




Requirements Workflow

Lecturer: Adel Vahdati



Four Steps of Requirements Capture

- List candidate requirements
 - Understand system context
 - Capture functional requirements
 - Capture nonfunctional requirements
- 



List Candidate Requirements

- **Prepare a 'Features' list:**

- Ideas that customers, users, analysts, and developers think are good for the systems

- **Each feature has:**

- Status (proposed, approved, etc)
 - Estimated cost to implement (like man-hours)
 - Priority (critical, important, optional)
 - Level of risk in implementation



Understand system context

- **Domain model**

- Important concepts of the context and relationships among them
- A glossary of terms for better communication
- Domain objects later transformed to classes

- **Business Model**

- Model the business processes of the organization
- Specify which processes are to be supported by the system



Capture functional requirements

- Use case model
 - Each use case describes a way of using the system by a user
 - Use case model contains all the use cases of the system
 - Interview users and customers to collect them
 - This model leads to analysis and design



Capture nonfunctional requirements

- System properties:
 - environmental and implementation constraints,
 - platform dependencies,
 - reliability,
 - timing constraints.
- Some nonfunctional requirements are relevant only to a certain use case.
- Supplementary requirements:
 - Nonfunctional requirements that cannot be applied to particular use cases




Capture Functional Requirements

➤ **Main Activity:**

1. Find actors and use cases
2. Prioritize use cases
3. Detail use cases
4. Prototype user interface
5. Structure the use-case model



1. Find actors and use cases: Objectives

- Delimit the system from its environment
 - Outline who and what (actors) will interact with the system and what functionality is expected from the system
 - Capture and define in a glossary common terms that are essential for describing the system
- 



1. Find actors and use cases: Steps

- **Finding the actors**

- At least one user who can enact the candidate actor
- Min. overlap between the roles played by different actors

- **Finding the use cases**

- A use-case should deliver an observable result that is of value to the particular actor – the initiating actor
- Avoid too small or too large use cases

- **Briefly describing each use case**

- A step-by-step description of what the system needs to do when interacting with the actor

- **Describing the use case model as a whole**

- Use diagrams and descriptions to explain the use-case model as a whole, and how they are related to each other
- Let the users/customers approve the use-case model through an informal review



2. Prioritize use cases

- The purpose is to provide input to the realization of use cases to determine which need to be developed in early iterations.
- MoSCoW rules are prevalently used for this purpose.



3. Detail use cases

- Describe the flow of events for each use case
- Structuring the use-case description
 - Choose a complete *basic path* from the start state to the end state and describe it in one section
 - Basic path: “normal” path
 - Describe the rest of the paths as alternatives of deviation from the basic path
 - Alternative paths are described in a separate section



3. Detail use cases:

What to include in use-case descriptions

- Define the start state and end states as precondition and post-conditions, respectively
- How and when the use case starts and ends
- The required order of actions
- Paths of execution that are not allowed
- Alternative path descriptions
- System interactions with the actor, explicitly specify what the system does and what the actor does
- Usage of objects, values, and resources of the system



3. Detail use cases:

Formalizing the use-case description

- For simple use cases with fewer states, textual description may be used
- For complex use cases
 - Use Activity diagrams to describe sequence of activities
 - Use state charts to describe the states and transitions between those states
 - Use Interaction Diagrams to describe how the actor (or actors) interacts with the system in the context of the use case



4. Prototype user interface

- Creating a logical user interface design
 - Determine what elements are needed from the user interfaces to enable the use cases for each actor
 - How should they be related to each other
 - What should they look like
 - How should they be manipulated
 - Use sticky notes (for elements) on a whiteboard



4. Prototype user interface

- Creating a physical user interface design and prototype
 - Sketch the constellation of user interface elements
 - Additional elements may be added to organize the elements (like windows, menus, etc)
 - Each actor should be provided with a well-integrated, easy-to-use, and consistent interface
 - Prototypes may be built for user validation



Structure the use-case model

- Identify shared descriptions of functionality
 - The actions that are common to or shared by several use cases (*Gen./Spec. Relationships*)
 - Identify additional and optional description of functionality
 - Identify *Extend* relationships: Additions to a use case's sequence of actions
 - Identify *Include* relationships between use cases: Commonalities among different use cases



Artifacts of requirements workflow

- **Actor:**

- Users who use the system, and
- external systems that interact with the system

- **Use cases**

- **Use Case Priorities**

- **Glossary:**

- Important and common terms used by analysts in describing the system

- **User Interface Prototype**



References



- Jacobson, I., Booch, G., Rumbaugh, G., *Unified Software Development Process*. Addison-Wesley, 1999.
- Ramsin, Raman. "Home." Department of Computer Science and Engineering, Sharif University of Technology. Accessed February 15, 2025. <https://sharif.edu/~ramsin/index.htm>.