Software Requirements Analysis - Use Cases

Zhiming Liu
zhiming.liu@swu.edu.cn
http://www.swu-rise.net.cn/zhiming.liu
Centre for Research and Innovation in Software Engineering
Southwest University
China

Requirements Modelling and Analysis

Modelling the real application world

- business processes — use cases
- real world concepts — classes
- real world objects — objects
- real-world structure — component interactions and class diagrams

Requirements Modelling and Analysis

Modelling the real application world

- business processes — use cases
- real world concepts — classes
- real world objects — objects
- real-world structure — component interactions and class diagrams

Software development = transforming models of real world to models of digital world

Requirements Modelling and Analysis

Modelling the real application world

- business processes — use cases
- real world concepts — classes
- real world objects — objects
- real-world structure — component interactions and class diagrams

Software development = transforming models of real world to models of digital world
Models need to be analysed and validated during the process of transformation

Use Case Driven Requirement Analsyis

Objectives

- understanding the domain to capture requirements
- create and represent functional and **nonfunctional specification**

Use Case Driven Requirement Analsyis

Objectives

- understanding the domain to capture requirements
- create and represent functional and **nonfunctional specification**

Concepts

- use cases for describing domain processes
- concepts, associations between concepts, objects, links between objects, attributes

Use Case Driven Requirement Analsyis

Objectives

- understanding the domain to capture requirements
- create and represent functional and **nonfunctional specification**

Concepts

- use cases for describing domain processes
- concepts, associations between concepts, objects, links between objects, attributes

Main Artefacts: Requirements Model (Specification)

- description of the use cases (useful but not rigorous)
- use-case diagrams,
- conceptual (class) diagram
- use case sequence diagrams
- use case operations, and
- contracts of use case operations

Use Cases

Informally speaking

- a **use case** is a story or a case of using a system by some **users** to carry out a **business process** (or a task)

A bit more precisely speaking,

- a **use case** describes the possible sequences of events of some **types of users** using some part of the system functionality to complete a process
- such a **type** of users is called an **actor**

Important Aspects of Use Cases

Use cases describe

- Functional requirements of the system from the actors perspective
- What do the actors do to use the system for realising an application task
- Each use case only uses part of the functionalities of the system, thus identifies a a component
- Actors form the external environment of the system

Important Aspects of Use Cases

Use cases describe

- Functional requirements of the system from the actors perspective
- What do the actors do to use the system for realising an application task
- Each use case only uses part of the functionalities of the system, thus identifies a a component
- Actors form the external environment of the system
- Consistent with David Parnas Four Value Model - $http://en.wikipedia.org/wiki/David_Parnas$

Important Aspects of Use Cases

Use cases describe

- Functional requirements of the system from the actors perspective
- What do the actors do to use the system for realising an application task
- Each use case only uses part of the functionalities of the system, thus identifies a a component
- Actors form the external environment of the system
- Consistent with David Parnas Four Value Model - $http://en.wikipedia.org/wiki/David_Parnas$
- Similar to Michael Jackson's Problem Frames - $http://en.wikipedia.org/wiki/Problem_frames_approach$

Important Aspects of Use Cases

Use cases describe

- Functional requirements of the system from the actors perspective
- What do the actors do to use the system for realising an application task
- Each use case only uses part of the functionalities of the system, thus identifies a a component
- Actors form the external environment of the system
- Consistent with David Parnas Four Value Model - $http : //en.wikipedia.org/wiki/David_Parnas$
- Similar to Michael Jackson's Problem Frames - $http : //en.wikipedia.org/wiki/Problem_frames_approach$
- Applicable to interface-based design of applications in digital-ecosystems

Actors

When performing use cases

- communicate with the system by sending messages to and receiving messages from the system – direct actors
- communicate with each other to exchange information
- can be human users, devices or digital/computing systems interacting with the system/component underdevelopment

How to Capture Use Cases?

- identify application processes
- identify actors involved in the use cases
- identify the *communication actions* that actors take when perform a use case

Focus on the actors that directly communicate with the system — direct actors

Example: in POST

- When perform **process a sale** with the trade system

Example: in POST

- When perform **process a sale** with the trade system
- what are the actors and what are their actions?

Example: in POST

- When perform **process a sale** with the trade system
- what are the actors and what are their actions?

- two actors must be involved: **Customer** and **Cashier**,
- the following sequence of actions must be performed:
  1. The Customer arrives at a **cash desk** with **items** to purchase.
  2. The **Cashier** *records* the purchase **items** and *collects* **payment**.
  3. On completion, the Customer leaves with the items.

Example: in POST

- When perform **process a sale** with the trade system
- what are the actors and what are their actions?

- two actors must be involved: **Customer** and **Cashier**,
- the following sequence of actions must be performed:
  1. The Customer arrives at a **cash desk** with **items** to purchase.
  2. The **Cashier** *records* the purchase **items** and *collects* **payment**.
  3. On completion, the Customer leaves with the items.

Each use case is a complete course of events in the system, seen from a user's perspective

Example - Library

To **register a library member** when the library system is used

Example - Library

To **register a library member** when the library system is used

- two actors: **librarian** and **member**
- the following sequence of events must be performed
  1. The **member** arrives at a **desk** with **identification**
  2. The **librarian** *records* the **personal details** and *issues* a **card**
  3. On completion, the member leaves with the card.

Example - Library

To **register a library member** when the library system is used

- two actors: **librarian** and **member**
- the following sequence of events must be performed
  1. The **member** arrives at a **desk** with **identification**
  2. The **librarian** *records* the **personal details** and *issues* a **card**
  3. On completion, the member leaves with the card.

Exercise: More use cases of the library system, their actors and actions

Incremental Analysis of Use Cases

- First write a **high level** use case description to obtain initial understanding of the overall process
- A **high-level use case** describe a process very briefly with actors and abstract actions that the actors perform

A high level use case can be presented in a structured format

Use case:	**Name of use case** (use a phrase starting with a verb)
Actors:	List of actors, indicating who initiates the use case.
Purpose:	Intention of the use case
Overview:	A brief description of the process

Example

Use case:	**Process Sale with Cash Payment**
Actors:	Customer (initiator), **Cashier**
Purpose:	Capture a **sale** and its **cash payment**
Overview:	A **customer** arrives at a checkout with **items** to purchase. The **cashier** *records* the purchase **items** and *collects* a **cash payment**. On completion, the Customer leaves with the items.
Cross References:	

Extended Use Cases

Stopped at this place

Extended Use Cases

Taking an high level use case and analyse it in details to write an *extended use case*

- what *input data* does an actor provide to the system in performing an action
- what *output data* does an actor receives after an action
- what does the system do when an actor performs an action: **what data to be updated, created, to be checked or found**
- what are the **main course of actions** and when *exceptions* may occur and what to do to handle them
- the **precondition** for the use case to start, and the **invariant property** to be preserved by the actions

Information in Extended Use Cases

- typical course of events: describes the general pattern of interactions carried out
- alternative courses of events: describes the patterns of interactions carried out under exceptional conditions.

An interaction between an actor and the system shows

- what an actor asks the system to do and with what input
- what should be done by the system to respond: what should be updated, checked, or output.

Identification and definition of functions, data, classes/objects and their relations

Structured Representation of Use Cases

High Level Description;
Precondition: // mostly ignored in informal analysis
Invariant: //mostly ignored in informal analysis
Typical Course of Events

Actor Action	**System Response**
Actions of the actors	Responses of the system

Alternative Courses

- Alternatives that may arise at *line_number*. Description of exception.

Process Sale With Cash Payment

Typical Course of Events

	Actor Action		**System Response**
1.	This use case begins when a **Customer** arrives a **Cash Desk** with **items** to purchase.		
2.	The **Cashier** *records* the **identifier** from each **item**.	**3.**	Determines the **item price** and *adds* the **item information** to the **running sale's transaction**.
	If there is more than one same item, the Cashier can enter the **quantity** as well.		The **description** and **price** of the current **item** are presented.

Continued

4. On completion of the item entry, the Cashier indicates to the Cash Desk that item entry is completed.

5 Calculates and presents the *sale total*.

6. Cashier tells the Customer the total.

7. The Customer gives a **cash payment**, possibly greater than the sale total.

8. The Cashier *records* the **cash received amount**.

9. Shows the **balance** due back to the Customer. Generate a **receipt**.

Continued

10. Cashier deposits the cash received and extracts the balance owing.
    Cashier gives the balance owing and the printed receipt to the Customer.
11. Logs the **completed sale**.
12. Customer leaves with the items purchased.

Alternative Courses

- Line 2: Invalid identifier entered. Indicate error.
- Line 7: Customer didn't have enough cash. Cancel sales transaction.

Working out is Brainstorming

A asks :	How does the buying items process start?
C says :	It starts when a customer brings the items to checkout
A asks :	Then what will happen next?
C says :	A cashier uses the identification of item to find out the information from the catalogue calculate the subtotal of the sale.
A thinks :	If the cashier enters the identification to the system, the system should determine the the price and calculate the current subtotal.
A asks :	What will happen when the last item is recorded?
C says :	The cashier calculates the total of the sale and tells the customer.

Continued

A thinks :	If the cashier indicates the system that entry of items is completed, the system should calculate and present the sale total
A asks :	What happens after the customer pay?
C says :	The Cashier notes down the amount received and calculates the balance due back to the customer complete the sale and logs the sale record in the store.
A thinks :	If the cashier records the amount received, into the system, it should calculate the balance and log the completed sale.

Assumptions

Make clear the assumptions made when analysis an use case

- Cash payments only, no inventory maintenance
- No tax calculations, no coupons
- Cashier does not have to log in; no access control
- No record maintained of a customer and her/his buying habits
- No control of the cash drawer.
- Name and address of store and date and time of sale are not shown on the receipt; Cashier ID and CashDesk ID are not shown on the receipt

Always begin the development with building a simple system and then let it evolve

Example - ATM

- Identify an actor

Example - ATM

- Identify an actor
- Bank Customer

Example - ATM

- Identify an actor
- Bank Customer
- Use cases involve Bank Customer?

Example - ATM

- Identify an actor
- Bank Customer
- Use cases involve Bank Customer?

1. Withdraw Money

Example - ATM

- Identify an actor
- Bank Customer
- Use cases involve Bank Customer?

1. Withdraw Money
2. Deposit Money, and

Example - ATM

- Identify an actor
- Bank Customer
- Use cases involve Bank Customer?

1. Withdraw Money
2. Deposit Money, and
3. Transfer between Accounts

Description of Withdraw Money (Exercise)

Use case:	**Withdraw Money**
Actors:	Bank Customer (initiator).
Purpose:	Capture a withdraw of money.
Overview:	A Bank Customer arrives at a bank. She identifies herself. She chooses from which account to withdraw money and specifies how much to withdraw. On completion, the Customer leaves with the money.

Expanded Version of Withdraw Money

1. This use case begins when a Customer arrives at a bank.

2. Customer inputs the identifier.

3. Checks the identification, and shows the description of the choice of services.

4. Customer indicates to withdraw some money.

5. Shows that Customer should indicate the amount and account from which to withdraw.

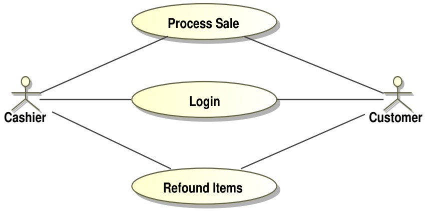
6. Customer enters the account and amount.

7. Deducts the amount from the account and dispenses the money.

8 On completion, Customer leaves with the money.

Use-Case Diagrams

A **use-case diagram** describes part of the *requirements model*, illustrating a set of use cases, actors, and their relationships



Use a use case diagram to group and relate use cases

Decompose Use Cases

Pay by Cash
Typical Course of Events

Actor Action	**System Response**
1. Customer gives a cash payment, possibly greater than the sale total.	
2. Cashier records the cash tendered.	**3.** Show the balance due back to the Customer.
4. Cashier deposits the cash received an extracts the balance owing. Cashier gives the balance owing and the printed receipt to the Customer.	

Contiuned

Alternative Courses

- Line 3. If cash the amount tendered is not enough, exception handling
- Line 4: Insufficient cash in drawer to pay balance. Ask for cash from supervisor.

Using exactly the same techniques in the creation of Pay by Cash, we can create two use cases Pay by Credit and Pay by Cheque.

Compose Use Cases

1. This use case begins with a Customer arrives at a CashDesk checkout with items to purchase.

2. Cashier records the identifier from each item.

If there is more than one of the same item, Cashier can enter the quantity as well.

3. Determines the item price and adds the item information to the running sales transaction.

The description and price of the current item are presented.

4. On completion of the item entry, Cashier indicates to the CashDesk that item entry is completed.

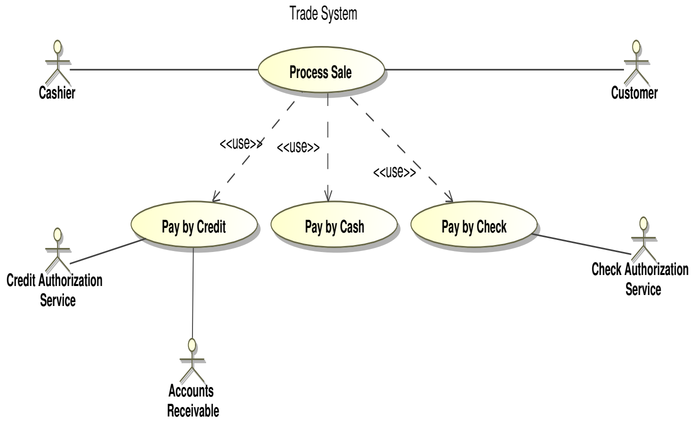
5 Calculates and presents the sale total.

6. Cashier tells Customer the total

Continued

7. Customer chooses payment method:
   (a) If cash payment, **initiate** *Pay by Cash*.
   (b) If credit payment, **initiate** *Pay by Credit*.
   (c) If cheque payment, **initiate** *Pay by Cheque*.
8. Logs the completed sale.
9. Prints a receipt.
10. Cashier gives the printed receipt to the Customer.
11. Customer leaves with the items purchased.

Relating Use Cases



Warning

Not always right to combine arbitrary two list of interactions to make a use case

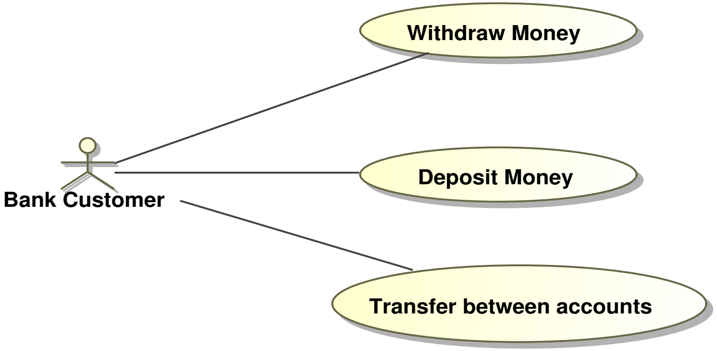
- Do not make “Borrow a Book” and “Return a Book” into one use case.
- Do not make “Add a member” and “Remove a member” into one use case
- Think whether they can be carried by an operator/user in one window and in one go.
- Think about them in terms of business processes

A use case is an end to end complete story using the system in carrying out a business process

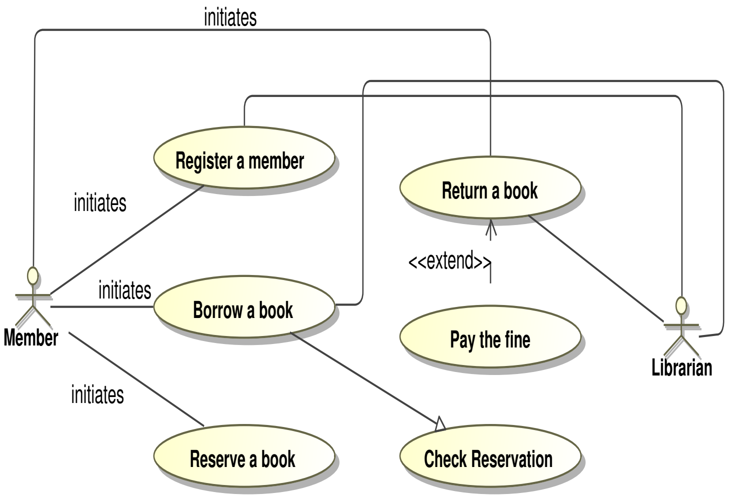
Use Case Analysis in Development Process

- Identify actors and use cases according to business processes
- Incremental use case capture and analysis: from abstract simplified high level, to detailed version with data to input, output, store and update
- Decompose/compose use cases
- Extended use cases for risk analysis
- Ranking use cases for project plan
- Present use cases clearly with structured presentations and use case diagrams

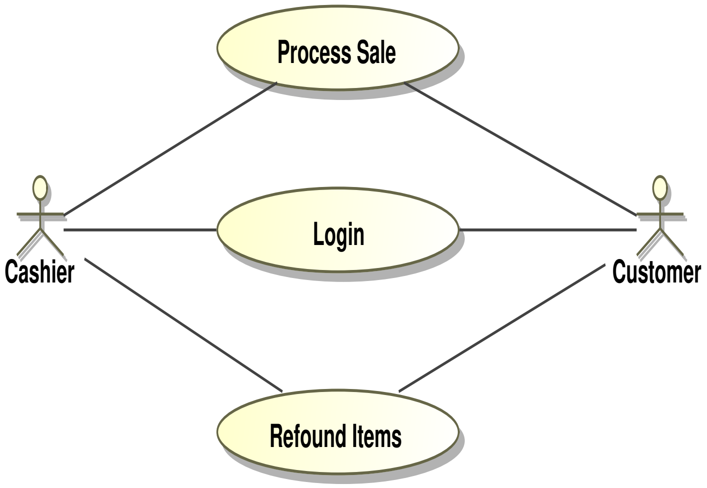
Example: ATM


Withdraw Money
Deposit Money
Bank Customer
Transfer between accounts

Example: Library



Example: Online Billing and Payment



Non-functional Requirements

constraints imposed on the system and the designer

- response time and memory requirements,
- operating system platforms,
- ease of use, interface metaphor,
- retail cost,
- fault-tolerance, and so on

- some cut across all use cases, such as the operating system platform.
- some may be related to particular functions of some use cases, such as response time of some functions.

Non-functional Requirements are now often called extra functionality, attributes

Example of Extra Functionality

Attribute	**Constraints**
response time	when recording an item, the description and price will appear in 5 seconds
interface metaphor	forms-metaphor windows and dialog boxes
fault-tolerance	must log authorised credit payments to accounts receivable within 24 hours, *even if power or device fails*
operating system platform	Microsoft Window XP and Linux

Why Use Cases

- represent the functional requirements
  - not only answer the question what the system should do but answer it in terms of what it should do for each type of users
  - not only identify the system functionalities but also the relationship among them
  - help the client, the users, and the developers agree on what the system should do and how to use the system
- important for identifying the objects involved in the application domain.
- placeholders for nonfunctional requirements
- important for project plan, system design and testing
- natural decomposition of the system, and support incremental compositional development and verification