Use Cases: Best Practices

• A use case is a set of scenarios tied together by a common user goal (UML Distilled, 3rd ed, p.99)
• A use case describes an interaction between an external role and the system, thereby documenting a major function that the system will perform.
• At minimum, a use case has a name, and a step-by-step description of a basic course of action that is necessary for the user to achieve an objective.

The Use of Use Cases

An Example of Use Case

Buy a Product

Main Success Scenario:
1. Customer browses catalog and selects items
2. Customer goes to check out
3. Customer fills in shipping information
4. System presents full pricing information
5. Customer fills in credit card information
6. System authorizes purchase
7. System confirms sale immediately
8. System sends confirming e-mail to customer

An Example of Use Case

Buy a Product (cont.)

Extensions:
3a: Customer is a regular customer
1. System displays current shipping, pricing, and billing information
2. Customer may accept or override these defaults
6a: System fails to authorize credit purchase
1. Customer may re-enter credit card information or may cancel the transaction

Seven Best Practices

• Scope the domain
• Scope your use cases
• Validate use cases as they emerge
• Define the requirements models you’ll need
• Determine the strategy you’ll use to elicit requirements
• Settle on a standard format for your use cases
• Develop a project glossary

Scope the Domain

• Determine and document the overall project scope early
• New use cases or alternative use cases will continue to be discovered after initial definition
  – Determine if they belong in the project, or the current, or the next iteration
  – Continually delineating project scope
  – Prevent scope creep
• Scope creep is often cited as the highest risk for any software development project.
Scope the Domain (cont.)

- At this point, the use cases will contain only the names, no details, yet.
- Name use cases during the inception phase (the first phase in a RUP)
- Make project scope decision, estimate project size if possible
- Compiling a survey list of use cases before detailing them

Scope the Domain (cont.)

- Generate good names
  - Format: action verb + [qualified] object. E.g. “Place order”, “Monitor network usage”
  - Avoid vague verbs such as do or process. Examples of inappropriate: “Do menu”, “Process list”, better to use “Print menu”, “Sort list”
  - Avoid low level verbs such as create, read, update, delete (known collectively as CRUD), get, or insert
  - The “object” part of the name can be a noun (e.g. inventory) or a qualified noun (e.g. in-sock inventory)
  - Make sure that the project glossary defines each object in the use case name
  - Add each object to the domain model (as a class, entity, or attribute)
  - Elicit roles and use cases concurrently, associating roles with use cases as you name each one

Scope Your Use Cases

- Carefully define the scope of each use case
- Make sure each use case addresses a single role goal, and not overly complex
- At the same time, avoid small, piecemeal use cases, e.g. CURD
- One approach is to frame each use case with triggering events and their responses
- For example: “Place order”
  - Triggering event: a customer placed an order
  - Responses: stores order information, send confirmation to the customer

Scope Your Use Cases (cont)

- Define events
  - Two types of events: business and temporal
  - Business event: high-level at unpredictable time, e.g. “customer orders a book”
    - Use name format “subject + verb + object”
  - Temporal event: driven by clock, e.g. “time to ship customer order”
    - Use name format “time to verb + object”

Validate Use Cases as They Emerge

- Validate the use cases: ensure that each one is necessary
- Validate use cases in the inception phase, answer the following questions
  - How does this use case help achieve our goals and vision?
  - Does this use case address some aspect of the problem?
  - Does this use case differentiate our product in some way?
  - Do we have use cases to address all the stakeholder and user groups we identified in our vision statement?
  - Which use cases will be implemented in our initial release?
- Any use case that does not align with the vision is potentially out of scope or lower in priority

Define the Requirements Models You’ll Need

- A requirements model is a set of models which acts as a blueprint for a software product.
- Each model may take different form, often supplemented with text
  - A diagram
  - A table
  - A list
Define the Requirements Models You’ll Need (cont)

• Examples of user requirements model
  – Event lists
  – Use cases
  – Context diagrams
  – Data models
  – Class models
  – Business rules
  – Role maps
  – Prototypes
  – Use case storyboards

Define the Requirements Models You’ll Need (cont)

• The primary purpose of a requirements model is to communicate.
• Defining requirements is a process of discovery for users and developers.
• Multiple views of a model
  – Behavior
  – Structure
  – Dynamics
  – Control

Define the Requirements Models You’ll Need (cont)

• Use cases: behavioral
• Class diagram: structural
• State diagram: dynamic
• Business rules (statement): control
• Multiple views of any one model

Determine Your Eliciting Strategy

• Conduct one-on-one or group interview
• Hold facilitated workshops
• Generate lists of scenarios
• Reuse existing requirements
• Prototype the user interfaces
• Observe end users in their work environment
• Conduct focus groups
• Send out questionnaires or market surveys
• Review regulations, procedures, and guidelines
• Mine customer complaints and help desk logs
• Conduct competitive analysis
• Use a combine of the above

Settle on a Standard Use Case Format

• Some sample use case formats, in order of increasing complexity
  – Use case name only
  – Use case name and a single-sentence goal statement
  – Use case brief description (two to five sentences)
  – Use case outline
  – Use case conversational format
  – Use case detailed description
• Different formats may be used in different phases
  – Inception phase: simple ones
  – Elaboration phase: more complex ones

Develop a Project Glossary

• Have a project glossary
• Define what each term means
• Facilitate the communications among team members, between different teams in the same project, and between the company and the customer
Tips for Writing Successful Use Cases

1. Develop your use cases iteratively
2. Involve users directly
3. Depict your use cases visually
4. Use your use cases to harvest nonfunctional requirements
5. Prioritize your use cases and use case scenarios
6. Trace your use cases
7. Verify your use cases