

Requirements Engineering

Functional and Non-Functional Requirements



Digital Skills & Jobs

Topics

- Definition and Relevance of Requirements
- Functional Requirements Artifacts: Overview, Vision Document and Glossary
- User Scenarios
 - User Story (US)
 - Use Case Model / Use Case (UC)
- System Sequence Diagram (SSD)
- Use Case Diagram (UCD)
- Example – Many Labs Project
- Non-Functional Requirements
 - Definition and where to capture
 - Artifact: Supplementary Specification
 - FURPS+ model

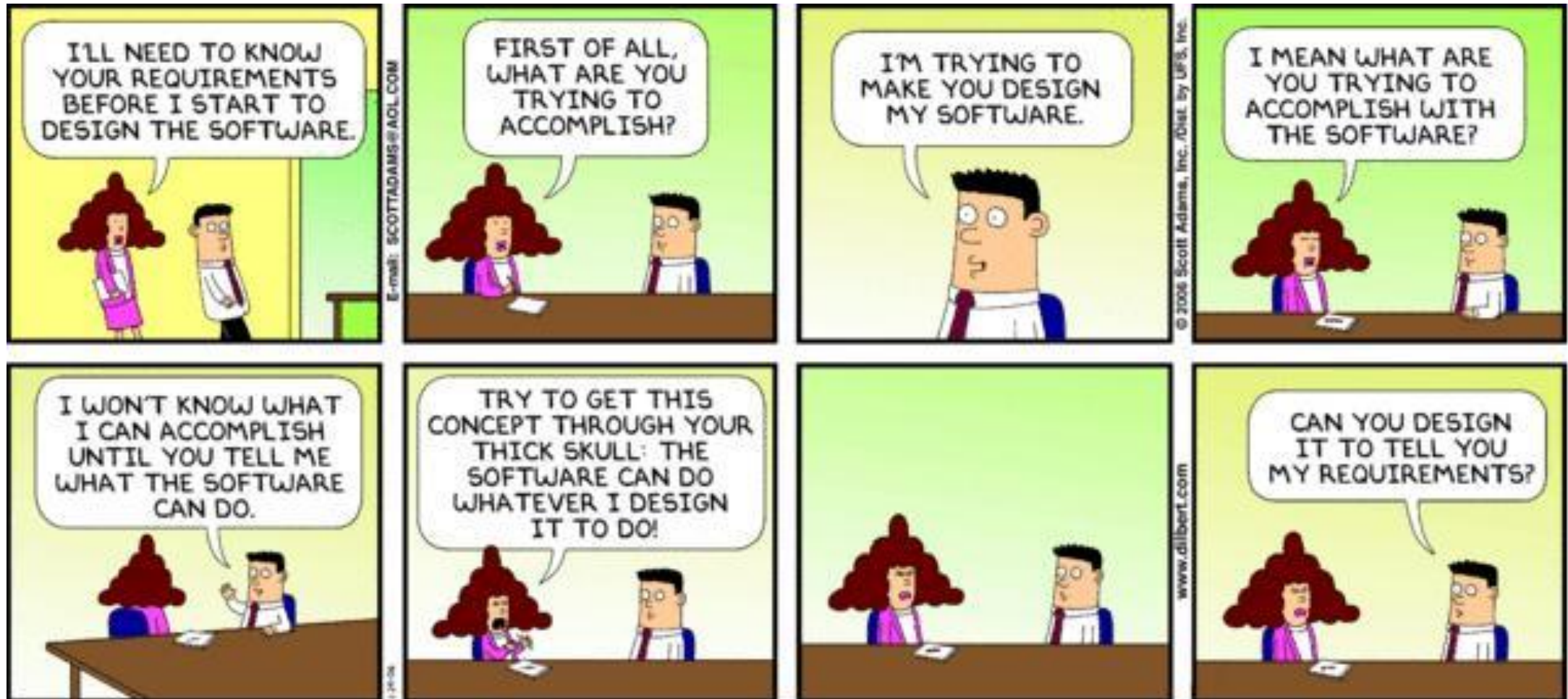
Requirements Definition

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 what to build. No part of the work so cripples the resulting system if done wrong. No other part is more 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

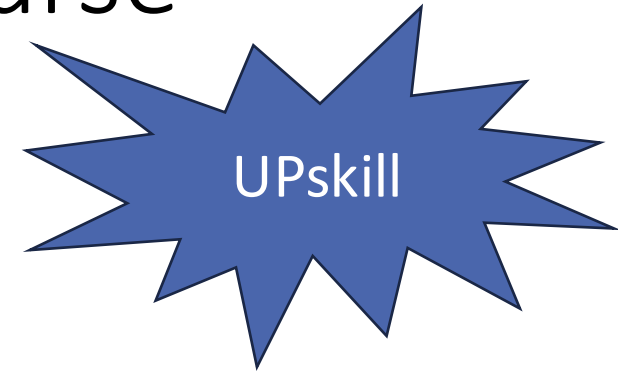


May occur in parallel
or be combined

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 this course

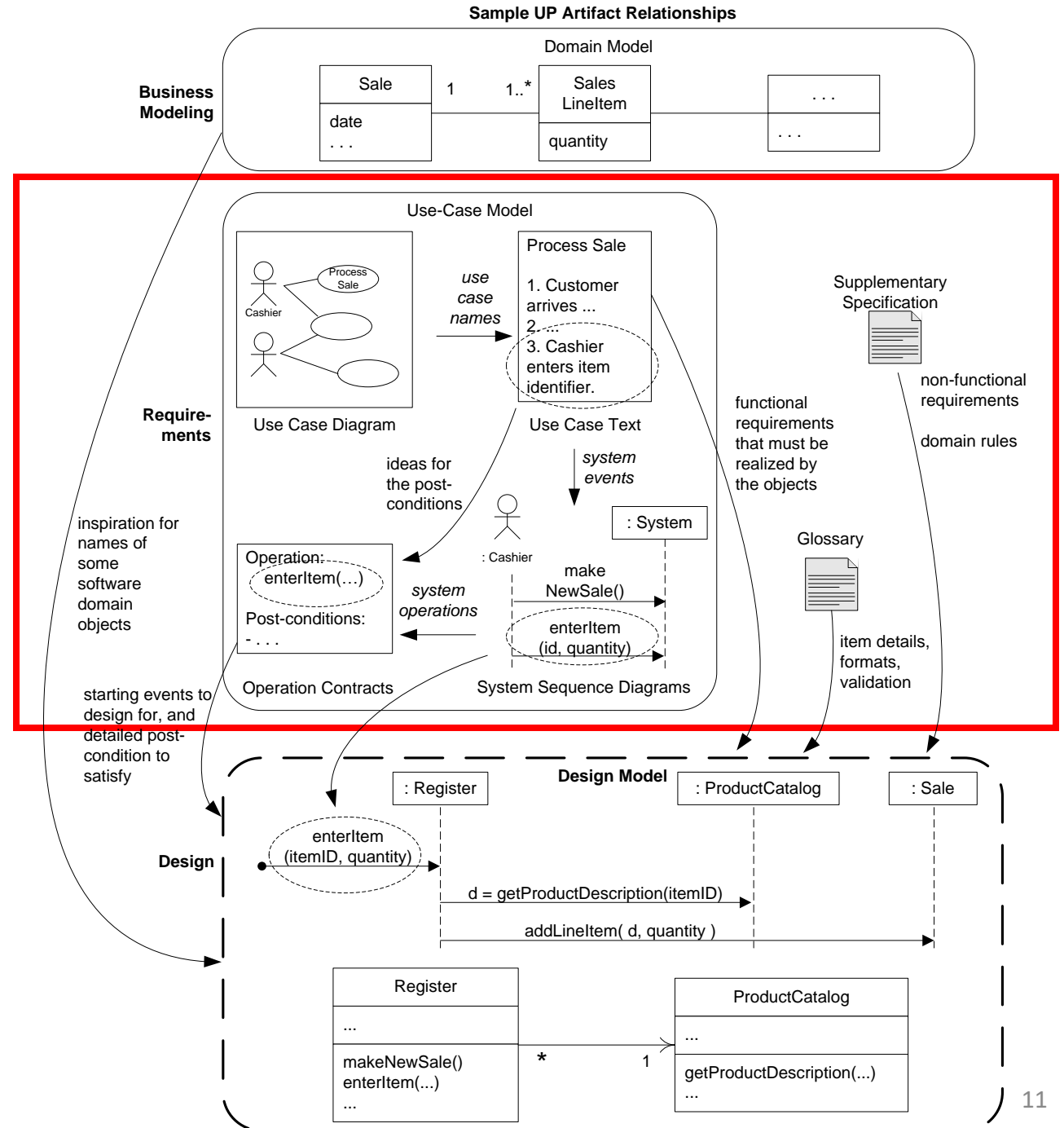


- Activity through which software requirements are:
 - Gathered from the client (Elicitation)
 - Identifying stakeholders
 - Recognizing multiple points of view (e.g. marketing vs. production) and interests
 - Specified (or documented)
 - Formal and informal documents/artifacts
 - Categorizing: functional 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

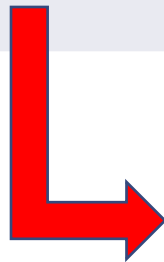
Glossary

- 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 Scenario* (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).

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. Customer tells Cashier they have a 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.
 - **US01** might be considered equivalent to the **UC main scenario**; while
 - **US02** might be considered equivalent to an **extension scenario** of the same UC (cf. extension 2-4a – fully-dressed use case example – on slides from Part One).

} Adopted in this course

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
 - Hard 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.

System Sequence Diagram (SSD)

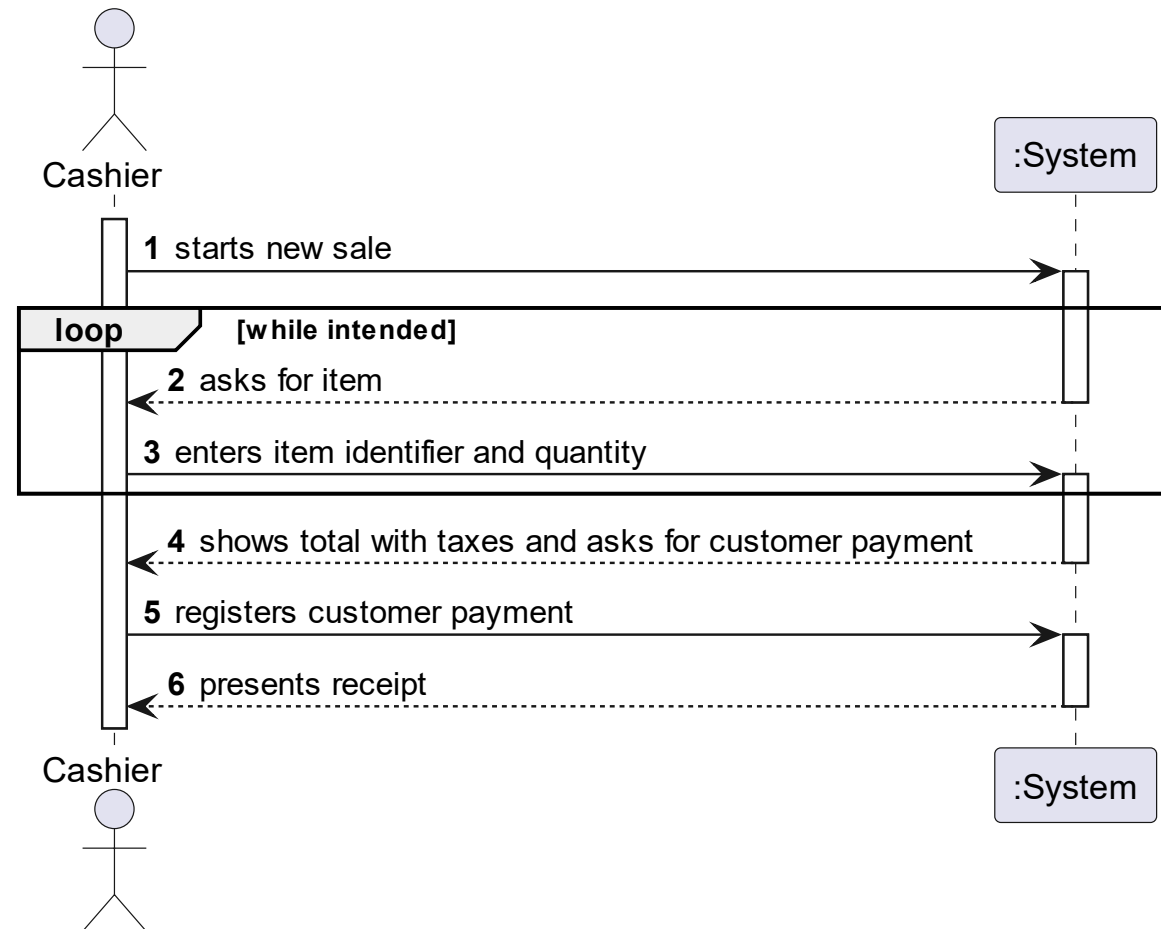
Artifacts

System Sequence Diagram (SSD)

- Use cases describe how actors interact with the software system
- SSD are visualizations of the interactions described in the use cases
 - Follows the UML notation to illustrate the actor's interactions
- SSD is part of the Use Case Model (UCM)
- Given a use case scenario, an SSD illustrates:
 - External actors that interact directly with the system
 - The system as a black box
 - The system events that the actor generates
 - The order of events according to the use case execution flow

SSD – Process Sale

NextGen POS
example



Use Case Diagram (UCD)

Artifacts

Use Case Granularity (1/2)

- What is the correct level for expressing use cases?
- Which use cases should be selected, namely regarding granularity?
 - Negotiate a contract with the supplier?
 - Process returns?
 - Login?
- Low-level use cases are useful when they correspond to repeated subtasks for multiple use cases (e.g. Login)

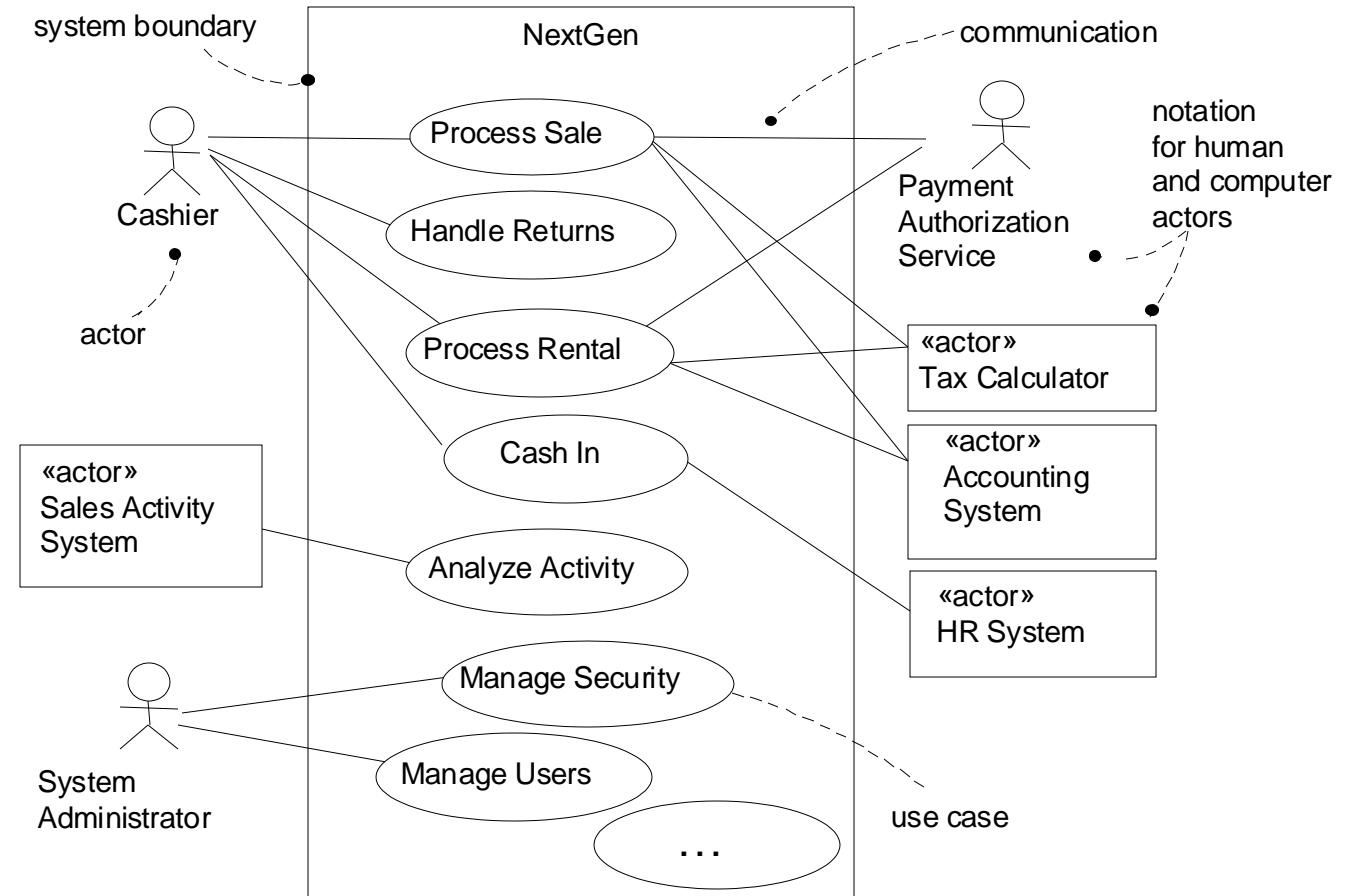
Use Case Granularity (2/2)

- Use cases should focus on the **Elementary Business Processes (EBP)**.
- An EBP is a task:
 - Performed by a person
 - In a specific location
 - At a certain time
 - In response to a business event
 - That adds measurable business value
 - That leaves data in a consistent state

Use Case Diagram (UCD)

*NextGen POS
example*

- Provides a visual perspective of the use cases
- Does not replace the entire text document
- “include” and “extend” relationships are less relevant



Many Labs Project

Requirements Engineering Example

Many Labs Project Specification

Many Labs

- Name: Many Labs – Clinical Analysis Management System
- The following slides:
 - provide some excerpts from the project specification
 - simulate some conversations with the software client (for demo purposes)

Warning: During the project development, conversations must be real and not simulated.

Project Specification Excerpt (1/4)

Many Labs

Many Labs is an English company that has a network of clinical analysis laboratories and that wants an application to manage the clinical analyses performed in its laboratories.

Many Labs is a company that operates in the English market. It has headquarters in London and has a network of clinical analysis laboratories in England where analysis of blood (samples are collected) are performed, as well as Covid-19 tests.

In England, Many Labs has exclusivity for Covid-19 tests throughout the territory, which means that no other company can perform this type of testing. All Many Labs clinical analysis laboratories perform clinical blood tests, and a subset of these laboratories also performs Covid-19 tests.

The set of Many Labs clinical analysis laboratories form a network that covers all England, and it is responsible for collecting samples and interacting with clients.

The samples collected by the network of laboratories are then sent to the chemical laboratory located in the company's headquarters and the chemical analysis are performed there.

Project Specification Excerpt (2/4)

Many Labs

Typically, the client arrives at one of the clinical analysis laboratories with a lab order prescribed by a doctor. Once there, a receptionist asks the client's citizen card number, the lab order (which contains the type of test and parameters to be measured), and registers in the application the test to be performed to that client.

Then, the client should wait until a medical lab technician calls him/her to collect the samples required to perform a given test.

All the tests (clinical blood tests and Covid-19 tests) performed by the network of laboratories are registered locally by the medical lab technicians who collect the samples. The samples are sent daily to the chemical laboratory where the chemical analyses are performed, and results obtained. When sampling (blood or swab) the medical lab technician records the samples in the system, associating the samples with the client/test, and identifying each sample with a barcode that is automatically generated using an external API.

Project Specification Excerpt (3/4)

Many Labs

At the company's headquarters, the clinical chemistry technologist receives the samples (delivered by a courier) and performs the chemical analysis, recording the results in the software application.

Each test is characterized by an internal code, an NHS code, a description that identifies the sample collection method, the date and time when the samples were collected, the date and time of the chemical analysis, the date and time of the diagnosis made by the specialist doctor, the date and time when the laboratory coordinator validated the test, and the test type.

Blood tests are frequently characterized by measuring several parameters which for presentation/reporting purposes are organized by categories. For example, parameters such as the number of Red Blood Cells (RBC), White Blood Cells (WBC) and Platelets (PLT) are usually presented under the blood count (Hemogram) category.

Project Specification Excerpt (4/4)

Many Labs

Covid tests are characterized by measuring a single parameter stating whether it is a positive or a negative result.

Despite being out of scope, the system should be developed having in mind the need to easily support other kinds of tests (e.g. urine). Regardless, such tests rely on measuring one or more parameters that can be grouped/organized by categories.

During system development, the team must:

1. Adopt best practices for identifying requirements and for OO software analysis and design;
2. Adopt recognized coding standards (e.g. CamelCase);
3. Use Javadoc to generate useful documentation for Java code.

All those who wish to use the application must be authenticated.

Glossary – Which are the important terms?

Many Labs

Typically, the client arrives at one of the clinical analysis laboratories with a lab order prescribed by a doctor. Once there, a receptionist asks the client's citizen card number, the lab order (which contains the type of test and parameters to be measured), and registers in the application the test to be performed to that client.

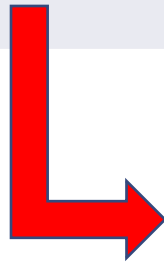
Then, the client should wait until a medical lab technician calls him/her to collect the samples required to perform a given test.

All the tests (clinical blood tests and Covid-19 tests) performed by the network of laboratories are registered locally by the medical lab technicians who collect the samples. The samples are sent daily to the chemical laboratory where the chemical analyses are performed, and results obtained. When sampling (blood or swab) the medical lab technician records the samples in the system, associating the samples with the client/test, and identifying each sample with a barcode that is automatically generated using an external API.

Glossary – *Many Labs example*

Many Labs

Term or Expression (EN)	Termo ou Expressão (PT)	Description (Definition and Information)
Client	Cliente	It refers to the person requesting a clinical analysis test.
MLT	MLT	Acronym for Medical Lab Technician.
Test	Exame	It refers to a clinical analysis test.
...



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

User Stories provided by the SW client

Many Labs

- **US04:** As a **receptionist** of the laboratory, I intend to **register a test** to be performed by a registered client.
- **US05:** As a **medical lab technician**, I want to **record the samples collected** in the scope of a given test.
- **US09:** As an **administrator**, I want to **specify a new type of test** and its collecting methods.
- **US10:** As an **administrator**, I want to **specify a new test parameter** and categorize it.
- **US11:** As an **administrator**, I want to **specify a new parameter category**.

Checking the User Stories

Many Labs

- Three distinct actors can be identified:
 - Receptionist
 - Medical Lab Technician
 - Administrator
- All US are missing 2 out of 3C:
 - Card → Got it!
 - Conversation → To do
 - Confirmation (i.e., the acceptance criteria) → To do
- Lack of details regarding the underlying business processes

US11: As an administrator, I want to specify a new parameter category. (1/4)

Many Labs

- What does the development team already know about it?
 - *“Blood tests are frequently characterized by measuring several parameters which for presentation/reporting purposes are organized by categories. For example, parameters such as the number of Red Blood Cells (RBC), White Blood Cells (RBC) and Platelets (PLT) are usually presented under the blood count (Hemogram) category.”*
 - *“Regardless, such tests rely on measuring one or more parameters that can be grouped/organized by categories.”*

US11: As an administrator, I want to specify a new parameter category. (2/4)

Many Labs

- What does the development team already know about it?
 - *“Blood tests are frequently characterized by measuring several parameters which for presentation/reporting purposes are organized by categories. For example, parameters such as the number of Red Blood Cells (RBC), White Blood Cells (RBC) and Platelets (PLT) are usually presented under the blood count (Hemogram) category.”*
 - *“Regardless, such tests rely on measuring one or more parameters that can be grouped/organized by categories.”*
- What does the development team still need to know? E.g.:
 - Which data characterizes a parameter category?
 - Which business rules apply to such data?

US11: As an administrator, I want to specify a new parameter category. (3/4)

Many Labs

- After some conversations with the software client, suppose that the development team got the following answers:
 - Which data characterizes a parameter category?
 - Just consider a code, a description and an NHS identifier.
 - Which business rules apply to such data?
 - The code must be unique, having 4 to 8 characters.
 - The description cannot be empty and must have a maximum of 40 characters.
 - The NHS identifier is not mandatory.

US11: As an administrator, I want to specify a new parameter category. (4/4)

Many Labs

- After some conversations with the software client, suppose the development team got the following answers:
 - Which data characterizes a parameter category?
 - Just consider a code, a description and an NHS identifier.
 - Which business rules apply to such data?
 - The code must be unique, having 4 to 8 characters.
 - The description cannot be empty and must have a maximum of 40 characters.
 - The NHS identifier is not mandatory.

In this course One User Story is equivalent to One Use Case (1:1) → Use cases are richer than user stories in details → Let's create the corresponding Use Case!

Use Case: Brief Format for UC11 (1/2)

Many Labs

- It assumes a high-level point of view

Id and Name
of the UC



UC11: Create a Parameter Category

UC Successful
Scenario



The administrator starts the definition of a new parameter category. The system asks for the required data (i.e. code, description, and NHS id). The administrator types the requested data. The system validates and presents the data to the administrator, asking him/her to confirm. The administrator confirms. The system records the data and informs the administrator of the operation's success.

Use Case: Brief Format for UC11 (2/2)

Many Labs

- It assumes a high-level point of view

Id and Name
of the UC



UC11: Create a Parameter Category

UC Successful
Scenario



The administrator starts the definition of a new parameter category. The system asks for the required data (i.e. code, description, and NHS id). The administrator types the requested data. The system validates and presents the data to the administrator, asking him/her to confirm. The administrator confirms. The system records the data and informs the administrator of the operation's success.

**The Use Case description should not include details about the system interface.
DO NOT write something like: “The user writes in the text box and clicks the OK button”!**

Use Case: Fully-Dressed Format for UC11 (1/4)

Many Labs

- **Primary actor**
 - Administrator
- **Stakeholders and Interests**
 - Administrator: Wants accurate, fast/easy entry as test parameters cannot be defined without a parameter category.
 - Many Labs: Wants to provide clients with a clear and well-organized/presented test report.
 - Client's Doctor: Wants a test report compliant with NHS regulations.
- **Preconditions**
 - Administrator is identified and authenticated.
- **Success Guarantee (or Postconditions)**
 - Parameter category is saved.

Use Case: Fully-Dressed Format for UC11 (2/4)

Many Labs

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

1. The Administrator starts the definition of a new parameter category.
2. The System asks for the required data (i.e. code, description, and NHS id).
3. The Administrator types the requested data.
4. The System validates and presents the data to the administrator, asking him/her to confirm.
5. The Administrator confirms.
6. The System records the data and informs the administrator of the operation's success.

Use Case: Fully-Dressed Format for UC11 (3/4)

Many Labs

- **Extensions (or Alternative Flows)**

- *a. At any time, the Administrator asks to cancel the creation of the parameter category.

- 1. The use case ends.

- 4a. Mandatory data is missing.

- 1. The System informs which data is missing.

- 2. The System allows the introduction of the missing data (step 3).

- 2a. The Administrator does not change the data.

- The use case ends.

- 4b. The System detects that the entered code already exists in the system.

- 1. The System alerts the Administrator.

- 2. The System allows the Administrator to change the code (step 3).

- 2a. The Administrator does not change the data.

- The use case ends.

...

Use Case: Fully-Dressed Format for UC11 (4/4)

Many Labs

- **Special Requirements**

- Code must be unique, having 4 to 8 characters.
- Description cannot be empty and must have a maximum of 40 characters.
- NHS identifier is not mandatory.

- **Technology and Data Variations List**

- (not identified)

- **Frequency of Occurrence**

- Very occasionally.

- **Open Issues**

- Is there any relationship between parameter categories and test types?
- What is the purpose of the NHS identifier?
- NHS identifier follows any format/structure?

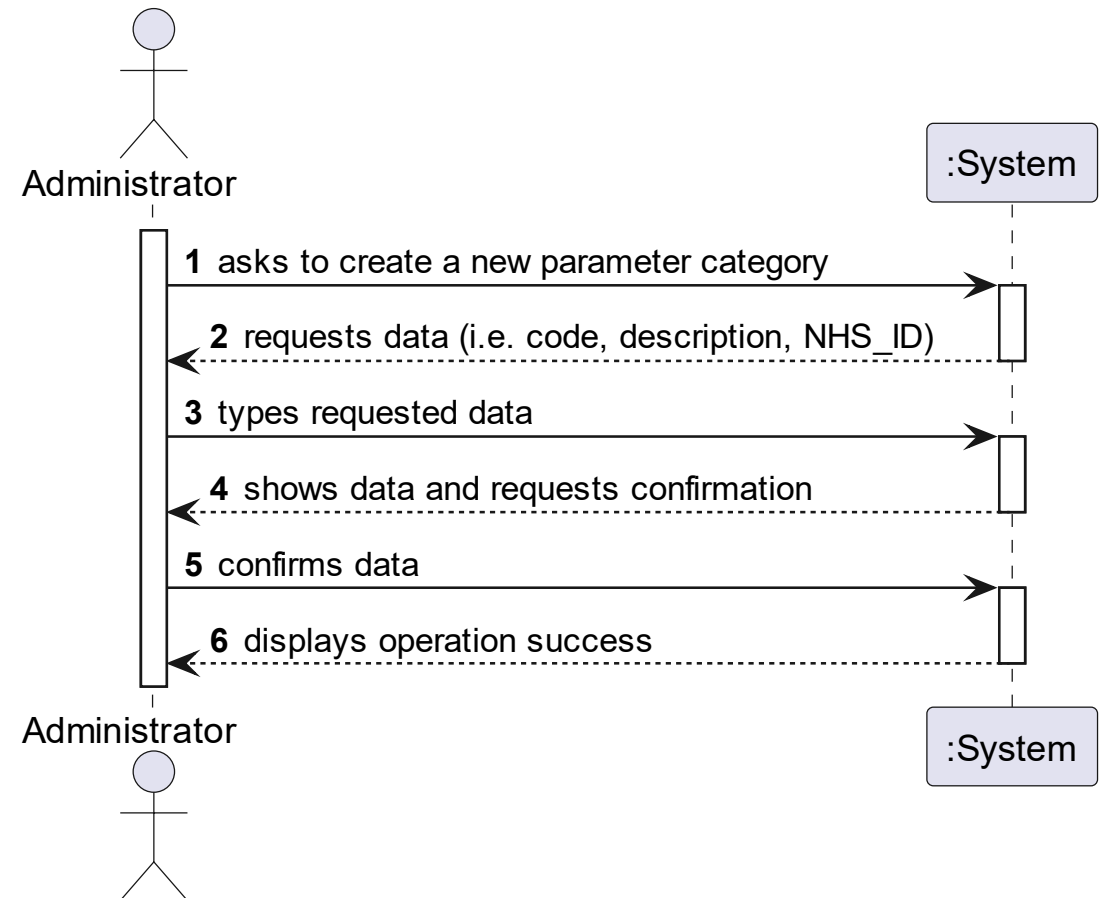
**Fosters more
conversations
with the SW Client**

SSD for UC11: Create a Parameter Category

Many Labs

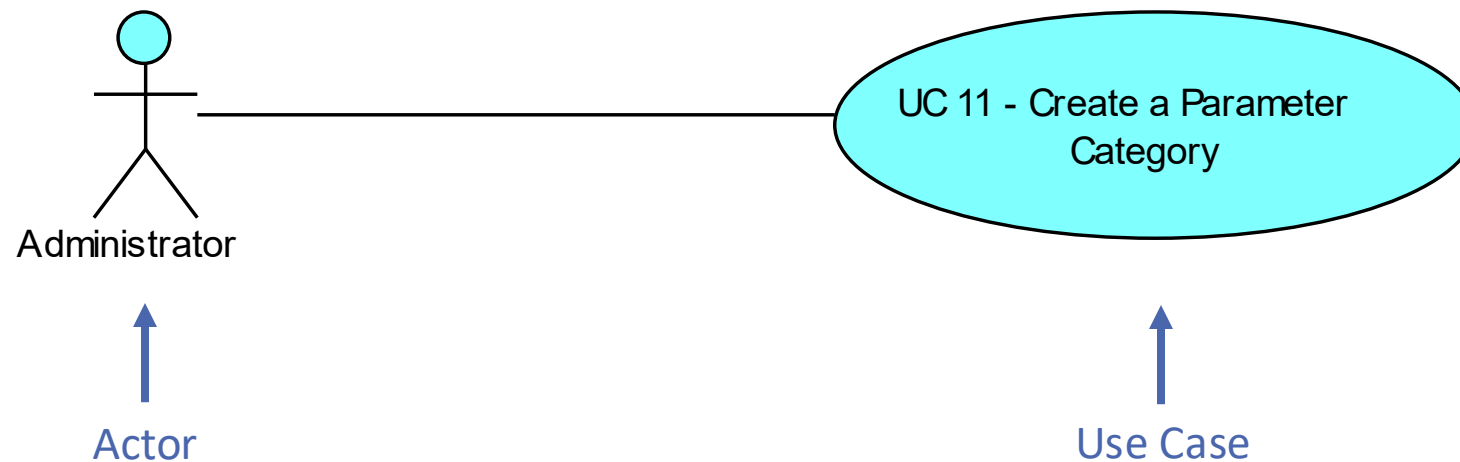
- **Main Success Scenario**

1. Administrator starts the definition of a new parameter category.
2. System asks for the required data (i.e. code, description, and NHS id).
3. Administrator types the requested data.
4. System validates and presents the data to the administrator, asking him/her to confirm.
5. Administrator confirms.
6. System records the data and informs the administrator of the operation's success.



Use Case Diagram (UCD)

Many Labs



PARTIAL VIEW, just with what has been worked so far!

The diagram should include ALL use cases.

Non-Functional Requirements

User Story and Use Case: Capturing Non-Functional Requirements

Non-Functional Requirements Definition

- Requirements that are not functionalities
- Aka **Quality Attributes**, but can also be constraints or business rules
- E.g. performance, reliability, usability

Where to Capture Non-Functional Requirements?

- Is the requirement specific to a given user scenario (US/UC)?
 - **Yes**
 - User Story: do it in the **Acceptance Criteria** section
 - Use Case: do it in appropriate UC sections (e.g. **Special Requirements**)
 - **No**
 - Do it in a **Supplementary Specification** document

Non-Functional Req. on a User Story

*NextGen POS
example*

- Add a section called **Acceptance Criteria** and record:
 - Quality attributes
 - Specific constraints and business rules
 - Variations on how something is done but not on what is done
- Some examples related to US Process Sale
 - AC1: System interaction occurs by means of a touch GUI.
 - AC2: Customer must be able to read text within 1 meter distance.
 - AC3: Credit authorization response within 30 seconds 90% of the time.
 - AC4: Item identifier might be entered by bar code scanner or keyboard.
 - AC5: Item identifier may be any UPC, EAN, JAN, or SKU coding scheme.

Non-Functional Req. on a Use Case

*NextGen POS
example*

- Add a **Special Requirements** section with:
 - Quality attributes
 - Specific constraints and business rules
- Some examples related to UC Process Sale
 - 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.
- Add a **Technology and Data Variations List** section with:
 - Variations on how something is done but not on what is done
- Some examples related to UC Process Sale (cf. fully-dressed use case example, on slides from Part One)
 - 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.

Supplementary Specification: FURPS+

Artifacts

Supplementary Specification

- Captures
 - Requirements not captured as/in user scenarios (US/UC)
 - Non-Functional requirements
 - Some functional requirements that are not an Elementary Business Process
- Organize requirements by categories
- Adopt **FURPS+** model

FURPS+

- FURPS+ is a classification system

Category (EN)	Categoria (PT)		
Functionality*	Funcionalidade*	Quality Attributes	Non-Functional Requirements
Usability	Usabilidade		
Reliability	Confiabilidade		
Performance	Desempenho		
Supportability	Suporte		
+: {implementation, interface, operations, packaging, legal}	+: {implementação interface, operações, empacotamento, legal}		

FURPS+ – Functionality (1/2)

- Includes features typically not captured as/in user scenarios (US/UC)

Function	Description
Auditing	Recording of additional data regarding system execution for audit purposes
Licensing	Adding services related to the acquisition, installation and monitoring of the software license
Localization	Possibility of multiple languages or other aspects related to the use of the software in different geographical points
Email	Adding services related to sending/receiving email
Help	Existence of informative support for users of the system

Adapted from Eeles, 2001.

FURPS+ – Functionality (2/2)

Function	Description
Printing	Facilities related with printing data manipulated by the system
Reporting	Support for generating reports
Security	Controlled access to certain system features or data
System management	Services facilitating application management in a distributed environment
Workflow	Support for managing the status of work items (e.g. what is approved, pending)

Adapted from Eeles, 2001.

FURPS+ – Usability

- Regards/Evaluates the user interface
- It has several subcategories, among them:
 - Prevention of errors entered by the user
 - Adequacy of the interface for different types of users
 - Aesthetics and design
 - Interface consistency

FURPS+ – Reliability

- Refers to integrity and conformity of the software
- Subcategories that can be considered:
 - Frequency and severity of system failures (Availability)
 - Disaster recovery possibility (Recoverability)
 - Accuracy of some calculus

FURPS+ – Performance

- Regards/Evaluates features related to:
 - Response time
 - System setup time
 - System start-up time
 - System shutdown time
 - System recovery time
 - System availability
 - Memory consumption
 - CPU usage
 - Load/Usage capacity
 - ...

FURPS+ – Supportability

- Regards/Evaluates characteristics concerned with:
 - Testability
 - Adaptability
 - Maintainability
 - Compatibility
 - Configurability
 - Installability
 - Scalability
 - Localizability
 - ...

FURPS+ – Others (+)

- Groups additional categories typically related with constraints
 - **Design**
 - specifies or constrains the options for designing a system (e.g. standards/patterns, development tools)
 - **Implementation**
 - specifies or constrains the coding or construction of a system (e.g. implementation languages, operating system)
 - **Interface**
 - specifies external items with which a system must interact, or constraints on formats or other factors used within such interaction
 - **Physical**
 - specifies a physical constraint imposed on the hardware used to house the system (e.g. material, size, weight)

Summary

- We've discussed the Requirements Engineering definition and need
- We've gone through some of the artifacts used to represent Functional Requirements
- The level of detail in the requirements specification generally increases throughout the project, at least for some of the requirements
- "It is certainly a myth that the requirements for large software projects are ever perfectly understood or perfectly specified."

Abran et al., 2001

References & Bibliography

- Larman, Craig; Applying UML and Patterns; Prentice Hall (3rd ed.); ISBN 978-0131489066
- Eeles, P. (2005). Capturing architectural requirements. Available on <http://www.ibm.com/developerworks/rational/library/4706.html>.
- Abran, A., Bourque, P., Dupuis, R., & Moore, J. W. (2001). Guide to the software engineering body of knowledge-SWEBOK. IEEE Press.
- Mike Cohn (2004). Advantages of User Stories for Requirements. Available on <https://www.mountangoatsoftware.com/articles/advantages-of-user-stories-for-requirements>
- Andrew, Stellman(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/>