

Lecture 6: Analysis and Design Descriptions

Kenneth M. Anderson
University of Colorado, Boulder
January 27, 2005



Overview

- Discuss the various types of system descriptions that can be used during analysis and design
 - Use Cases (in brief, more in Lecture 7)
 - Narratives
 - Scenarios
 - Conversations
 - Annotations
 - CRC Cards



Background

- During analysis and design, we will
 - capture requirements,
 - brainstorm candidate objects and roles,
 - consider trade-offs and design alternatives,
 - and make decisions
- In order to document these activities, we need various types of software artifacts
 - traditional requirements documents, UML diagrams, Use cases, etc.
- The format of these artifacts provide us with a means **to structure and capture** the information we are working so hard to create

User Perspective

- In analysis, as much as possible, we want to write our artifacts from the standpoint of a user
 - We will make frequent and consistent use of domain-related vocabulary and concepts
 - We will talk about the software system as a “black box”
 - We can describe **its inputs and its expected outputs** but we try to avoid discussing how the system will process or produce this information
- Use cases are a technique for maintaining this perspective
 - we identify the **different types of users** for our system
 - we then **develop tasks** for each of the different types of user

Actor

- More formally, a user is represented by an **actor**
 - Each use case can have one or more actors involved
 - An actor can be either a **human user or a software system**
- Actors have two defining characteristics
 - They are **external** to the system under design
 - They take **initiative** and **interact** with our system
- Typical types of users will include
 - Customers, “Front-Line” Employees, Administrators, Security Personnel, Managers, etc.

Use Cases

- Each use case describes a **single task for a particular actor**
 - The description typically includes **one “success” case and a number of extensions** that document “exceptional” conditions
- In our text book, three different types of use case are presented
 - narratives, scenarios, and conversations
- In lecture 7, we will see a more formal version of the scenario style of use case
- Use cases are used to capture functional requirements
 - They can be **annotated** to also describe non-functional requirements but typically **the focus is on functional requirements only**

Example: Word Processor

- ❁ Our textbook makes use of a word processor as an example domain for analysis and design descriptions
 - ❁ This domain has a number of real world counterparts, but be aware that this example is inherently “tool focused”
 - ❁ In the “real world”, you will be tackling **larger problem domains, understanding a specific problem** within that domain, and then **creating tools** (or a single software system) to address that problem
 - ❁ This domain has a number of primary concepts that will emerge during analysis and design discussions
 - ❁ Document, Page, Paragraph, Spell Checker, etc.

Use Case Narrative

- ❁ A **narrative** is a brief, high-level description of a user task; A narrative consists of typically one or two paragraphs of natural language text
 - ❁ Narratives are **highly informal** and typically leave out a lot of details that will need to be filled in at a later time
 - ❁ They are useful when discussing a new task for the first time

❁ Example

Documents can be saved in different file formats. When you save a new document, the default file format is used unless another is specified. When a Save Document operation has completed saving an existing document, the file represents accurately the document as displayed to the user upon saving

Narratives Discussed

- From the example, we can see that narratives
 - may not be labeled or otherwise have a title
 - may not explicitly identify an actor
 - may use undefined terms
- However, they allow for the **quick capture of functional requirements** and identify areas in the domain that require additional analysis and/or description
- For instance,
 - What is a file format? What's the default format? How is a "Save Document Operation" invoked?

Use Case Scenario

- A **scenario** describes a specific path (through a software system) that a user will take to complete a task
 - Each step will **describe** either an **action taken by the user** (or actor) or a **response generated by the system** under design
 - As we will see, each step should be kept **as simple as possible**
 - Use case scenarios require a particular writing style
 - We will cover guidelines for writing effective use cases in Lecture 7
 - The scenario should end with the **successful completion** of the given task

Example: Saving an HTML Doc

Save a document to an HTML File

1. The user commands the system to save a file.
2. The system presents a "Save File" dialog box.
3. If the file is being saved for the first time, the system will construct a name for the file using a default file extension and the first line of text in the document.
4. The user indicates the HTML document type from the dialog.
5. The system replaces the default extension with ".html"
6. The user selects a directory for the file.
7. The user clicks the "Save" button in the dialog box.
8. The software warns the user that some formatting information may be lost in the transformation to HTML. It gives the user a chance to cancel the operation.
9. The user asks for the operation to continue.
10. The software saves the document in HTML format in the location specified by the user.

Scenarios will typically contain a title and list the sequence of actions needed to successfully complete a task

Scenarios can be informal (like this example) or extremely formal

Scenarios can also indicate **extensions** that show how to **handle error conditions**

Use Case Conversations

- ❁ Use case **conversations** go one step further than **scenarios** to explicitly identify the system responsibilities that are implied by an actor's actions
- ❁ These responsibilities begin to reveal explicitly the functional requirements of the system under design
 - ❁ Recall that the creation of these responsibilities occurs within a **highly iterative process**; do not expect to get the responsibilities "right" the first time you write them down
- ❁ The responsibilities are **different** from the steps of a scenario since they are written from the **standpoint of the system** NOT the user
 - ❁ As such, conversations will typically **include details not found** in narratives and scenarios

Example








Save Document To File

User Actions	System Responsibilities
Indicate Save File	Display name of file to be saved Display contents of current directory, including files with same file extension If saving file for first time, construct default file name
Optionally, change directory	Redisplay contents of directory
Optionally, rename file	Rename file and display new file name
Optionally, change document format	Record document format Redisplay file extension Redisplay directory contents to match new extension
Indicate OK to Save	If formatting information will be lost, notify user Save document Redisplay contents if document format changed

User Actions appear on the left in **abbreviated form** (when compared to a use case narrative or scenario)

Responsibilities appear on the right and indicate **things the system must do** in response to the stated action

Annotations

-  Use cases can be annotated with many different types of information, including:
 -  Exceptional Conditions (see Lecture 7)
 -  Policies or Business Rules
 -  Design Notes: additional background information on system concepts
 -  Non-Functional Requirements
 -  Glossaries
-  We will see other examples in Lecture 7

CRC Cards

- ❁ Low fidelity method for brainstorming candidate object models
 - ❁ Information is written down on index cards or post-it notes
 - ❁ Keeps investment in any one object model low
 - ❁ If you don't like what you are seeing, rip up the cards and try again
- ❁ CRC stands for "Candidates, Responsibilities, and Collaborators"
- ❁ Affordances
 - ❁ Allows humans to take advantage of spatial dimension when performing analysis
 - ❁ Similar cards can be clustered physically, missing elements can be easily identified, collaborations can be formed and easily rearranged
 - ❁ Cards can be easily annotated and changed

Two Sides to a Card

- ❁ An index card has two sides; one is typically **blank**, the other is **lined**
- ❁ On the **unlined side** of the card, we
 - ❁ indicate the **candidate's name**
 - ❁ write an informal description of the **purpose** of the class
 - ❁ identify this candidate's **role stereotypes**
 - ❁ If this candidate participates in a **design pattern**, indicate its role within that pattern
- ❁ On the lined side of the card, we
 - ❁ **again** indicate the **candidate's name**
 - ❁ list its **responsibilities**
 - ❁ list its **collaborators**

Example

Unlined Side of Card

Document

Purpose: A Document acts as a container for graphics and text.

Patterns: Composite-component
Stereotypes: Structurer

Lined Side of Card

Document

Knows contents

Knows storage location

Inserts and removes text, graphics, and other elements

TextFlow

Next Lecture

🍷 Use Cases in more detail

🍷 Present information from Cockburn's Writing Effective Use Cases

🍷 Then, we will move on to chapters 3, 4, 5, and 6 in our textbook

🍷 Finding Classes

🍷 Finding Responsibilities

🍷 Designing Collaborations

🍷 Understanding Control Styles