3. Capturing Requirements and Use Case Model

Why is it hard to capture?

- We build it for others to use
- Users don’t have a clue?
  - Is there one user?
  - Each user does not think about the entire system
  - How to convert work into software?
  - Does not know what is wanted
  - Does not specify precisely either
  - How about an analyst to gather requirements?
- Hind sight is 20-20!
  - Seeing the system work leads to understanding
  - Changes suggested after the fact
What is the goal?

- To describe system requirements enough to
  - get customer agreement on the functionality
  - to do so in a language that is understood by customers
  - help the planning of the project iterations & release

Requirements Capture

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

- List candidate requirements
  - List of features/ideas from just about everyone
  - Kept until becomes requirement and is transformed
  - May keep information: status, cost, priority, risk level
  - Used for planning and estimation

Capture functional requirements

- Performed using the use-case model
  - Important to arrive at several versions of the user interfaces
  - Develop visualizations or prototypes for user to try out
Capture nonfunctional requirements

- System properties
- constraints
- performance
- platform
- Reliability

- Specified as part of domain model or use-cases
- Others part of supplementary requirements

Use case

- Specific way of using the system
- A complete course of events initiated by an actor
- Each actor performs several use cases
- Several use cases may begin with similar events
- Actor initiates a course of events that result in a complete use case
Flow of Events

- Textual description of sequence of actions
- Specifies what system does when use case performed

An Example System

- A recycling machine for bottles, cans and crates
- May be used by several customers simultaneously
- System registers types and number of each item returned
- Prints receipt & money on customers request
- Operator may ask for daily report
- Operator may change value of items returned
- Operator informed when malfunction detected
Finding the Actors

- If there is business model
  - who will use the system
    - one actor for each worker in business
    - one for each business actor (customer)
- If there is no business model, identify users into categories that represent actors
- Actors representing external users and actors for system maintenance and operations need to be identified

Finding the Actors...

- At least one user should enact candidate actor
  - helps find relevant actors
- Roles of actors must not be the same
  - minimum overlap between the roles of actors
- Takes a few rounds of discussion to find right ones
- Relevant names for actors essential to convey semantics
- Describe actor’s needs and responsibilities
Actors in Recycling Machine System

Finding Use-Cases

- Take actors one by one and suggest candidate use-cases
- Names for use-cases
  - must leads us to think of sequence of actions that adds value to an actor
  - often starts with a verb
- Use-Case must be complete and stand by itself
  - Result of value
  - to particular actor
Recycling Machine System: Use cases

Customer

Return Items

Generate Daily Report

Operator

Change Item /value

Briefly Describing Use-Case

• A couple of lines of descriptions of the actions performed by each Use-Case

• A step-by-step description of what the system needs to do when interacting with actor
Describing Use-Case Model as a whole

- A Diagram showing use-cases and related actors
- If and how use-cases relate to one another
- May be participating in one business use case
- May be those performed by one actor
- May be clustered into use-case packages

Generalization and Extends Use-Case

- A generalization use-case is an use-case that is inserted into other use-cases
  - It performs some sequence of actions that are part of other use-cases

- An Extends use-case is an use-case that extends the sequence of actions described in another use-case
Activity: Prioritize Use-Cases

- Determine which use-cases will be developed in early iterations and which in later iterations
- Results captured in architectural view of use-case model
- Used as input when planning development within an iteration

Activity: Detail a Use-Case

- Use-case model and use-case diagrams used as starting point
- Step-by-step description of each use case
  - Flow of Events
- How to Structure to specify all alternative paths?
- What to include in a description?
- How to formalize the description?
Structuring Use-Case Description

- States that use-case instances may enter and possible transitions between states
- Each such transitions is a sequence of actions
- May get complicated and intricate

- Precise and simple description needed
- Start with a Basic Path
  - start to end state in one section of description
- Alternate Paths described in separate section
  - If small, may inline in Basic Path
- Goal: Precise description that is easy to read

Flow of Events Example

Example:

- Basic Path for Return items is when user deposits items and asks receipt

- Alternative Path for the use case may consider an item getting stuck in the deposit slot
**Basic Path for Return Item**

Precondition: Customer wants to return cans, bottles or crates

1. Customer places each item in the machine
2. System increases received number of items as well as daily total for item type
3. Customer presses receipt button when done
4. System prints receipt with total returned items as well as total return sum

Alternative Paths:
In step 1, an item may get stuck in the slot. In this case, inform user about problem, alert operator, print receipt of transaction so far.

Postcondition: use-case ends when receipt button pressed

---

**Basic Path for Generate Daily Report**

Precondition: Operator wants daily report of items returned

1. System prints number of items received for each type
2. System prints total number of items received
3. System resets the number counts to zero

Postcondition: Daily figures reset when use-case terminates
Basic Path for Change Item/Value

Precondition: Operator wants to change items or value

1. Return value of each item may be changed
2. Size of each returnable item may be changed
3. New types of item may be added

Postcondition: -

What to include in description

- Start state as precondition
- How and when use-case starts (step 1)
- Order for the flow of events
- How and when use-case ends
- Possible end state as post condition
- Paths of execution that is not allowed
- Inlined alternative paths
- Alternative path descriptions extracted from basic path
- System interaction with actor
- Usage of objects, values, resources in system
- Explicitly describe what system does
Formalizing Descriptions

- For large use-cases with various alternatives, simple text description is not practical
- Statechart diagrams
  - express complex use-cases
  - Describes state of use-case and transitions
- Activity diagrams
  - Describe transition between states in more details of sequence of actions
  - Generalized form of SDL state transition diagrams
- Interaction diagrams
  - Describes interaction between an actor instance and an use-case instance

Activity: Prototype User Interface

- For each use-case, discern the need for user interfaces to enable the use case for actor
- This leads to logical user interface design
- We then develop physical user interface design
- Develop prototypes
  - illustrates how users can use system to perform use cases
The Essence of Use-Case Modeling

“By starting to specify what is needed before we decide how to realize it, we are compelled to understand the need before we try to realize them”