Requirements (2)

Jana Kostičová, 4.10.2023

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

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

- “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.

“Victorian novel”

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**

Combination - structured text

ID	Requirement	Priority	Estimates	…
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			
…	…			

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 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

…

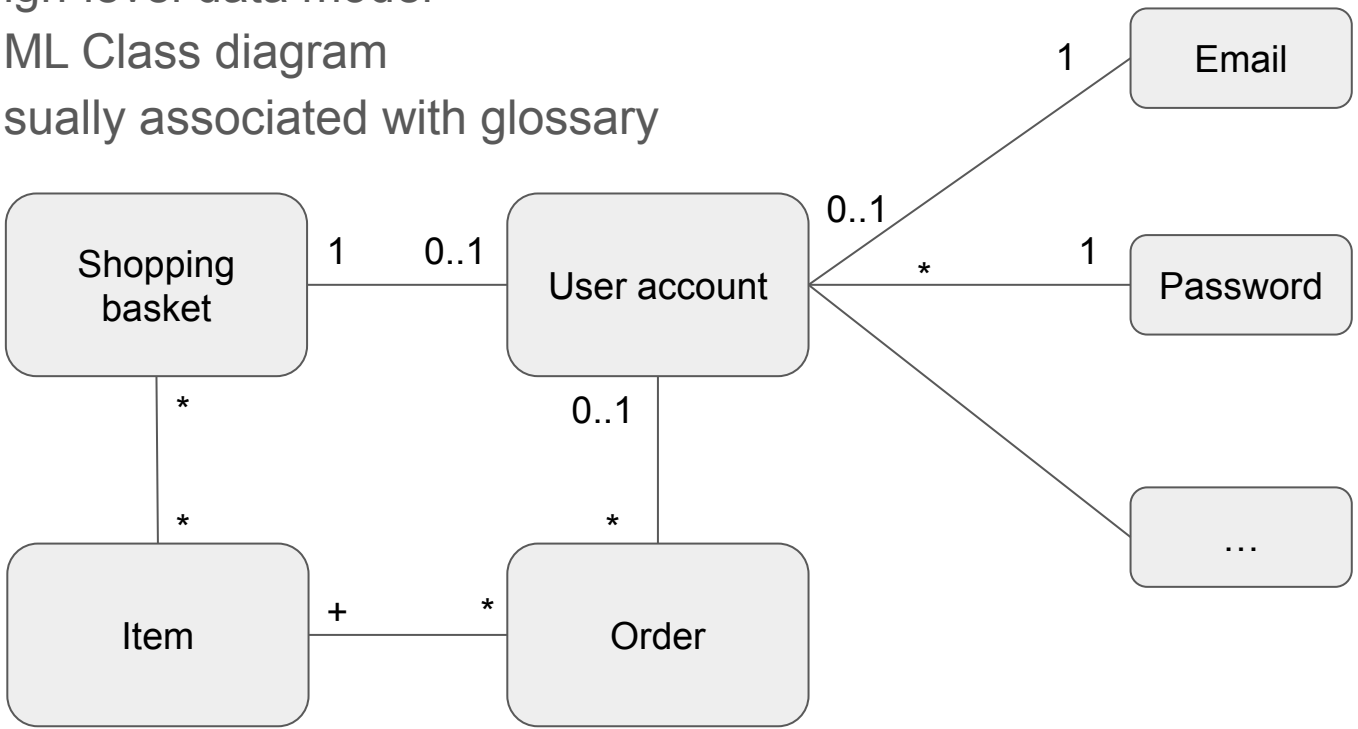
UML

Requirements definition can be supported by UML models, typically

- Use Case model
  - Use Case diagram + Use Case descriptions
- Data model - typically conceptual level
  - Class diagram

Conceptual model

- High-level data model
- UML Class diagram
- Usually associated with glossary

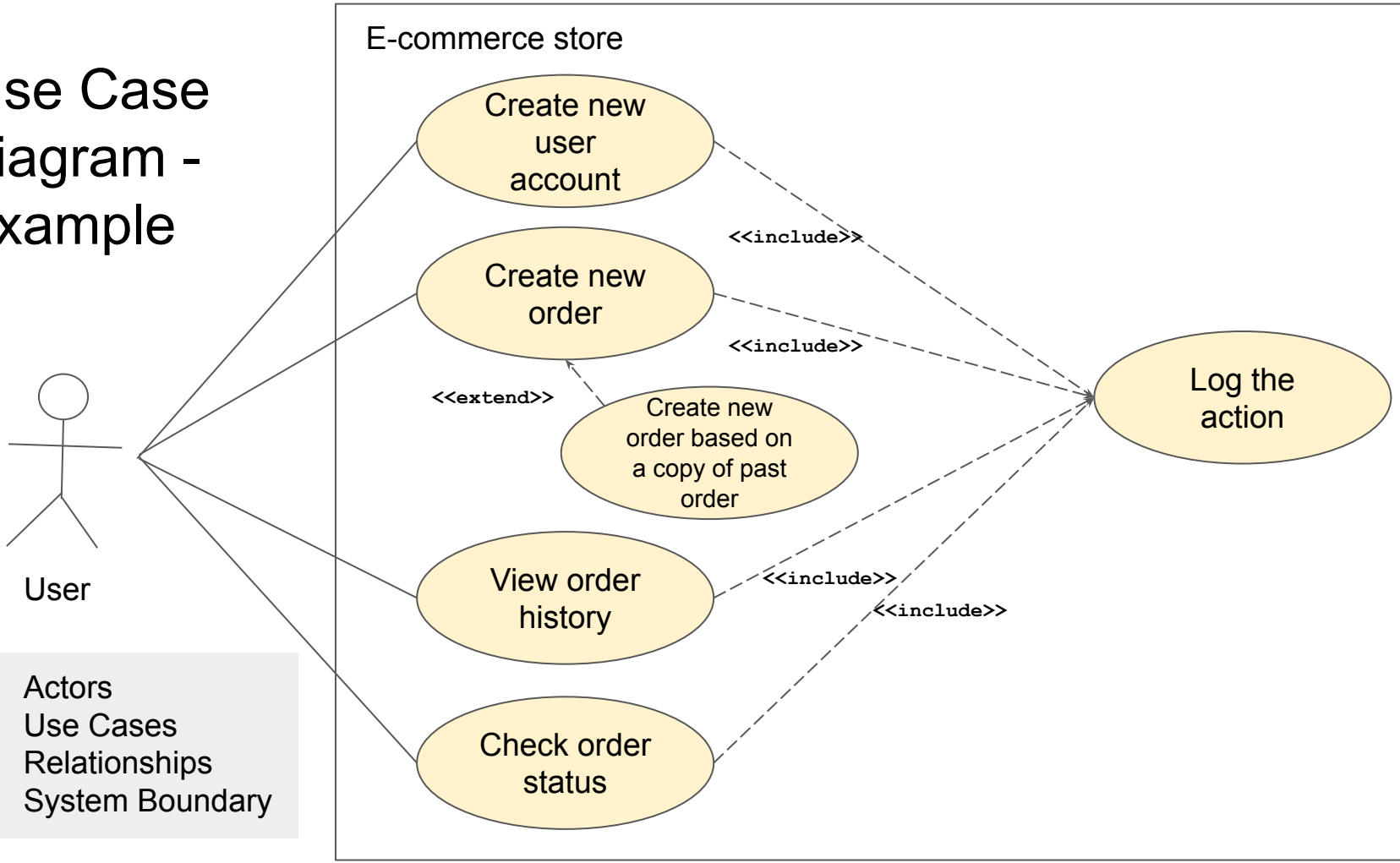


UML

Requirements definition can be supported by UML models, typically

- Use Case model
  - Use Case diagram + Use Case descriptions
- Data model - typically conceptual level
  - Class diagram

Use Case diagram - example



- Actors
- Use Cases
- Relationships
- System Boundary

Use Case diagram

Use Cases

= functionality of the system

- Inside the system boundary

Actors

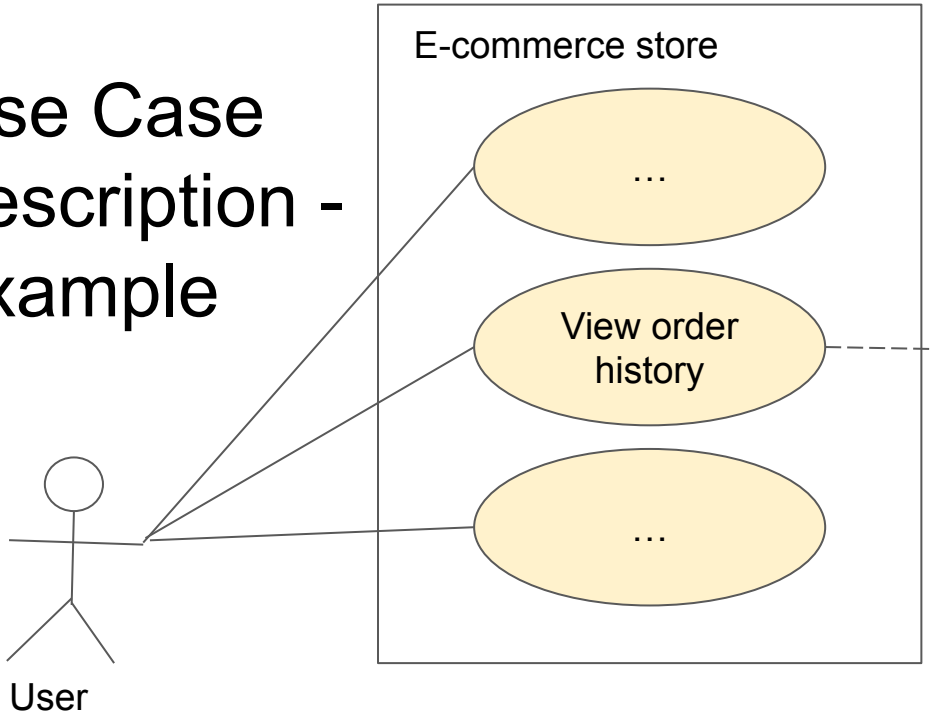
= entities that interacts with the system

- 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”
  - “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

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:

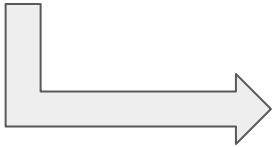
FR1
Create new user account with the following attributes: e-mail address, first name, last name, address line 1, address line 2, city, postal code, phone number,password, timestamp.

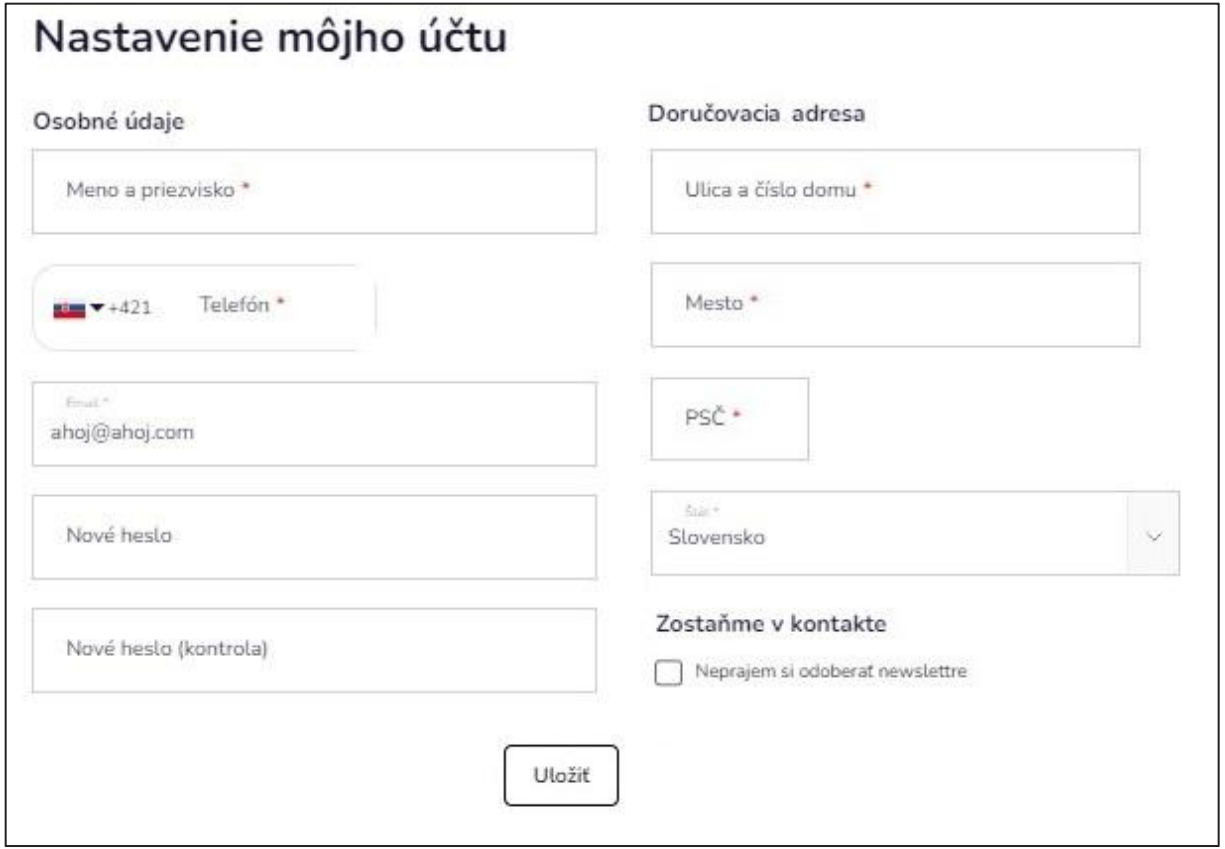
Mock-up: a visual representation or screenshot of how the final website or product will look

MU1:

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

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

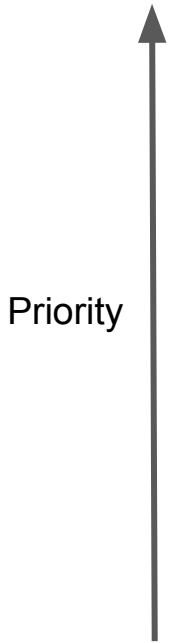
User stories - examples

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.

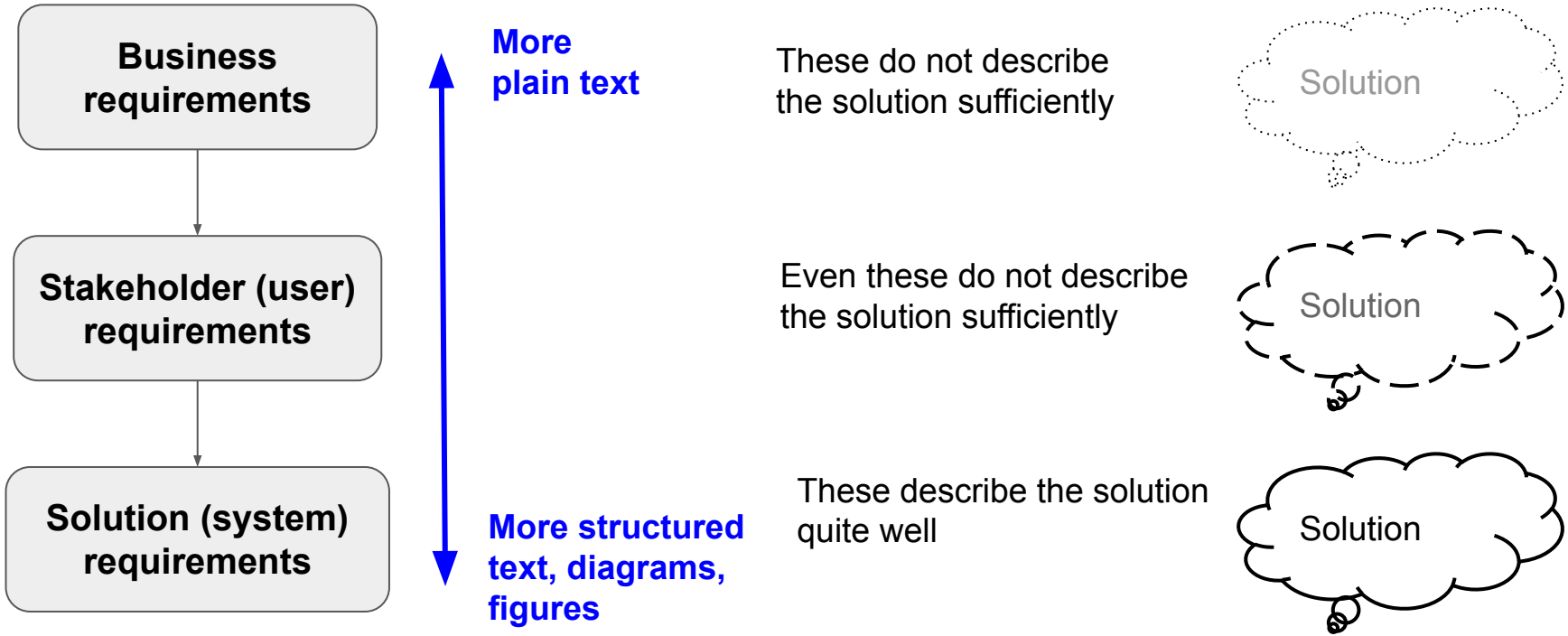
As a user
I want to create new order
so that I buy items.

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.



Three levels of requirements - requirements definition



Tools

- 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)
- 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
- Significant changes in requirements late in the analysis
- Poor traceability

Best practises

- 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)