Ch. 4: Part-B
Classification and Basic Behavioral Modeling in UML

H.S. Sarjoughian

CSE 460: Software Analysis and Design

School of Computing, Informatics and Decision Systems Engineering
Fulton Schools of Engineering

Arizona State University, Tempe, AZ, USA

Use-Cases

Use-case approach is found on the premise that analysis (identifying objects and classes) is facilitated by **understanding a system's use from the perspective of its (intended) users** such as analysts, designers, and testers

- A use-case captures functions that are visible to the user
- Each use-case represents a specific goal of the user
- Use-cases can be used for the whole system, part of the system (sub-system), or primitive components

Use-cases can be used to model **dynamic behavior** view of a system. It can be used to

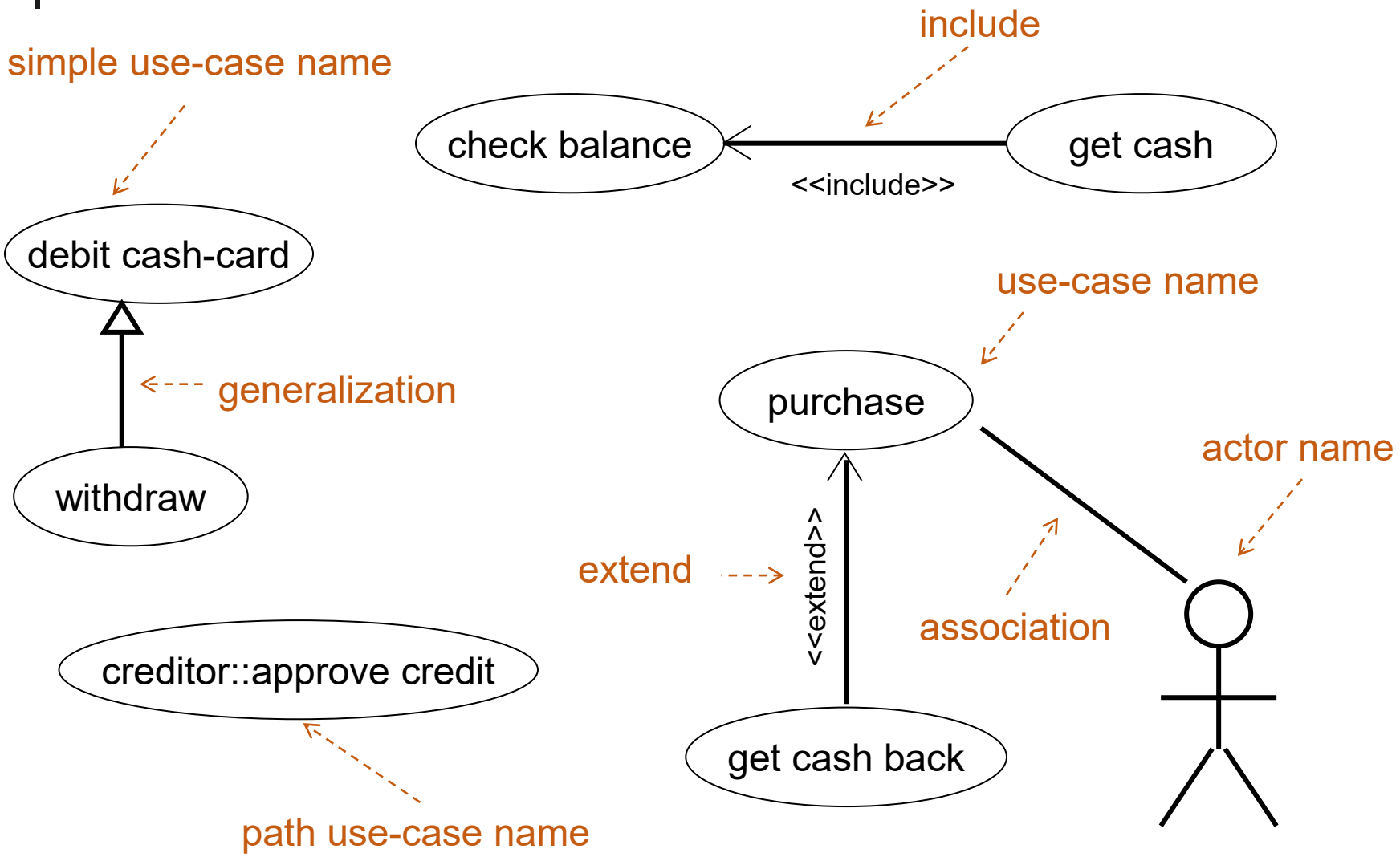
- model context of a system – specifying actors and the meaning of their roles given the external boundary of the system
- model requirements of a system – specifying what the system is (or expected) independent of how it might achieve it (describes what a system does, but not how!)

Use-Cases in UML

- **Semantics:** a use-case is a description of a set of sequences of actions a system performs to produce ***observable behavior*** to an actor.

  - use-case: a short verb phrase naming some behavior
  - actor: a role name a user plays when interacting with the system
  - association: signifies communication between actor & use-case
  - generalization: child use-case inherits behavior and meaning of the parent use-case
  - include: one use-case (referred to as base use-case) explicitly incorporates (includes) the behavior of another use-case (referred to as supplier use-case). The base use-case pulls in behavior from the supplier use-case
  - extend: one use-case (referred to as base use-case) implicitly incorporates the behavior of another use-case. The base use-case may standalone, but under certain conditions it may be extended.

UML Use-Cases Notation (Syntax)



Use-Cases and Scenarios

A use-case diagram represents a collection of scenarios describing the usages of a system by its actors

- Need to identify actors, a class of external entities (people or systems) that play user roles
- Scenarios provide a description of how the system is to be used. Each scenario is described from the point of view of an actor (a person or device) that interacts with the system

Each scenario can be used to answer queries such as

- What are the main tasks or functions performed by an actor?
- What system information will an actor acquire, produce, or change?
- Will an actor have to inform the system about changes in the external environments?
- Does an actor wish to be informed about unexpected changes?

Scenario: a specific set of actions that illustrates some behavior

Cash-Card Use-Case Example

Cash-Cards are like currency notes in value and can be used for the payment of purchases. The bank can issues one or more single Cash-Card to each of its customers *(Card-Owners.)* Cash-Cards need to be charged with some amount of money before they can be used with *Vendor-Machines* for purchasing goods.

For adding credit to the Cash-Card, the Card-Owner needs to use a *Bank-Machine* to add money to it. Typical activities that go between the Bank-Machine and the Card-Owner are as follows:

- Machine requests the Card-Owner to enter its PIN (Personal Identification Number)
- Having obtained and verified the Card-Owner PIN, it next requests the amount to be added to the Cash-Card
- Having obtained the amount, Bank-Machine accordingly credits the Cash-Card, shows balance and issues the transaction record to the customer

Having added credit to the Cash-Card, the Card-Owner can now use it for the payment of goods. The activities that can occur between the Card-Owner and the Vendor-Machine are as follows:

- The Vendor-Machine shows the total purchase amount and waits for the Card-Owner approval
- Upon the Card-Owner approval, Vendor-Machine debits the Cash-Card for that amount.

Purchase Scenario

1. Card-owner (customer) gives to the Grocery-Store-Register the items he wants to purchase

2. Grocery-Store-Register adds the items

3. The Card-Owner is shown the total amount for purchasing the items

4. Card-Owner pays for the items with his Cash-card
   a. Card-Owner swipes his cash-card
   b. Card-Owner enters his cash-card pin number
   c. Cash-card balance is verified that it has sufficient money for the amount required to purchase the items
   d. The amount of purchase is deducted from the Cash-card balance

Purchase Scenario (cont.)

5. Grocery-Store-Register give the items to the Card-Owner

Extension after step 4:

5. Card-Owner asks for cash back
6. Card-Owner show his picture ID to Grocery-Store-Register
7. Grocery-Store-Register verifies the picture ID
8. Grocery-Store-Register give cash to the Card-Owner

Cash-Card Use-Case Example (cont.)

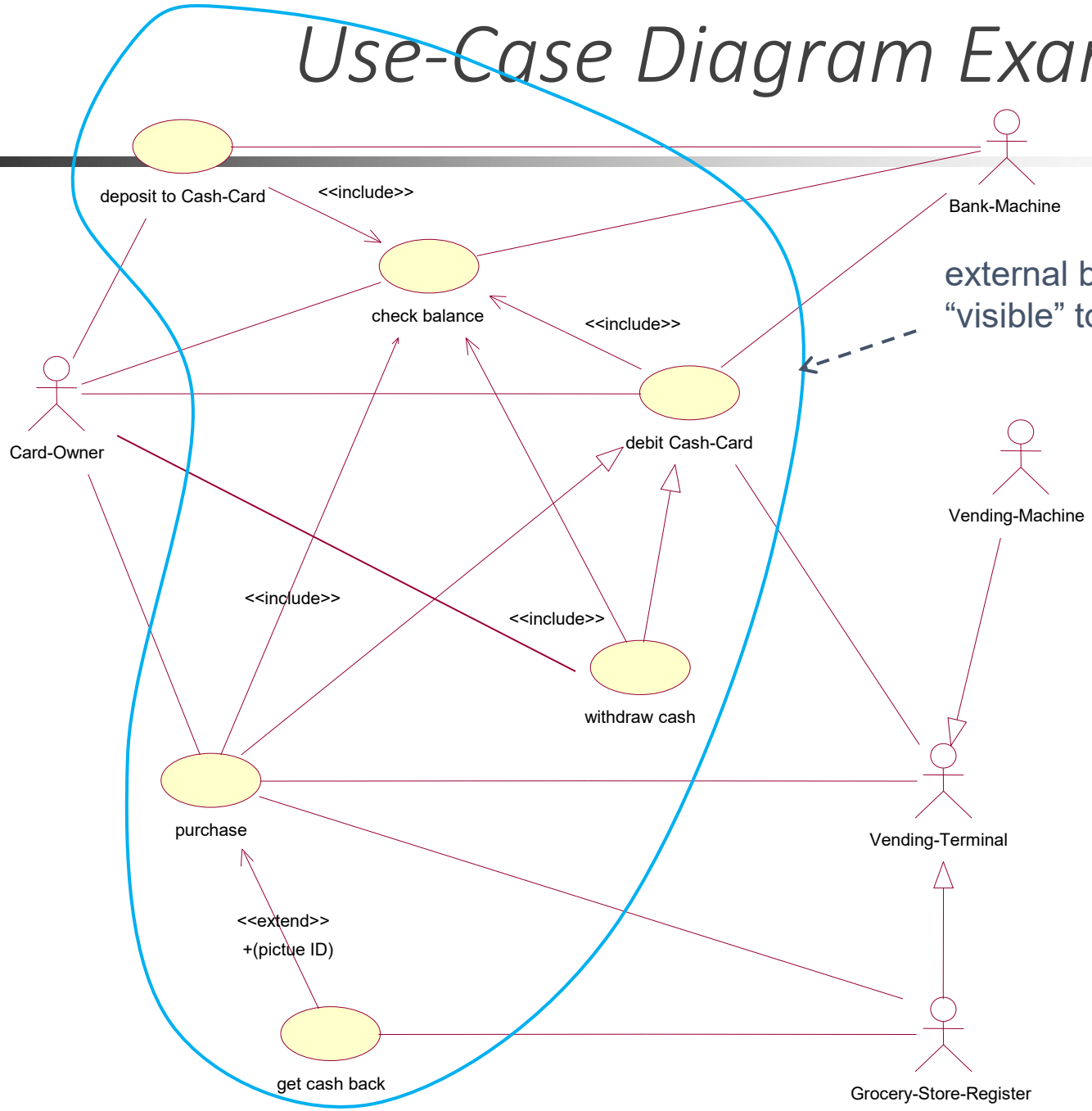
Actors:

- Card-Owner: a person who has been issued an ATM card
- Bank-Machine: an Automatic Machine Teller that can be used to do banking transactions
- Vending-Machine: a machine that has food items for purchase
- Vending-Terminal: a computerized terminal for people to purchase for items sold by a vending machine
- Grocery-Store-Register: a machine at a store selling groceries

Use-Cases:

- Deposit to Cash-Card: customer adds money to her Cash-Card
- Debit Cash-Card: withdraw money from her Cash-Card
- Check balance: check the balance on the Cash-Card
- Purchase: purchase items using Cash-Card from a Vending-Terminal
- Get cash back: get cash back from the Grocery-Store-Register after purchase
- Withdraw cash: withdraw cash from Cash-Car

Use-Case Diagram Example



Home Exercise

An authorized ASU affiliate (borrower) can search and borrow items from the library. The library owns a repository of books, journals, maps, and newspapers. The library also owns electronic subscription to journals, conference proceedings, magazines, and newspapers. The library has a number of self-checkout Library Automatic Teller Machines (LATM).

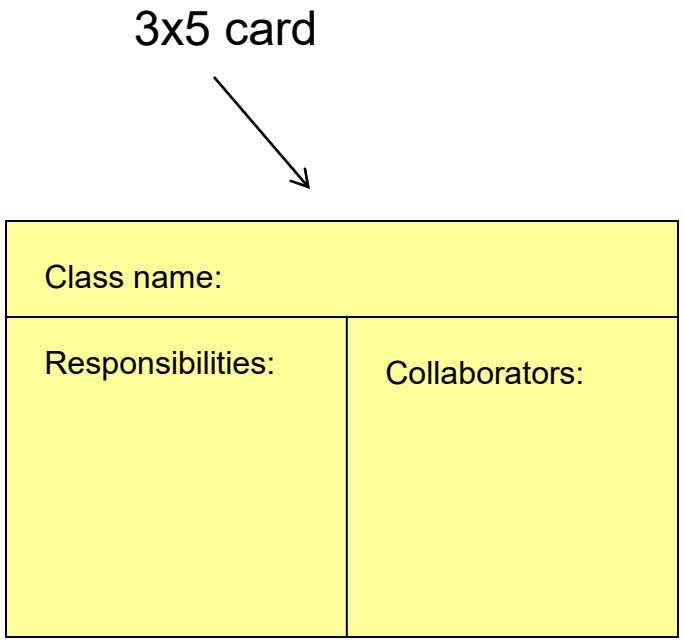
A borrower follows the following steps to check out hard copy items:

- Scan ID card for authorization
- Accept the terms of agreement for borrowing
- Scan the items to be checked out one at a time
- View whether an item can be checked out or not
- Self-register items to be checked out followed by demagnetization
- Receive a print out of due dates for the borrowed items

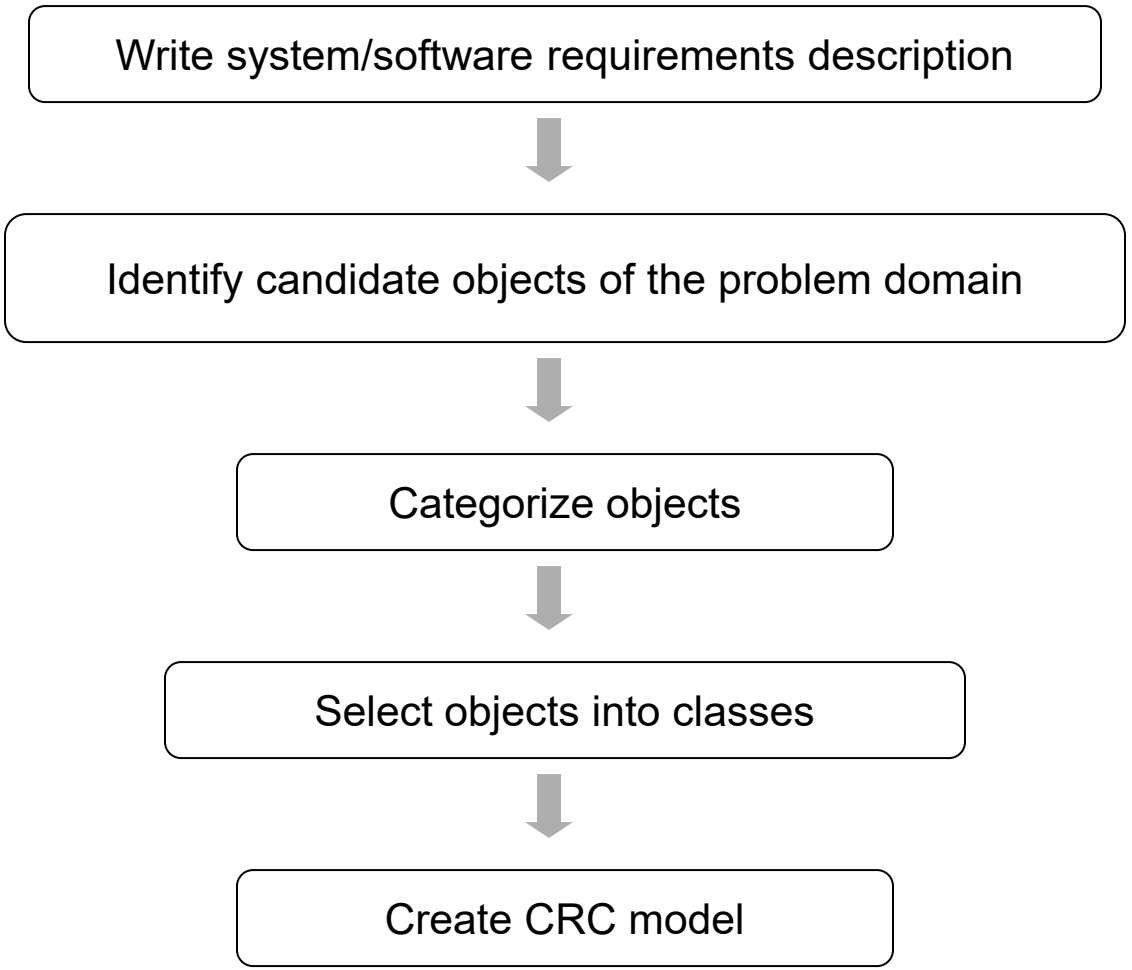
Each librarian may provide two services: (1) help borrowers locate items either in hardcopy or softcopy form and (2) check out hardcopy items to borrowers. Librarians use an on-line system to check a borrower's eligibility, records (address, affiliation (department or unit), and status (staff, students, and faculty) for loaning hardcopy items. The librarian must follow a step-by-step procedure as above.

Class/Responsibilities/Collaborators

- CRC cards are useful in analyzing scenarios – a scenario is an outline of events that elicits some system behavior
- Each CRC card captures name, respons-ibilities, and collaborators of a class
- CRC cards can be arranged to represent patterns of collaborations among a set of prototypical instances of classes
- CRC cards can be arranged to show generalization/specialization and aggregation hierarchies among classes



Procedure for OO Analysis



Iterate until all requirements have been accounted for!

Categories of Objects

- **External Entities:** Entities that are outside the scope of the software engineering activities
- **Things:** natural or physical objects which are tangible (e.g., can be manipulated for example in space and time) and within the scope of software engineering activities
- **Occurrences/Events:** activities that occur from time to time
- **Roles:** Responsibilities taken up by people or things
- **Organizational Units:** groups to which users belong
- **Places:** areas set aside for people or things
- **Computational Entities:** artificial entities with well-defined and predictable behavior

Selection Criteria

- Does this object encapsulate state information essential to meet the requirements?
- Does this object offer services needed to meet the requirements?
- Is this object simple enough to be considered as an attribute of a class?
- Do these objects have common attributes and operations to form a class?

Categorization and Selection Table

Candidate ,objects, classes	General Categorization (External Entities, Things, Occurrences/Events, Roles Organizational Units, Places, Computational	Does this object encapsulate state information essential to meet the requirements?	Does this object offer services needed to meet the requirements?	Is this object simple enough to be considered as an attribute of a class?	Do these objects have common attributes and operations to form a class?

A Description of FactoryWatch System

FactoryWatch software enables a plant supervisor to install a system to monitor the activities of people and machines in a non-intrusive manner. It requires placement of video cameras in various locations of the plant in which activities are to be monitored. The software is installed on a computer near a camera and performs interpretation of the image output streams. It uses the quantization principle to detect major changes in the images being generated and only sends information to a central database when such changes occur. The system operates in both real-time and non-real-time modes. In real-time mode, the system can alert the plant supervisor to an abnormal situation in production who can send out a technician to correct the problem. In non-real-time mode, the supervisor can examine the record of images from one or more cameras in the database to try to understand how a problem has arisen and how it might be prevented in the future.

Categorization and Selection Table Example

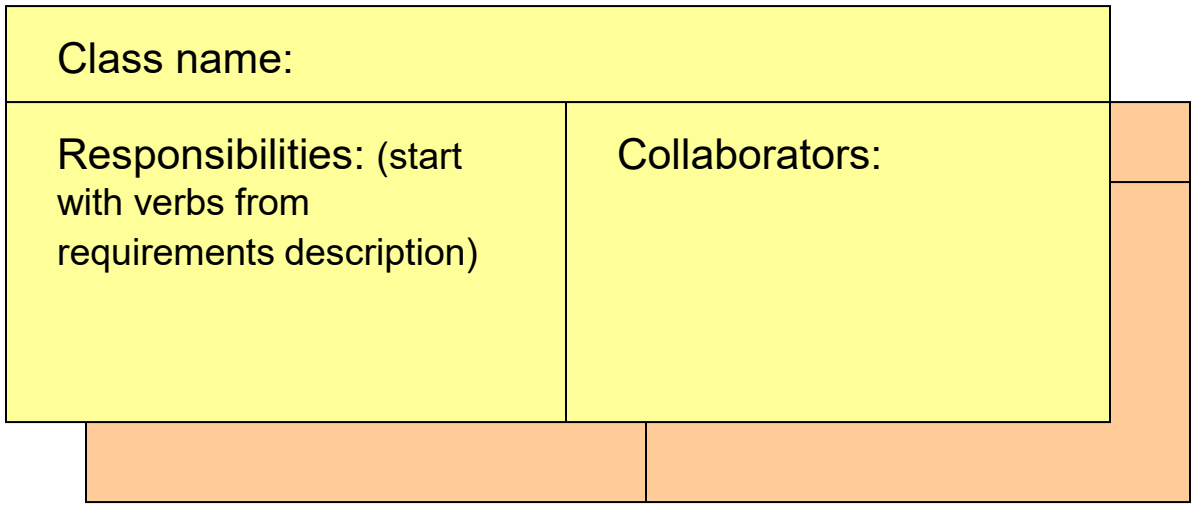
Candidate , Objects, classes	General Categorization (External Entities, Things, Occurrences/Events, Roles, Organizational Units, Places, Computational Entities)	Does this object encapsulate state information essential to meet the requirements?	Does this object offer services needed to meet the requirements?	Is this object simple enough to be considered as an attribute of a class?	Do these objects have common attributes and operations to form a class?
Camera	External entity	yes	yes	no	yes
Camera size	N/A	no	yes	yes	no
Work cell	External entity	yes	yes	no	yes
Image	Thing	yes	yes	no	no
Supervisor	Role	yes	yes	no	yes
Quantizer	Computational entity	yes	yes	no	yes
Message	Events	yes/no	yes	yes	yes/no
Display panels	Computational entity	yes	yes	no	yes
Camera location	Places	yes	yes	yes	no
etc.					

CRC Model

Create Class-Responsibilities-Collaborators (CRC) model

Identify responsibilities
(operations to be carried out by the object requested by others or itself)

Identify collaborators
(other objects it interacts with directly)



Camera Class

Camera	
Responsibilities: ▪ Take images from factory floor at some time interval with some specified resolution ▪ Send images to designated targets	Collaborators: ▪ Camera location ▪ Message

Summary

- Identification of objects/classes is fundamental to the object-oriented analysis and design activities
- Classical categorization, conceptual clustering, and prototype theory are foundational classification (clustering) techniques
- Classification is an incremental and iterative process
- Classification can be carried out from object-oriented and/or classical structured worldviews in methodical settings
- Classical structured approaches such as FAST and QDF are complementary to object-oriented methods and techniques
- Use-cases provide a user/client intensive understanding of the external view of the software system and problem domain
- CRC provides an approach for discovery/invention of classes and object and their relationships

References

- *Object-Oriented Analysis and Design with Applications, 3rd Edition, G. Booch, et al. 2007, Addison Wesley, 2007*
- *OMG Unified Modeling Language Specification, UML Standard, http://www.omg.org/technology/documents/formal/uml.htm, Computer Associates International, Inc., 2001*
- *The Unified Modeling Language User Guide,G. Booch, J. Rumbaugh, I. Jacobson, Addison Wesley Object Technology Series, 1999*
- *Software Engineering: A Practioner's Approach, 8th Edition, R.S. Pressman, McGraw Hill, 2000*

UML: Behind the Scenes

Meta-models for behavioral specification languages

Use-Case Abstract Syntax

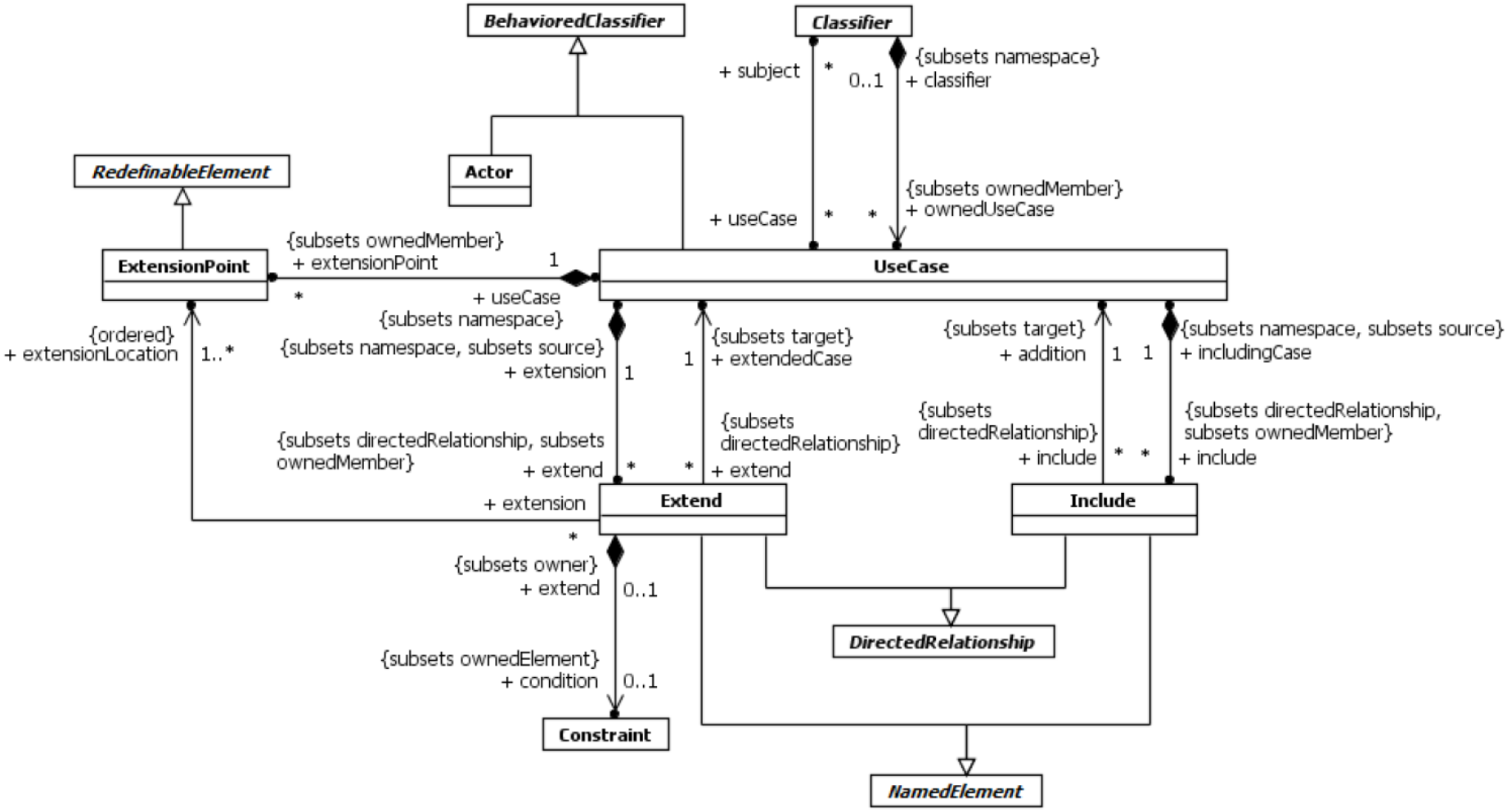


Figure 18.1 UseCases