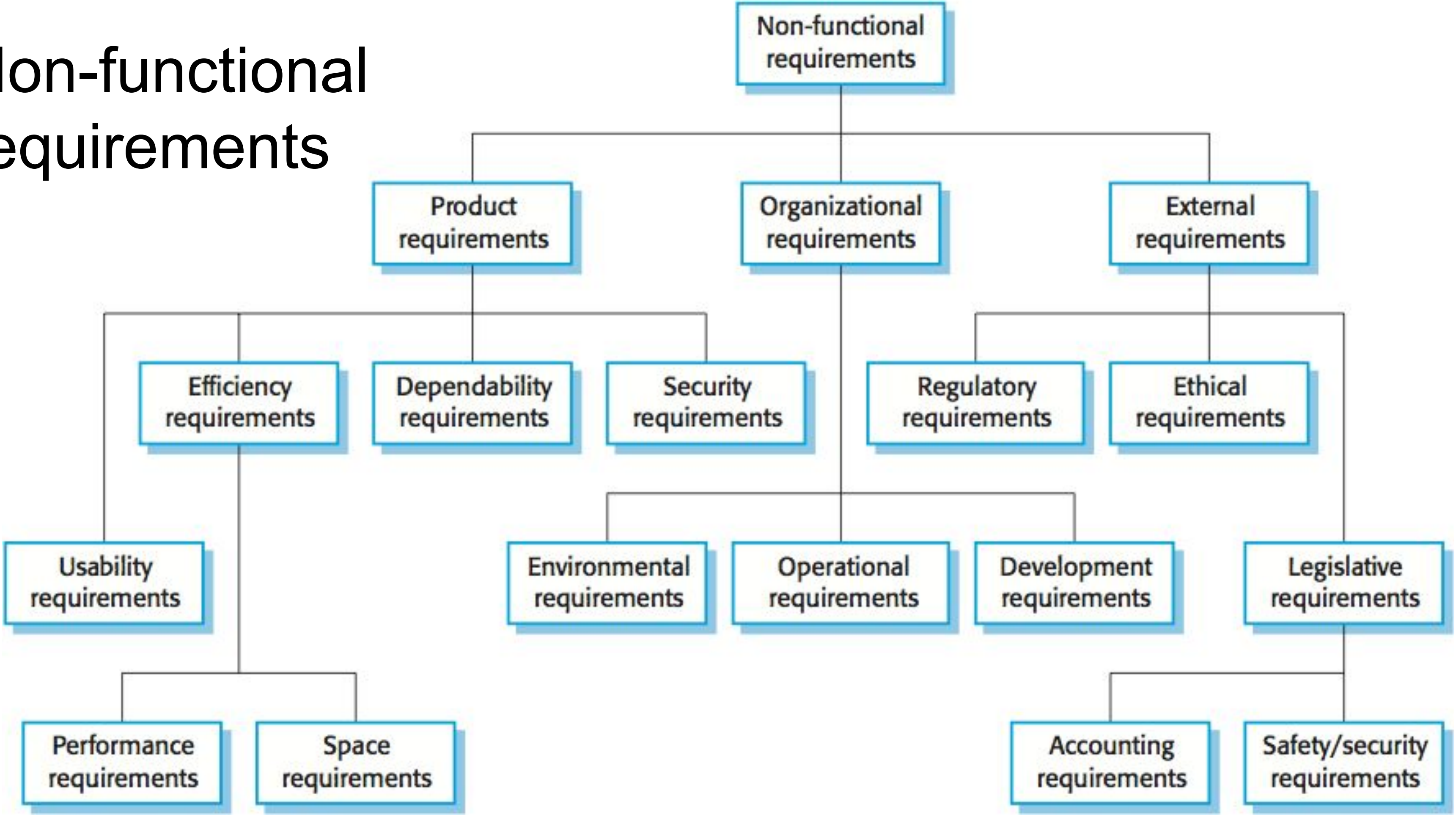
Non-functional requirements

- 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

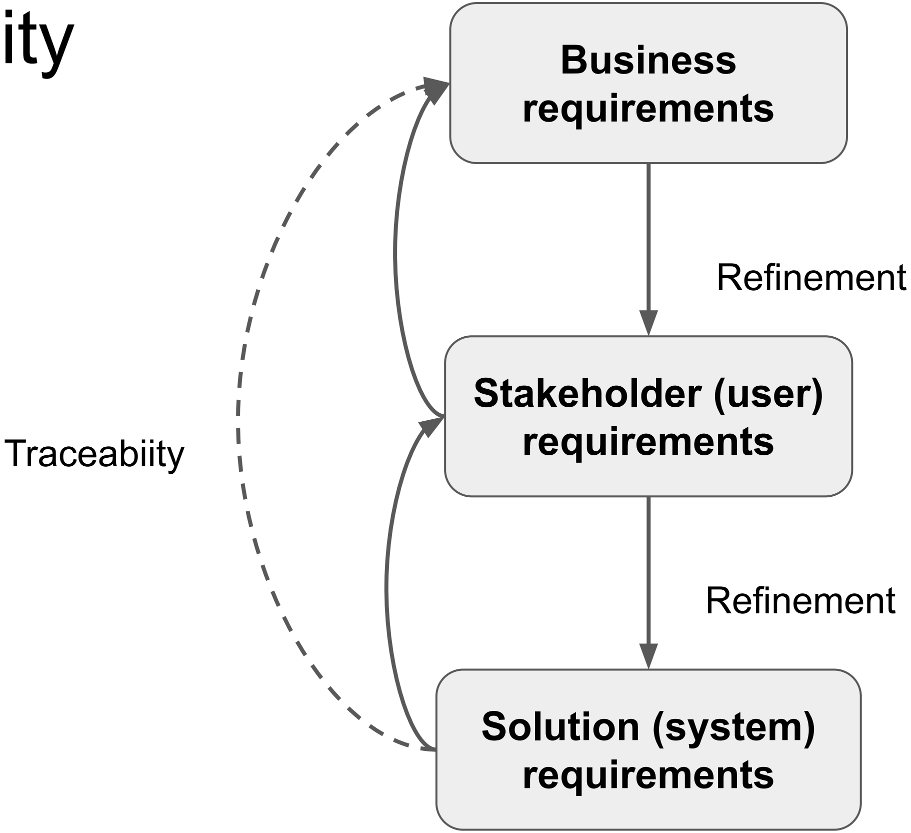


More examples

- 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.

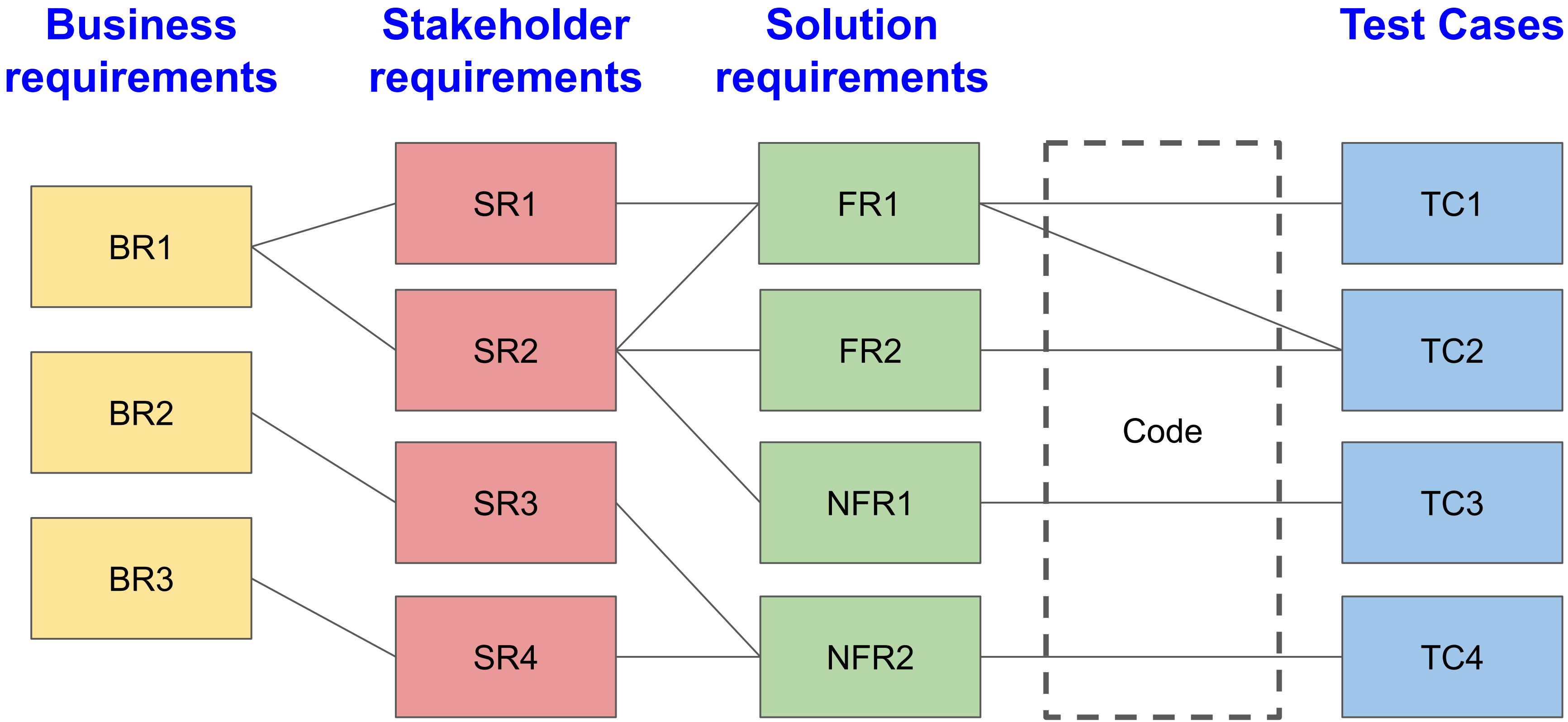
- Domain requirement:
  - The deceleration of the train shall be computed as: D (train) = D (control) + D (gradient), where D (gradient) is 9.81ms2 * compensated gradient/alpha and the values of 9.81ms2 /alpha are known for different types of train.

Requirements traceability



Business requirements
Refinement
Stakeholder (user) requirements
Traceabiity
Refinement
Solution (system) requirements

Requirements traceability



Business requirements

Stakeholder requirements

Solution requirements

Test Cases

BR1

BR2

BR3

SR1

SR2

SR3

SR4

FR1

FR2

NFR1

NFR2

Code

TC1

TC2

TC3

TC4

Why do we need requirements?

Input for next phases

- Architecture and design, implementation, validation and verification, maintenance

Input for supporting activities

- Documentation, project management, …

Input for establishing a contract

- Basis for a bid for a contract
- Part of the contract (scope definition - what will be delivered)

Audience for requirements:

- ➔ People participating in these next phases / supporting activities, or establishing a contract
- ➔ Also people validating the requirements

How the requirements are written?

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.”

Software requirements definition

- Output of “Requirements” phase
- Also **Software Requirements Specification (SRS)**

- Typically structured text supported by figures/models/diagrams
  - UML, BPMN, E-R diagrams, GUI model (mockups), …

Various approaches

- Unstructured text - “Victorian novel”
  - Massive narrative sequential description, seldom used today
- Flat catalogue of requirements
  - Often used, not optimal
- Combination - structured text

Modeling languages

- UML
  - UML does not provide means to define non-functional requirements
  - Customer may have poor knowledge of UML…
- BPMN
  - Business process modeling (overlap with business analysis)

User stories (agile approaches)

Usually a **combination** of these approaches.

It is important to take into account the **audience** so that they are able to read and understand the requirements definition.

Unstructured text

The ecommerce store will be expanded with user accounts so that each account will contain the email address of the given user. Our marketing department will be able to reach users by email with various marketing campaigns if the user gives such permission. This way we expect to increase repeat orders. See also our internal Reports that show number of repeat orders in last 12 months and Case study that show how account management can help to increase repeat orders. ….

The e-commerce store will provide the possibility to create a user account, log in to this account, log out of this account. Before creating an account, the user must agree to the storage of his personal data in accordance with (GDPR). ....

- Difficult to distinguish specific requirements and to track their attributes (priority / progress / estimates)
- Useful e.g., when explaining the general context and relationships between particular levels of requirements (especially BRs <-> SRs)

Flat catalogue

ID	Requirement	Priority	Estimates	…
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			
SR1	USER: Create new user account.			
SR2	USER: View order history.			
SR3	USER: Check order status.			
SR4	USER: Create new order.			
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			
…	…			

- Easy to distinguish specific requirements and track their attributes
- Difficult to explain relationships between requirements (especially BRs <-> SRs)

Combination - structured text

ID	Requirement	Priority	Estimates	…
BR1	Reduce incorrectly processed orders by **50%** by the end of next quarter			
BR2	increase repeat orders from user by **10%** within six months after deployment			

Current state based on Internal reports

- BR1: 90% of incorrectly processed orders are caused by **incorrect data filled by user** when creating an order (incorrect e-mail address, incorrect delivery address, incorrect phone number)
- BR2: Only 30% of customers ordered repeatedly within last 12 months

Case study

Extending the system with the user accounts

- BR1: can decrease number of incorrectly processed orders due to incorrect user data by 60%
- BR2: can increase number of repeat orders by 10-15% (together with marketing campaigns)

⇨ **Extending the system with user accounts will fulfill BR1 and BR2**

Glossary

- Use the same terms for the same concepts throughout the whole requirements definition
- It makes it easier to understand the requirements
- Examples
  - System vs. e-commerce store vs e-shop
  - User vs customer vs buyer
  - Item vs product
  - Shopping basket vs shopping cart vs cart

User roles may be described separately from the glossary

- Requirements with inconsistent terms:
  - The system shall enable the customer to insert items into the shopping basket.
  - The e-shop shall enable the buyer to remove products from the cart.

Glossary - example

Order

- A request for delivery of a group of items

Item

- A product to be sold

Shopping basket

- A container for storing items that the user is considering ordering

…

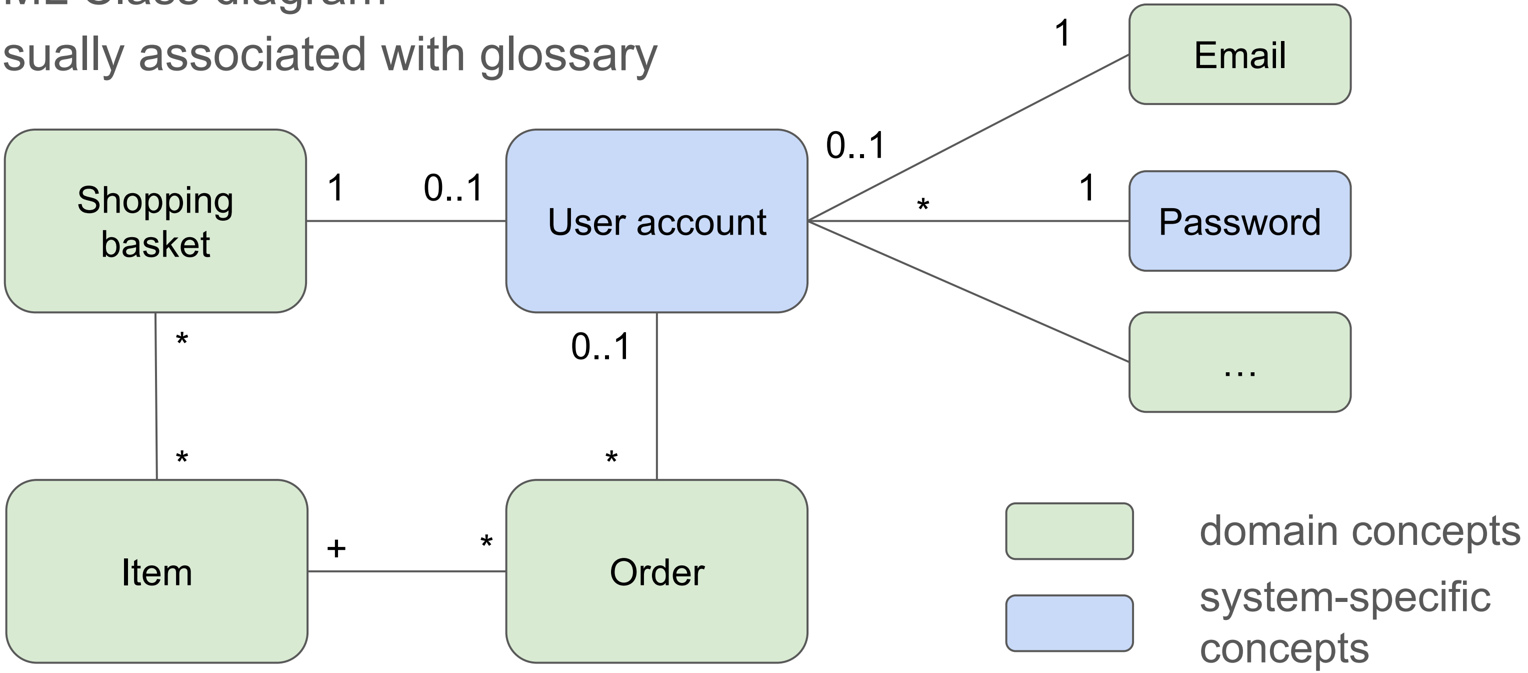
UML

Requirements definition can be supported by UML models, typically

- Data model
  - Conceptual level
  - Class diagram
- Use Case model
  - Functional requirements
  - Use Case diagram + Use Case descriptions
  - Activity diagram

Conceptual model

- High-level data model, both domain and system concepts
- UML Class diagram
- Usually associated with glossary

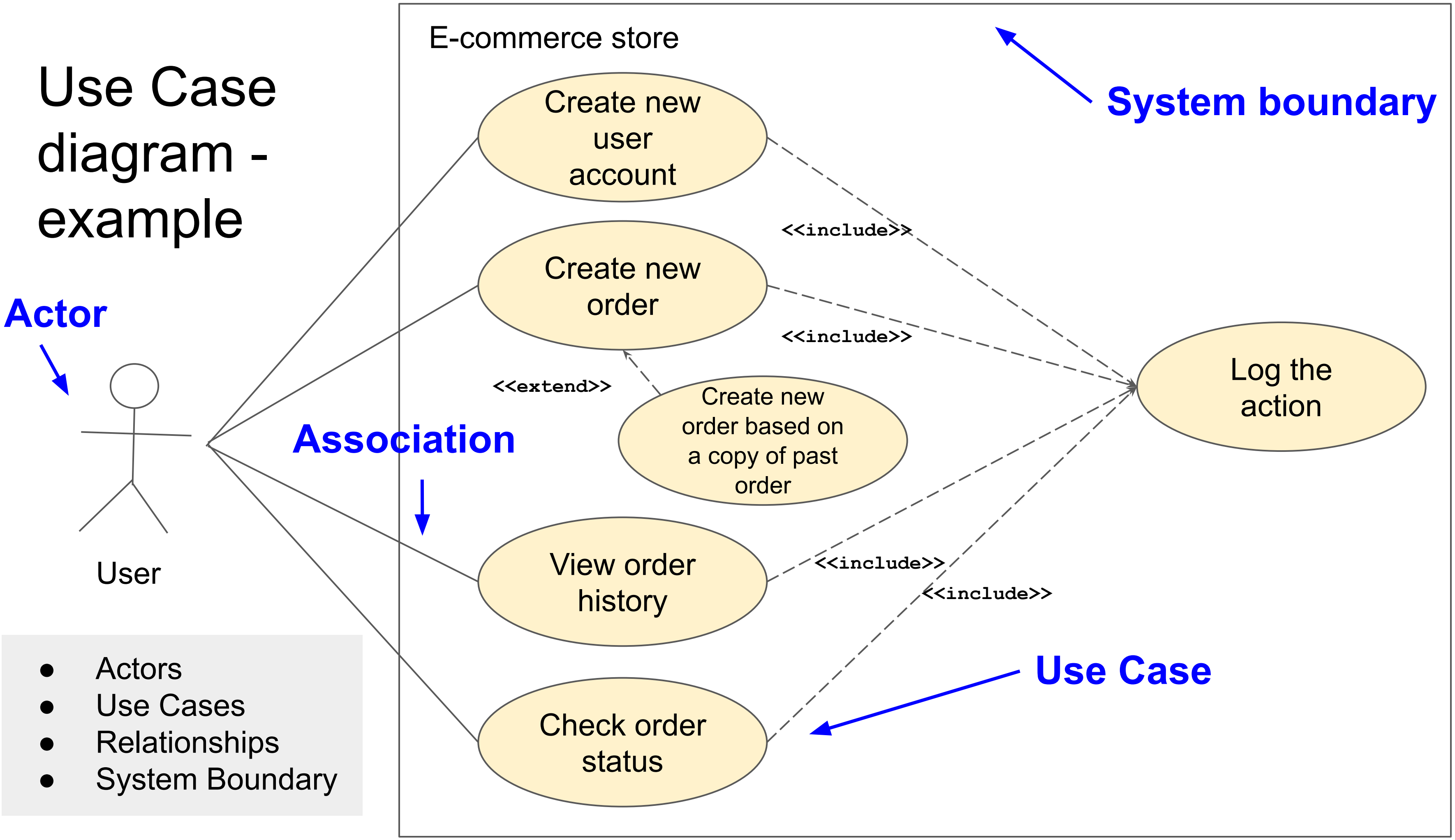


UML

Requirements definition can be supported by UML models, typically

- Data model
    - Domain / conceptual level
    - Class diagram
- Use Case model
    - Functional requirements
    - Use Case diagram + Use Case descriptions
    - Activity diagrams

Use Case diagram - example



- Actors
- Use Cases
- Relationships
- System Boundary

Use Case diagram

Use Cases

= functionality of the system

- Inside the system boundary

Actors

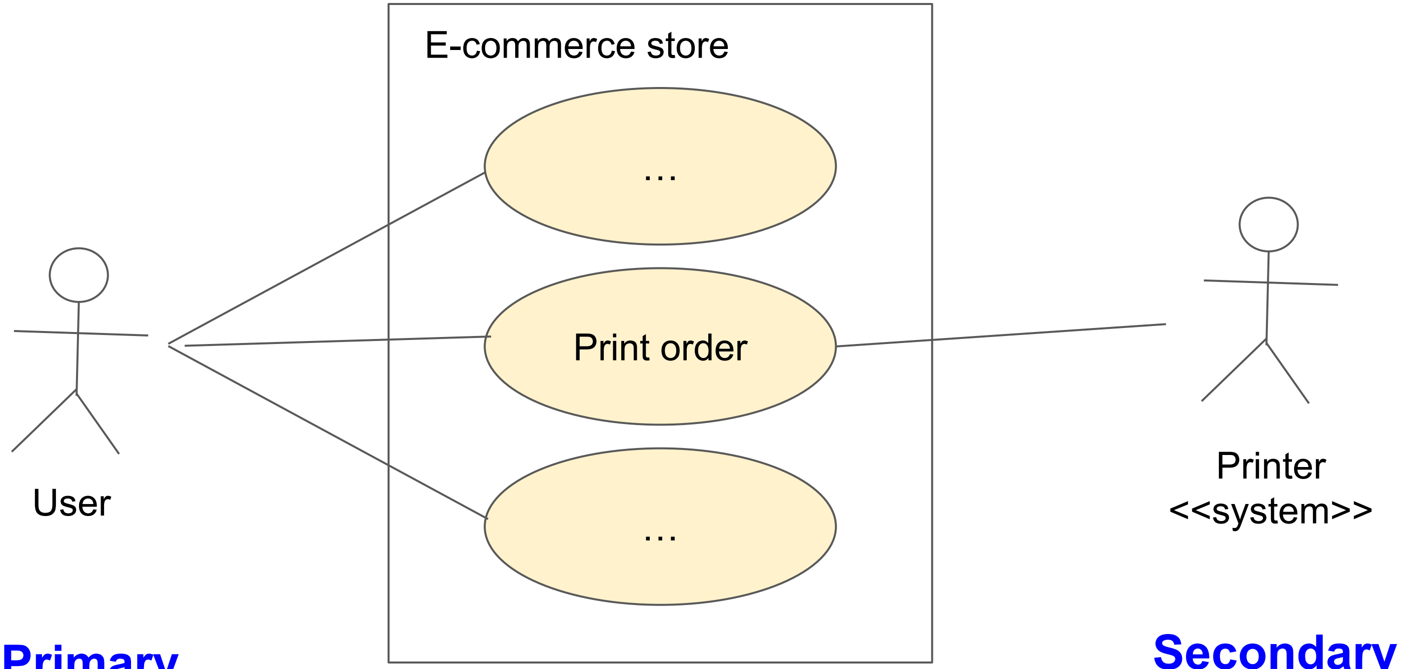
= entities that interacts with the system

- Always outside the system boundary
- Human users or other systems (<<system>> stereotype)
- Primary vs. secondary actors

Relationships

- Actor - Actor
  - Generalization
- Actor - Use Case
  - Association (represents interaction)
- Use Case - Use Case
  - Generalization
  - “Extend” dependency
  - “Include” dependency

Primary vs secondary actor



Primary

Secondary

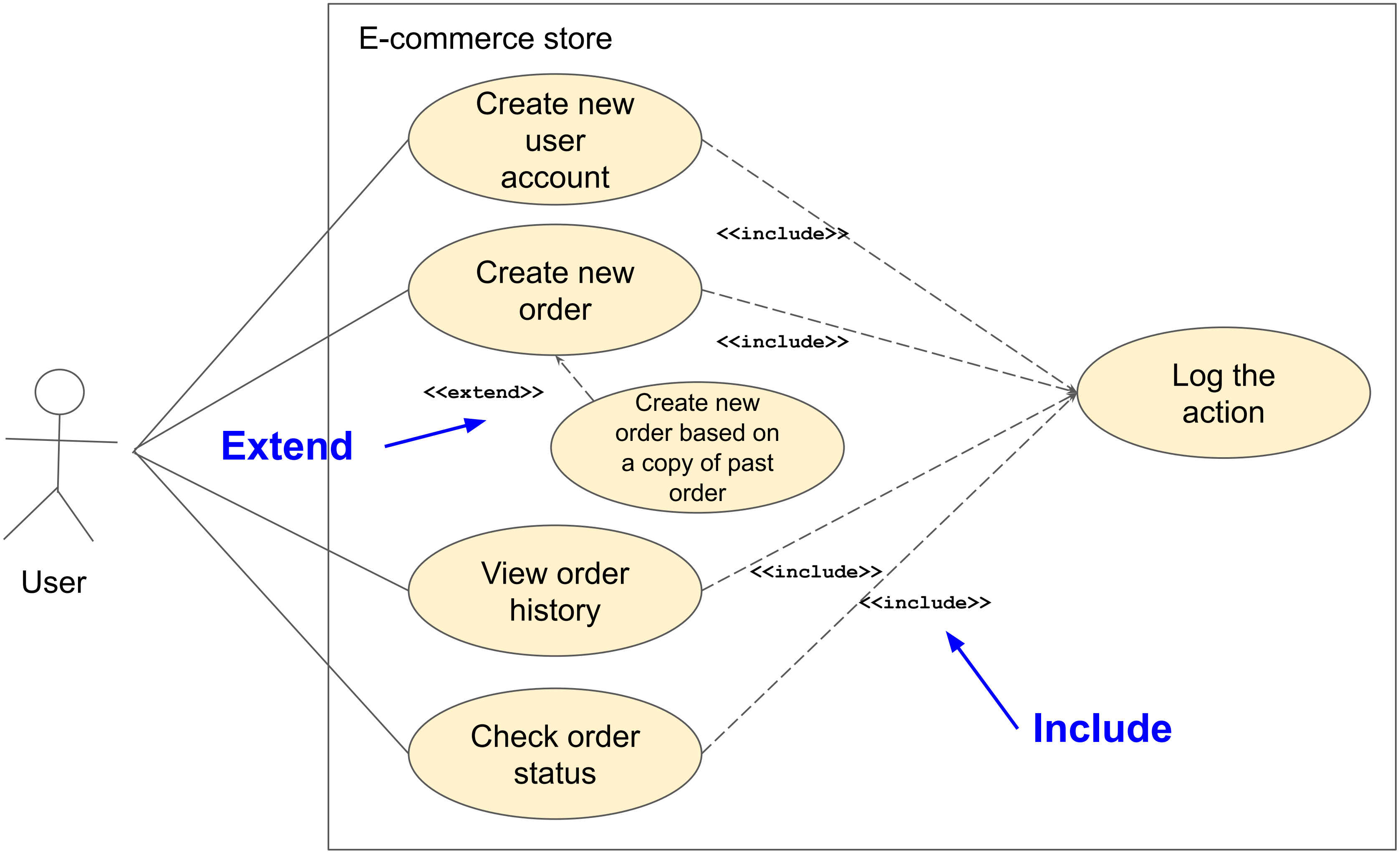
Include vs extend dependency

“Include” dependency

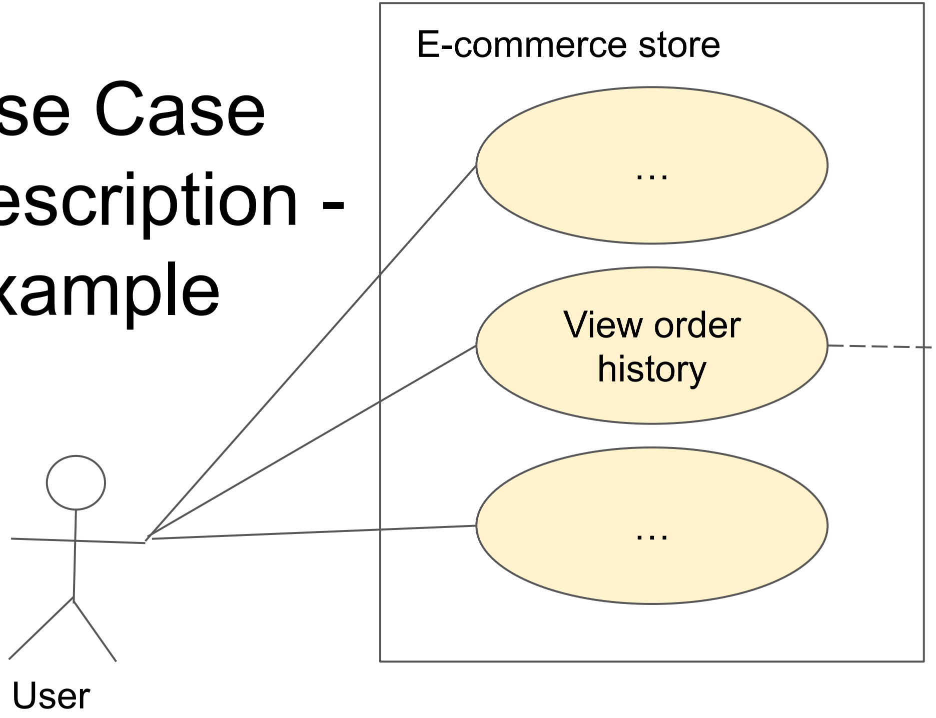
- An including use case always contains the behavior defined in another, included (base), use case. Included use case can be seen as subroutine.

“Extend” dependency

- The behavior defined in the extending use case can be inserted into the behavior defined in the extended use case


E-commerce store
Create new user account
Create new order
Create new order based on a copy of past order
View order history
Check order status
Log the action
User
<<include>>
<<include>>
<<extend>>
<<include>>
<<include>>
Extend
Include

Use Case description - example



View order history

Goal:
To display all orders associated with given user account
Preconditions:
User is logged into their user account
Postconditions:
The list of all orders associated with given user account is displayed

Steps:

1. If no orders are associated with given user account, inform the user that there are no orders.
2. If one or more orders are associated with given user account, display the list of these orders sorted by order time from newest to oldest. For each order display order ID, order status and list of ordered items. For each ordered item display …

- Too verbose
- Difficult to maintain

Activity diagrams

Example (“Checkout” functionality)

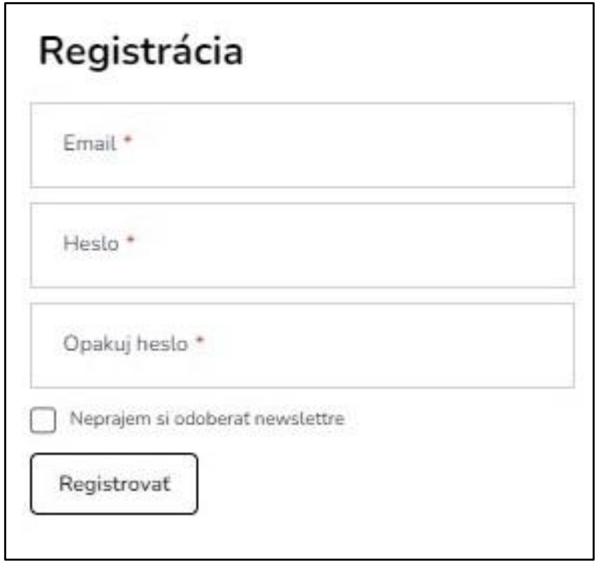
Requirements definition vs. UX outputs

- Solution requirements are typically associated with mock-ups
- Mock-ups vs. requirements
  - A single mock-up may cover one or more requirements
  - A single requirement may be covered by one or more mock-ups (or even no mock-up at all)
  - Mockups may use different glossary than requirements definition
- Example:

  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.

Mock-up: a visual representation or screenshot of how the final website or product will look

MU1:



Registrácia

Email *

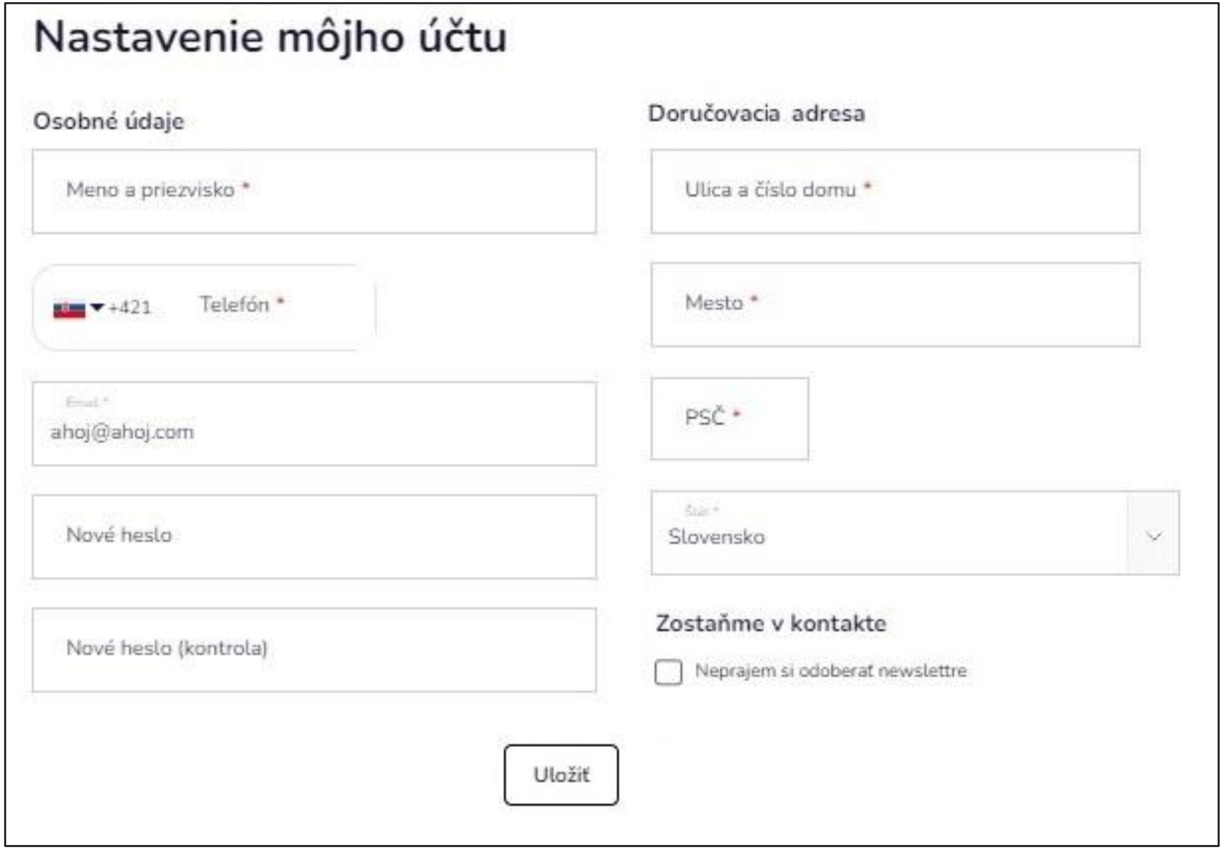
Heslo *

Opakuj heslo *

□ Neprajem si odoberať newslettre

Registrovať

MU2:



Nastavenie môjho účtu

Osobné údaje

Meno a priezvisko *

+421 Telefón *

Email *
ahoj@ahoj.com

Nové heslo

Nové heslo (kontrola)

Doručovacia adresa

Ulica a číslo domu *

Mesto *

PSČ *

Štát *
Slovensko

Zostaňme v kontakte

□ Neprajem si odoberať newslettre

Uložiť

Agile methodologies

- Software development runs in short, flexible iterations

SCRUM

- Self-organizing team, 5-9 people responsible for the product

  (scrum master, product owner, developers, UX, QA)

- Iteration: 2-4 weeks
- Input for the team: product backlog = prioritized list of “user stories”

Scrum assumes that the product backlog contains the requirements for the product, but does not specify where they come from.

- Typically some pre-analysis has to be completed outside SCRUM

User stories

“A user story is an informal, general explanation of a software feature written from the perspective of the end user or customer”

- Roughly correspond to the level of stakeholder requirements, but often mixture of levels
- Alistair Cockburn (1998): "A user story is a promise for a conversation."
- Common template
    - `As a <role> I want to <capability>, so that <receive benefit>`
    - “So that” part optional

- User stories = placeholders for further discussion, can be split / refined to more detailed specification if needed
- Placed into product backlog and prioritized

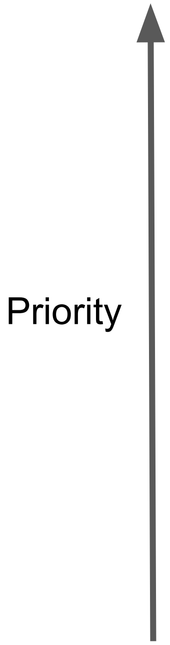
User stories - examples

As a user
I want to create new user account
so that I do not need to enter my data repeatedly and I am entitled to various discounts.

As a user
I want to create new order
so that I buy items.

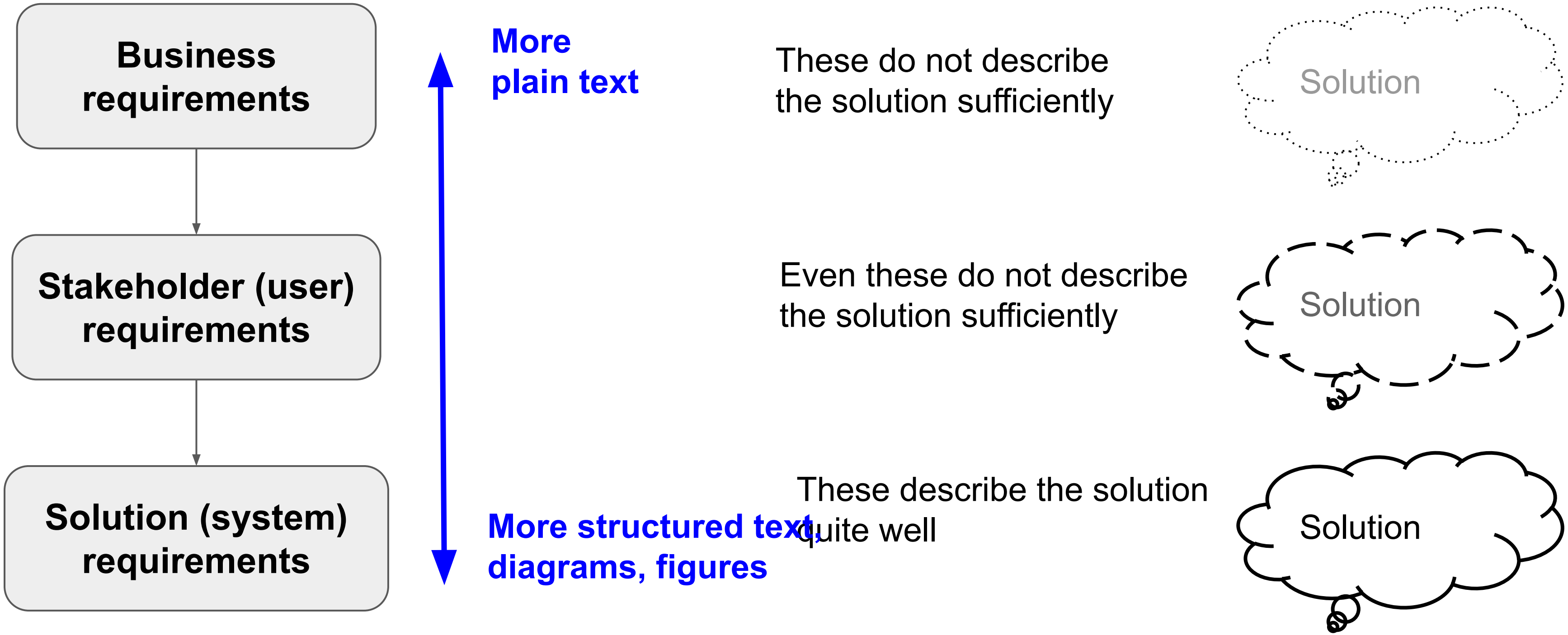
As a user
I want to view order history
so that I buy again items that I liked.

As a user
I want to check order status
so that I see if there is some issue with order processing.



Priority

Three levels of requirements - requirements definition



Business requirements

Stakeholder (user) requirements

Solution (system) requirements

More plain text

More structured text, diagrams, figures

These do not describe the solution sufficiently

Solution

Even these do not describe the solution sufficiently

Solution

These describe the solution quite well

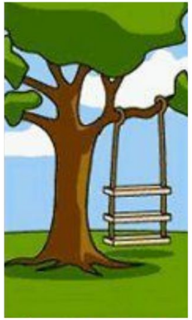
Solution

Tools

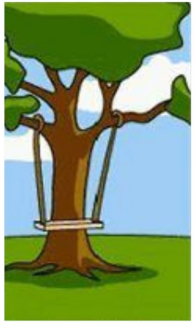
- CASE (Computer-aided software engineering tool)
  - More difficult to create and maintain the specification
  - Provides complete system description
  - Example: Enterprise Architect
- Collaborative Software, Wiki (e.g., Atlassian Confluence)
- Issue tracking system (e.g. Atlassian Jira)

Requirements problems

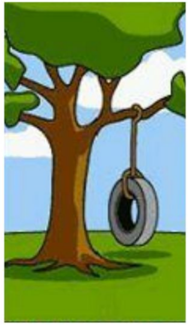
- Stakeholders don't understand what they want
    - Even if they know what they want, they cannot describe it
    - Even if they can describe it, they often describes solution rather than real need



How the customer explained it.



What we did.



What the customer really needed.

- Unclear business requirements
- Missing stakeholders
- Unavailable stakeholders
- Significant changes in requirements late in the analysis
- Poor traceability

Best practices

- Insist on clear **business requirements**
- Gather requirements from **all stakeholders**
- Take into account **all types** of requirements
- Avoid **grey zones**
- Document requirements **accurately** and **consistently**
- Only accept **traceable requirements**
- Validate requirements with **all stakeholders**

Further reading

- Ian Sommerville: Software Engineering (10th edition)
- IEEE STANDARD 830-1998 - IEEE Recommended Practice for Software Requirements Specifications
- Karl Wiegers, Joy Beatty: Software Requirements (3nd Edition)
- Dean Leffingwell: Agile software requirements
- Alistair Cockburn: Writing Effective Use Cases
- Alistair Cockburn: Agile Software Development (2nd edition)