

Gathering and Specifying Requirements

Part Two of Two

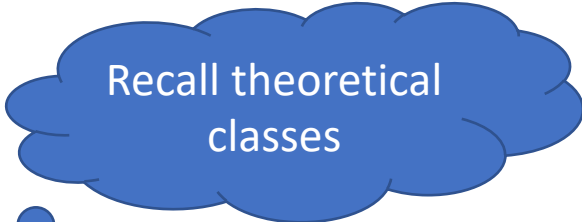
Topics

- Continuation of Functional Requirements Artifacts
 - System Sequence Diagram (SSD)
 - Use Case Diagram (UCD)
 - Use Case Model (UCM)
- Non-Functional Requirement Artifacts
 - Supplementary Specification – FURPS+
- Recommended Approach for the Project Development

Artifacts

System Sequence Diagram (SSD)

System Sequence Diagram (SSD)



Recall theoretical classes

- 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
- 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 follows the Use Case order

Example of an SSD for a Brief Format UC

UC 11 - Create a Parameter Category

The administrator starts the definition of a new parameter category.

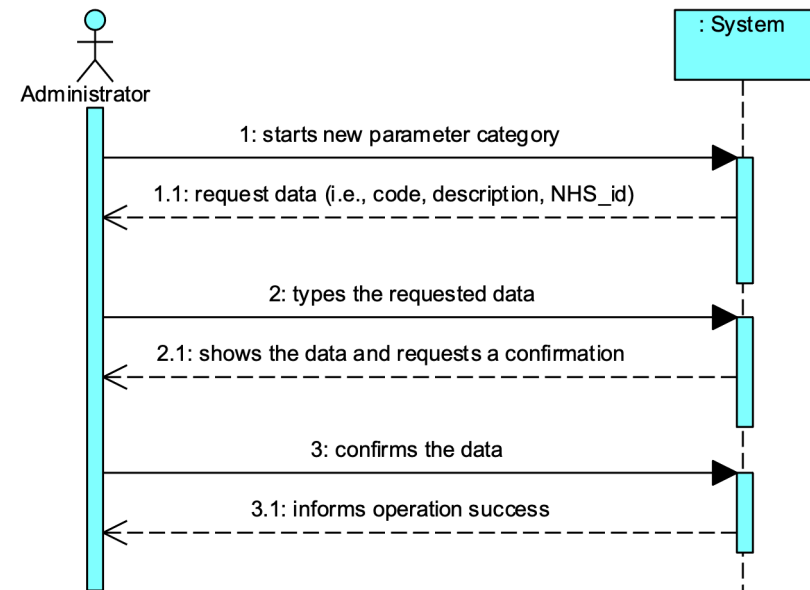
The system requests the required data (i.e., code, description, and NHS id).

The administrator types the requested data.

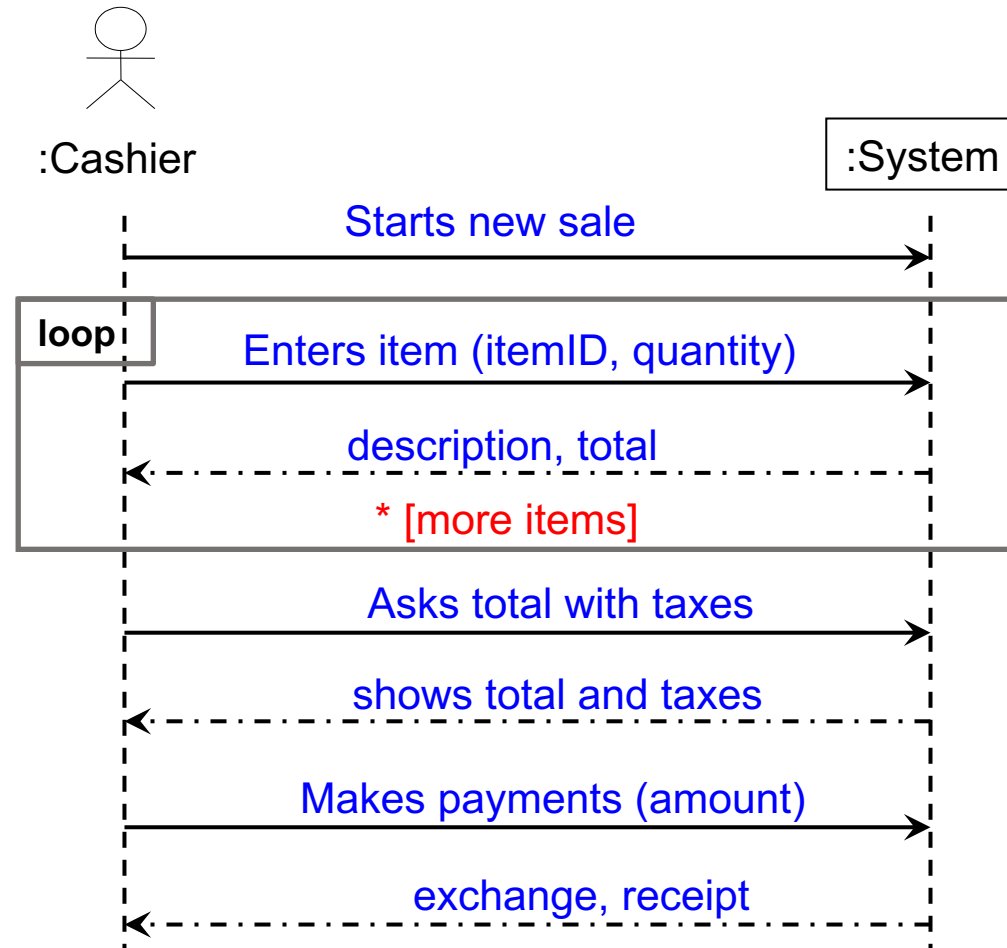
The system validates and presents the data to the administrator, asking her/him to confirm.

The administrator confirms. The system records the data and informs the administrator of the operation's success.

Visual Paradigm Standard (paulomaio(Instituto Superior de Engenharia do Porto))



SSD – A Loop Example

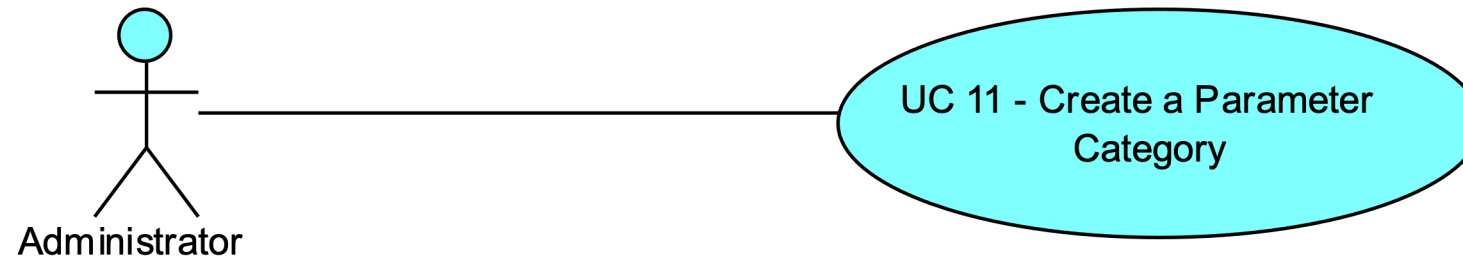


Artifacts

Use-Case Diagram (UCD)

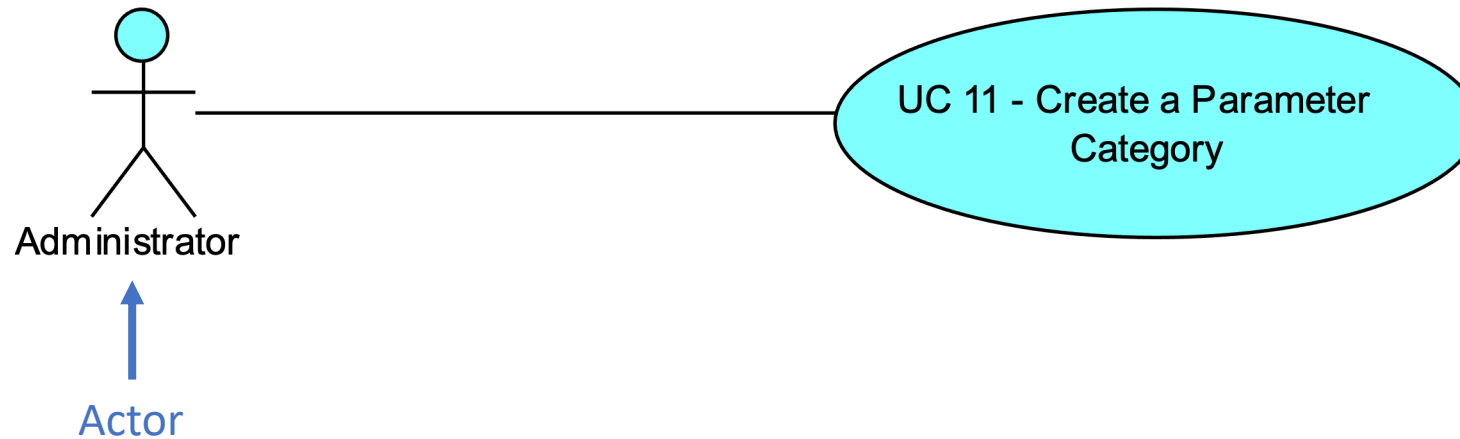
Use Case Diagram (UCD)

- It serves to provide a visual perspective of the use cases
- It does not replace the entire text document



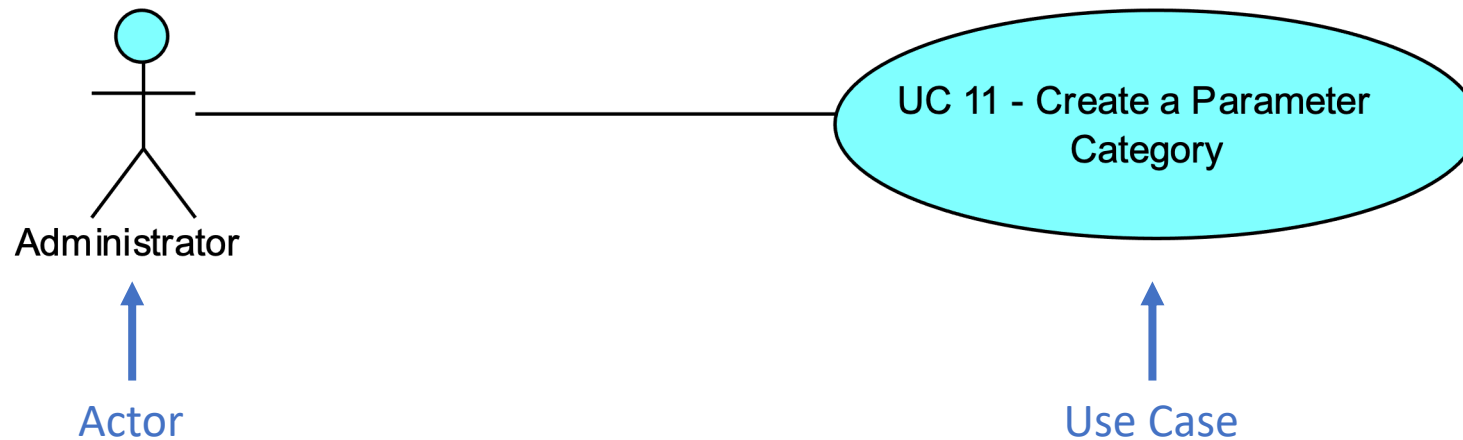
Use Case Diagram (UCD)

- It serves to provide a visual perspective of the use cases
- It does not replace the entire text document



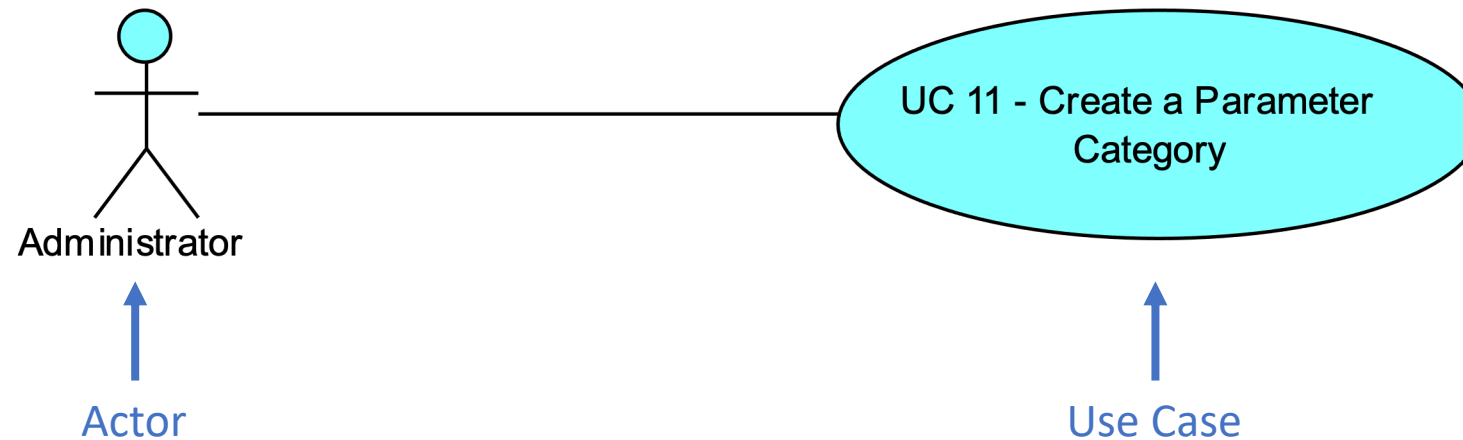
Use Case Diagram (UCD)

- It serves to provide a visual perspective of the use cases
- It does not replace the entire text document



Use Case Diagram (UCD)

- It serves to provide a visual perspective of the use cases
- It does not replace the entire text document

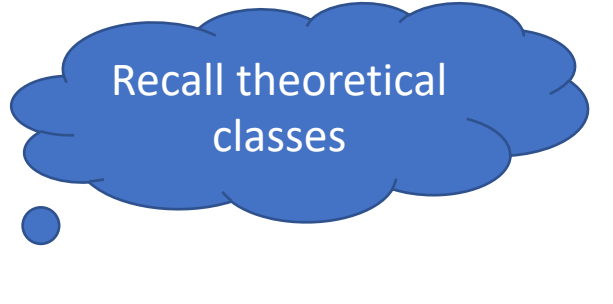


Partial view, just with what has been worked so far! The diagram should include all use cases.

Artifact

Use-Case Model (UCM)

Use-Case Model (UCM)



Recall theoretical classes

- A model is an abstraction of something. It allows some understanding before its construction or modification
- It promotes the understanding and description of requirements (especially the ones involving users). It includes in particular:
 - Use Case Diagram (UCD)
 - Use Cases (UC)
 - Sequence System Diagrams (SSD)
 - Operation Contracts

Exercise

Exercise

- Check US 10
 - US10: As an administrator, I want to specify a new test parameter and categorize it.
- Create the corresponding Use Case

Exercise: US 10 → UC 10

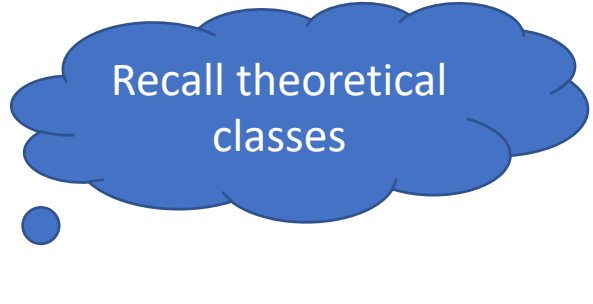
- What the development team already knows about it?
 - ???
- What the development team still needs to know?
 - ???
- From the conversations with the SW client, what did you find out?
 - (in this exercise, you can fake a few conversations based on the business context)
 - ???
- Elaborate
 - UC in the fully-dressed format
 - SSD corresponding to the main success scenario of the UC
- Update Use-Cases Diagram

To do.

Artifacts

US and UC: Capturing Non-Functional Requirements

Non-Functional Requirements



Recall theoretical classes

- A.k.a. Quality Attributes
- But it might be also constraints and/or business rules
- E.g., performance, reliability, usability
- **Where to capture non-functional requirements?**
 - Is the requirement specific of a given user scenario?
 - **Yes**
 - User Story: do it in an Acceptance Criteria section
 - Use Case: do it in a UC section (e.g., special requirements)
 - **No**
 - Do it in a Supplementary Specification document

Non-Functional Requirements: US vs. UC

- On a User Story

- At the “Acceptance Criteria” section
 - Quality attributes
 - Specific constraints and/or business rules
 - Variations on **how** something is done but **not** on **what** is done

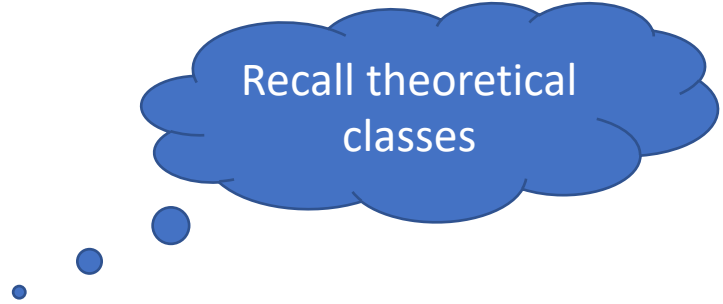
- On a Use Case

- At the “Special Requirements” section
 - Quality attributes
 - Specific constraints and/or business rules
- At the “Technology and Data Variations List” section
 - Variations on **how** something is done but **not** on **what** is done

Artifacts

Supplementary Specification

Supplementary Specification



Recall theoretical classes

- Captures
 - Requirements not captured as/in User Scenarios
 - Non-Functional requirements
 - Some functional requirements that are not an Elementary Business Process
- Organize requirements by categories
- Adopt **FURPS+** model

FURPS+

Recall theoretical classes

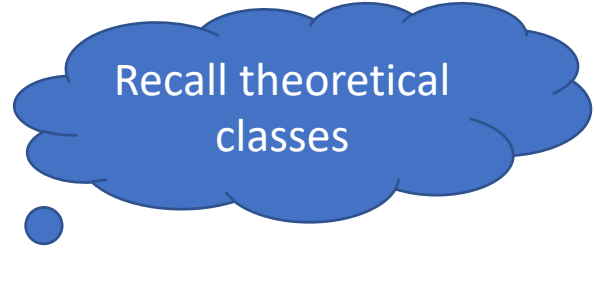
- It is a classification system

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

Quality Attributes

Non-Functional Requirements

FURPS+ – Functionality (1/2)



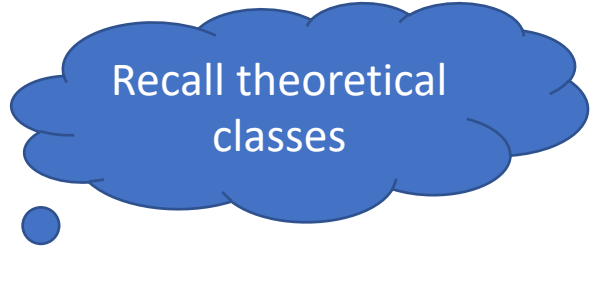
Recall theoretical classes

- Features typically not captured in the user scenarios

Function	Description/Example
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, 2005).

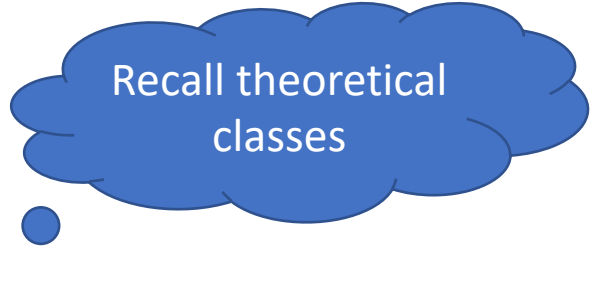
FURPS+ – Functionality (2/2)



Recall theoretical classes

Function	Description/Example
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)

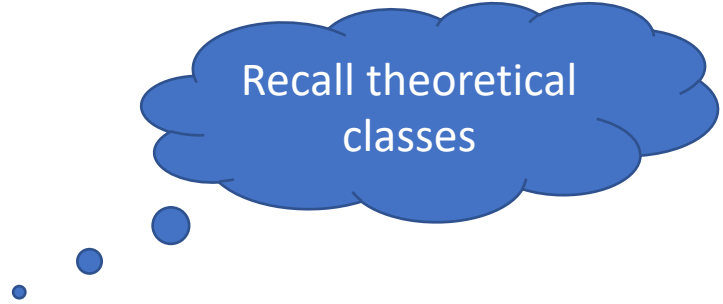
FURPS+ – Usability



Recall theoretical classes

- 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

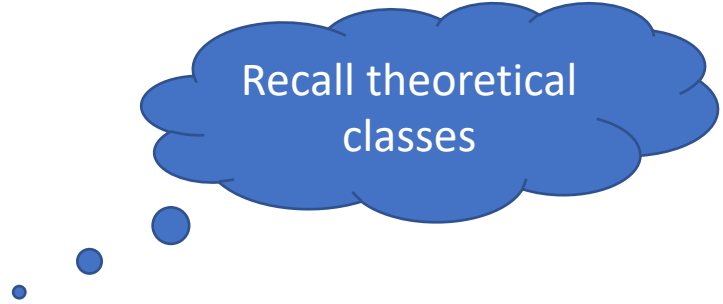
FURPS+ – Reliability



Recall theoretical classes

- Refers to
 - Integrity
 - Conformity
- The requirements that can be considered are
 - Frequency and severity of system failures
 - Disaster recovery possibility
 - Accuracy of some calculus

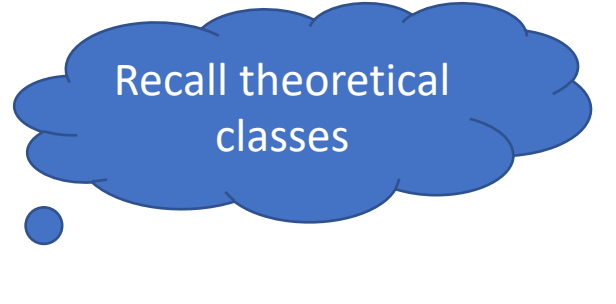
FURPS+ – Performance



Recall theoretical classes

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

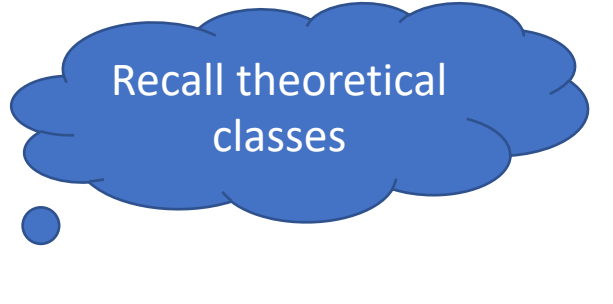
FURPS+ – Supportability



Recall theoretical classes

- Regards/Evaluates characteristics concerned with (e.g.,)
 - Testability
 - Adaptability
 - Maintainability
 - Compatibility
 - Configurability
 - Installability
 - Scalability
 - Localizability

FURPS+ – Others (+)



Recall theoretical classes

- Groups additional categories typically related with constraints
 - Design
 - specifies or constrains the options for designing a system
 - Implementation
 - specifies or constrains the coding or construction of a system
 - Interface
 - specifies an external item with which a system must interact, or constraints on formats or other factors used within such an interaction
 - Physical Requirements
 - specifies a physical constraint imposed on the hardware used to house the system

Exercise

FURPS+

Supplementary Specification of the Project in Development

- Functionality
 - Security
 - “All those who wish to use the application must be authenticated.”
 - ...
 - ...
- Usability
 - ???
 - ...
- Reliability
 - ???
 - ...

To do

Supplementary Specification of the Project in Development (cont.)

- Performance
 - ???
 - ...
- Supportability
 - ???
 - ...
- Others (+)
 - Design
 - ???
 - ...
 - Implementation
 - ???
 - ...

To do

Requirements – Approach for the Project Development

A Pragmatic and Simplified Approach

For each US/UC:

1. Recall the US (Index Card) → The 1st C
2. Collect existing information about the US
3. Analyze the US and systematize what you still need to know
4. Summarize and record conversations with the SW client
5. Record Acceptance Criteria (explicit and implicit) → The 3rd C
6. Find out dependencies for other US/UC (which US must be working in order to this one have success) → Helps planning the work
7. Clearly identify input and output data. Distinguish between typed data and data that can/should be input by selection
8. Elaborate a plausible SSD for the US

} The 2nd C

→ Avoiding UC verbosity without losing too much

The Approach in Practice

1. Recall the US

2. Collect existing information about the US

3. Analyze the US and systematize what you still need to know → Interact with the client

4. Summarize and record conversations with the SW client

US 006 - To create a Task

1. Requirements Engineering

1.1. User Story Description

As an organization employee, I want to create a new task in order to be further published.

1.2. Customer Specifications and Clarifications

From the specifications document:

Each task is characterized by having a unique reference per organization, a designation, an informal and a technical description, an estimated duration and cost as well as the its classifying task category.

As long as it is not published, access to the task is exclusive to the employees of the respective organization.

From the client clarifications:

Question: Which is the unit of measurement used to estimate duration?

Answer: Duration is estimated in days.

Question: Monetary data is expressed in any particular currency?

Answer: Monetary data (e.g. estimated cost of a task) is indicated in POTs (virtual currency internal to the platform).

The Approach in Practice (cont.)

5. Record Acceptance Criteria
(explicit and implicit)

6. Find out dependencies for other
US/UC (which US must be working
in order to this one have success)

7. Clearly identify input and output
data related to the US. Distinguish
between typed data and data that
can/should be input by selection
(not typed).

1.3. Acceptance Criteria

- **AC1:** All required fiels must be filled in.
- **AC2:** Task reference must have at least 5 alphanumeric chars.
- **AC3:** When creating a task with an already existing reference, the system must reject such operation and the user must have the change to modify the typed reference.

1.4. Found out Dependencies

- There is a dependency to "US003 Create a task category" since at least a task category must exist to classify the task being created.

1.5 Input and Output Data

Input Data:

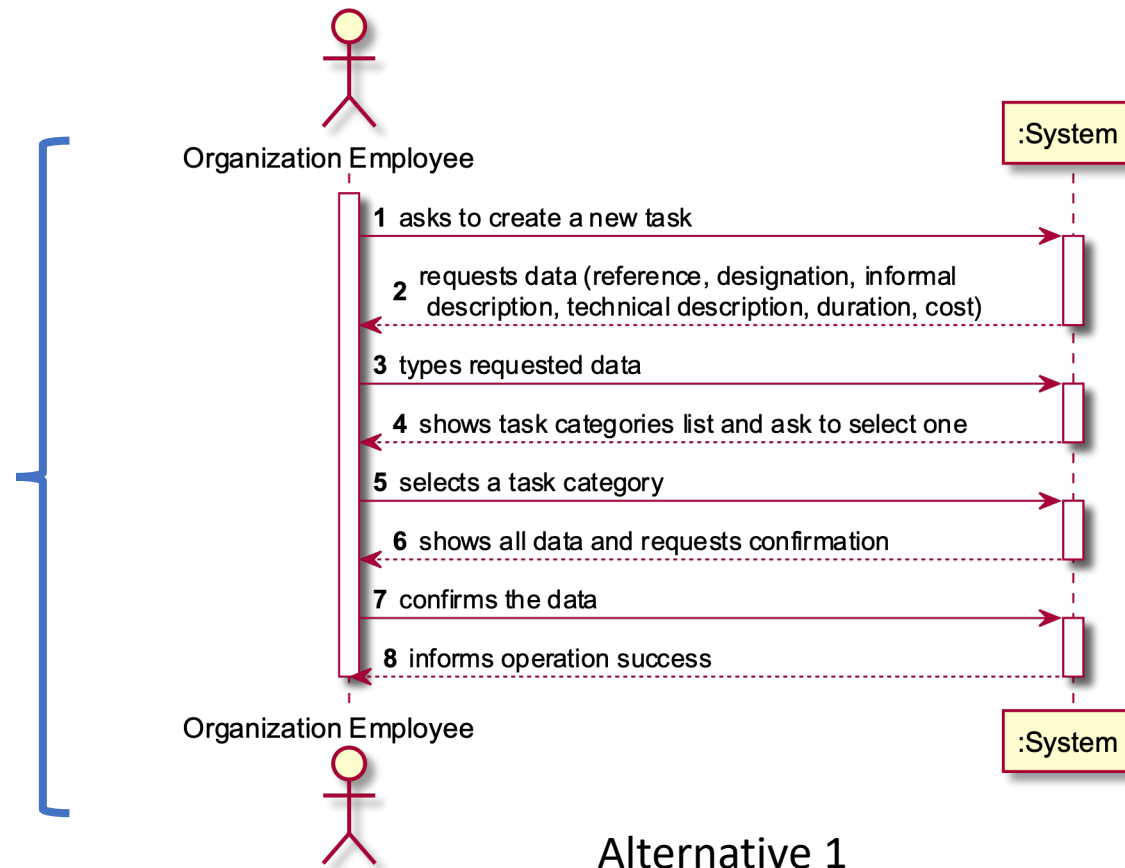
- Typed data:
 - a reference,
 - a designation,
 - an informal description
 - a technical description
 - an estimated duration
 - an estimated cost
- Selected data:
 - Classifying task category

Output Data:

- List of existing task categories
- (In)Success of the operation


The Approach in Practice (cont.)

8. Elaborate a plausible SSD for the US



Summary

- Regarding engineering requirements, you need to:
 - Elaborate and keep up-to-date a Glossary
 - Elaborate and keep up-to-date a Supplementary Specification (FURPS+)
 - For each US/UC
 - Check/Analyse/Interpret the US/UC
 - Carry out conversations with the client regarding the US/UC open issues
 - Record a summary of those conversations
 - Find out dependencies to other US/UC
 - Identify input/output data
 - Elaborate a System Sequence Diagram
 - Elaborate and keep up-to-date a Use-Cases Diagram



Adopt the
suggested
approach

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/>