UNIFIED MODELING LANGUAGE (UML)

02-1. UML & USE CASE DIAGRAM

1

Content

2

Discussion

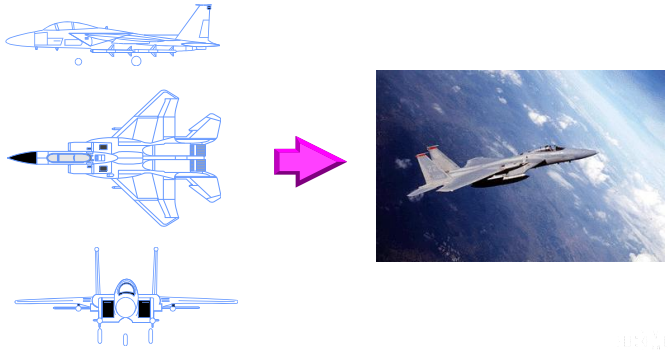
- You have a complicated object in the real world, e.g. an airplane

- How can you make it?
- How can you know its structure / design?
- …

3

1.1. What Is a Model?

- A model is a simplification of reality.



4

5

Why Model?

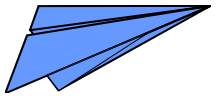
- Modeling achieves four aims:
  - Helps you to **visualize** the system we want to build.
  - Allows you to specify the **structure** or **behavior** of a system.
  - Gives you a **template** that instructs you in constructing a system.
  - Document the **decisions** you have made.
- You build models of complex systems because you cannot comprehend such a system in its entirety.
- You build models to better understand the system you are developing.

5

6

Discussion

- How do you build a paper airplane?
- If it cannot fly, what will you do?



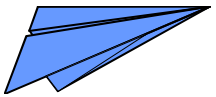
- What about a fighter jet?

6

7

The Importance of Modeling

Less Important ⟷ More Important



Paper Airplane

Fighter Jet

7

8

Software Teams Often Do Not Model

- Many software teams build applications with the approach like they were building paper airplanes
  - Start coding immediately after receiving project requirements
  - Work fast at beginning but very slow after that
  - Do not have architecture
  - Doomed to failure
- Modeling is the key to successful projects

8

1.2. Why UML?

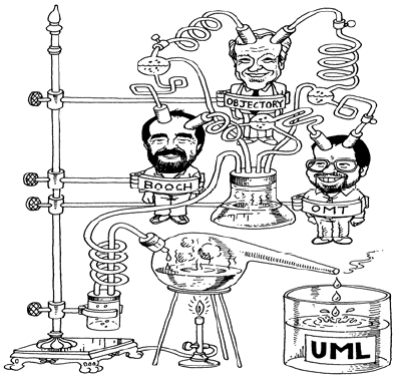
- 1980s: classical structural analysis and design
- 1990s: object-oriented analysis and design
- Mid-1990s: > 50 object-oriented methods with many design formats
  - Fusion, Shlaer-Mellor, ROOM, Class-Relation,Wirfs-Brock, Coad-Yourdon, MOSES, Syntropy, BOOM, OOSD, OSA, BON, Catalysis, COMMA, HOOD, Ooram, DOORS…

➔ A **unified** modeling language is **indispensable**

9

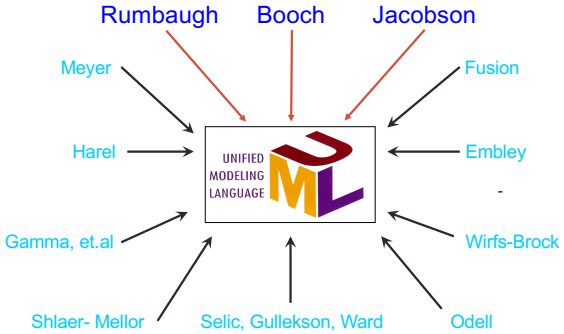
UML is a standardized modeling language

- An Object Management Group (OMG) standard.
- By 3 experts in Rational Software
  - Booch91 (Grady Booch): Conception, Architecture
  - OOSE (Ivar Jacobson): Use cases
  - OMT (Jim Rumbaugh): Analysis



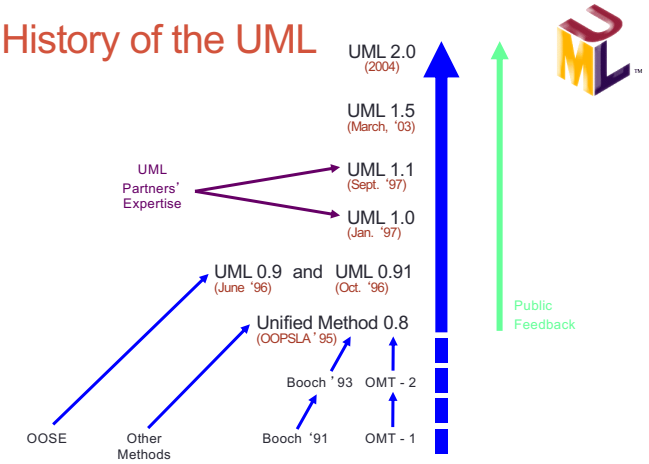
10

Inputs to the UML



11

History of the UML



12

1.3. What Is the UML?

- The UML is a language for
  - Visualizing
  - Specifying
  - Constructing
  - Documenting

  the artifacts of a software-intensive system.

13

The UML Is a Language for Visualizing

- Communicating conceptual models to others is prone to error unless everyone involved speaks the same language.
- There are things about a software system you can't understand unless you build models.
- An explicit model facilitates communication.

14

The UML Is a Language for Specifying

- The UML builds models that are precise, unambiguous, and complete.

15

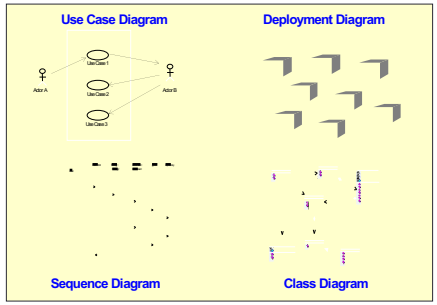
The UML Is a Language for Constructing

- UML models can be directly connected to a variety of programming languages.
  - Maps to Java, C++, Visual Basic, and so on
  - Tables in a RDBMS or persistent store in an OODBMS
  - Permits forward engineering
  - Permits reverse engineering

16

The UML Is a Language for Documenting

- The UML addresses documentation of system architecture, requirements, tests, project planning, and release management.



17

Content

1. UML Overview
2. Requirement modeling with use-case
3. Use case diagrams

18

Purpose of Requirement

- Establish and maintain agreement with the customers and other stakeholders on what the software should do.
- Give software developers a better understanding of the requirements of the software.
- Delimit the software.
- Provide a basis for planning the technical contents of the iterations.
- Provide a basis for estimating cost and time to develop the software.
- Define a user interface of the software.

19

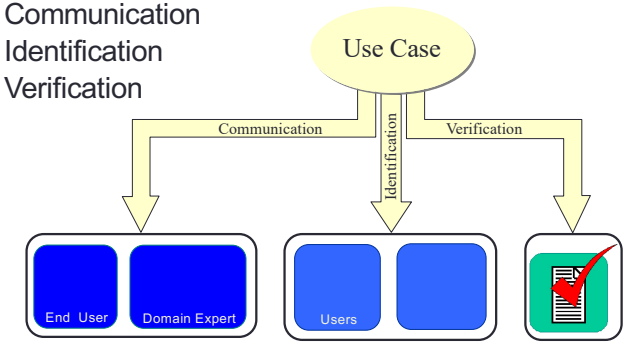
What Is Software Behavior?

- Software behavior is how a software acts and reacts.
  - It comprises the actions and activities of a software.
- Software behavior is captured in use cases.
  - Use cases describe the interactions between the software and (parts of) its environment.

20

Benefits of a Use-Case Model

- Communication
- Identification
- Verification



21

Major Concepts in Use-Case Modeling

- An actor represents anything that interacts with the software.

Actor

- A use case describes a sequence of events, performed by the software, that yields an observable result of value to a particular actor.

22

Content

23

3.1. Actors

- Actors represent roles a user of the software can play
  - They can represent a human, a machine, or another software
  - They can be a peripheral device or even database
- They can actively interchange information with the software
  - They can be a giver of information
  - They can be a passive recipient of information
- Actors are not part of the software
  - Actors are EXTERNAL

Actor

24

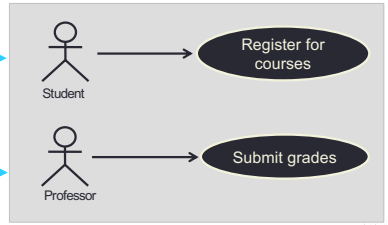
Actors and Roles

Charlie: Is employed as a math professor and is an economics undergraduate.

Jodie: Is a science undergraduate.

Charlie and Jodie both act as a Student.

Charlie also acts as a Professor.



25

Internet banking system

- The internet banking system, allowing interbank network, communicates with bank customers via a web application. To perform transactions, customers have to log in the software. Customers may change password or view personal information.
- Customers can select any of transaction types: transfer (internal and in interbank network), balance inquiries, transaction history inquiries, electric receipt payment (via EVN software), online saving.
- In the transfer transaction, after receiving enough information from the customer, the software asks the bank consortium to process the request. The bank consortium forwards the request to the appropriate bank. The bank then processes and responses to the bank consortium which in turn notifies the result to the software.
- The bank officers may create new account for a customer, reset password, view transaction history of a customer.

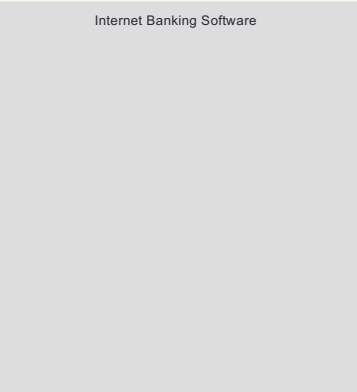
26

Some guideline to extract actors

- Pay attention to a noun in the problem description, and then extract a subject of action as a Actor.
- Ensure that there are no any excesses and deficiencies between the problem description and Actors extracted.
- Actor names
  - should clearly convey the actor's role
  - good actor names describe their responsibilities

27

Exercise: Find actors



28

3.2. Use Cases

- Define a set of use-case instances, where each instance is a sequence of actions a software performs that yields an observable result of value to a particular actor.
  - A use case models a **dialogue** between one or more actors and the software
  - A use case describes the **actions the software takes** to deliver something of value to the actor

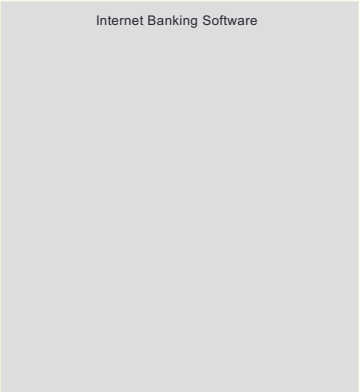
29

Some guidelines to extract use cases

- Pay attention to a verb in the problem description, and then extract a series of Actions as a UC.
- Ensure that there are no any excesses and deficiencies between the problem description and Use cases extracted.
- Check the consistency between Use Cases and related Actors.
- Conduct a survey to learn whether customers, business representatives, analysts, and developers all understand the names and descriptions of the use cases

30

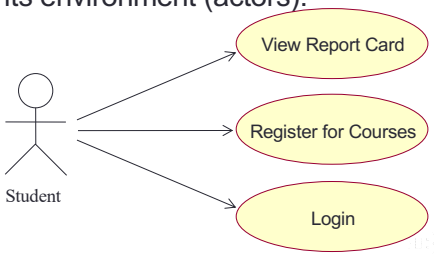
Exercise: Find use cases



31

3.3. Use-Case Diagram

- A diagram modeling the dynamic aspects of softwares that describes a software's functional requirements in terms of use cases.
- A model of the software's intended functions (use cases) and its environment (actors).



32

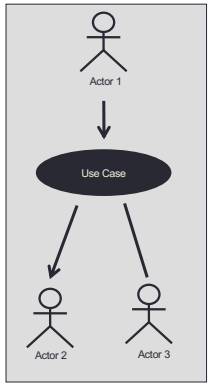
Association between actor and use case

- Establish the actors that interact with related use cases
  - Associations clarify the **communication** between the actor and use case.
  - Association indicate that the actor and the use case instance of the software communicate with one another, each one able to **send and receive messages**.
- The arrow head is <u>optional</u> but it's commonly used to denote the <u>initiator</u>.

Actor

33

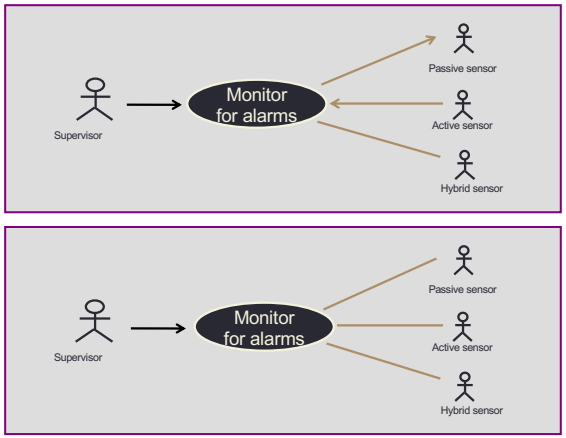
Communicates-Association



- A channel of communication between an actor and a use case.
- A line is used to represent a communicates-association.
  - An arrowhead indicates who initiates each interaction.
  - No arrowhead indicates either end **can** initiate each interaction.

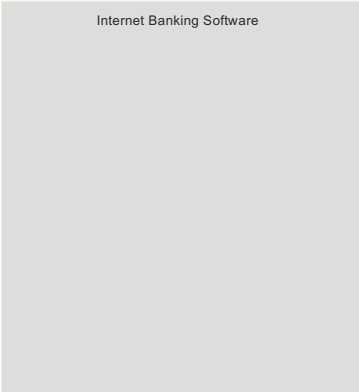
34

Arrowhead Conventions



35

Exercise: Draw use case diagram



36

Question?

37