CS211 Lecture: Use Cases and Initial Functional Tests

last revised August 21, 2007

Objectives:

1. To become familiar use cases / use case diagrams / use case flows of events
2. To become familiar with early specification of functional test cases

Materials:

1. Projectable: Figure 3.5 in book
2. Video projection: Address Book and ATM examples on the web
3. Projectable: Figure 3.6 in the book

I. Introduction

A. As we pointed out at the start of the course, there are many different processes that can be followed in software development (e.g. waterfall life cycle, RUP, etc).

B. Regardless of what process is followed, however, certain tasks will need to be done as part of the development process *per se* - whether all at once, iteratively, or incrementally. In fact, activities like these will be part of any situation in which one uses his/her professional skills to help solve someone else's problem - not just when creating software or even in a computer field.

1. Establishing Requirements: The goal of this is to spell out what constitutes a satisfactory solution to the problem.

2. Analysis. The goal of this is to *understand* the problem. The key question is "What?".

3. Design. The goal of this is to develop the *overall structure* of a solution to the problem in terms of individual, buildable components and their relationships to one another. The key question is "How?".

4. Implementation. The goal of this task is to actually *build* the system as designed.

5. Installation / Maintenance / Retirement

   All of these must be done in a context of commitment to Quality Assurance - *ensuring* that the individual components and the system as a whole do what they are supposed to do (which may involve identifying their shortcomings and fixing them.)

C. Last class addressed establishing requirements. Today's focus is on analysis. The goal of the analysis is a thorough *understanding* of the problem.

1. Obviously, this must take place in the mind of the analyst(s); but it also must be documented in some form, especially if - as is often the case - people other than the analyst(s) are involved in later phases of the project.

2. Typically, this documentation takes the form of one or more *models* of the system being analyzed.

   a) A model is simply a representation of a system.

   b) For complex problems, more than one model is often needed - each focusing on a different aspect of the system.

D. Over the years, software engineers have developed a large array of analysis tools that can be used for modeling a system.

1. Each tool is useful for modeling certain aspects of a system; and different sets of tools will be appropriate for different problems. Frequently, in fact, several different tools will be utilized to capture different aspects of a given problem.

   Example: A carpenter learns to use a wide variety of tools, though he may need only a few for any particular project.

2. Some of these tools are discussed in the software engineering course or other courses

3. There are also numerous publications and workshops that deal with various tools.

4. In this lecture we will consider use cases, As we shall see, use cases are not only used for analysis, but will also serve to drive the remainder of the process. There is a certain "seamlessness" to OO methodologies in this regard.

5. We will also talk about the early establishment of functional tests.

6. Our next lecture will deal with a topic that sits right on the border between analysis and design; identifying classes.

E. The book uses the "Wheels" system as a continuing example. In addition, we will use two other continuing examples, plus the video store system you are building as a semester project.

1. One is already familiar to many of you - the development of a <u>very simple</u> address book which we used for the last project in CS112.

   SHOW ON WEB

2. The other concerns developing software to simulate an automated teller machine (ATM).

   SHOW ON WEB

   GO TO REQUIREMENTS PAGE

   <u>Note</u>: The requirements in this example are somewhat more detailed and specific than would typically be available at the outset of a project. Normally, much of the knowledge we need at this point comes from conversations with the client. This example document embodies some of the fruit of what would typically come from such study.

   What kind of requirements are these?

   ASK

   What kinds of non-functional requirements might be appropriate for a system like this?

   ASK

   DEMONSTRATION: Example system on the web

   a) Run it

   b) Show structure of pages

II. Use Cases

A. The *use case* approach was developed by Ivar Jacobson. It is discussed at some length in the assigned chapter of the book.

1. What aspect of a system does the use case model? (QC question a)

   ASK

   It's functionality. [answer p. 36 top]

   Note: the book discusses use cases as part of the requirements specification stage of system design. I am treating it as part of analysis. The difference really reflects the fact that there is not a clear line between requirements and analysis - in fact, some writers speak of "requirements analysis"

2. The use case model consists of a set of complementary elements. Can you name them? (QC question b) What is each?

   ASK [answer p. 40 first paragraph under Use Case Diagram]

   a) The use case diagram - which we will discuss shortly

   b) Scenarios - which we met in conjunction with requirements engineering, and will discuss further below

   c) Actors

   An actor is a type of person - or sometimes another system - that must <u>use</u> the system that is to be built.

   (1) Actors are not specific people - they are specific <u>roles</u> filled by people. Thus, the same person may need to be regarded as two or more actors if he/she relates to the system in two or more roles.

   (2) Actors are not part of the system - they are outside the system boundary and <u>use</u> the system.

   (3) Example: who are the actors for the Wheels system?

   ASK

   (a) Adminstrator

   (b) Receptionist

(c) (Figure 3.14 also shows a Customer as an actor, but the explanatory text makes it clear that this would only make sense in a very high level view which considers the adminstratr and receptionist part of the system.) In general, the customer would not be considered an actor because the customer does not use the system directly; rather, the customer interacts with the receptionist who actually uses the system.

(4) Example: who would be the actors for the video store system you are building?

ASK

(a) The store clerk

(b) The manager

Note that the requirements distinguish these roles by specifying that some tasks can only be performed by the manager. At the same time, note that the <u>person</u> who is the manager can sometimes also function in the <u>role</u> of a clerk.

Again, a customer would not be considered an actor, because the customer does not interact with the system directly. Can you think of any circumstances under which a customer might be considered an actor?

ASK

If the customer interacted directly with the system - e.g. if the store provided some sort of "self serve" checkout, or an interactive kiosk where a customer could inquire about availability or where to find an item

(5) Actors may be classified as primary and secondary.

(a) The primary actor(s) are the actors for whom the system is built.

(b) Secondary actors are other actors who must also make use of the system.

Example: ATM use case diagram. Note that, in this case, a customer is considered an actor because a customer does interact directly with an ATM. The operator - who is responsible for refilling the receipt printer and the cash dispenser, removing deposited envelopes, etc. is also an actor - but a secondary one, since the system is not built simply to provide the operator with a job!

(6) Sometimes, we will also discover passive actors - external systems (or, more rarely, individuals) that the system being built makes use of to do its job, rather than vice versa.

Example: in the ATM system, the Bank.

d) Use cases - ways in which the actors use the system.

Here, the absolutely crucial idea is that use cases relate to actor <u>goals</u> - i.e. what might an actor say if you were to ask “what are you trying to accomplish?”. (Beware the tendency to make use cases too small - a use case should correspond to what the actor would perceive as a complete task)

(1) Suggest two ways of identifying use cases (QC question c)

ASK [Answer p. 42]

(a) Use cases can be identified from the actors - by asking “what are this actor’s goals when using the system?” [each goal becoming a separate use case]

Example: In the Wheels system, the receptionist uses the system to issue bikes, return bikes, and handle inquiries. The interview with Annie also reveals the need for the system to maintain customer information so that the receptionist does not have to write it out by hand each time a bike is rented.

Note: recall that use cases deal with <u>what</u> the system does, not <u>how</u> it does it; therefore, they are organized based on <u>functionality</u>, not system structure

(b) Use cases can be identified from scenarios - a use case is a generalization of a scenarios having a common goal.

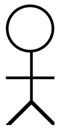
B. The use cases for a system are typically represented by one or more <u>use case diagrams.</u> The use case diagram is one nine standard types of UML diagram. The basic elements of a use case diagram are

USE ATM DIAGRAM AS AN EXAMPLE

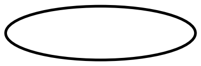
1. A box representing the system boundary. Use cases are inside the boundary; actors are outside.

   Note: often, the symbol is often actually omitted, since the it doesn't convey any new information.

2. The symbol for an actor



3. The symbol for a use case



4. Relationships, connecting actors and use cases they use, and connecting related use cases

   You can use four types of relationships in a use case diagram; list them and give a brief description of each one (QC Question h)

   Example: ATM System use case diagram

   a) The relationship between an actor and a use case - a solid line. (This is technically called a communication association). Some technical notes:

      (1) An actor often is associated with multiple use cases, and a single use case may be associated with more than one actor (though care should be used here, since actors are roles. For example, in the ATM system we would not associate ATM Sessions with the Operator actor - even though an Operator can certainly be a bank customer who uses the ATM - because when an Operator does so, he/she is in fact functioning in the Customer role

      (2) A single use case can be associated with more than one actor. There is, however, some difference of style between different writers on this point. My own preference is to associate a use case only with those actors with whom it interacts directly - e.g. in the ATM System, a Transaction is associated with both the Customer and the Bank, because it does interact with the latter.

b) Use case diagrams can also depict relationships between use cases. There are three general ways in which use cases can be related to each other

   (1) The generalization relationship is used when one use case generalizes several similar use cases. (For example, in the ATM system the transaction use case is a generalization of the various specific types of transaction: cash withdrawal, deposit, etc.)

   Generalization is denoted by the use of the “isa” triangle.

   Note: the book discusses this later in the chapter

   (2) The « include » relationship is used when one use case is contained within another use case. This can arise in one of two ways:

      (a) A given use case may be both “stand alone” and also subordinate to another case.

      Example: in the Wheels system, maintaining the customer lilst is a stand alone case, but can also be invoked while issuing a bike to a customer if the customer is new.

      Example: in the Video store system, you will might have a use case called “pay a late charge”, which might be invoked stand-alone, but could also be included as part of the process of checking out or returning an item

      (b) A given high-level goal may contain an arbitrary number of subordinate goals.

      Example, in the ATM system the use cases representing individual transactions are included within the use case for an overall customer session - a customer can perform any number of transactions during the course of one session.

   (3) The « extend » relationship is used when one use case may be extended (under some circumstances) by some special behavior that is complex enough to warrant treating separately.

   Example, in the ATM system the use case for handling the reentry of an invalid PIN extends the basic transaction use case. Because the handling of an invalid PIN entails a measure of complexity (e.g. multiple re-entry of the PIN is possible), it is treated as a separate use case.

(4) The distinction between « include » and « extend » may sometimes seem a bit unclear.

(a) « include » is used when the included case <u>always</u> occurs in the including case, and it represents a goal the user might actually have, and/or os common behavior that occurs in multiple places.

Example: in an ATM Session, Transaction is an inclusion, since it represents a customer goal, and every Session includes one or more transactions.

(b) « extend » is used when the extension is extraordinary (and therefore does not always occur) and is not actually a user goal.

Example: in the ATM Session, "Invalid PIN" is an extension, because it is hard to imagine a customer having entering an invalid PIN as a goal, and this does not normally occur as part of an ATM Session.

(5) In UML terminology, « include » and « exclude » are called <u>stereotypes</u>. You can think of them as words having special, technical meanings. In UML, stereotypes are always enclosed in guillemets - the technical name for the brackets «, ».

C. For each use case, we need to write up a description of what needs to take place. This can be done with varying degrees of formality.

1. A simple approach is to create a *flow of events* that describes that case, incorporating the user's stated requirements.

a) Example: Figure 3.5 in the book

PROJECT

b) This approach is used for the use cases in the two online examples

SHOW in AddressBook and ATM Examples

c) Some of the ATM flows of events illustrate an important point about use cases: they typically follow a normal pattern, but have to allow for variations in individual cases.

(1) ATM Session: The customer may enter an incorrect PIN.

(2) ATM Transaction: The bank may disallow the transaction for other reasons (e.g. insufficient funds for a withdrawal, or the customer does not have an account of the specific type chosen).

(3) ATM Transaction: The customer may cancel the transaction in the early stages by pushing the "Cancel" button.

etc.

d) This gets at a distinction between the scenarios developed during requirements elicitation and use cases:

(1) QC Question (d) : What is the relationship between scenarios and use cases?

ASK [answer p. 43 2nd paragraph]

Use cases are a generalization of scenarios, attempting to cover all the possibilities.

(2) QC Question (e): It would be impractical to write scenarios for every possible sequence of events. What would be a practical set of scenarios for a developer to write?

ASK [answer p. 44 middle]

A representative set, including

(a) A typical case (or cases)

(b) Obvious variations on the norm

(c) Circumstances under which the user’s goal is not achieved.

Example: What scenarios might be explored as part of developing the ATM Session use case?

ASK

i) A typical session with one transaction

ii) A typical session with multiple transactions

iii) A session where the customer has to re-enter his/her PIN due to a typo

        iv) A session which fails because the customer does not know the PIN

    e) Sometimes, a general use case has various specializations.

    EXAMPLE: "Transaction" generalizes four different specific kinds of transaction: withdrawal, deposit, transfer, and inquiry.

    GO OVER WITHDRAWAL ONLINE

2. There are also more formal approaches to documenting use cases. The book gives an example using a "two column" approach, with the actor actions listed in one column and the system response in another.

   Figure 3.6

   PROJECT

   Go over explanation in Figure 3.7. Note how the high level description of Figure 3.5 has become the Overview of Figure 3.6

3. It is possible to use an even more formal style - the so-called "usecases.org" or "fully dressed" format.

   In addition to what is in the example in the book, this includes discussion of:

    a) Stakeholders and interests - what stakeholders are affected by the case (either as an actor or otherwise) and what legitimate interests do they have.

    Example: Who are the stakeholders and what are their interests for the "Issue bike" use case of the Wheels system?

    ASK

    (1) The customer - wants to get desired bike; wants appropriate record of rental and deposit paid

    (2) The clerk - wants straightforward, accurate, efficient process

    (3) The store - wants to have an accurate rental of the bike; wants to collect appropriate rental and deposit (in cash or via accurate transmission to credit card agency); wants some fault tolerance in case of inability to communicate with credit card company

    (4) Tax authorities - want to get appropriate taxes credited

(5) Credit card company (if credit card used for payment) - wants accurate information for billing, transmitted electronically in proper format

b) Preconditions - things that must be true before the use case is started.

Example: What would be appropriate preconditions for the “Handle bike return” use case of the Wheels system?

ASK

(1) (Assuming some sort of login procedure is used for clerks) - clerk is properly logged in

(2) Bike being returned has been rented from Wheels

c) Postconditions (success guarantee) - what will be guaranteed to be true by proper execution of the use case

Example: What would be appropriate postconditions for the “Issue Bike” use case of Wheels system?

ASK

(1) Rental of bike is correctly recorded by the system

(2) Customer has receipt

(3) Payment is correctly processed (cash or credit card)

(4) Tax is properly recorded

d) Special requirements

Example: For the “Issue bike” use case of the wheels system: perhaps a device for scanning credit cards and recording signature

e) Technology and Data variations

Example: For the “Issue bike” use case of the wheels system: perhaps appropriate to consider possibility of using a bar scanner for the bikes

f) Open issues: items that still need thought

4. Regardless of what format one uses, there are certain things that are very important to bear in mind about a use case description:

   a) A use case description gives a <u>functional</u> view of the system: <u>what</u> happens, not <u>how</u> it is done. It therefore deals with things like what input is entered, what output appears, and what information is stored.

   b) A use case description needs to clearly spell out how the use case is started - e.g. what the actor does to initiate it.

D. Sometimes it is helpful to recognize that different use cases have different levels of significance.

   1. Often there will be one or a few <u>central use cases</u> in a system - use cases that get right to the heart of what the system is about.

      Example: what would be the central use case(s) for the Video Store problem?

      ASK

      Probably rent item and return item

   2. Sometimes, there will be one or more <u>high risk use cases</u> - use cases which are essential to the functioning of the system, but about which there is some uncertainty as to whether they can actually be realized.

      Example: a text book I used in a previous year had an example of an ambulance dispatching system that uses speech recognition

   3. Sometimes, there will be one or more <u>high value</u> use cases that, while not essential to the functioning of the system, have a certain “gee-whiz” value in terms of marketing it

      Example: that book cited an example of a use case for an online stock exchange system that allows a user to watch the changes in value of a stock over time graphically displayed.

E. Class Exercise: Do 3.1 (page 65) - Identify actors, and use cases, then draw use case diagram based on given scenarios (Authors’ answer page 353)

III. Functional Test Case Specifications

A. At this point, it is also possible to write specifications for a set of initial functional tests, intended to verify that the implemented system carries out the use cases.

1. The ability to write these tests at this point serves as a check on the completeness of the use case specifications.

2. The test cases can also serve to clarify the use cases.

   EXAMPLE: Show Initial Functional Test Cases for the ATM System

B. Note the following:

1. This is not a plan for thorough testing for all possible illegitimate inputs. (We will talk later about what's involved in that sort of testing.) Such tests could be planned at this point - and perhaps should be - but since we're not talking about thorough testing until later, for pedagogical reasons this example is fairly simple.

2. The focus is on <u>functional</u> testing - i.e. verifying that the system performs the functions specified in the use cases.

C. For each test case, it is necessary to specify certain things:

1. The purpose of the test case - what is it testing.

2. The initial conditions for the test.

3. The input to be supplied.

4. The expected output