

Requirements Engineering

Part One of Two

ISEP / LETI / ESOF

Topics

- Definition and Relevance of Requirements
- Functional Requirements Artifacts (Part One)
 - Overview
 - Vision Document
 - Glossary
- User Scenarios
 - User Story (US)
 - Use Case Model / Use Case (UC)
- User Story (US) vs. Use Case (UC)

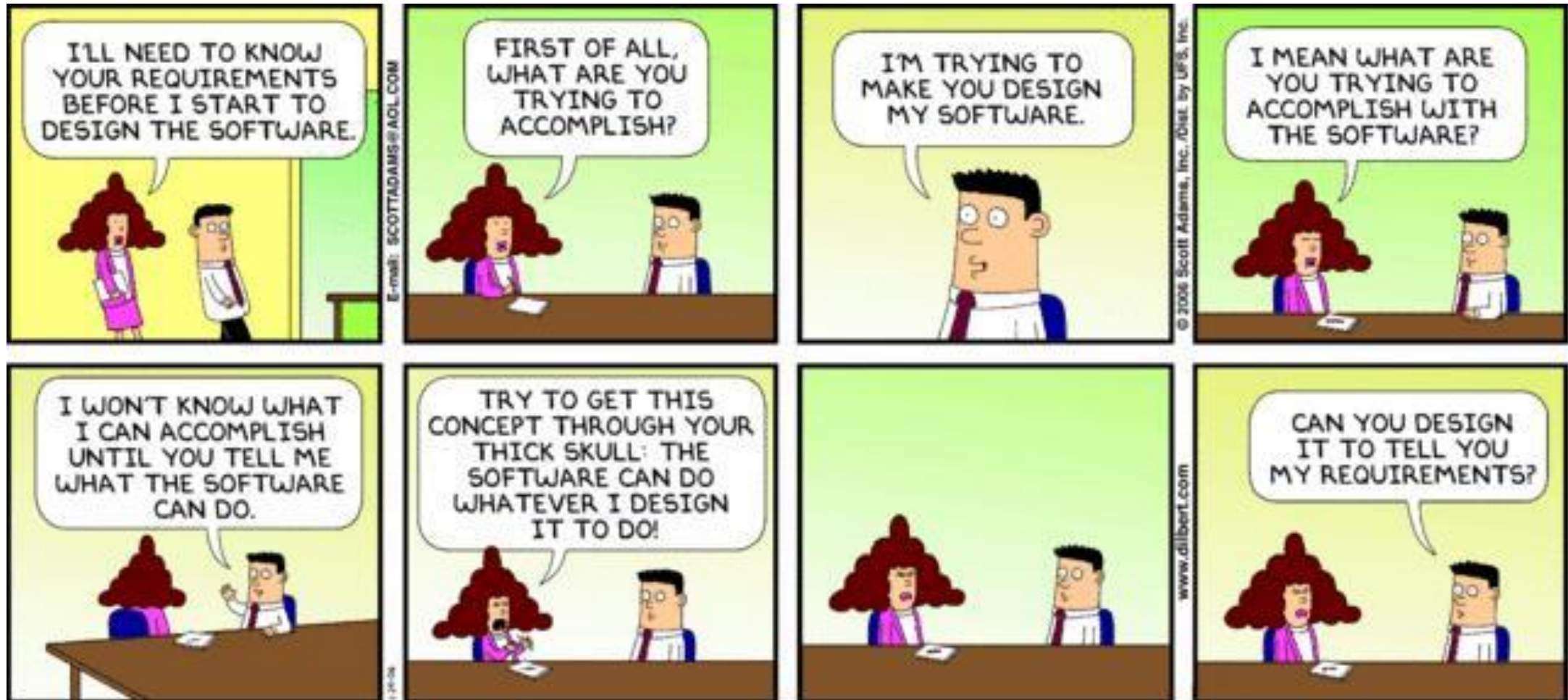
Requirements Definition

Requirements Definition

“Requirements are **capabilities and conditions** to which the system
– and more broadly, the project – must conform.”

Ivar Jacobson

Developing Software without Requirements?



The Relevance of Requirements

“The hardest single part of building a software system is deciding precisely what to build. (...) **No other part of the work so cripples the resulting system if done wrong.** No other part is as difficult to rectify later.”

Fred Brooks

Requirements Engineering

- These are all the **techniques, skills and methods** used to collect Functional and Non-Functional Requirements that the system to be developed must have, so that users (and other parties involved) can achieve their goals
- Main Activities/Tasks:
 - Project creation
 - Elicitation
 - Interpretation and Structuring
 - aka Elaboration and Specification
 - Negotiation
 - Verification & Validation
 - Change Management
 - Tracing



May occur in parallel
or be combined

Gathering Requirements

- It involves communicating with the client, system/end users and other parties having a stake in the software to be developed
- Some techniques:
 - All kind of interviews (e.g. oral vs. written, group vs. individual)
 - Surveys and questionnaires
 - Brainstorming
 - Task analysis and observations
 - Document analysis
 - Domain analysis
 - UI and/or system analysis

Requirements Engineering in ESOFT

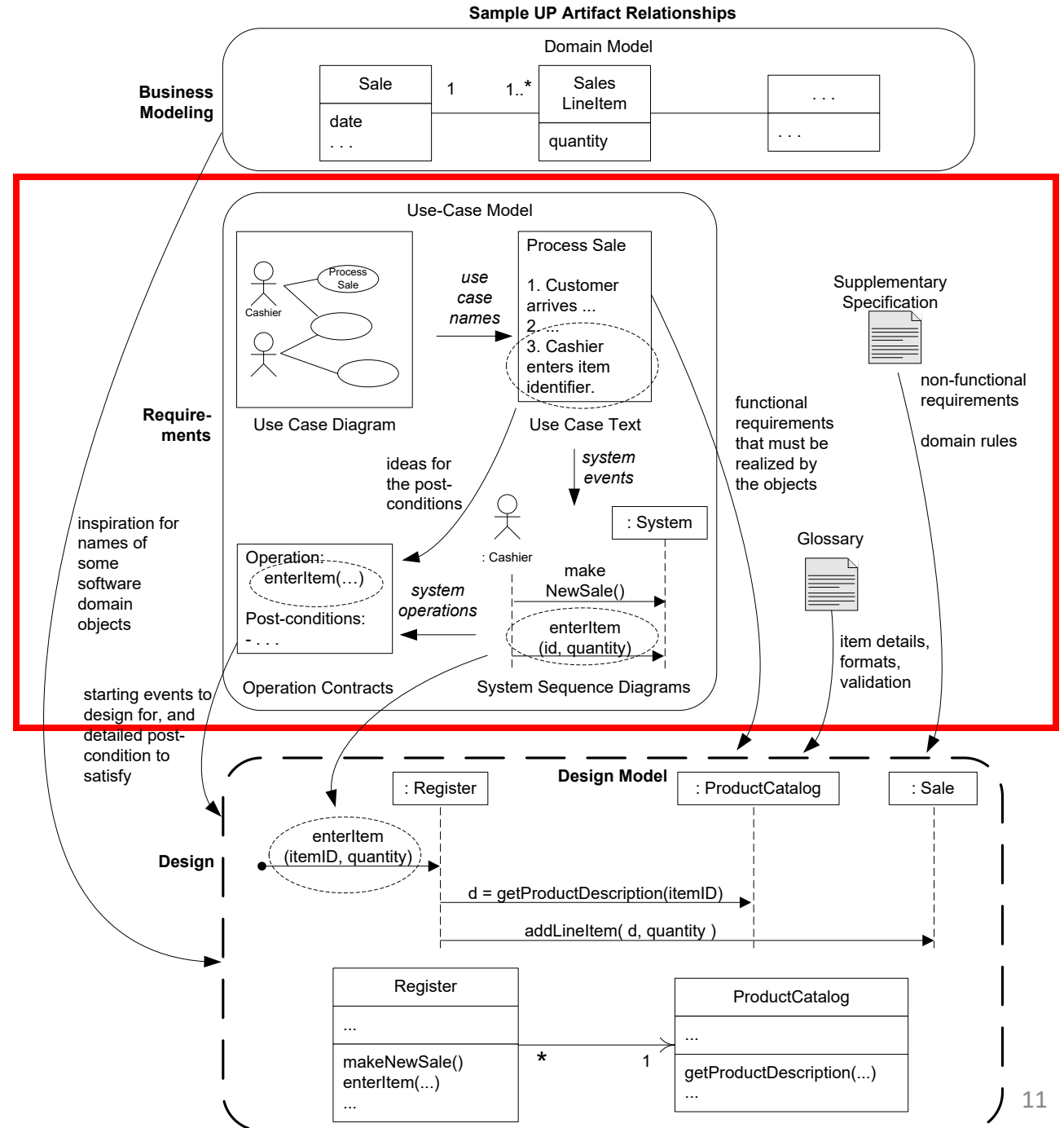


- Activity through which software requirements are:
 - Gathered from the client (Elicitation)
 - Identifying stakeholders
 - Recognizing multiple points of view (e.g. marketing vs. production) and interests
 - Specified (or documented)
 - Formal and informal documents/artifacts
 - Categorizing: functional vs. non-functional
 - Negotiated
 - Addressing conflicting requirements
 - Addressing limited (business) resources
 - Ranking and prioritizing requirements
 - Validated
 - Ready to go?

Artifacts

Overview

Artifacts Overview



Developing Requirements Artifacts

Synergy: each artifact helps to clarify the other

1. Write a brief outline of the **vision document**
2. Identify **user scenarios** and user objectives
3. Write some of the user scenarios
4. Start **supplementary specification**
5. Refine the vision, summarizing and systematizing the information from the previous ones

Common Requirements Artifacts

- Vision Document
- Glossary
- User Scenarios
 - User Stories (US)
 - Use Case Model (UCM)
- Supplementary Specification
- System operation contracts
- ...

Vision Document

Artifacts

Vision Document

It includes the big ideas about:

- Why the project was proposed
- What are the problems
- Who are the stakeholders
- What are the needs
- Perspective of the proposed solution

Vision Document

*NextGen POS
example*

- Review: (version, date, description, author)
- Introduction:
 - We envision a fault-tolerant POS application, with the flexibility to support varying business rules... and the integration of multiple remote/ third party systems
- Positioning:
 - Business opportunity (what are existing POS systems unable to do): ???
 - “Problem Statement” (problems caused by lack of characteristics): ???
 - “Product Position Statement” (for whom the system is, noticeable characteristics, in which it is different from the competitors): ???
- Description of interested parties (distinguish between User and Non-User):
 - Summary: ???
 - Objectives: ???
- Generic Product Description: ???
 - Product perspective: ???
 - Product benefits: ???
 - Cost and Price: ???
 - Licensing and installation: ???
- Summary of system features: ???
- Other requirements and restrictions: ???

Glossary

Artifacts

Glossary

- In its simplest form, it is just a **list of terms** and their meanings in the business scope
 - Interact with the client about the intended meaning of the terms
 - External sources of information might be also used
- Aims to facilitate communication between:
 - Members of the development team
 - Development team and the client
- It can detail any element: an attribute of an object or terms used in other artifacts
 - Although it is only used for terms that are important in the project and whose ambiguity is intended to be reduced or eliminated

Glossary

*NextGen POS
example*

Term or Expression (EN)	Termo ou Expressão (PT)	Description (Definition and Information)
Cashier	Caixa	Employee who operates the POS during the sale.
POS	POS	Acronym for Point-Of-Sale. A kind of cash register with extra features.
Sale	Venda	Business process in which goods are traded with the customer, who pays for the goods.
...



Optional. Used to clarify terminology when the development team uses more than one language (e.g. English, Portuguese)

Glossary – Some Rules

- Glossary terms must be placed in **alphabetical order**
- A term must appear in its **singular form** in the glossary
 - E.g. it should have “**sale**” and not “**sales**”
- Abbreviations must also be included
 - E.g. “POS” as acronym for “Point-Of-Sale”
- Terms with the same meaning must also be in the Glossary
 - In the description of the term itself; or
 - In another entry in the Glossary

Glossary – Starting and Ending

- When to start?
 - It should start very early, but, as with other artifacts, it can and should be modified many times over the project development
- When is it done?
 - New terms may be added over time, but some definitions may also be refined and new details may be added as they become known

User Scenarios for capturing Functional Requirements

User Scenarios

- Aim to capture the goals of the system from the perspective of the application's end-users
 - Focused on what the end-users need to do in their day-to-day job
 - Based on **Roles** and/or *personas*
- An application end-user is a system actor
- **Actor** – something with behavior, like a person, a computer system, or an organization

User Scenarios are usually captured by...

- **User Story** – a short description of a functionality told from the perspective of a user that is valuable to either a user or a customer of the software system

and/or

- **Use Case** – a text story of how the system is used to achieve a certain business objective

User Story (US)

Capturing Functional Requirements

User Story

- A user story is a short, simple description of a feature told from the perspective of the person who desires the new functionality, usually a user or a customer of the system.
- Uses informal natural language and domain/business jargon
- Follows a common and well-known template:
 - *As a <user role>, I <want to do something>.*
 - *As a <user role>, I <want to do something>, so that <benefit>.*
- Examples:
 - As a cashier, I want to process a sale of a customer.
 - As a cashier, I want to handle a product return made by a customer.
- Occasionally, it may be used to express non-functional requirements too.

User Story – More than a Small Text Snippet

- Each user story is comprised by three aspects, aka 3C:
 - **Card** – a written description of the story
 - Usually written on index cards
 - Can be used both for planning and as a reminder
 - **Conversation** – about the story that serves to flesh out the details of the story
 - Values verbal communication with the client
 - Although some notes must be recorded
 - **Confirmation** – Acceptance criteria that can be used to determine when a story is complete

Use Case Model / Use Case (UC)

Capturing Functional Requirements

Use Case Model (UCM)

- A model is an abstraction of something
- Allows some understanding before its construction or modification
- Promotes the understanding and description of requirements (especially the ones involving users)
- The UCM includes:
 - **Use Case Diagram (UCD)**
 - **Use Cases (UC)**, including **System Sequence Diagrams (SSD)**
 - Operation Contracts (defining the system behavior in terms of state changes)

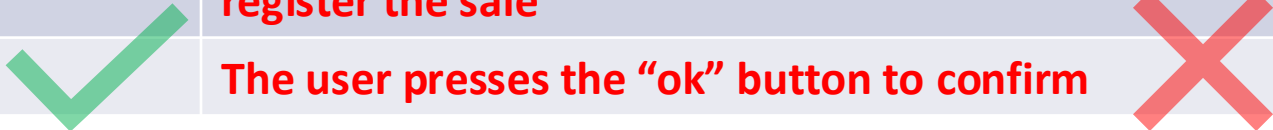
Use Case Model – Some Definitions

- **Use Case**
 - a text story of how the system is used to achieve a certain business objective
- **Scenario**
 - a specific sequence of actions and interactions between an actor and the system
 - a path followed in a use case
 - each use case is a collection of related success and failure scenarios
- **Actor**
 - something with behavior, like as a person, a computer system, or an organization
- **Use Case Model**
 - the set of all use cases developed within the scope of Requirements Engineering activity
 - promotes understanding and description of requirements (especially the functional ones)

Rules for writing Use Cases

- Try to answer the question: “How does the use of the system support users, in a visible way, to achieve their goals?”
- Use cases emphasize functional requirements
- Use cases are text documents, not diagrams
- Use cases based on black boxes are recommended – they describe “**what**” (responsibilities) and **not** “**how**” (procedures)

	what	how
Example 1	The system records the sale	The system executes the following SQL command to register the sale
Example 2	The user confirms	The user presses the “ok” button to confirm



Use Case: Brief Format

*NextGen POS
example*

- It assumes a **high-level point of view**
- Has a Title (ID and name of the UC)
- Has a paragraph describing the main success scenario
- In the Inception phase, write most use cases in this form

Process Sale:

A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a receipt from the system and then leaves with the items.

Use Case: Casual Format

*NextGen POS
example*

- Multiple paragraphs for various scenarios
- In the Inception phase, write some use cases in this form

Process Sale:

Main Scenario: A customer arrives at a checkout with items to purchase...

Alternative Scenarios:

- If the system rejects debit card...
- If the customer does not have sufficient funds...

Use Case: Fully-dressed Format

- Has several sections
 - Primary actor
 - Stakeholders and Interests
 - Preconditions
 - Success Guarantee (or Postconditions)
 - Main Success Scenario (or Basic Flow)
 - Extensions (or Alternative Flows)
 - Special Requirements
 - Technology and Data Variations List
 - Frequency of Occurrence
 - Open Issues
- In the Elaboration phase, write use cases in this format

Fully-dressed Use Case (1/4)

*NextGen POS
example*

- **Primary actor**
 - Cashier
- **Stakeholders and Interests**
 - Cashier: Wants accurate, fast entry, and no payment errors, as cash drawer shortages are deducted from his/her salary.
 - Salesperson: Wants up-to-date sales commissions.
 - Customer: Wants a fast service with minimal effort. Wants an easily visible display of entered items and prices. Wants proof of purchase to support returns.
 - Company: ???
 - Manager: ???
 - Government Tax Agencies: ???
 - Payment Authorization Service: ???
- **Preconditions**
 - Cashier is identified and authenticated.
- **Success Guarantee (or Postconditions)**
 - Sale is saved. Tax is correctly calculated. Accounting and Inventory are updated. Commissions recorded. Receipt is generated. Payment authorization approvals are recorded.

Fully-dressed Use Case (2/4)

NextGen POS
example

- **Main Success Scenario¹ (or Basic Flow)**

1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. The price is calculated from a set of price rules.

Cashier repeats steps 3-4 until indicates done.

5. System presents total with taxes calculated.
6. Cashier tells Customer the total and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).

¹ extended version

Fully-dressed Use Case (3/4)

*NextGen POS
example*

- **Extensions (or Alternative Flows)**

- *a. At any time, the Manager requests an override operation:

- 1. System enters Manager-authorized mode.
 - 2. Manager or Cashier performs one Manager-mode operation (e.g. cash balance change, resume a suspended sale on another register, void a sale).
 - 3. System reverts to Cashier-authorized mode.

- 2-4a.** The Customer informs the Cashier of their tax-exempt status (e.g. seniors, native peoples)

- 1. Cashier verifies and then enters tax-exempt status code.
 - 2. System records status (which it will use during tax calculations).

- 3a.** Invalid item ID (not found in system):

- ...

- 3b.** Item requires manual category and price entry (such as flowers or cards with a price on them):

- 1. Cashier enters special manual category code, plus the price.

- ...

Fully-dressed Use Case (4/4)

*NextGen POS
example*

- **Special Requirements**

- Touch GUI on a large screen. Text must be visible from 1 meter.
- Credit authorization response within 30 seconds 90% of the time.
- Pluggable business rules to be insertable at steps 3 and 7.
- ...

- **Technology and Data Variations List**

- 3a. Item identifier entered by bar code scanner (if bar code is present) or keyboard.
- 3b. Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.
- 7a. Credit account information entered by card reader or keyboard.
- ...

- **Frequency of Occurrence**

- Could be nearly continuous.

- **Open Issues**

- What are the tax law variations?
- What customization is needed for different businesses?
- ...

User Story vs. Use Case

US & UC – Most common relations

- **One UC** is equivalent to **One US (1:1)** and vice-versa
 - UC: Process Sale
 - US: As a cashier, I want to process a sale of a customer.
- **One UC** is equivalent to **Several US (1:*)**, where:
 - Each US represents just one success scenario of the UC
 - UC extensions are also used to describe successful but secondary paths/scenarios
- Example (UC) 1 : * (US):
 - UC: Process Sale
 - **US1**: As a cashier, I want to process a regular sale of a customer.
 - **US2**: As a cashier, I want to tax-exempt an item during the sale processing.
 - **US1** might be considered equivalent to the **UC main scenario**; while
 - **US2** might be considered equivalent to an **extension scenario** of the same UC (cf. extension 2-4a – fully-dressed use case example)

} Adopted in ESOF

US & UC – Similarities and differences (1/3)

- Both are focused on achieving a particular goal for a user
 - **US** is more targeted to capturing who, what and **why of a functionality**
 - **UC** is more targeted to capturing who, what and **how the system plays (the flow)**
 - **US** is more about **user needs** while **UC** is more about **behavior to meet needs**
- Degree of detail
 - **US** are normally, and purposely, **vaguer** → **Lack of details**
 - **US** are meant to **promote** elicit **conversations** with the client
 - **UC** shows **how user and system interact** with each other → **Richer in details**
 - **UC** are a **more structured approach** demanding more up-front details

US & UC – Similarities and differences (2/3)

- Size
 - US/UC are sized to deliver business value
 - **US** are more **suitable for sprints/iterations**
 - **UC** are commonly **split across multiple sprints/iterations**
 - one UC being equivalent to several US
- Communication
 - **UC emphasize written communication**, which is often very imprecise
 - Harder to elaborate → More verbose
 - **US emphasize verbal communication**, becoming easier to clarify something
 - Easier/faster to elaborate → Easier to read


US & UC – Similarities and differences (3/3)

- Longevity
 - US are intended to outlive the sprint in which they are added to the software
 - However, it is possible to archive US cards
 - UC are often permanent artifacts that continue to exist as long as the product is under active development or maintenance
- Usefulness for planning
 - **US** are typically **smaller and easier to estimate** difficulty and time-consuming
 - **UC** are generally **too large** (especially when matching several US) and are therefore **harder to estimate**

When to use US or UC?

- It **depends on** the:

- **Client of the software** to be developed
- **Type and size of software product** to be developed
- **Organization** on which the software will be developed
- Size of the **development team**
- Software **development process** to be adopted



Also, on cultural and personal experience factors of each one involved

- Best Rule of Thumb:

- Have a discussion with all stakeholders involved in the software product
- Check the pros and cons of each one to decide

Why not combine US and UC?

- They are not mutual exclusive
- Each one has their own unique benefits
- **There is no winner** between US and UC comparison
- Combining both might have its payoffs
 - Must be done strategically
 - Extra details provided by UC might be relevant

Afterall, what really matters is choosing whatever helps
to successfully deliver the software product.

Summary

- We've discussed the Requirements Engineering definition and need
- We've gone through some of the artifacts used to represent Functional Requirements
- Next, we'll focus on other artifacts for the same purpose and some that are used for Non-Functional Requirements

Bibliography

- Cohn, M. (2004). Advantages of User Stories Over Requirements and Use Cases. Available on: <https://www.mountangoatsoftware.com/articles/advantages-of-user-stories-for-requirements>
- Eeles, P. (2001). Capturing architectural requirements. Available on: https://www.researchgate.net/publication/329760910_Capturing_Architectural_Requirements
- Fowler, M. (2003). UML Distilled (3rd ed.). Addison-Wesley. ISBN: 978-0-321-19368-1
- Larman, C. (2004). Applying UML and Patterns (3rd ed.). Prentice Hall. ISBN: 978-0-131-48906-6
- Passing, J. (2008). Requirements Engineering in the Rational Unified Process
- Stellman, A. (2009). Requirements 101: User Stories vs. Use Cases. Available on: <https://www.stellman-greene.com/2009/05/03/requirements-101-user-stories-vs-use-cases>