Requirements Engineering

Part Two of Two

ISEP / LETI / ESOFT

Topics

- System Sequence Diagram (SSD)
- Recommended Approach for ESOFT Project
- Functional Requirements Artifacts (Part Two)
  - Use Case Diagram (UCD)
- Non-Functional Requirements
  - Definition and where to capture
  - Artifact: Supplementary Specification
    - FURPS+ model

System Sequence Diagram (SSD)

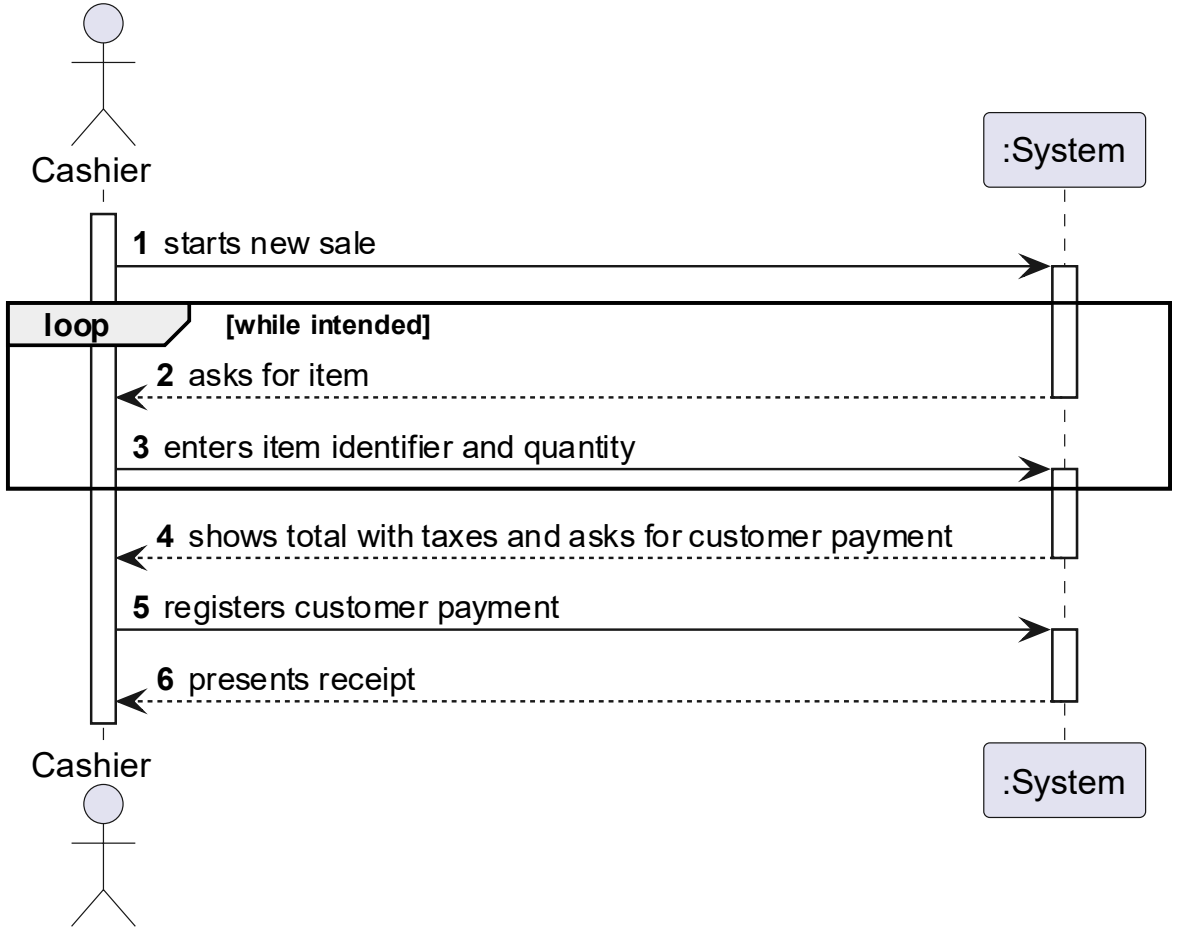
Artifacts

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



Recommended Approach for ESOFT Project

A Pragmatic and Simplified Approach

ESOFT

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

   The 2nd C

5. Record Acceptance Criteria (explicit and implicit) → The 3rd C
6. Find out dependencies on other US/UC (Answer the question: "Which US must be working for this to have success?") → Helps with work planning
7. Clearly identify input and output data; distinguish between typed data and data that can/must be entered by selection
8. Elaborate an SSD for the US

The Approach in Practice (1/3)

Platform for Outsourcing Tasks

ESOFT

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 SW Client
4. Summarize and record conversations with the SW Client

US006 - 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 a 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 POT (virtual currency internal to the platform).

The Approach in Practice (2/3)

Platform for Outsourcing Tasks

5. Record Acceptance Criteria (explicit and implicit)

6. Find out dependencies on other US/UC (Answer the question: “Which US must be working for this to have success?”)

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

1.3. Acceptance Criteria

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

1.4. Found out Dependencies

- There is a dependency on "US003 - Create a task category" as there must be at least one task category 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:
  - a task category

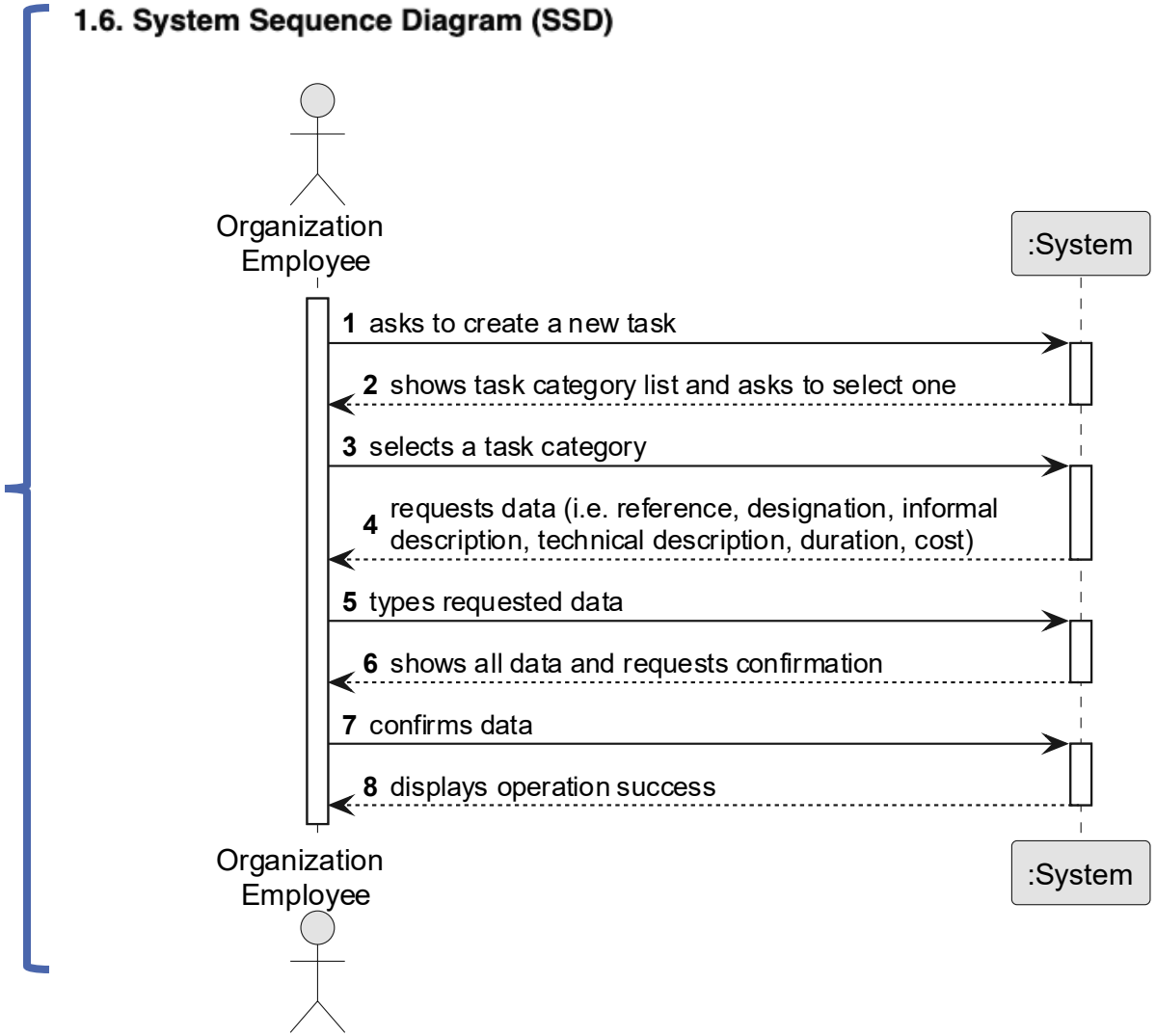
Output Data:

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

The Approach in Practice (3/3)

Platform for Outsourcing Tasks

8. Elaborate an SSD for the US



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)

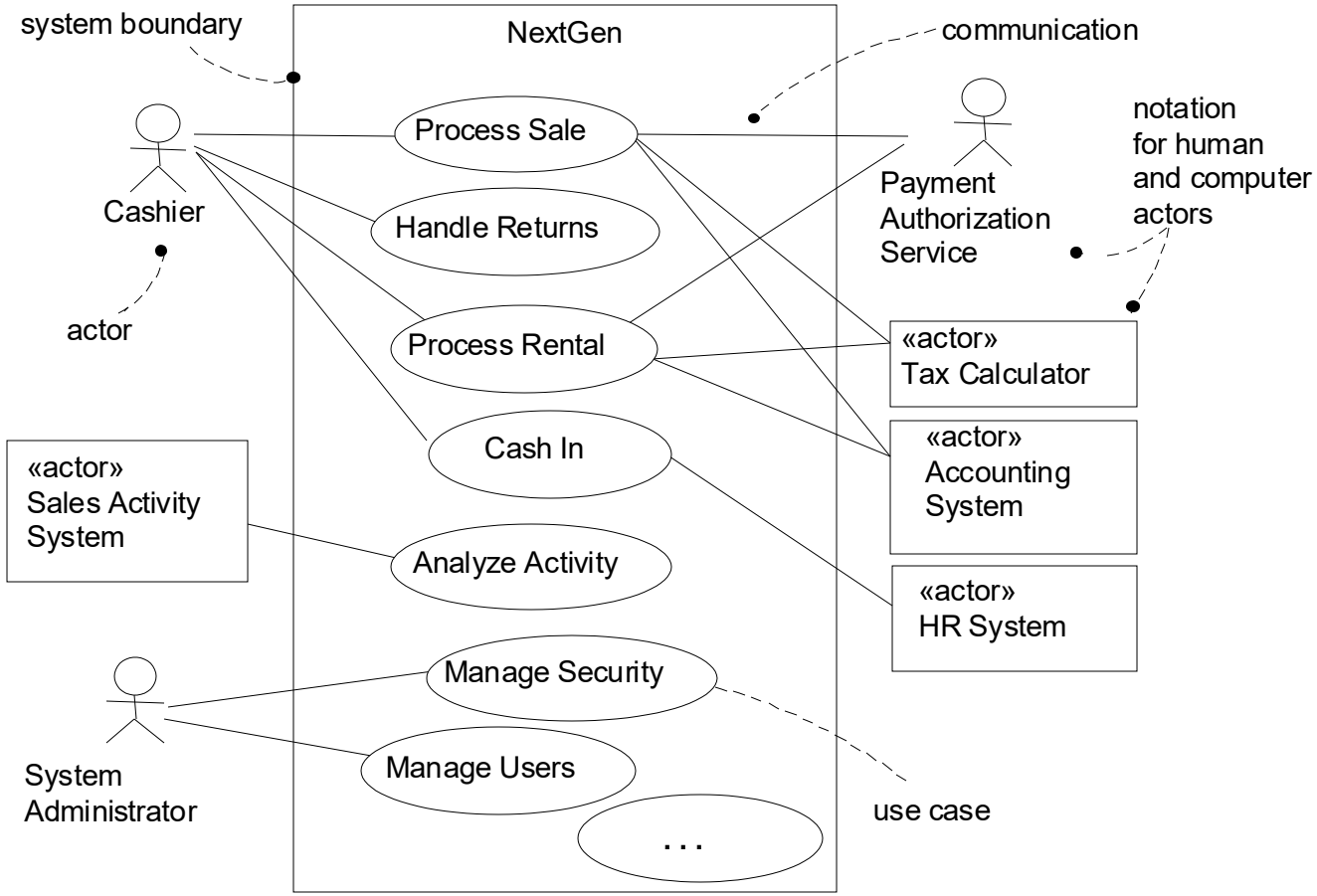
- Use cases should focus on the **Elementary Business Processes** (EBP).

- 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



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

- Organizes requirements by categories

- Adopts **FURPS+** model

FURPS+

- FURPS+ is a classification system

Category (EN)	Categoria (PT)
Functionality	Funcionalidade
Usability	Usabilidade
Reliability	Confiabilidade
Performance	Desempenho
Supportability	Suporte
+: design, implementation, interface and physical constraints	+: restrições de desenho, implementação, interface e físicas

Quality Attributes

Non-Functional Requirements

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

Adapted from Eeles, 2001.

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

Adapted from Eeles, 2001.

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
  - …

Adapted from Eeles, 2001.

FURPS+ – Supportability

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

Adapted from Eeles, 2001.

FURPS+ – Others (+)

- Groups additional categories typically related with <u>constraints</u>
  - **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)

Adapted from Eeles, 2001.

Summary

- 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*

Bibliography

- Cohn, M. (2004). Advantages of User Stories Over Requirements and Use Cases. Available on: https://www.mountaingoatsoftware.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