ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

SYSTEMS ANALYSIS AND DESIGN

Lecturers: Nguyen Thanh Binh – Nguyen Quang Vu – Le Viet Truong – Le Thi Bich Tra – Vo Van Luong – Nguyen Thi Hanh

Faculty of Computer Science

Vietnam - Korea University of Information and Communication Technology (VKU)

http://vku.udn.vn/

Requirement modelling

- **Requirement**
- **Use-case diagram**

Software Development Activities

Requirements Gathering

Define requirement specification

Analysis

Define the conceptual model

Design

Design the solution / software plan

Implementation

Code the system based on the design

Integration and Test

Prove that the system meets the requirements

Deployment

Installation and training

Maintenance

Post-install review
Support docs
Active support

Requirement

- The systems development process transforms the existing (as is) system into the proposed (to be) system
- Requirements determination
  - The single most critical step of the entire SDLC
  - Changes can be made easily in this stage
  - Most (>50%) system failures are due to problems with requirements
  - The iterative process of OOSAD is effective because:
    - Small batches of requirements can be identified and implemented incrementally
    - The system will evolve over time

Requirements Determination

- Purpose: to convert high level business requirements (from the system request) into detailed requirements that can be used as inputs for creating models
- What is a requirement?
  - A statement of what the system must do or a characteristic it must have
  - Will later evolve into a technical description of how the system will be implemented
- Types:
  - Functional: relates to a process or data
  - Non-functional: relates to performance or usability

Non-Functional Requirements

Requirement type	Example
Operational	• The system should be able to fit in a pocket or purse • The system should be able to integrate with the existing inventory system.
Performance	• Any interaction between the user and the system should not exceed 2 seconds. • The system should receive updated inventory information every 15 minutes.
Security	• Only direct managers can see personnel records of staff • Customers can see their order history only during business hours.
Cultural & Political	• The system should be able to distinguish between United States and European currency • The system shall comply with insurance industry standards.

Requirements Definition

- Functional & non-functional requirements listed in outline format
- May be prioritized
- Provides information needed in subsequent workflows
- Defines the scope of the system

Determining Requirements

- Business & IT personnel need to collaborate
- Strategies for effective results:
  - Business Process Analysis (BPA)
  - Business Process Improvement (BPI)
  - Business Process Re-Engineering (BPR)

Determining Requirements

- Requirements are best determined by systems analysts ***and*** business people together
- Strategies for analyzing the requirements
  - Business Process Analysis (BPA)
  - Business Process Improvement (BPI)
  - Business Process Reengineering (BPR)
- Techniques for identifying requirements
  - Interviews, questionnaires and/or observation
  - Joint application development (JAD)
  - Document analysis

Creating a Requirements Definition

- Determine the types of functional and non-functional requirements applicable to the project
- Use requirements-gathering techniques to collect details
- Analysts work with users to verify, change and prioritize each requirement
- Continue this process through analysis workflow, but be careful of scope creep
- Requirements that meet a need but are not within the current scope can be added to a list of future enhancements

Problems in Requirements Determination

- Analyst may not have access to the correct users
- Requirements specifications may be inadequate
- Some requirements may not be known in the beginning
- Verifying and validating requirements can be difficult

Requirements Analysis Strategies

- Business Process Automation (BPA)
  - Least amount of change to the current system
  - Use computer technology to automate some portions
- Business Process Improvement (BPI)
  - Moderate amount of change is required
  - Designed to improve efficiency of the current system
- Business Process Reengineering (BPR)
  - Most amount of change—a complete makeover
  - Focus is on the to-be system—little time spent on the current system

Business Process Automation

- Techniques
  - Problem analysis
    - Ask users to identify problems with the current system
    - Ask users how they would solve these problems
    - Good for improving efficiency or ease-of-use
  - Root cause analysis
    - Focus is on the cause of a problem, not its solution
    - Create a prioritized list of problems
    - Try to determine their causes
    - Once the causes are known, solutions can be developed

Business Process Improvement

- Techniques:
  - Duration analysis
    - Determine the time required to complete each step in a business process
    - Compare this to the total time required for the entire process
    - Large differences suggest problems that might be solved by:
      - Integrating some steps together
      - Performing some steps simultaneously (in parallel)
  - Activity-based costing—same as duration analysis but applied to costs
  - Informal benchmarking—analyzes similar processes in other successful organizations

Business Process Reengineering

- Institutes maximum change: “Out with the old and in with the new”
- Techniques:
  - Outcome analysis—what does the customer want in the end?
  - Technology analysis—apply new technologies to business processes and identify benefits
  - Activity elimination—eliminate each activity in a business process in a “force-fit” exercise

Selecting An Appropriate Strategy

	Business Process Automation	Business Process Improvement	Business Process Reengineering
Potential business value	Low–moderate	Moderate	High
Project cost	Low	Low–moderate	High
Breadth of analysis	Narrow	Narrow–moderate	Very broad
Risk	Low–moderate	Low–moderate	Very high

Requirements Gathering Techniques

- Process is used to:
  - Uncover all requirements (those uncovered late in the process are more difficult to incorporate)
  - Build support and trust among users
- Which technique(s) to use?
  - Interviews
  - Joint Application Development (JAD)
  - Questionnaires
  - Document analysis
  - Observation

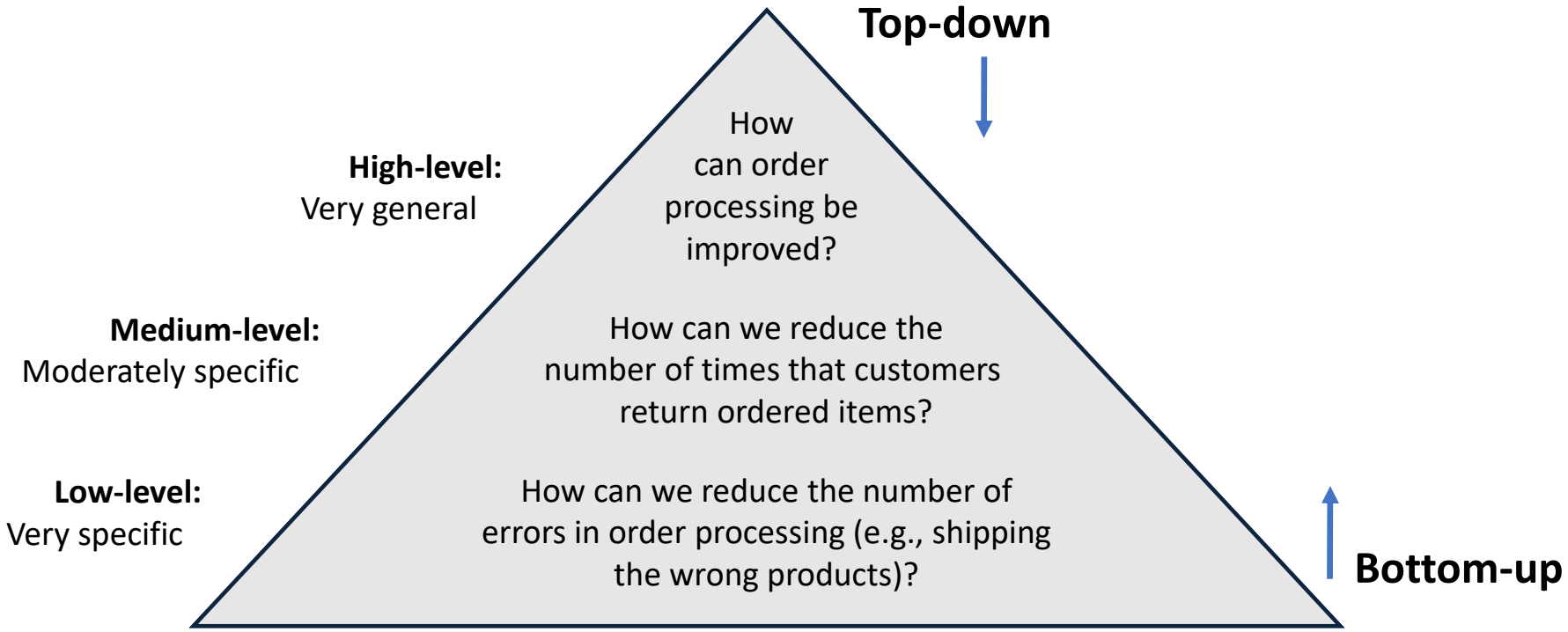
Interviews

- Most popular technique—if you need to know something, just ask
- Process:
  - Select people to interview and create a schedule
  - Design interview questions (Open-ended, closed-ended, & probing types of questions)
  - Prepare for the interview (Unstructured vs. structured interview organized in a logical order)
  - Conduct the interview (Top-down vs. bottom-up)
  - Follow-up after the interview

Question Types

Types of Questions	Examples
Closed-ended questions	• How many telephone orders are received per day? • How do customers place orders? • What information is missing from the monthly sales report?
Open-ended questions	• What do you think about the current system? • What are some of the problems you face on a daily basis? • What are some of the improvements you would like to see in a new system?
Probing questions	• Why? • Can you give me an example? • Can you explain that in a bit more detail?

Interviewing Strategies



Post-Interview

- Prepare notes and send to the interviewee for verification

Interview Notes Approved by: Linda Estey

Person Interviewed: Linda Estey,
Director, Human Resources

Interviewer: Barbara Wixom

Purpose of Interview:

- Understand reports produced for Human Resources by the current system
- Determine information requirements for future system

Summary of Interview:

- Sample reports of all current HR reports are attached to this report. The information that is not used and missing information are noted on the reports.
- Two biggest problems with the current system are:
  1. The data are too old (the HR Department needs information within two days of month end; currently information is provided to them after a three-week delay)
  2. The data are of poor quality (often reports must be reconciled with departmental HR database)
- The most common data errors found in the current system include incorrect job level information and missing salary information.

Open Items:

- Get current employee roster report from Mary Skudrna (extension 4355).
- Verify calculations used to determine vacation time with Mary Skudrna.
- Schedule interview with Jim Wack (extension 2337) regarding the reasons for data quality problems.

Detailed Notes: See attached transcript.

Joint Application Development (JAD)

- Joint user-analyst meeting hosted by a facilitator
  - 10 to 20 users
  - 1 to 2 scribes as needed to record the session
  - Usually in a specially prepared room
- Meetings can be held electronically and anonymously
  - Reduces problems in group settings
  - Can be held remotely
- Sessions require careful planning to be successful
  - Users may need to bring documents or user manuals
  - Ground rules should be established

Questionnaires

- A set of written questions used to obtain information from individuals
- May be paper based or electronic (e.g., web based)
- Common uses:
  - Large numbers of people
  - Need both information and opinions
  - When designing for use outside the organization (customers, vendors, etc.)
- Typical response rates: < 50% (paper); < 30% (Web)

Questionnaire Steps

- Select the participants
  - Identify the population
  - Use representative samples for large populations
- Designing the questionnaire
  - Careful question selection
  - Remove ambiguities
- Administering the questionnaire
  - Working to get good response rate
  - Offer an incentive (e.g., a free pen)
- Questionnaire follow-up
  - Send results to participants
  - Send a thank-you

Good Questionnaire Design

- Begin with non-threatening and interesting questions
- Group items into logically coherent sections
- No important items at the very end
- Do not crowd a page with too many items
- Avoid abbreviations
- Avoid biased or suggestive items or terms
- Number questions to avoid confusion
- Pretest to identify confusing questions
- Provide anonymity to respondents

Document Analysis

- Provides information about the “as-is” system
- Review technical documents when available
- Review typical user documents:
  - Forms
  - Reports
  - Policy manuals
- Look for user additions to forms
- Look for unused form elements

Observation

- Users/managers often don’t remember everything they do
- Checks validity of information gathered in other ways
- Behaviors may change when people are watched
  - Workers tend to be very careful when watched
  - Keep a low profile
  - Try not to interrupt or influence workers
- Be careful not to ignore periodic activities
  - Weekly … Monthly … Annually

Requirements-Gathering Techniques Compared

- A combination of techniques may be used
- Document analysis & observation require little training; JAD sessions can be very challenging

	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to-be	As-is, improvements, to-be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low–Medium	Low	Low	Low–Medium

Alternative Techniques

- Concept Maps
  - Represent meaningful relationships between concepts
  - Focus individuals on a small number of key ideas
- Story Cards & Task Lists
  - Associated with agile development methods
  - File cards with a single requirement
  - Each requirement is discussed
    - How much effort is required to implement it
    - A task list is created for each requirement (story)
    - Large requirements can be split into smaller sections

The System Proposal

- Combines all material created in planning and analysis
- Included sections:
  - Executive summary
    - Provides all critical information is summary form
    - Helps busy executives determine which sections they need to read in more detail
  - The system request
  - The workplan
  - The feasibility analysis
  - The requirements definition
  - Current models of the system (expected to evolve)

System Proposal Template

1. **Table of Contents**
2. **Executive Summary**

   A summary of all the essential information in the proposal so a busy executive can read it quickly and decide what parts of the proposal to read in more depth.
3. **System Request**

   The revised system request form (see Chapter 2).
4. **Workplan**

   The original workplan, revised after having completed analysis (see Chapter 2).
5. **Feasibility Analysis**

   A revised feasibility analysis, using the information from analysis (see Chapter 2).
6. **Requirements Definition**

   A list of the functional and nonfunctional business requirements for the system (this chapter).
7. **Functional Model**

   An activity diagram, a set of use case descriptions, and a use case diagram that illustrate the basic processes or external functionality that the system needs to support (see Chapter 4).
8. **Structural Models**

   A set of CRC cards, class diagram and object diagrams that describe the structural aspects of the to-be system (see Chapter 5). This may also include structural models of the current as-is system that will be replaced.
9. **Behavioral Models**

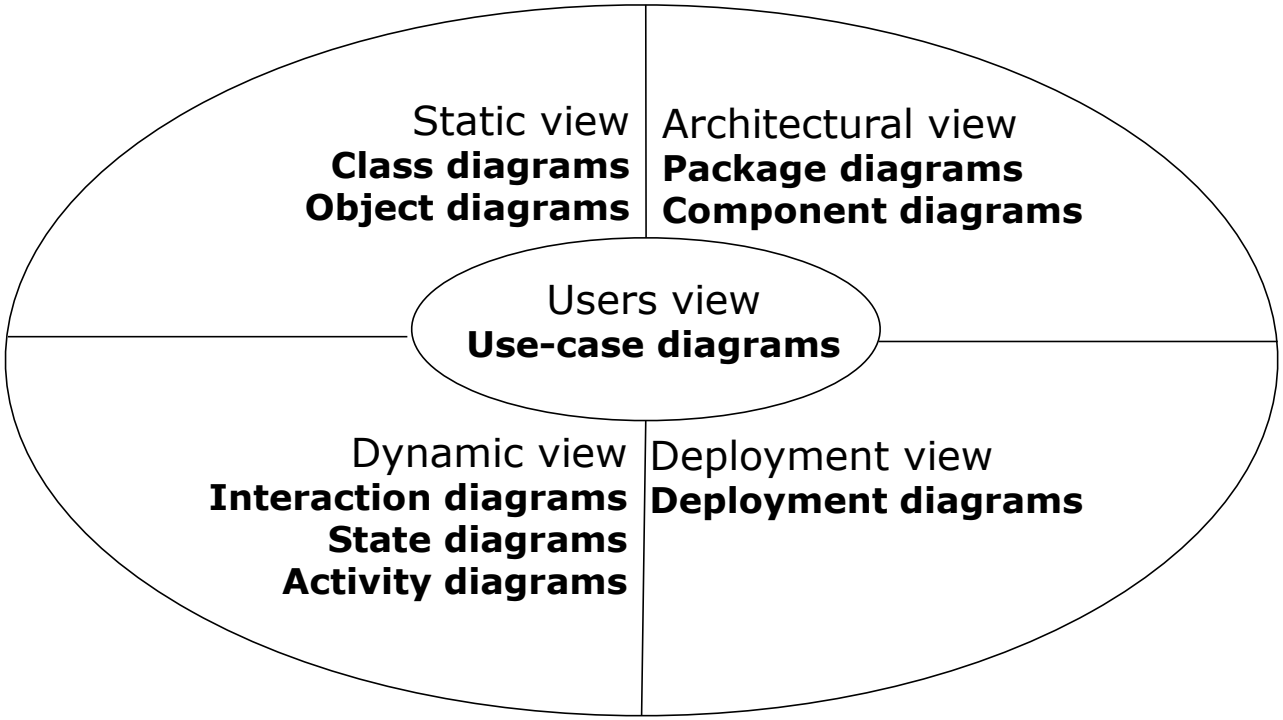
   A set of sequence diagrams, communication diagrams, behavioral state machines, and a CRUD matrix that describe the internal behavior of the to-be system (see Chapter 6). This may include behavioral models of the as-is system that will be replaced.

Appendices

These contain additional material relevant to the proposal, often used to support the recommended system. This might include results of a questionnaire survey or interviews, industry reports and statistics, and so on.

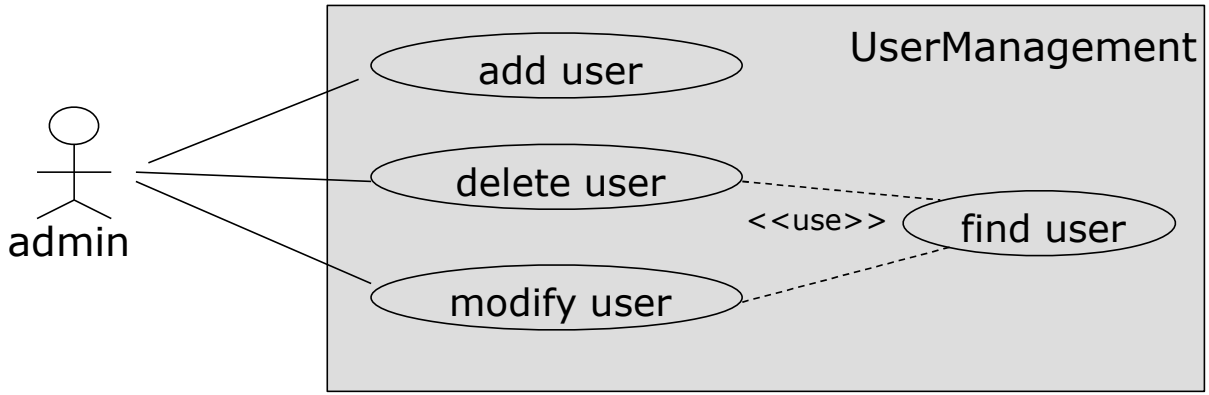
Requirement Modelling

Views


Static view
Class diagrams
Object diagrams
Architectural view
Package diagrams
Component diagrams
Users view
Use-case diagrams
Dynamic view
Interaction diagrams
State diagrams
Activity diagrams
Deployment view
Deployment diagrams

Use-case diagram

- The first step in requirement analysis is to determine use-cases of the system
- Use-case diagrams
  - allow to **represent the functionalities of the system in the users view**
  - allow to delimit the boundary of the system



User-centred design

- The development of a system should always be centred around the needs of users
  - Understand who are the users
  - Understand the tasks performed by the users
  - Make sure that users are involved in the decision-making process
  - Design the interface well following the needs of the user
  - Users will need to evaluate prototypes and return their comments

Cash register at the supermarket

Interest of user-centred design

- Meets the actual requirements
- Reduce costs related to changes or maintenance
- Allow to better define the properties in the development
- Reduce learning time
- Reduce training and supporting costs
- Allow efficient use
- Making the system more attractive and better suited to its market

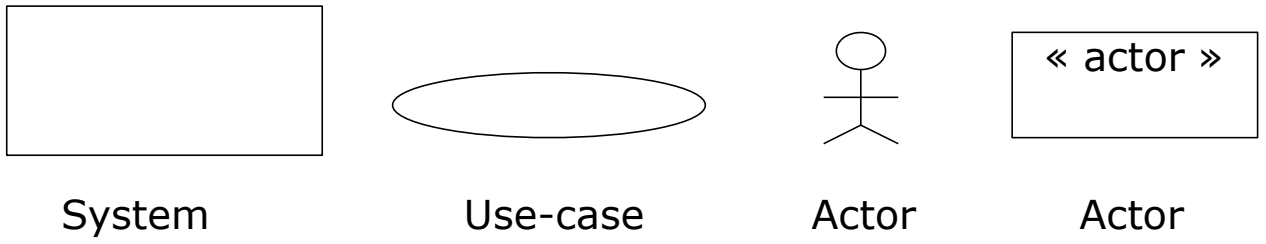
Determining users' characteristics

- Good questions
  - What are their goals?
  - How will they use the software?
  - What is their level of computer literacy?
  - What are their psychological characteristics?
  - What are their habits?

Use-case diagrams

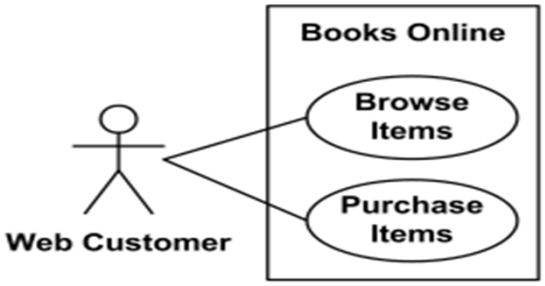
- A use-case diagram consists of three parts
  - The system
  - The use-case
  - The actor

- Graphical representation

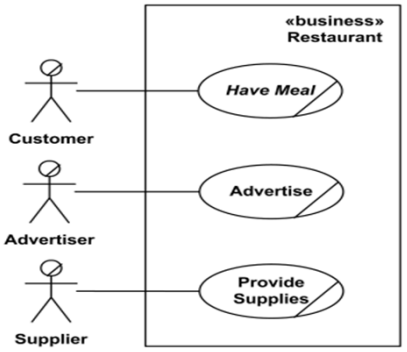


System

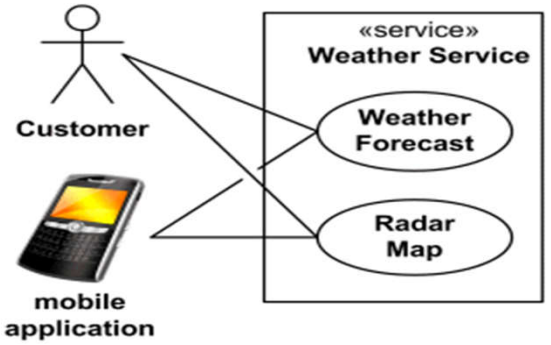
- The system can be any system, not only the software system
- It defines the boundary of the system in a clear and precise manner
  - Not too ambitious
  - Only determine basic functionalities
  - Build a well defined architecture
  - Additional functionality can be added during development



"Books Online" system



"Restaurant" system



"Weather Service" system

Use-case

- A use-case is a typical interaction or a typical sequence of interactions between the system and its environment
- The objective of a use-case is to model the system
  - according to the perspective of user interacting with the system
  - to accomplish their objectives
- A use-case may can be either large or small
- Example: developing a tool for text processing
  - Some possible use-cases
    - Create a new document
    - Modify an existing documents
    - Delete a document
    - Input new text, …

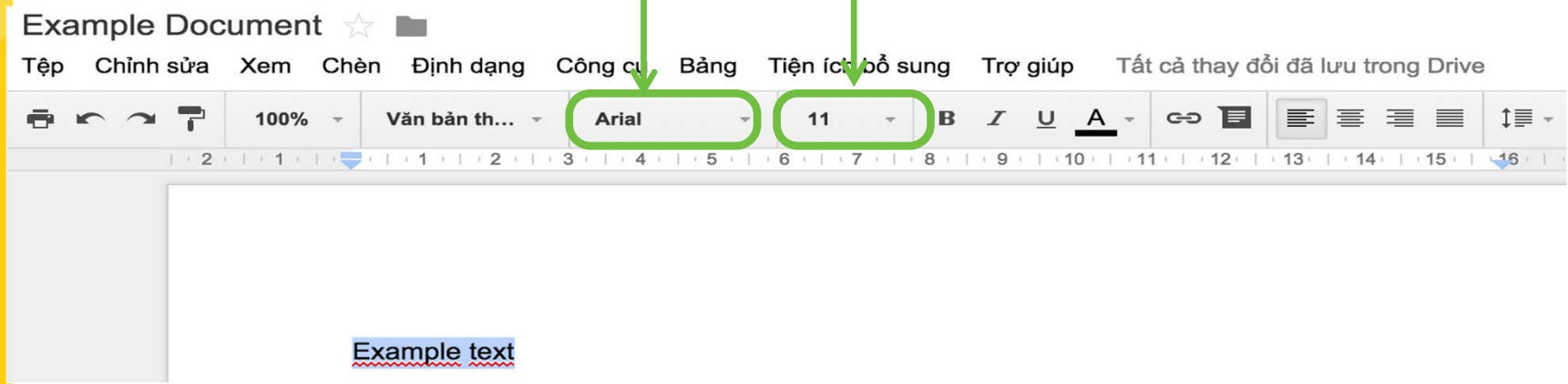
Text processing tool

Use-case

- A use-case needs to
  - always correspond to a high level objective
  - describe the interaction between the user and the system, not the operations that the system should perform
  - cover all the steps to follow in performing a given task
  - be written, to the possible extent, independently of the user interface
  - include only the interactions with the system

Use-case

- **Objectives** and **interactions**
  - **Objectives** of users: what users expect from the system
  - **Interactions** with the system: mechanisms to meet those objectives
  - Define the objectives then determine the interactions to achieve objectives
  - Example
    - Objective: define the document style
    - Interactions: choose the font, choose sizes, choose the page layout, …



Use-case

- Example: developing of an ATM system
- Some interactions in the following scenario
  - Insert the card
  - Enter the PIN code
  - Choose the amount to be withdrawn
  - Confirm the amount
  - Take the card
  - Take the money
  - Take the receipt

- **Is each interaction a use-case?**

Use-case

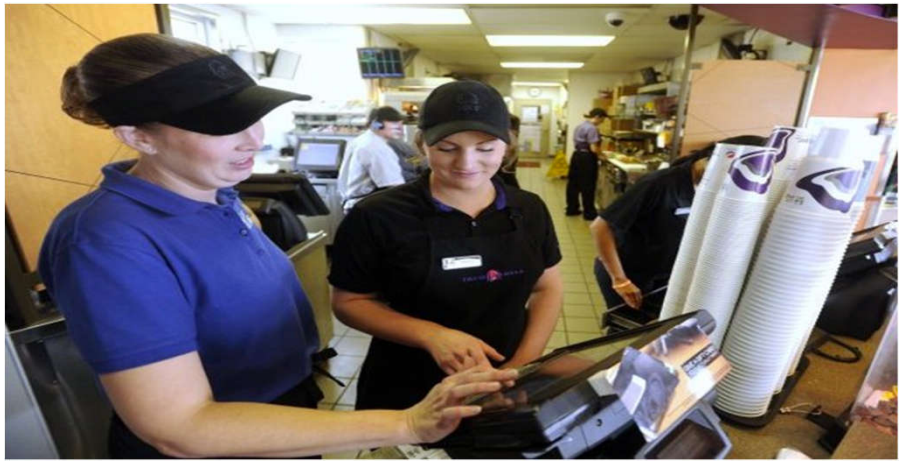
- Example (continue)
  - The answer is no
  - Since some interactions such as “confirm the amount” do not meet a goal of the user
  - The goal of the user in this case is to **withdraw money**: this is a use-case

Actors

- An **actor** is a **role** played by the user or an external entity during interaction with the system
- Who or what uses the system
- Actors communicate with the system by sending and receiving messages
- Example
  - Develop a system of cash register at the supermarket
    - Possible actors
      - Client
      - Cashier
      - Manager
      - Inventory manager

Actors

- Distinguishing two notions: **actor** and **user**
  - Multiple users may correspond to a single actor
    - Different cashiers play the same role in the system
  - A user may correspond to several actors
    - A user can simultaneously be a cashier and a manager in the system

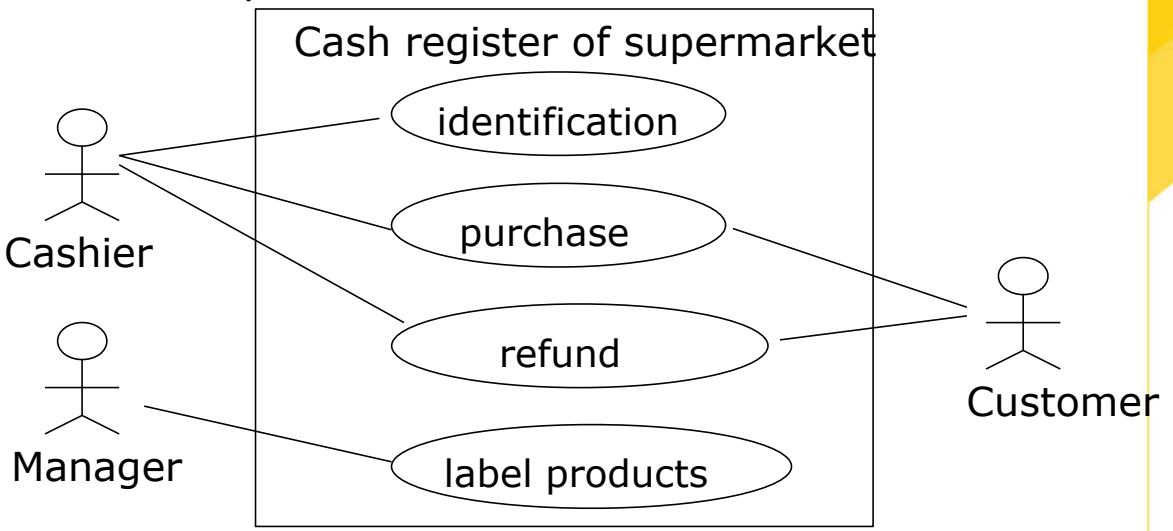


cashier and manager

cashier and customer

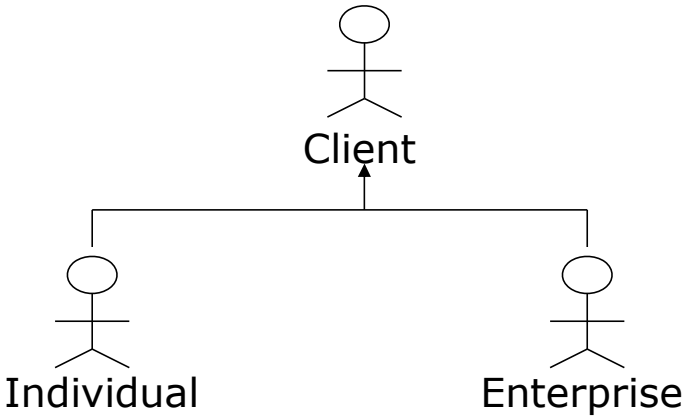
Actors

- Questions for identifying the system actors
  - Who will use the main features of the system?
  - Who will need the support of the system to perform its tasks?
  - Who should update, administer and maintain the system?
  - Does the system interacts with other systems?
  - Who or what has interests on the results of the system?



Relations between the actors

- Inheritance between actors



Use-case specification

- Typical specification of a use-case
  - **Use-case**: name of a use-case often begins with a verb
  - **Actors**: list of stakeholders concerning the use-case
  - **Objective**: objective of the use-case
  - **Description**: a brief description of treatment to achieve

- Example
  - **Use-case**: purchase products
  - **Actors**: Client, Cashier
  - **Objective**: describe a purchase of products by the customer with cash payment
  - **Description**: The client comes in the box with the selected products. The cashier encodes products, announces the total. The customer pays. The cashier registers the payment.

Use-case specification

- The use-case specification may add
  - the references concerning the specification of the requirement
  - the pre- and post-conditions of the use-case

- Example
  - **Use-case**: purchase products
  - **Actors**: Client, Cashier
  - **Objective**: describe a purchase of products by the customer with cash payment
  - **References**: R1.2, R3.4
  - **Pre-conditions**: the cashier is identified and authorized
  - **Post-conditions**: the purchase is registered, the payment is made, the receipt is printed
  - **Description**: The clients comes in the box with the selected products. The cashier encodes products, announces the total. The customer pays. The cashier registers the payments.

Use-case specification

- A use-case can be specified by adding **scenarios**
- A scenario describes the specific actions of the actors in the system
- A scenario consists of **main interactions** and **exceptional interactions**
- The actions can be divided into **two flows**
  - Flow of actions concerning the **actors**
  - Flow of actions concerning the **systems**
- Example
  - A scenario for “purchase products” use-case

Use-case specification

- Main interactions of “purchase products” scenario

Actions of actor	Actions of system
1. The customer comes to the cash desk with the products to buy	
2. The cashier encodes the identifier of each product If a product has more than one item, the cashier inputs the number of items	3. The cash desk displays the description and price of the product This number is displayed
4. After having encoded all of the products, the cashier signals the end of the purchase	5. The cash desk calculates and displays the total amount that the customer has to pay
6. The cashier announces the total amount to the customer	
7. The customer pays	
8. The cashier input the amount of money paid by the customer	9. The cash desk displays the balance

Use-case specification

- Main interactions of “purchase products” scenario (continue)

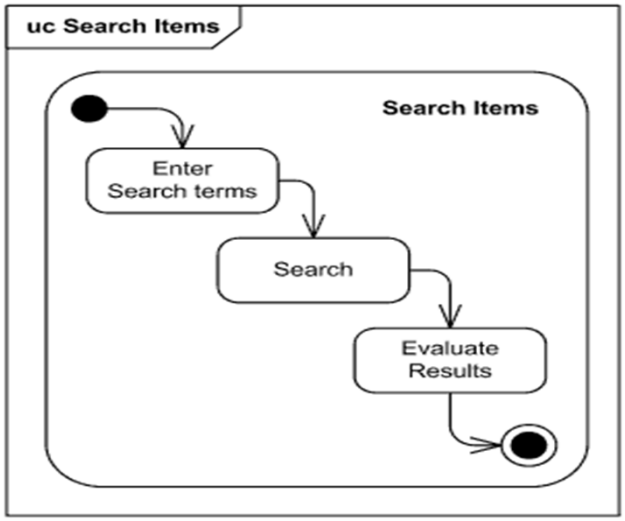
Actions of actor	Actions of system
	10. The cash desk prints the receipt
11. The cashier gives change to the customer and the receipt	12. The cash desk saves the purchase
13. The customer leaves the cash desk with the bought products	

- Exceptional interactions of “purchase products” scenario

Actions of actor	Actions of system
	3. The product identifier is not correct, the system displays the error
7. The customer doesn’t have enough money. The cashier cancel the purchase	

Use-case specification

- Remarks
  - The use-case's specification format is only a proposal. Therefore, it is not strict
  - The interactions are described in more detail for important use-cases
  - Use-case's interaction can also be described using activity diagram, state diagram or interaction diagram



Use-case's interactions described in activity diagram

Use-cases identification techniques

- **Software Developer write requirements specification themselves**
  - Lack of human reactions (future users of the system)

- **Interview**

User interview

User interview

Use-cases identification techniques

- Workshop *(Organize meetings)*
  - Meeting of all the concerned people of the system to be developed
    - Customers, Users, Software developers
    - Everyone gives their ideas
      - List all the possible actors, use-cases
      - Analyze and describe briefly each use-case
    - Model the use-cases and actors

_Workshop

  - Remarks
    - Don’t try to search for all the use-cases
      - Other use-cases can appear in the development process

Relationships between use-cases

- Two types of relationship between use-cases
  - Extension
  - Inclusion

- "extension" relationship
  - Used to specify the **optional interactions**
    - These are **exceptional cases**
  - The case where a use-case is similar to another but it includes **additional actions**
  - The extending use-case must list all the actions in the main use-case and also the supplementary actions

Relationships between use-cases

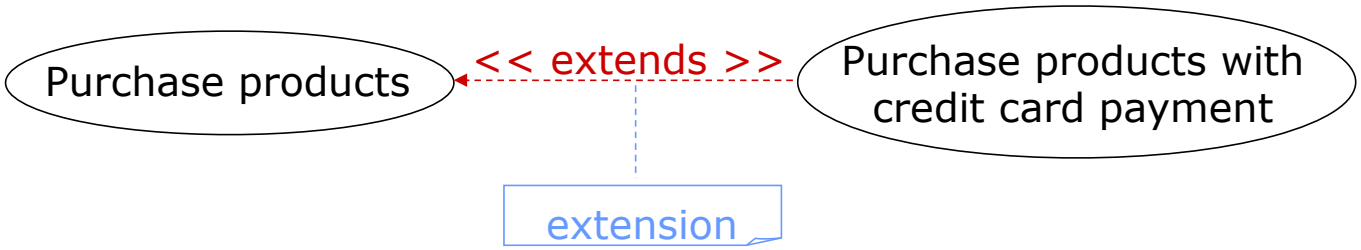
- "extension" relationship
  - Example: "purchase product with payment by credit card" use-case
    - **Use-case**: purchase products
    - **Actors**: Customer, Cashier
    - **Objective**: describe a purchase of products by the customer with payment by credit card
    - **Description**: The customer comes to checkout with selected products.

The cashier encodes products, announces the total amount. The customer gives his credit card. The cashier inserts the credit card into the system. The customer types the PIN code. The system verifies the card and then deducts the total of the card.

- This use-case is a variation of the "purchase products" use-case but adds actions relating to the use of credit card.

Relationships between use-cases

- "extension" relationship
  - "Purchase products with credit card payment" use-case is an extension of the "Purchase products" use-case
  - Notation



  - **Remarks**: If a use-case is associated with an actor, all extensions are also associated with this actor. This is expressed implicitly in the use-case diagrams.

Relationships between use-cases

- “inclusion” relationship
  - describes a series of joint actions in several cases of different usages
  - if several use-cases share **the same sequence of actions** and this common part is intended to meet a clearly defined goal then the part is described in a separate use-case
  - helps to avoid repeating the same details in different use-cases

Relationships between use-cases

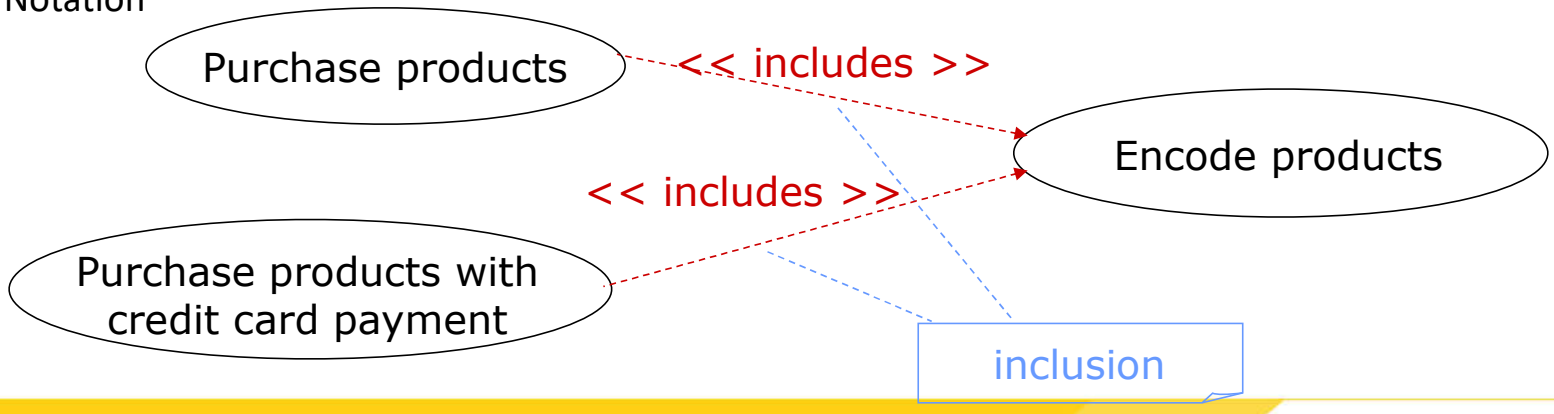
- Example of “inclusion” relationship
  - Suppose we have two use-cases “purchase product” and “purchase products with credit card payment”
    - Both use-cases have the same sequence of actions of encoding products that can be described by the “encode products” use-case

Actions of actor	Actions of system
• The customer comes to the cash desk with the products to buy	
• The cashier encodes the identifier of each product If a product has more than one item, the cashier inputs the number of items	• The cash desk displays the description and price of the product This number is displayed
• After having encoded all of the products, the cashier signals the end of the purchase	• The cash desk calculates and displays the total amount that the customer has to pay
• The cashier announces the total amount to the customer	

actions of encoding products

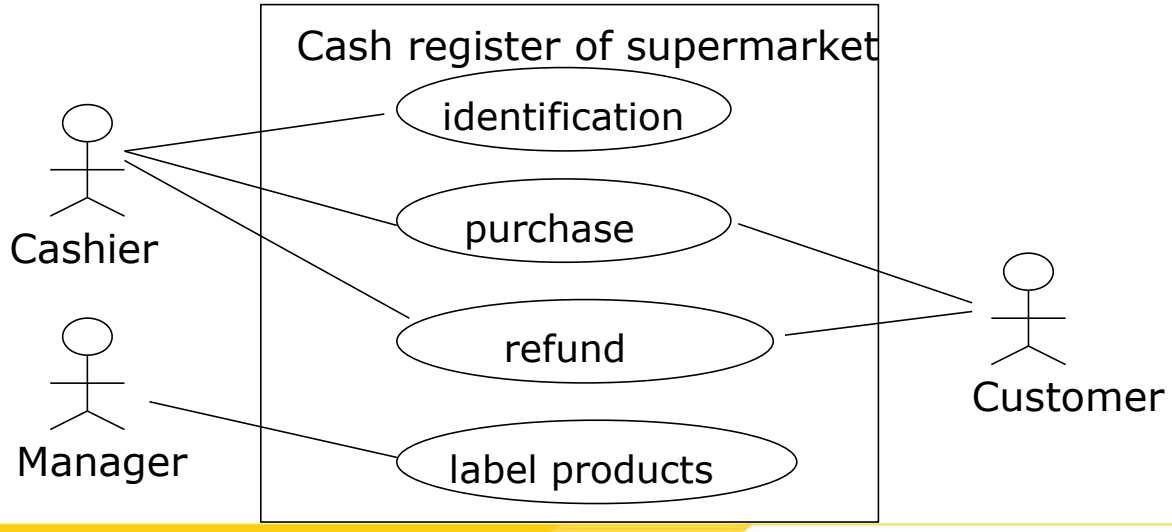
Relationships between use-cases

- "inclusion" relationship
  - Example (continue)
    - "encode products" use-case
      - **Use-case**: encode products
      - **Actor**: Customer, Cashier
      - **Objective**: describe the encoding of the products bought by a customer
      - **Description**: The customer comes to checkout with the selected products. The cashier encodes products, announces the total amount to the customer.
    - Notation



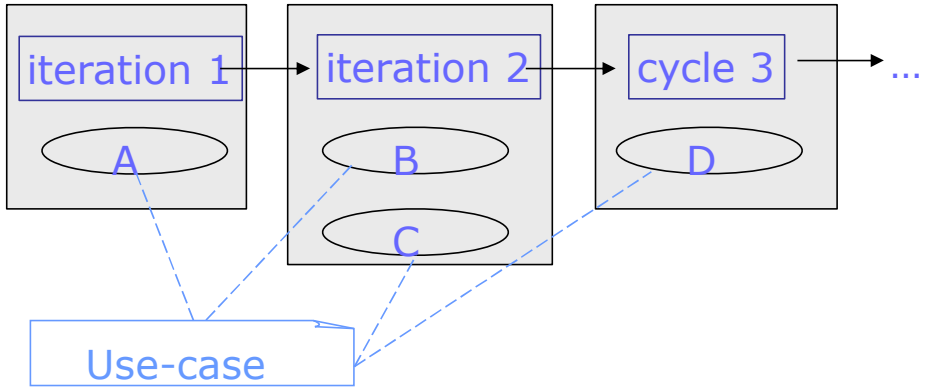
Building use-case diagrams

- A use-case diagram describes the relationships between the use-cases and actors of the system
- The steps to build a use-case diagram
  - Define the limits of the system
  - Identify the actors
  - Identify the use-cases (UCs)
  - Define the relationships between Actors-UCs; UC-UC;
  - Verify the diagrams



Classification of use-cases

- Assign the use-cases to iterations of development process



- How to assign use-cases to iterations of development process?
  - Use-cases should be implemented in the order of importance. For example:
    - Use-cases that may contain risks
    - Use-cases that build the architecture of the software
    - Use-cases that realise a large part of the system functionality
    - Use-cases that require new technology or significant research
    - Use-cases that have great interests by the customer

Example

- An automated teller machine (**ATM**) is a banking subsystem that provides bank customers with access to financial transactions in a public space without the need for a cashier, clerk, or bank teller.
  - *Customer* uses bank/VISA ATM to *Check Balances* of his/her bank accounts, *Deposit Funds*, *Withdraw Cash* and/or *put money*
- On most bank ATMs, the customer is authenticated by inserting a plastic ATM card and entering a personal identification number (PIN).
- Operator need to regularly put money into the machine, get checks, get cards
- Identify actors, use cases, and use case diagrams

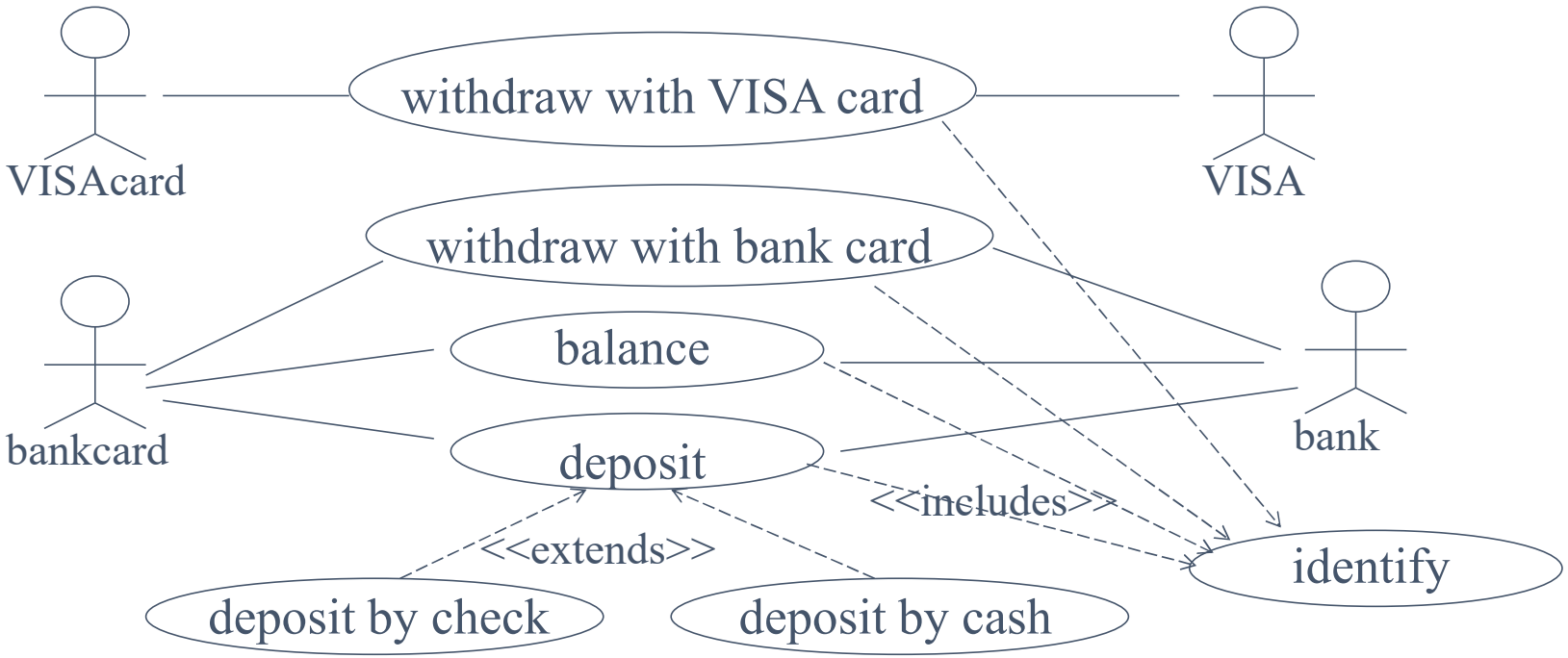
Example

- Actors
  - Bankcard
  - VISAcard
  - Operator
  - VISA system
  - Bank IS

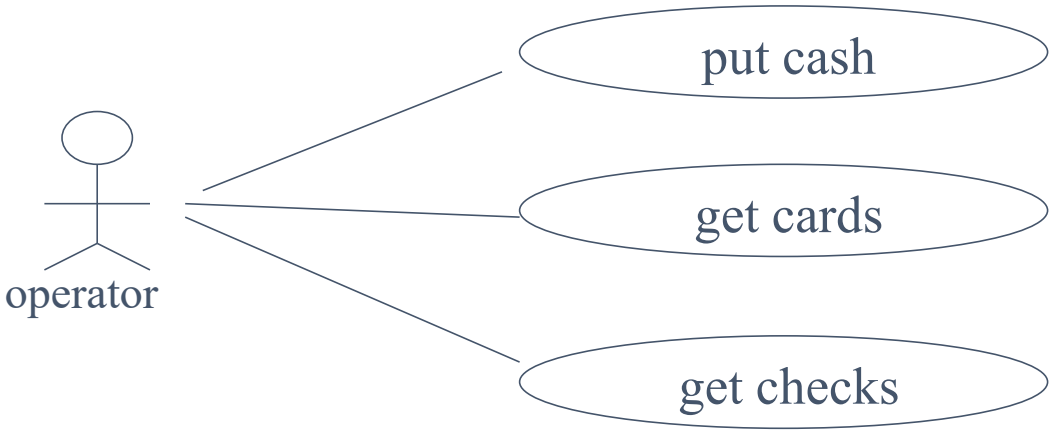
Example

- Use-cases
  - withdraw by bankcard
  - withdraw by VISAcard
  - PIN Identify
  - Balance
  - Deposit (by check or by cash)
  - put cash (into machine)
  - get cards
  - get checks

Example


VISAcard
withdraw with VISA card
VISA
withdraw with bank card
bankcard
balance
bank
deposit
<<includes>>
<<extends>>
identify
deposit by check
deposit by cash

Example



Project

- Divide groups of 4-5 students
- Each group chooses a problem
- Guide students through the steps of project
- Describe the problem requirement

Project (2)

- Divide groups of 4-5 students
- Each group chooses a problem
- Building use case diagrams: Choose one of the following tools:
  - Microsoft Visio
  - StarUML: http://staruml.io/
  - Argo UML: https://argouml.jaleco.com/
  - Lucidchart: https://www.lucidchart.com/pages/examples/uml_diagram_tool
  - Or Others...

Chapter 4.
Requirement Modelling