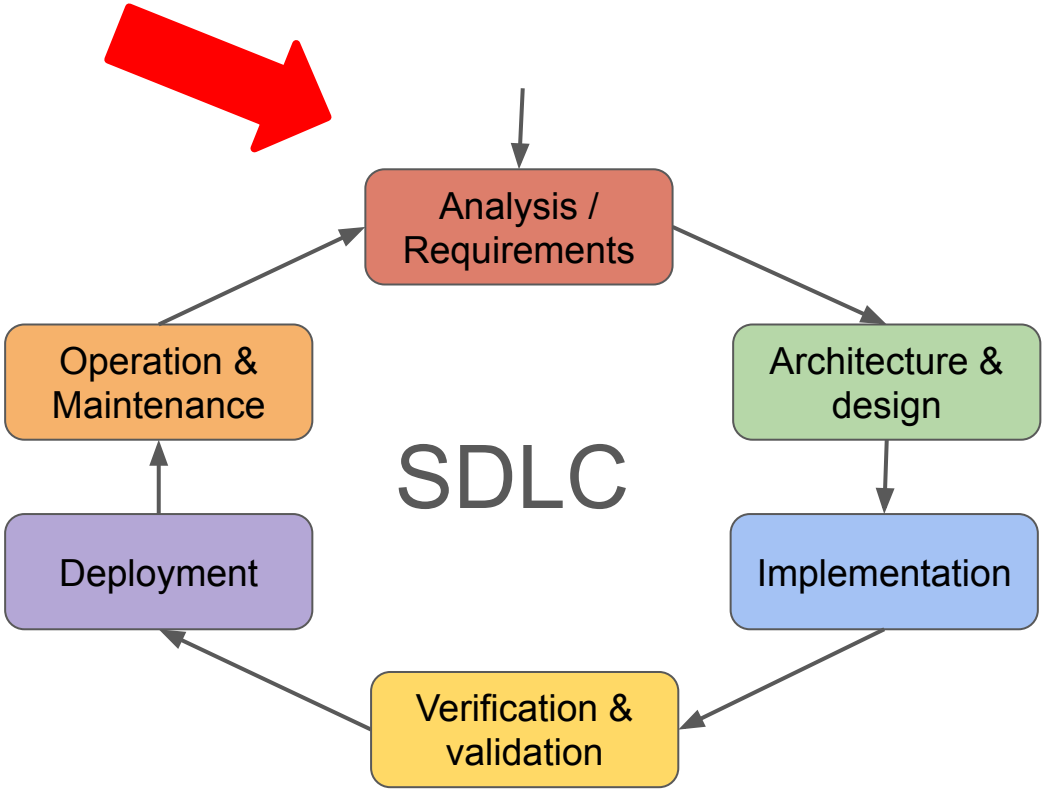
Requirements


Analysis / Requirements
Architecture & design
Implementation
Verification & validation
Deployment
Operation & Maintenance
SDLC

Terminology

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

Informally also

- “IT analysis”
- “Software analysis”
- “IT business analysis” (typically 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
    -

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

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 analysis approach

Traditional approach

- Requirements analysis: a clearly identifiable phase before the implementation
- More commonly used for large systems

Agile approach

- Requirements are elicited concurrently as the system is developed

Hybrid approaches

- Some requirements are analyzed up front (core features, constraints) in a separate phase, while others evolve incrementally through agile iterations

We start with the traditional approach.

Levels of Requirements

The term “requirement” is not used consistently in the SW industry. In some cases, a requirement is simply **a high-level, abstract statement** of a service that a system should provide or a constraint on a system. At the other extreme, it is a **detailed formal definition** of a system function.

Some of the problems that arise during the requirements engineering process are a result of failing to make a clear separation between these different levels of description.

(Sommerville, 2015)

- We will consider 3 levels of requirements, based on BABOK guide

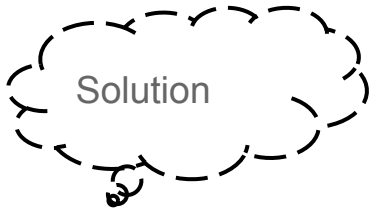
Three levels of requirements (BABOK guide)

Business requirements

These do not describe the solution sufficiently

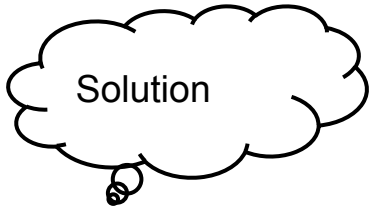
Stakeholder (user) requirements

Even these do not describe the solution sufficiently



Solution (system) requirements

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



Three levels of requirements

Business requirements

- High-level, abstract statements in nature language
- Written for management (but also basis for next phases)

Stakeholder (user) requirements

- Based on real 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
- Written for development team, architects (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.

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

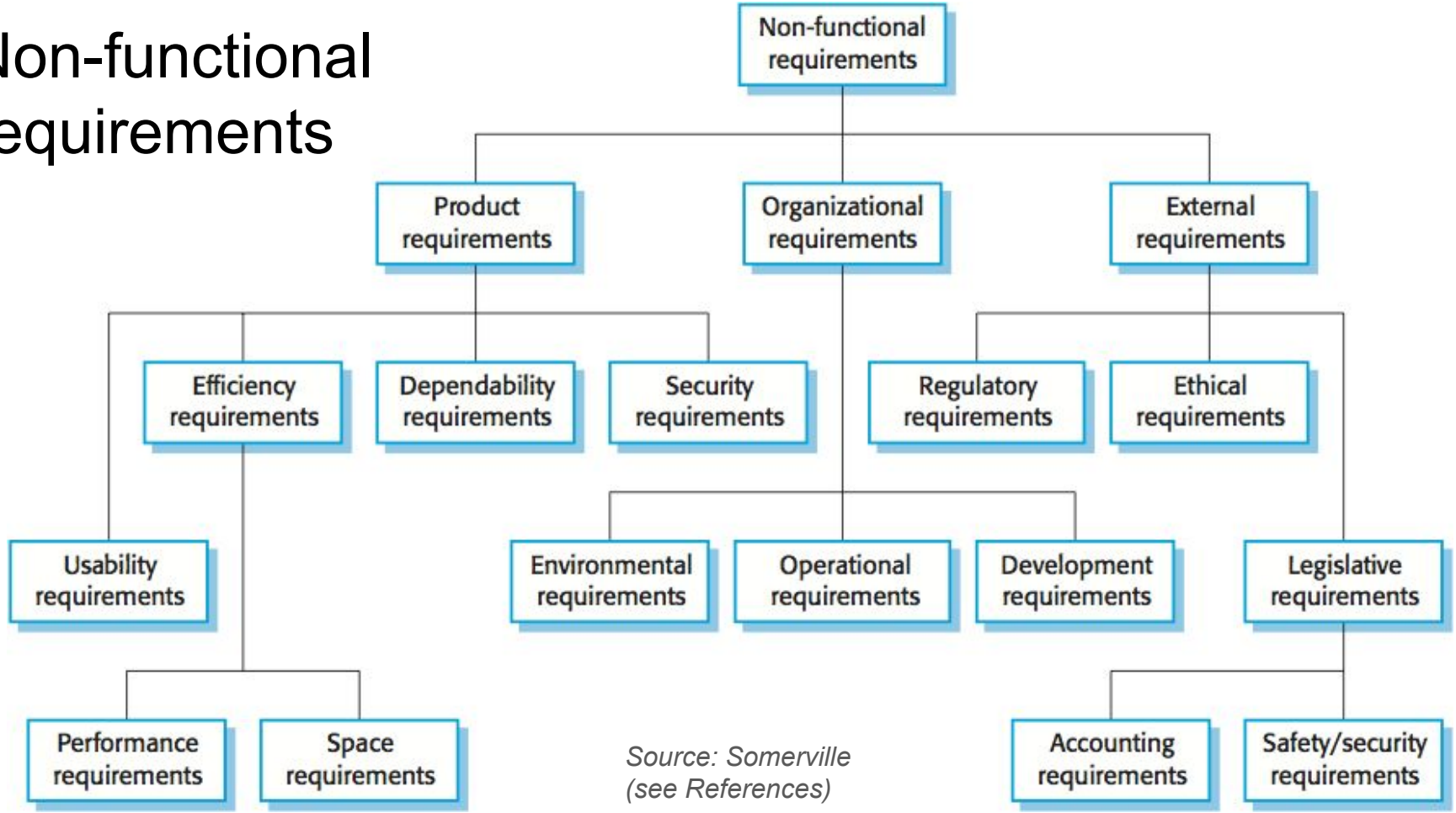
Non-functional requirements

- 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



Source: Somerville (see References)

Examples

Business

BR1: Achieve at least a 10% increase in repeat purchases within the first year by enabling customers to browse, select, and purchase products.
BR2: The platform should attract new customers and generate at least 15% of total revenue online in the first year

Stakeholder (user)

SR1: The customer must be able to create and use a user account.
SR2: The customer must be able to view order history.
SR3: The inventory manager must be notified automatically when stock of a product falls below a defined threshold..

Solution (system)

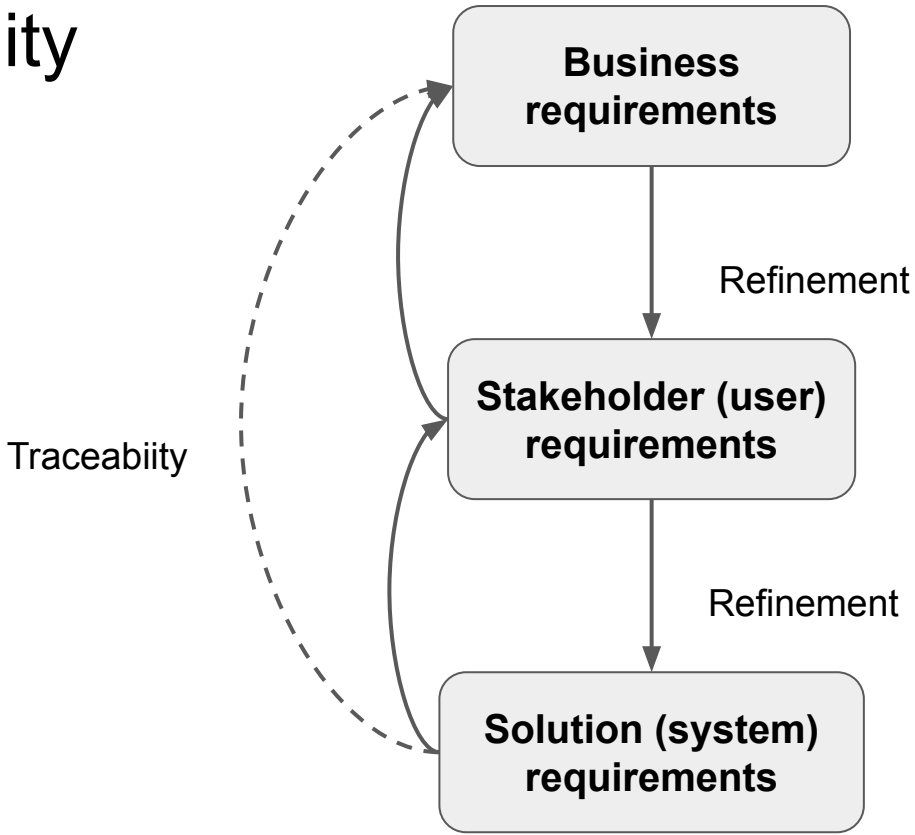
FR1: The system shall allow the creation of a new user account with the following attributes:e-mail address, first name, last name, address line 1, address line 2, city, postal code, ….
FR2: The system shall allow users to log in using e-mail and password.
NFR1: The system shall enforce passwords of at least 8 characters.
NFR2: The system shall run on all Java platforms including 64-bit versions.

More examples

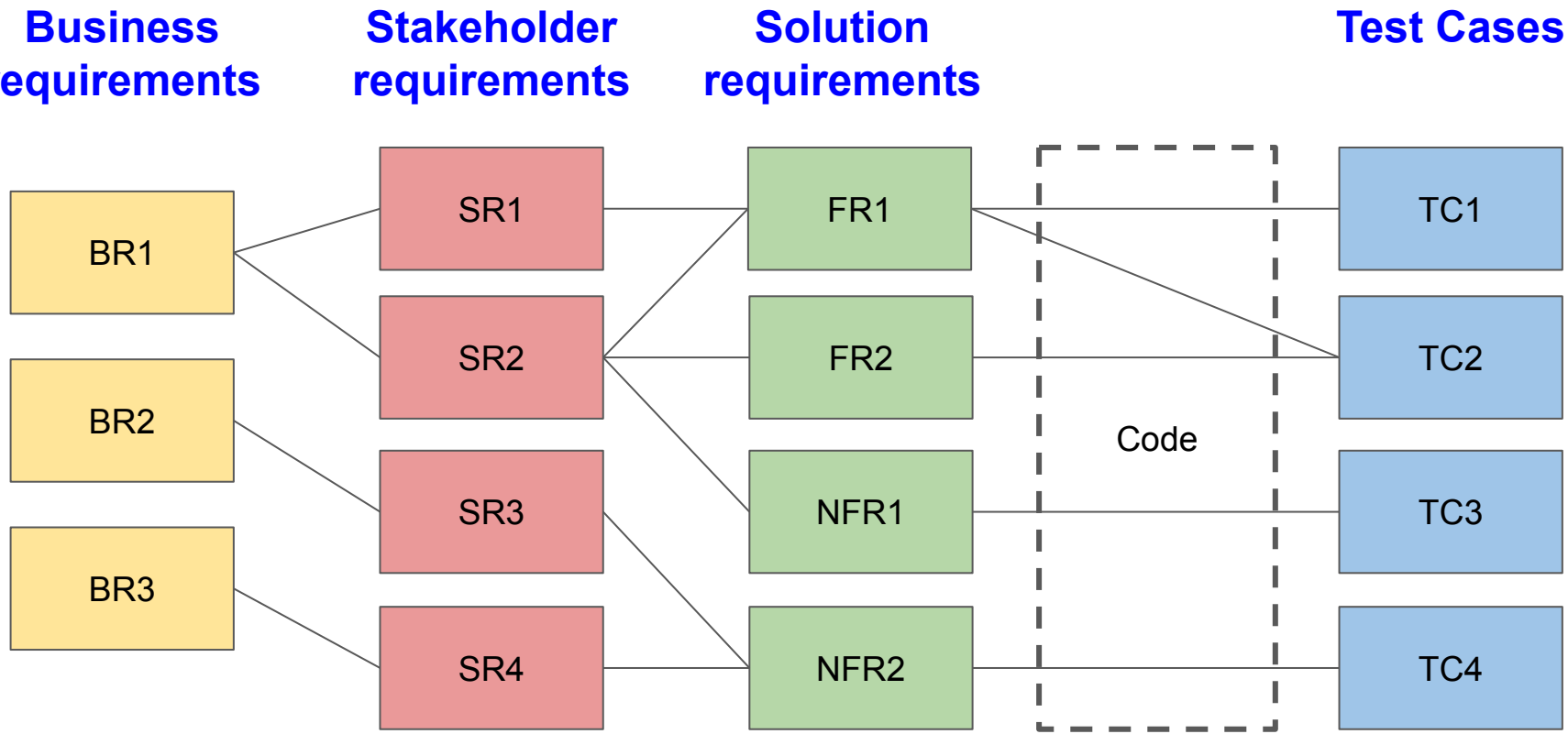
- Non-functional requirements:
  - PRODUCT REQUIREMENT: The system's user interface shall be implemented in HTML5, without the use of frames or Java applets.
  - ORGANIZATIONAL REQUIREMENT: The system development process and deliverable documents shall conform to the process and deliverables defined 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

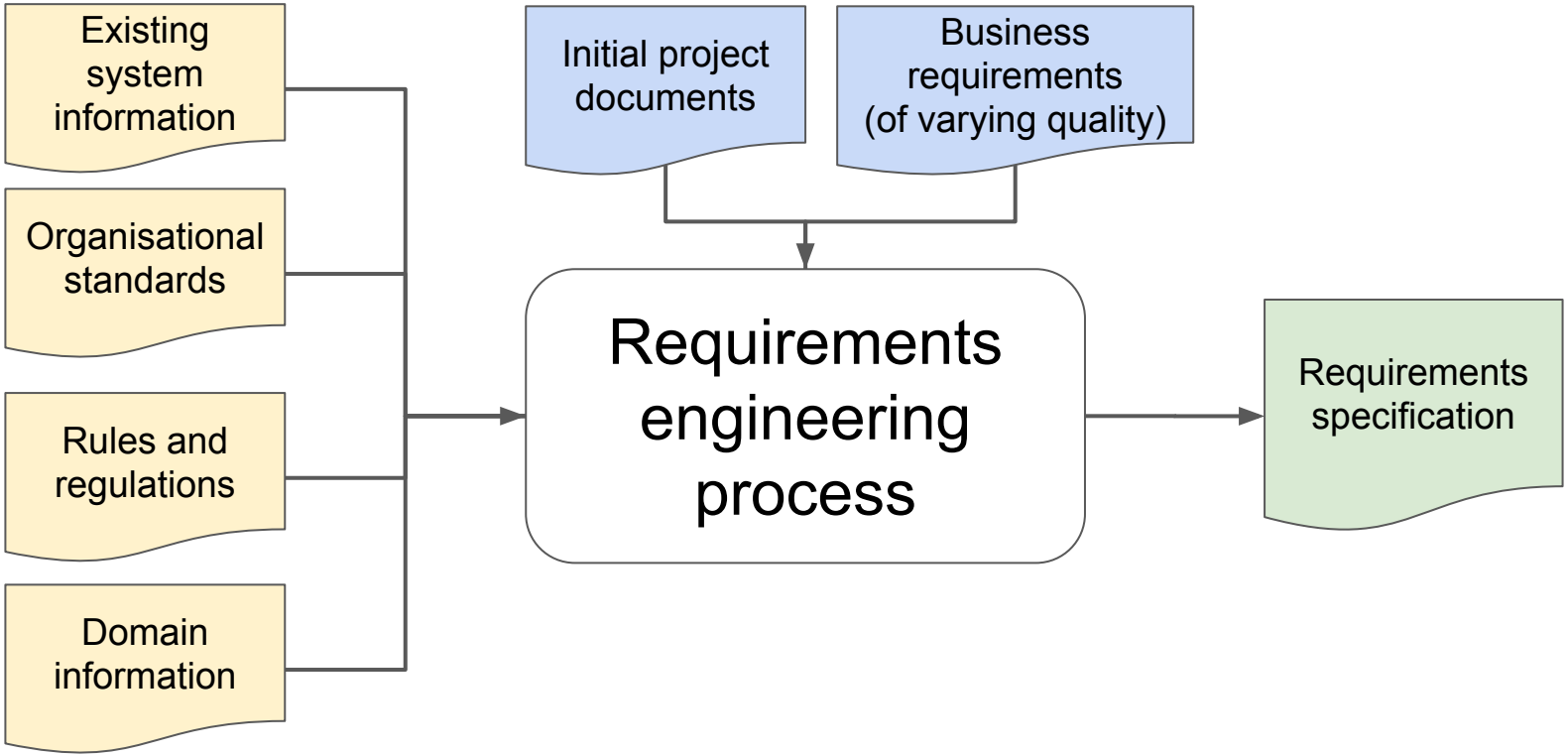


Requirements traceability



Requirements Engineering Process

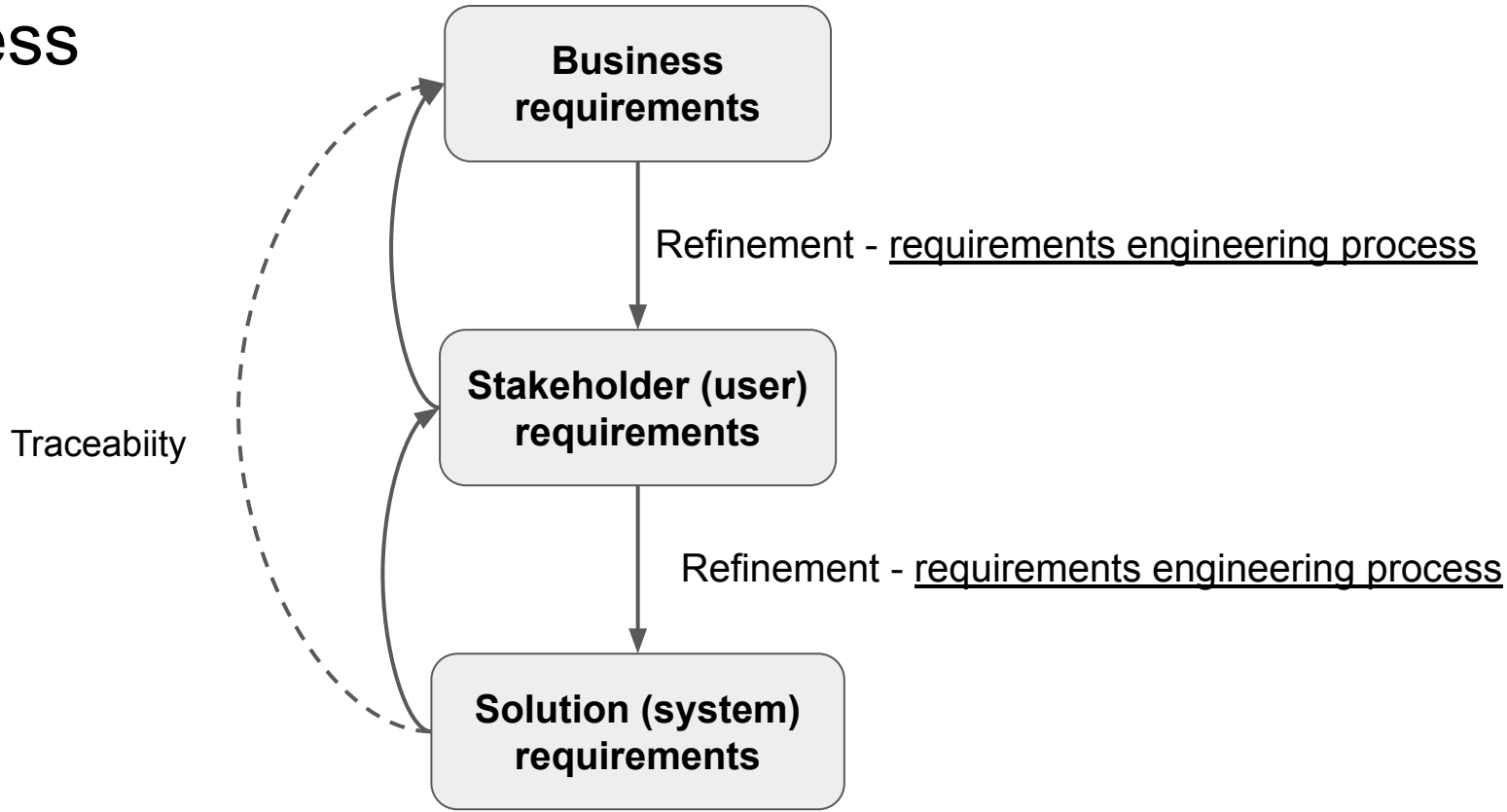
Inputs and outputs



Initial project documents

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

Requirements levels vs process



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

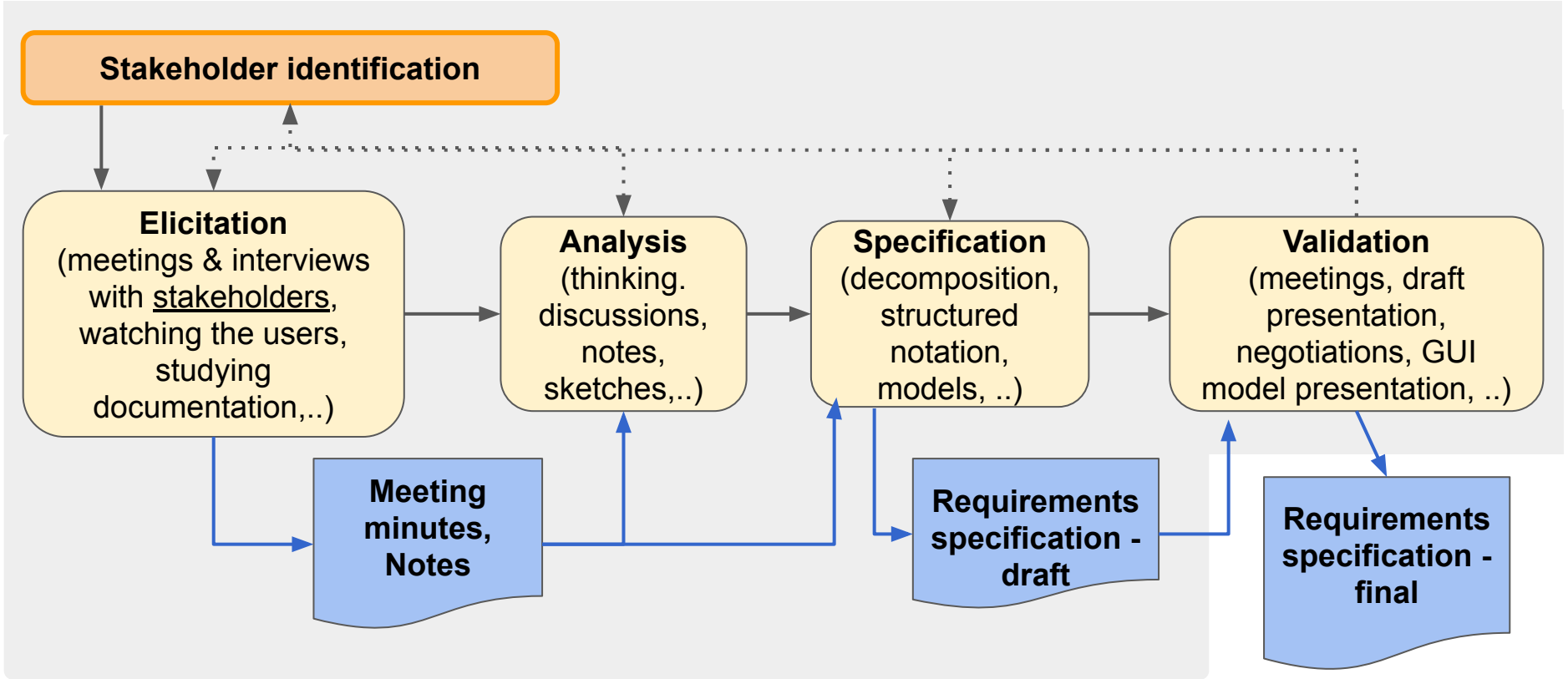
- Basis for a bid for a contract
- Part of the contract (scope definition - what will be delivered)

Audience for requirements:

➔ People participating in these next phases / supporting activities, or establishing a contract

➔ Also people validating the requirements

Requirements engineering process



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 (see next slide)
  - 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 - literature

IEEE Std 29148-2018

Stakeholders include, but are not limited to:

- end users, end user organizations,
- supporters,
- developers, producers,
- trainers, maintainers,
- disposers, acquirers, operators,
- customers,
- supplier organizations,
- accreditors and regulatory bodies

BABOK guide

- Customer
- Domain Subject Matter Expert
- End User
- Implementation Subject Matter Expert
- Sponsor
- Other

SWENG book

- Users
- Customers
- Market analysts
- Regulators
- Software engineers

Karl Wiegers' checklist: DOCX

❗ End users vs customers

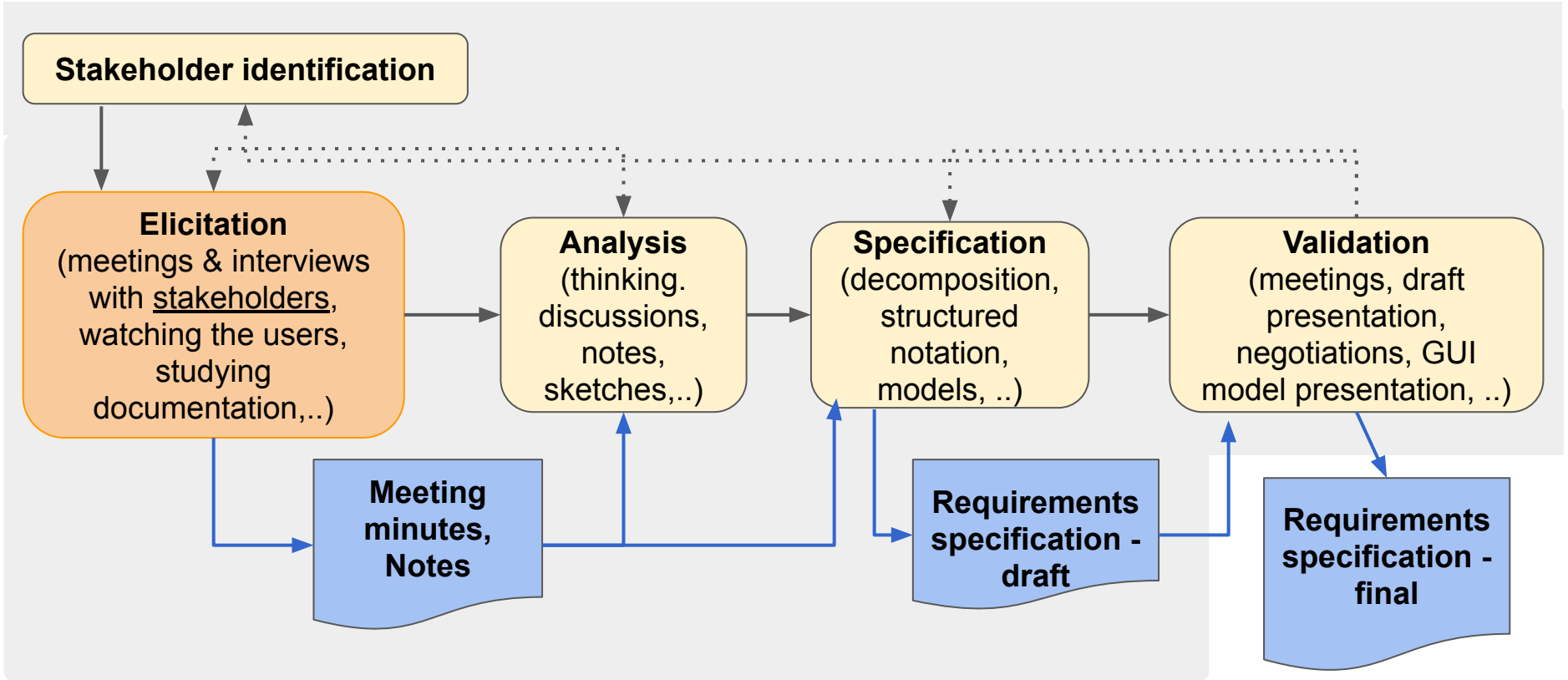
See References

Stakeholder examples - summary

- Project customer
- Product / service end users
  - May have internal representative (often product manager)
- Company management / various levels
  - Sponsor, project owner, …
- Project team members
  - Project manager, Analysts, UX designers, Architects, Developers, Testers, Document writers, …
- Other teams
  - Sales representative, Marketing representative, Legal dept. representative, IT support, IT operations, ….
- Other external entities
  - Regulators, suppliers, ..

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

Requirements engineering process



Elicitation phase - techniques

Active techniques:

- Interviews - structured (with prepared questions), semi-structured, open ended
- Workshops - multiple stakeholders in collaborative session
- Brainstorming - group activity to generate a wide range of ideas quickly
- Questionnaires / Surveys
- Prototyping - building & discussing paper sketches / wireframes / mockups / SW models

Indirect techniques:

- Domain understanding
- Observation - watching users in the real environment
- Document analysis
- Current system analysis

AS-IS vs TO-BE analysis !

Elicitation phase - common issues

- Stakeholders don’t know what they want from a system
- Stakeholders describe SOLUTION that does not solve their real NEED
- Stakeholders are busy and it is difficult to “catch” them
- Changing requirements
- Language and terminology differences (business vs IT vs analyst)
- Conflicting requirements / priorities among stakeholders
- Undefined decision makers
- Incomplete or missing documentation
- Difficulties in getting access to existing systems

Example

We want lightweight suitcases

Customers

Detailed research on materials, design, prototyping, testing, …

Development team
(implementing the requirement blindly)

➔ Super lightweight
➔ Higher cost (expensive materials)

How successful were the new suitcases?

Example

Nobody buys them!

Customers have bought suitcases from competitors that were cheaper and heavier, but they hadi **wheels.**

- The original requirement described a SOLUTION
- Real NEED:: “We want suitcases that are easy to transport”
- The solutions described by customers were over-engineered and did not even meet their real needs!
    - If they put heavy load into the lightweight suitcases, they were again hard to move

Source: Wiegers, Beatty (see References)

Five whys method

- A tool for root-cause analysis
- Comes from Toyota Motor Corporation
- The number of whys may be higher or lower depending on the complexity of the analysis

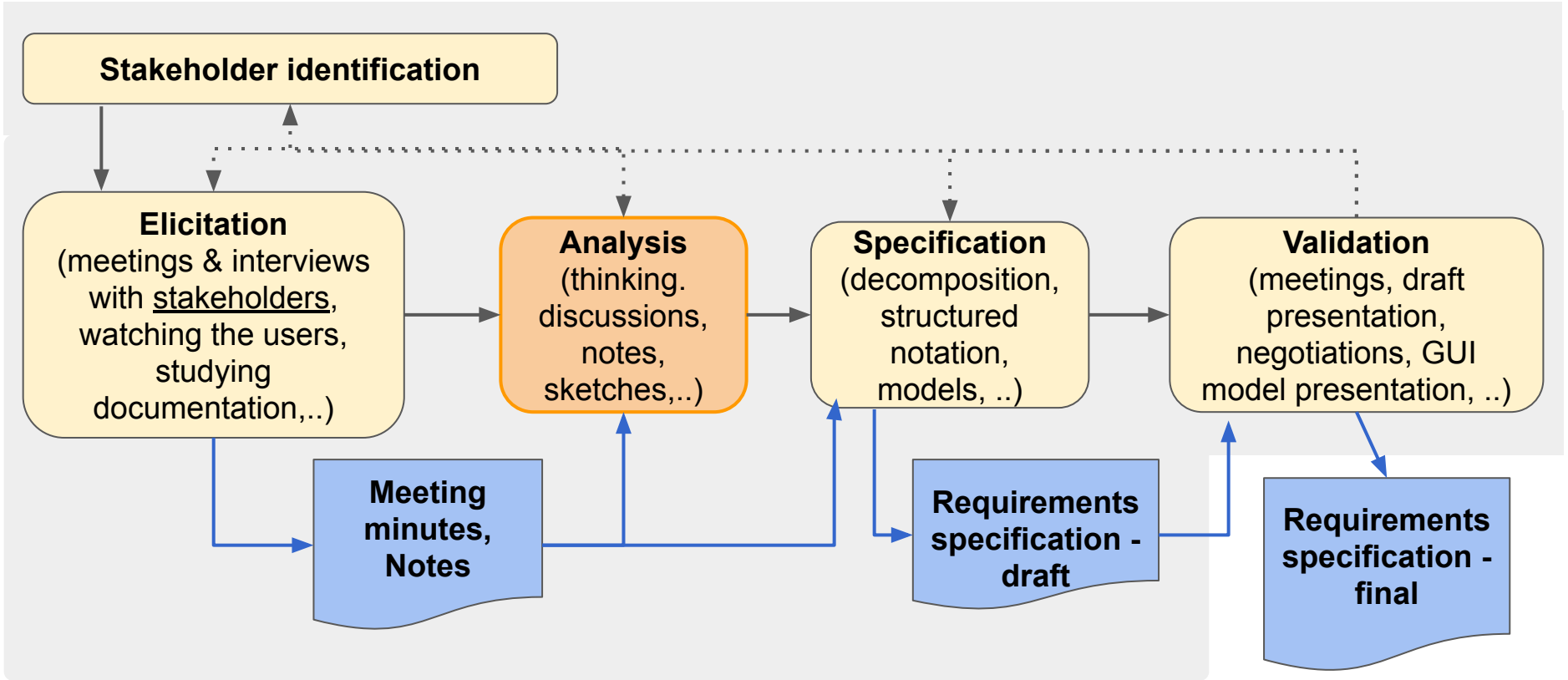
“We need lightweight suitcases.” -> REQUIREMENT DESCRIBING SOLUTION

WHY?

“Because we want suitcases that are easy to transport.” -> REAL NEED

Now that we know the real need, we can start thinking about a suitable solution…

Requirements engineering process



Analysis phase

Goal: to analyze the requirements gathered so far and resolve identified issues

TYPICAL ISSUES	SOLUTION
Ambiguous, incomplete	Clarify / complete or re-elicitate
Conflicting requirements (with each other or with constraints)	Resolve conflicts / modify
Not aligned with business requirements / stakeholder real needs	Remove / modify / re-elicitate
Infeasible requirements (technical, financial, schedule, …)	Remove / modify
Unclear or under-specified areas (“grey zones”)	Gather / complete / refine

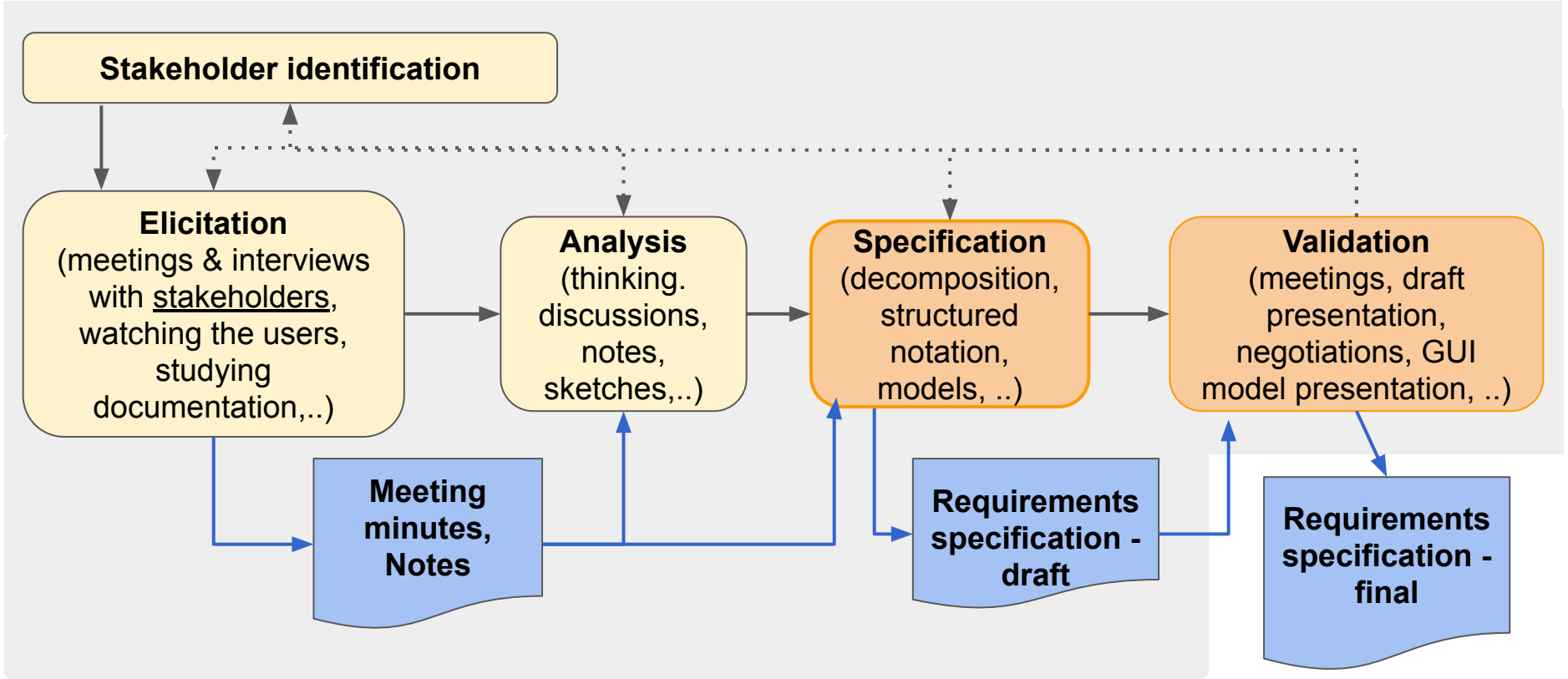
- Local vs global analysis
- Further discussions with stakeholders typically overlap with another round of elicitation
- Karl Wieger’s Analysis phase checklist: DOCX

Requirements quality characteristics

- Clear and understandable
  - Written in plain, precise language so that all stakeholders interpret it the same way
- Unambiguous
  - Each requirement can only be interpreted in one way
- Consistent
  - No internal conflicts between requirements; terms and concepts are used consistently
- Design-independent
  - Should not prescribe GUI (**!**), architecture, or technology (exceptions in NFRs if prescribed by constraints)
- Complete
  - Fully covers the required scope, leaving no grey zones or implicit assumptions
- Traceable
  - Aligned with business requirements / real stakeholder needs
- Feasible
  - Can be implemented within known technical, business, and project constraints.
- Verifiable / testable
  - It must be possible to check objectively whether the requirement is satisfied
- Detailed appropriately
  - Provide enough input for subsequent design and implementation phases (or contract if relevant)
- Prioritized

References: BABOK guide, IEEE Std 29148-2018, Karl Wiegers' checklist

Requirements engineering process



Specification & Validation phase

Specification

- How the requirements are written -> we'll deal with this in detail

Validation

- Final checks on requirements quality (see quality characteristics)
  - Similar to tasks in "analysis phase", but the input should be polished and complete, minimizing the need to revisit prior phases
- Documentation check
- Might include formal approval

Requirements Specification

How the requirements are written?

There is much variation in how they are written and presented

Software Requirements Definition

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

FORM: Typically structured text + figures / models / diagrams

- UML, BPMN, E-R diagrams, ad-hoc models
  (drawback: intended audience has poor knowledge of used notation)

CONTENT: Requirements, Glossary, Domain model*, GUI model,

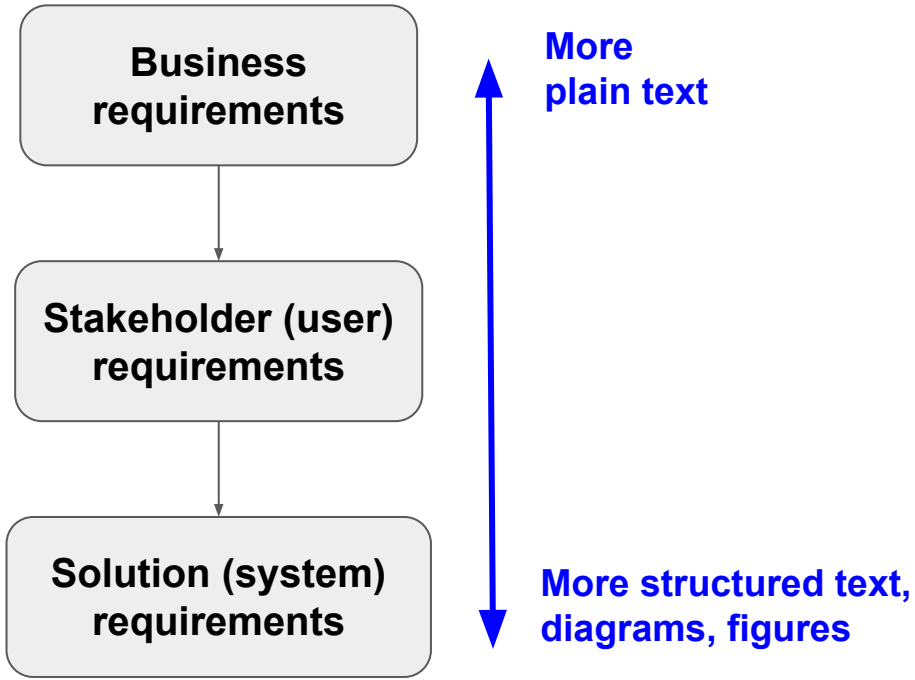
\+ Stakeholder and environment description, Constraints and assumptions, High-level data model

Agile / hybrid approaches - more lightweight documentation

❗ It is important to take into account the **audience** so that they are able to read and understand the requirements definition (or the part that is intended for them)

* Note: The domain model (& business process models) are often created already during overall business analysis

Three levels of requirements - requirements definition



Requirements slicing

Vertical slice:
= A specific level of requirements (BRs, SRs, FRs / NFRs)

Horizontal slice
= Requirements for one delivery iteration, across levels

We'll take a look at ….

Basic approaches

- Requirements - structured text
- Glossary - structured text
- Domain model - UML class diagram
- GUI model - ad-hoc modeling

Specific approaches

- Functional requirements - UML use case diagram / model, Activity diagram
- Agile requirements - user stories

Requirements - structured text

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

Unstructured text - example

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). ....

❌ Lacks any structure, difficult to distinguish specific requirements and to track their attributes (priority / progress / estimates)

Flat catalogue - example 1

ID	Requirement	Priority	Estimates	…
BR1	Achieve at least a 10% increase in repeat purchases within the first year by enabling customers to browse, select, and purchase products.			
BR2	The platform should attract new customers and generate at least 15% of total revenue online in the first year.			
SR1	The customer must be able to create a new user account.			
SR2	The customer must be able to view order history.			
SR3	The inventory manager must be notified automatically when stock of a product falls below a defined threshold..			
FR1	The system shall allow the creation of a new user account with the following attributes:e-mail address, first name, last name, address line 1, address line 2, city, postal code, ….			
FR2	The system shall allow users to log in using e-mail and password.			
NFR1	The system shall enforce passwords of at least 8 characters.			
NFR2	The system shall run on all Java platforms including 64-bit versions.			
…	…			

❌ Difficult to read and maintain for larger systems.
No traceability.

Flat catalogue - example 2

ID	Requirement	Priority	Estimates	Parent reqs
BR1	Achieve at least a 10% increase in repeat purchases within the first year by enabling customers to browse, select, and purchase products.			-
BR2	The platform should attract new customers and generate at least 15% of total revenue online in the first year.			-
SR1	The customer must be able to create and use a user account.			BR1
SR2	The customer must be able to view order history.			BR1
SR3	The inventory manager must be notified automatically when stock of a product falls below a defined threshold..			BR2
FR1	The system shall allow the creation of a new user account with the following attributes:e-mail address, first name, last name, address line 1, address line 2, city, postal code, ….			SR1
FR2	The system shall allow users to log in using e-mail and password.			SR1
NFR1	The system shall enforce passwords of at least 8 characters.			
NFR2	The system shall run on all Java platforms including 64-bit versions.			
…	…			

- Column Parent reqs - may contain multiple parent requirements
- Poor support for detailed rationale
- Still difficult to read and maintain

Structured (hierarchical) text - example

1. Business Requirements

BR1. Achieve at least a 10% increase in repeat purchases within the first year.
Priority: High

BR2. The platform should attract new customers and generate at least 15% of total revenue online.
Priority: High

2. Stakeholder Requirements

SR1. The customer must be able to create and use a user account.
→ supports BR1 (+ rationale)
Priority: Medium

…

3. System Requirements

3.1 Functional

FR1. The system shall allow creation of a new user account with fields: e-mail, name, address, etc.
→ supports SR1 (+ rationale)

…

3.2 Non-functional

NFR1. The system shall enforce passwords of at least 8 characters.
→ supports BRX (+ rationale)

…

- ➔ Hierarchy is mostly based on vertical/horizontal slicing, logical grouping or a combination
- ➔ Possibility to add both attributes and explanatory unstructured paragraphs if needed
- ➔ Tool support (different views of the same requirements, sorting / filtering)

We'll take a look at ….

Basic approaches

- Requirements - structured text
- Glossary - structured text
- Conceptual model - UML class diagram
- GUI mockups - ad-hoc modeling

Specific approaches

- Functional requirements - UML use case diagram / model, Activity diagram
- Agile requirements - user stories

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
- 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.

User roles may be described separately from the glossary

Glossary - example

Order
A request by a customer to purchase one or more products, typically including delivery and payment information.
Product
A tangible or intangible item that can be offered for sale in the e-commerce store
Shopping cart
A temporary container for storing products that a customer intends to purchase.
Shopping cart item (or item)
A specific product along with the quantity selected, representing one line in the shopping cart.
Customer
A user of the e-shop who can browse products, add them to the shopping cart, and place orders.

….

We’ll take a look at ….

Basic approaches

- Requirements - structured text
- Glossary - structured text
- Domain - UML class diagram
- GUI mockups - ad-hoc modeling

Specific approaches

- Functional requirements - UML use case diagram / model, Activity diagram
- Agile requirements - user stories

Domain (conceptual) model

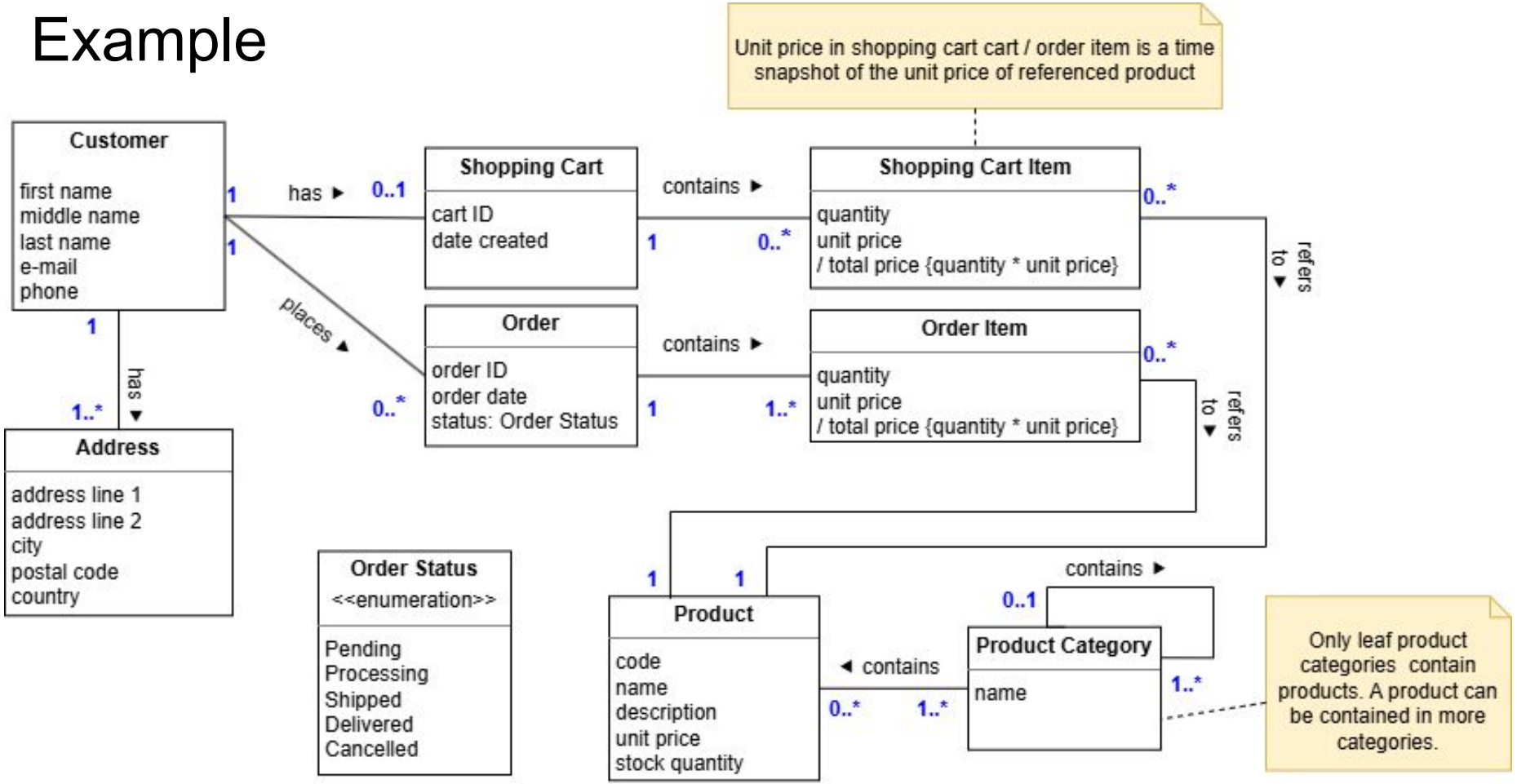
- High-level static model, visualization of domain concepts
- UML Class diagram, E-R diagram
- Aligned with glossary

-> See also PTS1

UML class diagram

- Conceptual classes = domain concepts
  - Attributes
  - Operations - different views:
    - No operations in domain models (e.g. Larman 2004)
    - Operations are allowed, but describing responsibilities, not interfaces (Fowler 2003)
- Associations = relations between domain concepts
  - Cardinalities
  - Association names with “reading direction”
  - Roles
  - …

Example


Unit price in shopping cart cart / order item is a time snapshot of the unit price of referenced product
Customer
first name
middle name
last name
e-mail
phone
1
has
0..1
Shopping Cart
cart ID
date created
contains
1
0..*
Shopping Cart Item
quantity
unit price
/ total price {quantity * unit price}
0..*
refers to
1
places
0..*
Order
order ID
order date
status: Order Status
contains
1
1..*
Order Item
quantity
unit price
/ total price {quantity * unit price}
0..*
refers to
1
has
1..*
Address
address line 1
address line 2
city
postal code
country
Order Status
<<enumeration>>
Pending
Processing
Shipped
Delivered
Cancelled
1
1
Product
code
name
description
unit price
stock quantity
contains
0..*
1..*
contains
0..1
Product Category
name
1..*
Only leaf product categories contain products. A product can be contained in more categories.

We'll take a look at ….

Basic approaches

- Requirements - structured text
- Glossary - structured text
- Domain model - UML class diagram
- GUI model - ad-hoc modeling

Specific approaches

- Functional requirements - UML use case diagram / model, Activity diagram
- Agile requirements - user stories

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)
  - Final mockups may use different glossary than requirements definition

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

Example:

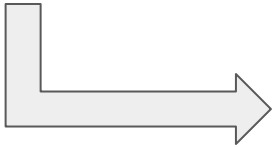
FR1: The system shall allow the creation of new user account with the following attributes: e-mail address, first name, last name, address line 1, address line 2, city, postal code, phone number,password, timestamp. ⟺ **MU1, MU2**

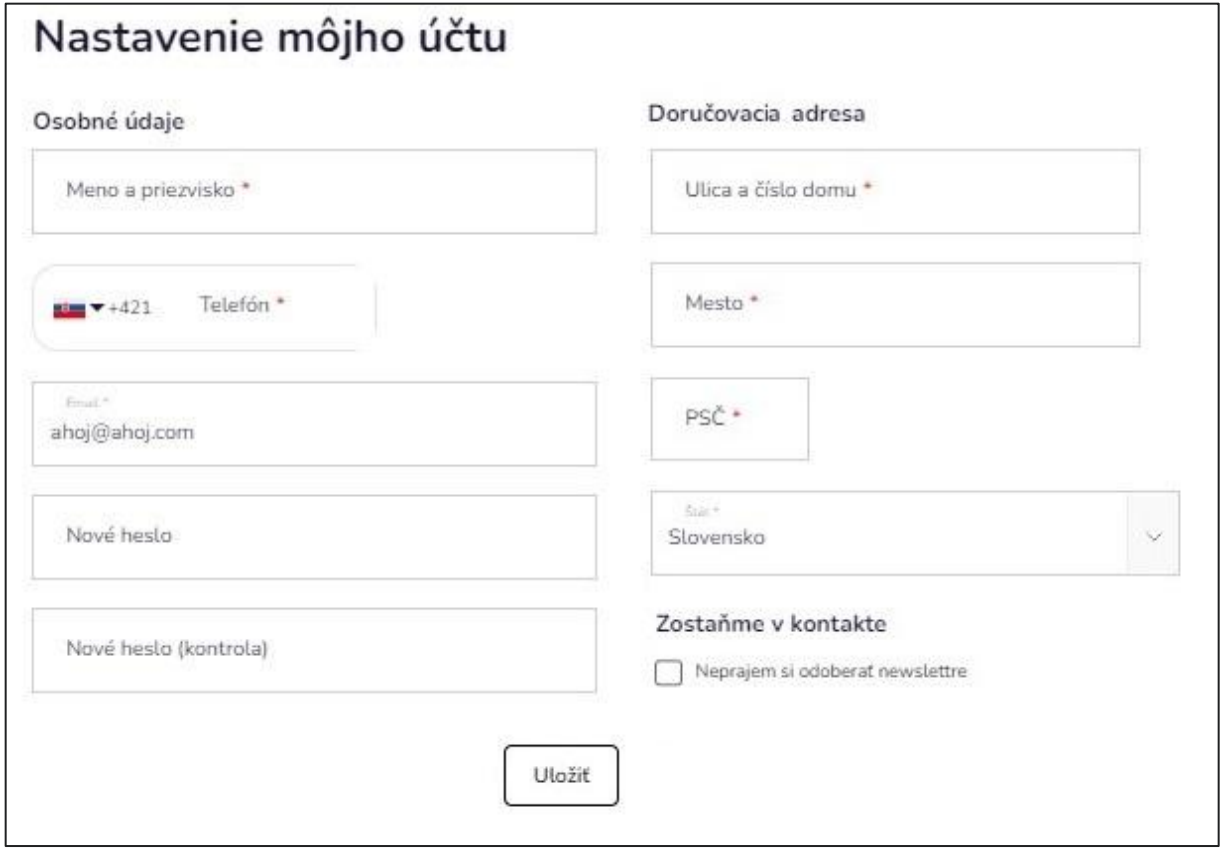
NFR1: The system shall enforce passwords of at least 8 characters. ⟺ **MU1**

MU1:

MU2:


Registrácia
Email *
Heslo *
Opakuj heslo *
Neprajem si odoberať newslettre
Registrovať




Nastavenie môjho účtu
Osobné údaje
Meno a priezvisko *
+421
Telefón *
ahoj@ahoj.com
Nové heslo
Nové heslo (kontrola)
Doručovacia adresa
Ulica a číslo domu *
Mesto *
PSČ *
Slovensko
Zostaňme v kontakte
Neprajem si odoberať newslettre
Uložiť

We'll take a look at ….

Basic approaches

- Requirements - structured text (possibly with tool support)
- Glossary - structured text
- Domain model - UML class diagram
- GUI model - ad-hoc modeling

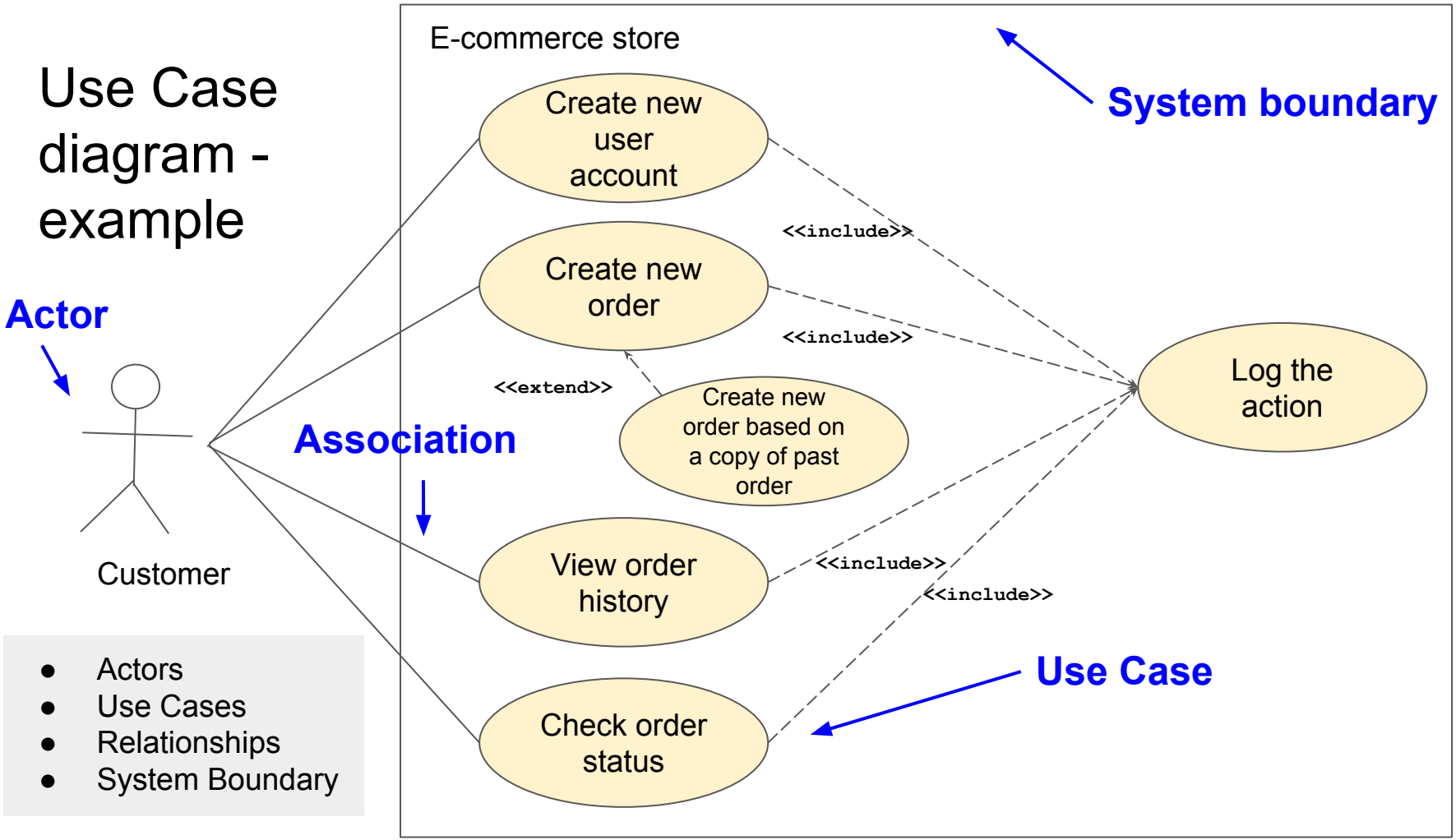
Specific approaches

- Functional requirements - UML use case diagram / model, Activity diagram
- Agile requirements - user stories

Use Case model

- Use Case diagram + Use Case descriptions
- Can be supported by Activity / State / Sequence diagrams (or other

Use Case diagram - example



- Actors
- Use Cases
- Relationships
- System Boundary

Use Case diagram

Use Cases

= functionality of the system

- Inside the system boundary

Actors

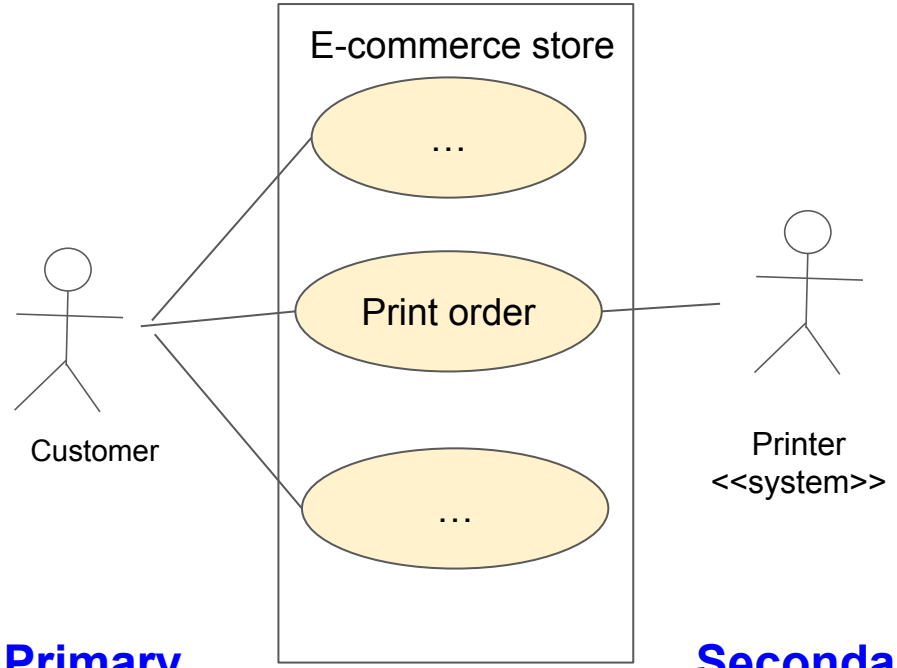
= entities that interacts with the system

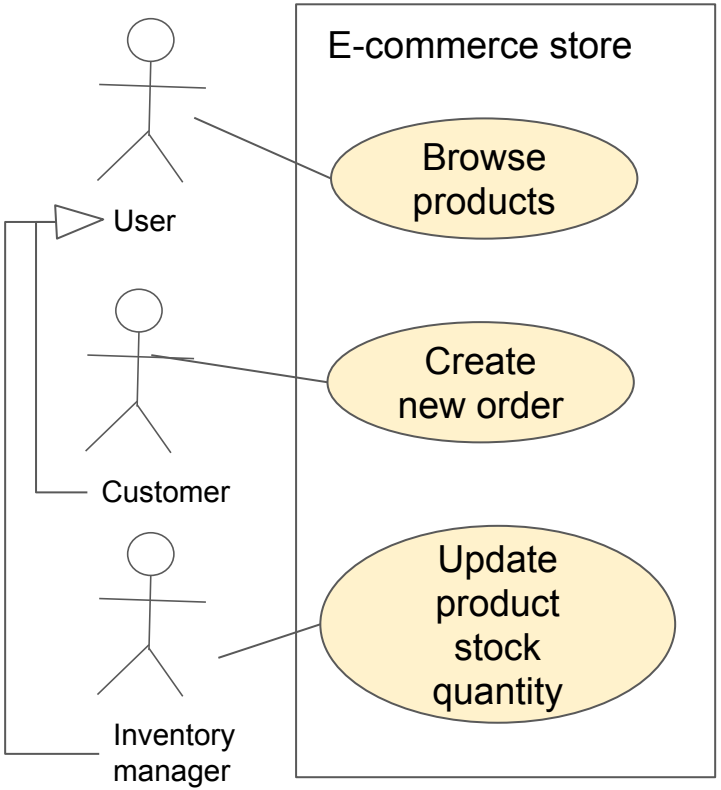
- Always outside the system boundary
- Human users or other systems (<<system>> stereotype)
- Primary vs. secondary actors

Relationships

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

Primary vs secondary actor, actor inheritance





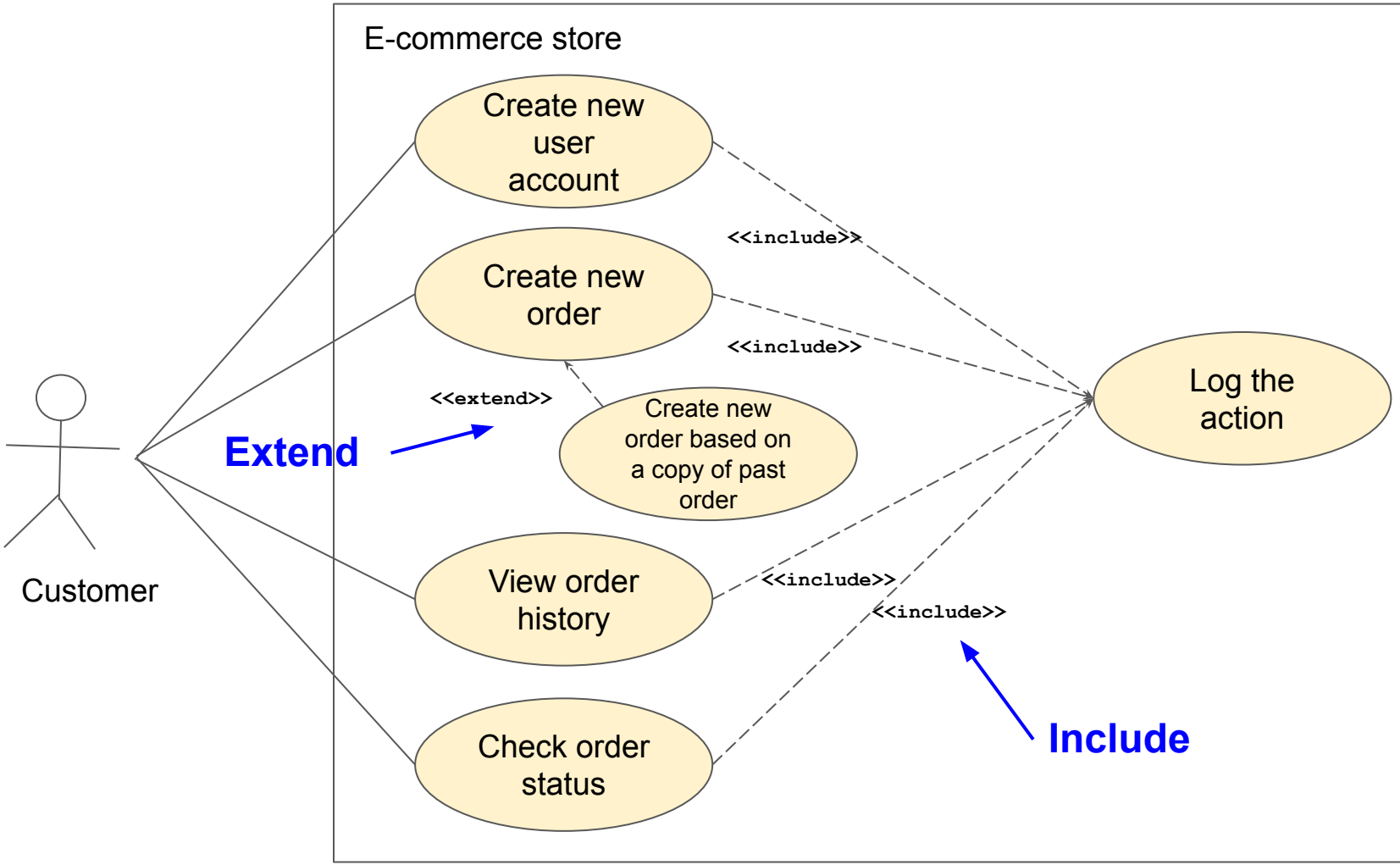
Include vs extend dependency

“Include” dependency

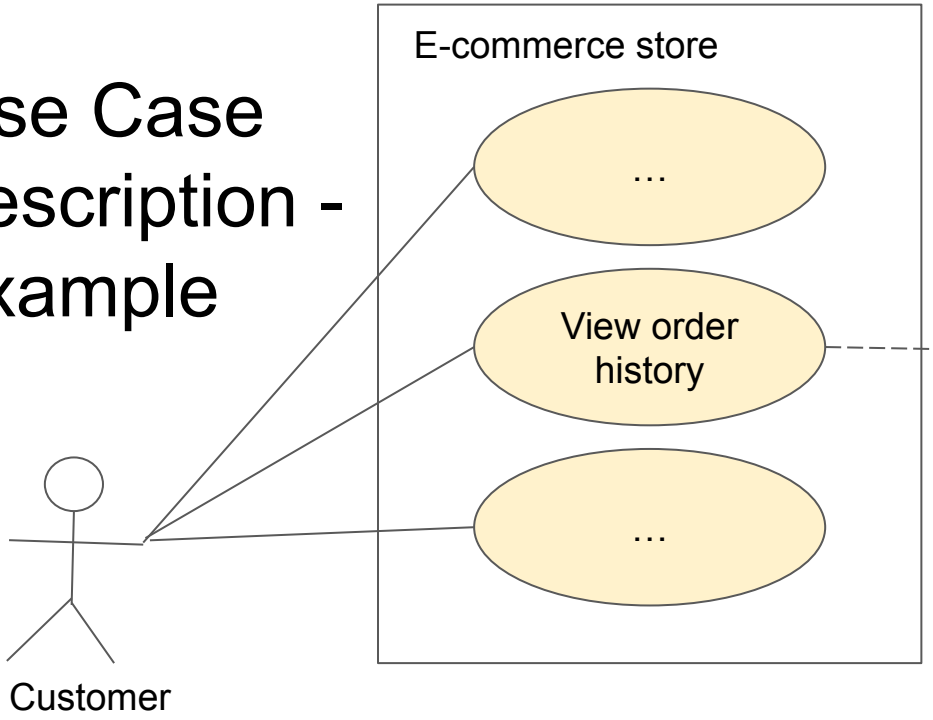
- An including use case always contains the behavior defined in another, included (base), use case. Included use case can be seen as subroutine.

“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
Customer
<<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:
Customer 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 …

- UC diagram - only overview, no details
- UC description - complete specification
- Issues: Might get verbose and difficult to maintain

Activity diagrams

Example (“Checkout” functionality)

We'll take a look at ….

Basic approaches

- Requirements - structured text
- Glossary - structured text
- Domain model - UML class diagram
- GUI model - ad-hoc modeling

Specific approaches

- Functional requirements - UML use case diagram / model, Activity diagram
- Agile requirements - user stories

Agile methodologies

- Software development runs in short, flexible iterations
- Lightweight documentation

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 a part of the 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”

- Somewhere between stakeholder and solution requirements
  - They capture needs from the user’s perspective but often include hints about how the system should behave (especially in acceptance criteria)
- 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, supplemented by acceptance criteria

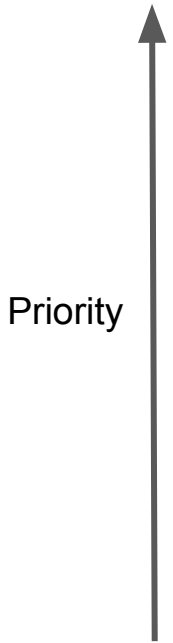
User stories - examples

As a customer
I want to create new user account
so that I do not need to enter my data repeatedly.

As a customer
I want to create new order
so that I can purchase the products I want efficiently.

As a customer
I want to view order history
so that I can easily repurchase items I liked.

As an inventory manager
I want to be notified automatically when stock of a product falls below a defined threshold
so that I can timely order more products from suppliers.



Acceptance criteria

- Conditions that the given feature must fulfill in order to be accepted by stakeholders
- Provides more details about User story

Example:

As a customer **I want to** view order history **so that** I can easily repurchase items I liked.

- Customer can access a page showing all their past orders, sorted by most recent.
- Each order displays order ID, date, status, total amount, and items.
- Items have a “Buy Again” button that adds them to the shopping cart.
- If an item is out of stock, the customer is notified and cannot add it.
- Only the authenticated customer can view their orders.

Tools

- Document-based
  - MS Word / Excel, Google Docs / Sheets, ..
  - Example: Book E-Commerce System (Michigan State University example)
  - Convenient for formal requirement documents (e.g., in case of contracts)
- CASE (Computer-aided software engineering tool)
  - Enterprise Architect,.. EA Screenshots
  - More difficult to create and maintain the specification
  - Provides complete system description
- Collaborative Software, Wiki
  - Atlassian Confluence, .. Medium article with short example
- Agile / issue tracking system
  - Atlassian Jira, Azure DevOps, Trello

Key best practices

- Insist on clear **business requirements**
- Gather requirements from **all stakeholders** and validate requirements with **all stakeholders**
- Only accept **traceable r**equirements
- Only accept requirements that solve **real stakeholder needs**
- Take into account **all types** of requirements
- Avoid **grey zones**
- Document requirements **accurately** and **consistently**

References

- Ian Sommerville: Software Engineering (10th edition), 2016
- Karl Wiegers, Joy Beatty: Software Requirements (3nd Edition), 2013
- Karl Wiegers' checklists & templates: https://softwareregs.com/ -> Downloads
- Dean Leffingwell: Agile software requirements, 2011
- Alistair Cockburn: Writing Effective Use Cases, 2000
- Alistair Cockburn: Agile Software Development (2nd edition), 2006
- A Guide to the Business Analysis Body of Knowledge v3 (BABOK® Guide), 2015
- IEEE Std 29148-2018 (Systems and Software Engineering — Life Cycle Processes — Requirements Engineering), 2018
- Craig Larman: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd edition), 2004
- Martin Fowler: UML Distilled: A Brief Guide to the Standard Object Modeling Language 3rd Edition, 2003