MASTERING USE CASES

CAPTURING FUNCTIONAL REQUIREMENTS FOR INTERACTIVE APPLICATIONS

2nd ACM SIGCHI Symposium on
Engineering Interactive Computing Systems

Daniel Sinnig, Concordia University

Homa Javahery, IBM Pacific Development Center

Berlin, June 20, 2010

Tutorial Information

Objective: Practice and theory around writing quality use cases

Duration: 180min

Web: http://www.cs.concordia.ca/d_sinnig/EICSUCTutorial

- Detailed Tutorial Slides
- References
- Follow-up tutorials

Contact:

Daniel Sinnig, PhD (d_sinnig@cs.concordia.ca)

Homa Javahery, PhD (homa@ca.ibm.com)

Roadmap

Part I – Use Case Basics

a. Introduction
b. Requirements Engineering
c. Stand-alone Use Case Model
d. Integrated Use Case Model

Roadmap

Part II – Advanced Topics

a. Guidelines and Checklists
b. Use Cases and the User Interface
c. Tool Support
d. Use Cases and Concurrency

Roadmap

Part III – Discussion

Part I – Basic Topics

(a) Introduction

History

- 1986: Ivar Jacobson coined the term **use case** (result of his PhD thesis "Concepts for Modeling Large Real Time Systems")
- 1992: Ivar Jacobson defined the **Use Case Driven Approach** in his book "Object-oriented Software Engineering: A Use-Case Driven Approach"
- 2003: Use cases have been adopted by the Rational Unified Process
- Have become a key activity in mainstream software development

Common Definitions

"A use case as a specific way of using the system by using some part of the functionality." (Ivar Jacobson)

"A use case is a collection of possible sequences of interactions between the system under discussion and its Users (or Actors), relating to a particular goal. " (Alistair Cockburn)

"A use case represents a series of interactions between an outside entity and the system, which ends by providing a business value." (Kulak and Guiney)

"Use cases represent the things of value that the system performs for its actors. […] Use cases have a name and […] detailed descriptions, about how the actors use the system to do something they consider important." (Bittner and Spence)

Use Cases – What are they good for?

- Contract Statement
- Requirements Specification
- Test Case Generation
- Model-driven Design
- etc…

This tutorial focuses on Use Cases as part of
(functional) Requirements Specification

Purpose of this Tutorial

To provide you with a practical guide on writing quality use cases, with a focus on:

- Treating use cases as part of an integrated and comprehensive requirements documentation
- Bridging the gap between functional and user interface requirements
- Integrating usability into the development process

Part I – Basic Topics

(b) Requirements Engineering

What is Software Engineering?

- Systematic approach to achieving high quality software
- Development process is iterative and incremental
- Gear application to the needs of stakeholders

What is Requirements Engineering?

- Divided into disciplines
  - Problem and root cause analysis
  - Requirements elicitation
  - Domain modeling and requirements specification
  - Validation
- Difficult and extremely important
- Often underestimated and poorly done

Separation of Concerns

- **Key Idea**: Separate different concepts into distinct but interrelated artifacts
- **To:** Eliminate redundancies and inconsistencies in documents
- **Which Results In:** Improved Maintainability
- **And Improved:** Efficiency and product lifecycle

Traceability

"Ability to describe and follow the life of an artifact (work product)"

- Requires effective **change management**
- Detects and prevents inconsistencies in documents
- Improves maintainability

Part I – Basic Topics

(c) Stand-alone Use Case Model

The Use Case Model

- Medium of choice for capturing **functional requirements**
- Grouped into **use case packages**
- Defines several use cases, which capture interaction between actors and system under development as **scenarios**
- Use cases are organized into three parts:
  - Properties
  - Main Success Scenario (Normal Flow)
  - Extensions (Alternative Flow)

Main Success Scenario (Normal Flow)

- Denotes most common way that primary actor achieves use case goal
- First step typically initiated by primary actor
- Last step typically performed by the system
  - leads to the fulfillment of use case goal

Extensions (Alternative Flows)

- Specify alternative scenarios which **may** or **may not** lead to the fulfillment of the use case goal
- Each extension starts with a condition and a reference to a use case step
- **Exhaustive modeling** of Extensions is indispensable to capturing full system behavior

"Bugs lurk in corners and congregate at boundaries." — Boris Beizer

Use Case Actors

- **Actors** represent users or entities that interact with the system. By definition, actors are outside of the system boundary.

- **Primary Actor:**
  - Typically a user
  - Initiates the use case in order to accomplish a pre-set goal

- **Secondary Actor:**
  - Plays the role of supporting the execution of the use case
  - Participates in the interaction later

Linking Use Cases

- A use case may **include** sub-use cases or may **extend** an existing use case
- Use Case Include:
  - Invokes a sub-use case
  - Often an extension is needed if the sub-use case terminates unsuccessfully
- Use Case Extend:
  - Expands an existing base use case with additional interactions
  - This relationship is often used to model optional or seemingly unrelated behavior relative to the goal of the base use case

Current Literature on Use Cases

- Use case model is often treated in isolation
- Proposed templates and best practices are too 'academic' and do not scale
- Integration with related requirements and design model (especially HCI) is vague at best
- Tool Support is not discussed

Part I – Basic Topics

(d) Integrated Use Case Model

Integrated Use Case Model

- Expressive, descriptive, scalable and maintainable
- Support for iterative and evolutionary development
- Support for collaborative development
- Traceability links to related artifacts

The Big Picture

- Software Specification consists of several interlinked work products

Stage	**Work Products**
Analysis	Domain Model, Actor-Goal List, Personas and User Profiles, Feature List, etc.
Requirements	Business Rules, **Use Case Model**, etc.
Design	Software Architectural Model, UI Specification, Task Models, etc.
Testing	Heuristic Evaluation, Functional Test Plans, etc.

Software Specification

- Consists of definitions and references
- Dependency chain: Analysis ← Requirements ← Design ← Testing
- Example: Use Case Model
  - **Defines** a set of use cases
  - **References** Business needs, domain objects, actors, features, etc…
  - Does **not** reference UI specs, text plans, etc…
- Evolution of work products traced through configuration management, notes, versioning, etc…

Exercise 1

Building Assessment Application (BAA)

A portfolio manager for buildings and their environmental evaluation

Let's sketch the following:

- Domain Model
- Use Case Model
- Business Rules
- User Messages

Part II – Advanced Topics

(a) Guidelines and Checklists

Basic Use Case Guidelines

- Use simple grammar:
  - Subject … verb … direct object
  - Use an expressive verb in present tense, for example: The system deducts the amount from the account
- Specify whether the system or actor performs the use case step
- Write from a commentator's view
- Show the process moving forward
- Keep UI details out! Show intent, not actions

Common Mistake

- Use Cases should **not** contain UI details
- This is problematic because the UI is more prone to change compared to core functionality

Use Case: Buy XYZ

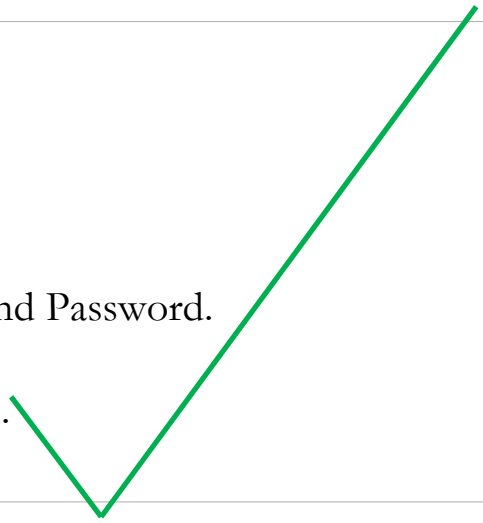
- **Level:** user-goal
- **Primary Actor:** Customer
- **Main Flow:**
    1. System presents ID and Password screen with two input fields.
    2. Customer types ID and password into system and then clicks the "Okay" button at the bottom of the screen.
    3. System validates user.
    4. Customer types last name first in the "last name" field, then first name

Solution

- Abstract the Use Case from UI Details
- Results in less clutter and redundancies, reduced maintenance overhead

Use Case: Buy XYZ

- **Level:** user-goal
- **Primary Actor:** Customer
- **Main Flow:**
    1. Customer accesses system with ID and Password.
    2. System validates user.
    3. Customer provides name and address.
    4. …



Other Mistakes

Mistakes	**Implication**	**Solution**
Use Case is not goal-oriented	Does not reflect user needs	Make use of an actor – goal list
Duplication of information	Results in inconsistencies	Use case refactoring
Incorrect abstraction level (too many details)	Maintenance overhead	Show process moving forward
Conditional Statements	Clutter and poor readability	Use alternative flows (extensions)

Detailed Authoring Guidelines

- Guidelines suggested for various aspects of use case authoring
  - Structure Guidelines
  - Contents Guidelines
  - Completeness Checks
  - Process Guidelines
- Refer to cheat-sheet of use case guidelines (distributed during tutorial)

Exercise 2

Building Assessment Application (BAA)

A portfolio manager for buildings and their environmental evaluation

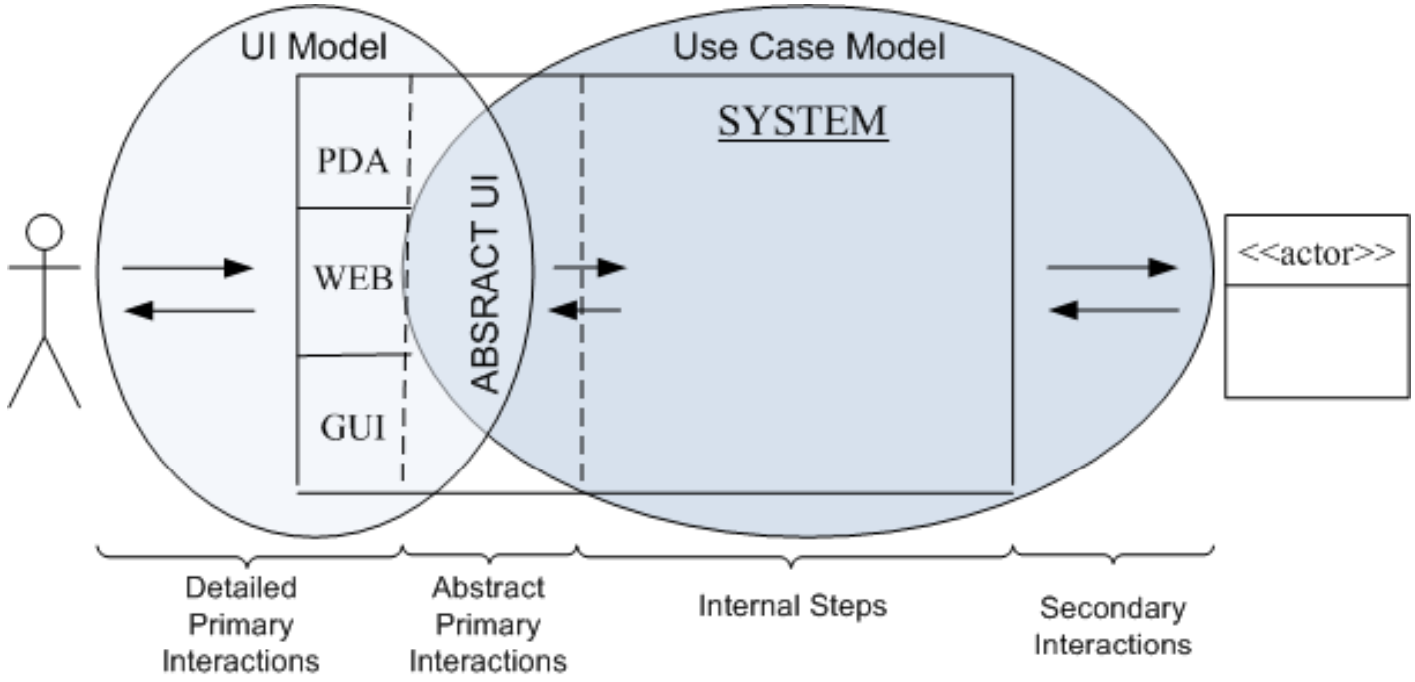
Let's apply our guidelines

- Evaluate the BAA use case model

Part II – Advanced Topics

(b) Integration with HCI Models

Single Use Case Model – Many Interfaces



Problem

- User Interface Design methods are often poorly related to standard software engineering practices, causing
  - Duplication of effort
  - Increased maintenance overhead
  - Neglect of usability aspects

- Both are equally important; thus should be addressed in an integrated manner

From Actors to Personas

- Identify human actors, roles and user groups
- Examples:
  - Consultant . Reasonably tech-savvy, has investigated BAA before, but is not completely comfortable with it.
  - Owner. Comfortable with computers, interacts with BAA for informational purposes only.
  - Building Manager. Comfortable with computers and software tools, is a BAA return user (but not expert).
- Extract personas based on key user groups

Personas

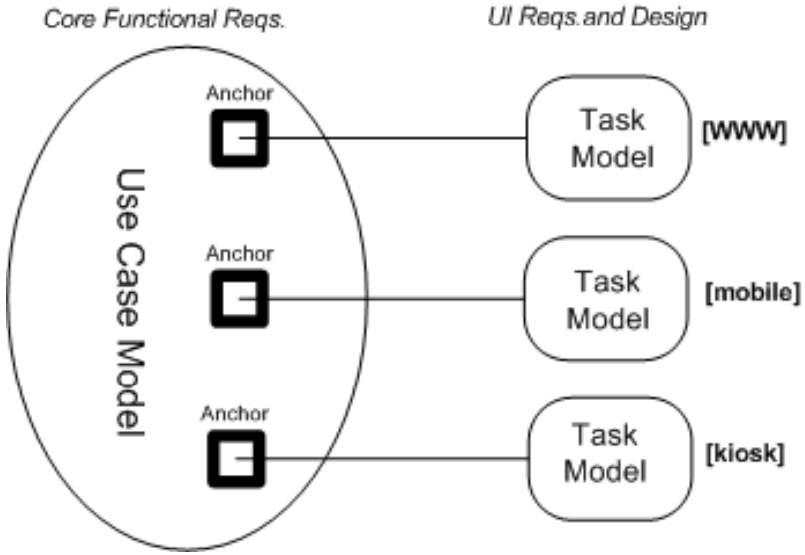
- Personas are an engaging technique to capture user details
- Helps focus design concept and effort to actual users of the system
- Format:
  - General summary
  - Profile (age, gender, work experience, computer and application experience)
  - Work environment (people, expectations, constraints)
  - Goals, tasks and design implication
  - Behaviour
  - Scenario examples
  - Specific attributes relevant to project

Example: Manager Persona (subset)

Goals	**Tasks**	**Design Implications**
Effectively manage building	• prepare management plans • ensure building maintenance • provide reports on upgrading options	Software tools must be easy to access, intuitive to use and should not slow down her workflow.
Supervise relations with stakeholders	• supervise building staff & contractors • maintain good tenant-owner relations • address stakeholder concerns	Needs effective and efficient tools for personnel management, communication and collaboration.
Perform administrative and financial functions	• adminster legal agreements • prepare budgets and statements • obtain quotes from contractors • collect rent and ensure market value	Needs access to current information and services, and suitable support tools.

From Use Cases to Task Models

- Use case model captures functional requirements independent of any UI details
- Task models extend the use case model by refining the UI-related steps
- One Task Model for each UI-type



Task Models for UI Design

- Task modeling is a well understood technique supporting user-centered design
- Task set is primary input to the UI design stage
- Like use cases, task specifications describe user's interaction with the system
- Primary purpose is to capture way users achieve a goal when interacting with system
- Goals captured in personas drive design decisions, but interface is modeled based on tasks

Applying Best Practices: UI Design Patterns

- Best practices for various aspects of UI design
- Examples
  - **Navigation:** Process Funnel, Clear entry points
  - **Displaying Data:** Sortable table and grid, Tree table
  - **Forms and Controls:** Progressive disclosure, Illustrated choices
- Selected based on functional and UI requirements

Exercise 3

Building Assessment Application (BAA)

A portfolio manager for buildings and their environmental evaluation

Know your users and their tasks!

- Personas
- Tasks

Part II – Advanced Topics

(c) Tool Support

DEMO

MS Word 2007 

Caliber DefineIT Borland

Rational RequisitePro 

Critical Evaluation of Tools

TOOL / **CRITERIA**	MS Word	Caliber DefineIT	Requisite Pro
ease of use			
structured use case writing			
expressiveness			
embedding w/ related artifacts			
traceability/impact analysis			
history keeping			
etc...			

Part II – Advanced Topics

(d) Use Cases and Concurrency

Think outside the box – Leverage full potential of use cases.

Examples:

- Multiple users and concurrent access
- Standards affecting interaction capabilities (healthcare industry)
- Separation of concerns and use case refactoring

Here we focus on modeling concurrency management strategies in use cases

Why Thinking about Concurrency?

- **Context:** Enterprise applications
  - Concurrent access to shared data volumes
- **Concurrency** related interactions often **overlooked** during requirements phase
- Leads to significant **usability problems**

What proof? Consider the following scenario…

Using the flight reservation system "Get-You-There", customer **Paul** wants to **book a flight** from Montreal to Vancouver. After entering his flight criteria, the system suggests a set of suitable flights. **Paul selects a flight** and, out of excitement for getting such a great air fare, runs into the kitchen to **tell his wife about his good fortune**. In the mean time, **Frank**, who is also **interested in** flying from Montreal to Vancouver, coincidentally selects the **same flight** as Paul. Frank lives alone and, without any delays, proceeds to select an appropriate seat and purchases the flight using his credit card. After successful validation of his credit limit, the **system issues an electronic ticket**. While Frank already enjoys a cool beverage to celebrate his upcoming vacation, **Paul returns** to his computer in order to finalize his booking. He **selects a seat** and **submits payment information**. Unfortunately, in Paul's case, the system indicates that **seats in the selected booking class are already fully booked**. As it turns out, **Frank forestalled Paul** and booked the last available seat in the booking class. **Paul, frustrated**, returns to the kitchen to tell his wife that he will never use the system again.

Conclusions from Scenario:

- **Optimistic** concurrency resolution strategy **inappropriate**
- **Pessimistic** concurrency management strategy would have **increased usability**
- **Which approach is best?** It is up to all stakeholders, especially, targeted end users to decide.

Where to Capture Concurrency Management Strategy?

- Each Strategy requires specific interactions
  - Pessimistic concurrency management strategy: interactions related to **conflict prevention**
  - Optimistic concurrency management strategy: interactions related to **conflict resolution**

- **Capture in Use Case Models**

How to Capture Concurrency Management Strategy?

Proposed Terminology

- TRANSACTIONAL RESOURCE: Domain entity that is directly or indirectly either created, read, modified or deleted by business transaction.

- CONFLICT PREVENTION STAGE: Interval during which a transaction has exclusive access to its TRs.

- COMMIT and ABORT: Last sub-step of a transaction, and indicates whether effects of transaction become visible or transaction is rolled back.

Part III – Discussion

Thank You for Your Attention.

Daniel Sinnig
d_sinnig@cs.concordia.ca

Homa Javahery
homa@ca.ibm.com