System Analysis and Design

(IT3120E)

Quang Nhat Nguyen

quang.nguyennhat@hust.edu.vn

Hanoi University of Science and Technology

School of Information and Communication Technology

Academic year 2022-2023

Content:

- Introduction of object-oriented system analysis and design
- **Introduction of the modeling language UML**
- Introduction of software development process
- Analysis of the environment and needs
- Function analysis
- Structure analysis
- Interaction analysis
- Behavior analysis
- Design of the system's overall architecture
- Class detail design
- User interface design
- Data design

Introduction of modeling language UML

- Development history of the modeling language UML
- UML views
- Diagrams used in UML
- Free UML modeling tools

Development history of UML (1)

- UML language (i.e., Unified Modeling Language) is an object-oriented modeling notation system
- 1975-1990:
  - There are many object-oriented MHH languages developed
- 1990-1994:
  - More than 50 methods of object-oriented development, including 3 following famous ones:
    - OOD - Object Oriented Design (Grady Booch)
    - OOSE - Object Oriented Software Engineering (Ivar Jacobson)
    - OMT - Object Modeling Technique (Jim Rumbaugh)

Development history of UML (2)

- Oct. 1994: Rumbaugh and Booch conducted a UML project in Rational, building a unified methodology based on two methods Booch 93 and OMT-2
- 1995: Jacobson joined the project
- Oct. 1995: UML draft version (version 0)
- Jun. 1996: UML version 0.9
- Jan. 1997: IBM and SoftTeam combined with members => Version 1.1
- Nov. 14, 1997: UML 1.1 is recognized as a standard by OMG (Object Management Group)
- Jun. 1998: UML 1.2
- Oct. 1998: UML 1.3
- May 2001: UML 1.4
- Jun. 2003: UML 2.0

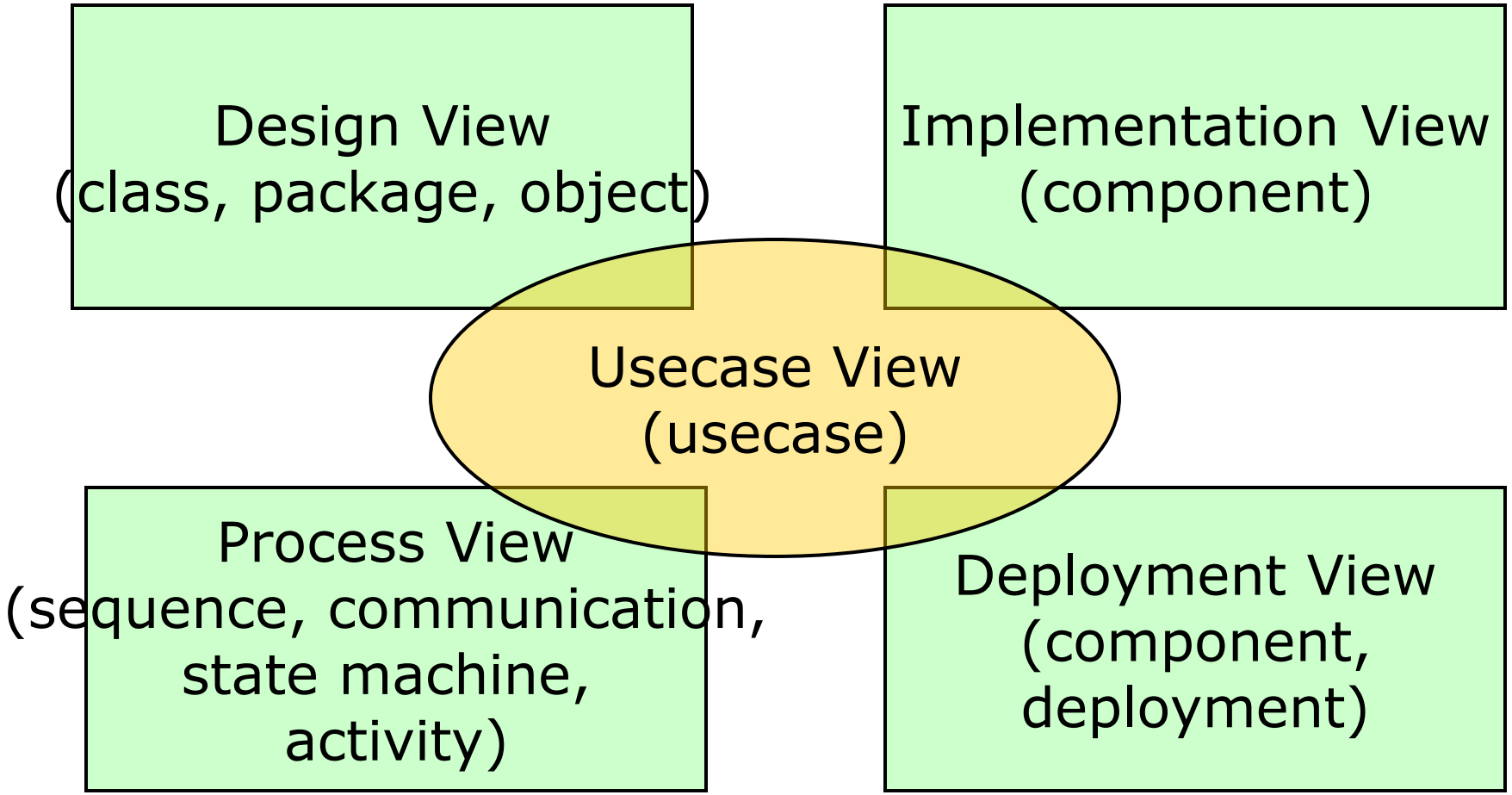
UML – A modeling language

- UML is a language for:
  - Visualization,
  - Specification,
  - Construction,
  - Documentation
- Can be used in any system development process
- Throughout the system development lifecycle
- Used by different implementation technologies

UML views (1)

- UML provides different models to describe a system
- Each model can only describe the system from a certain point of view
- UML provides 5 different views of a system
- Each view is made by a number of diagrams (models)
- A diagram (model) may belong to (i.e., is used by) several (>1) views

UML views (2)


Design View
(class, package, object)
Implementation View
(component)
Usecase View
(usecase)
Process View
(sequence, communication,
state machine,
activity)
Deployment View
(component,
deployment)

UML views (3)

- **Usecase View**
  - Is the view from the outside looking at the system
  - As the point of view of *end users*, *analysts*, *testers*
  - Does not reflect the internal organization, but only clarifies the main/important functions that the system must satisfy for users
  - Static models: Usecase diagram
  - Dynamic models: Communication diagram, State machine diagram, and Activity diagram

UML views (4)

- **Design View**
  - Also known as the Logical View
  - It is a view inside the system (structure), showing the tasks of the system
  - Is the view of the *system designer*
  - Static models: Class diagram, Object diagram
  - Dynamic models: Communication diagram, State machine diagram, Activity diagram

UML views (5)

- **Process View**
  - Also known as the Parallel View
  - Reflects the control processes, the execution processes, showing the synchronous operation of the system
  - Shown (used) with the diagrams like in the Design View, focusing on active classes
    - Active class: A class that represents for control/execution processes

UML views (6)

- **Implementation View**
  - Also known as the Component View
  - Is the view on the release form of the software
  - Shows relatively independent components and files that can be assembled to make the system run
  - Static models: Component diagram
  - Dynamic models: Communication diagram, State machine diagram, Activity diagram

UML views (7)

- **Deployment View**
  - A view of the hardware architecture and infrastructure on which the system is deployed
  - Specify the distribution and arrangement of the system's components on hardware units and infrastructure platforms
  - Static models: Deployment diagram
  - Dynamic models: Communication diagram, State machine diagram, Activity diagram

UML views (8)

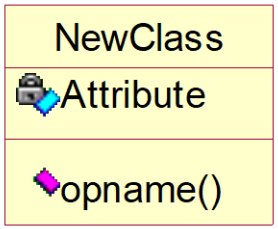
- Each role in the system development process (e.g., analyst, designer, integrator, tester, end user, etc.) is usually interested only in a certain view of the system
- 5 views relate and complement each other
- The Usecase View influence (i.e., relate to) the 4 remaining views

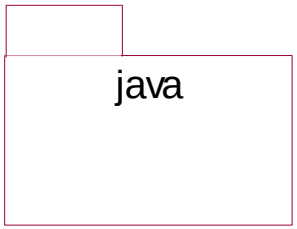
Diagrams in UML 2.0

- **Structure diagrams**:
  - ⇨ *Class diagram*
  - ⇨ *Object diagram*
  - ⇨ *Deployment diagram*
  - ⇨ *Package diagram*
  - ⇨ *Component diagram*
  - ⇨ Composite structure diagram
- **Behavior diagrams**:
  - ⇨ *Usecase diagram*
  - ⇨ *Activity diagram*
  - ⇨ *Sequence diagram*
  - ⇨ *Communication diagram*
  - ⇨ *State machine diagram*
  - ⇨ Timing diagram
  - ⇨ Interaction overview diagram

Elements of a diagram

- Nodes
  - Are elements of the model (diagram)
  - Represented by a 2D graphic form
  - Examples: Class, Package





- Edges (links/paths)
  - Are elements of the model (diagram)
  - Represented by a linear graphic form
  - Examples: Association, Dependency, Generalization

Add meaning to a diagram

- Specification
  - A textual statement of syntax and semantics
- Adornment
  - Role, Multiplicity, Qualifier, etc.
- Stereotype
  - String of characters, enclosed in quotes (i.e., <<…>>)
- Property and Tagged value
  - Give more information to an element of the diagram
  - Example: {label=value}, {label_boolean}
- Constraint
  - Add conditions/constraints to an element of the diagram

Modeling with UML

- Modeling the system from multiple views
  - Can use 5 views to the system
  - Depending on the system is small/large, simple/complex => Decide to describe (i.e., to model) the system by the suitable views
- Modeling the system by different abstraction levels
  - Depends on the phases (of the system development process) and user needs
  - Can be at logical/overview level or detailed level

Free UML modeling tools (1)

- **ArgoUML** (https://argouml-tigris-org.github.io/tigris/argouml/)
  - Diagram-to-source-code generation for: C++, C#, Java, PHP4, PHP5, Ruby
  - Source-code-to-diagram generation (i.e., reverse engineering) for: Java
- **BOUML** (http://www.bouml.fr/)
  - Diagram-to-source-code generation for: C++, Java, PHP, IDL, Python, MySQL
  - Source-code-to-diagram generation for: C++, Java, PHP, MySQL
- **NClass** (http://nclass.sourceforge.net/)
  - Diagram-to-source-code generation for: C#, Java
  - Source-code-to-diagram generation for: C#, Java
- **Umbrello UML Modeller** (https://umbrello.kde.org/)
  - Diagram-to-source-code generation for: C++, Java, Perl, PHP, Python
  - Source-code-to-diagram generation for: C++, IDL, Pascal/Delphi, Ada, Python, Java

Free UML modeling tools (2)

- **WhiteStarUML** (https://sourceforge.net/projects/whitestaruml/)
  - Diagram-to-source-code generation for: Java, C#, C++, SQL
  - Source-code-to-diagram generation for: Java, C#, C++, SQL
- **Open ModelSphere** (http://www.modelsphere.com/org/)
  - Diagram-to-source-code generation for: Java, SQL
  - Source-code-to-diagram generation for: Java
- **Modelio** (https://www.modelio.org/)
  - Diagram-to-source-code generation for: Java
  - Source-code-to-diagram generation for: Java
- **Dia** (https://wiki.gnome.org/Apps/Dia/)
  - Diagram-to-source-code generation for: Python, C++, JavaScript, Pascal, Java, PHP
- **Papyrus** (http://www.eclipse.org/papyrus/)
  - Diagram-to-source-code generation for: C/C++, Java

Free UML modeling tools (3)

- **UML Designer** (http://www.umldesigner.org/)
- **UMLet** (http://www.umlet.com/)
- **Violet UML Editor** (http://alexdp.free.fr/violetumleditor/page.php)
- **PlantUML** (http://plantuml.com/)
- **Astah** (https://astah.net/products/free-student-license/)
- **Visual Paradigm** (https://www.visual-paradigm.com/solution/freeumltool/)
- **MetaUML** (https://github.com/ogheorghies/MetaUML/wiki)
- **TinyUML** (https://sourceforge.net/projects/tinyuml/)

Free GUI design tools

- **Pencil** (https://pencil.evolus.vn/)
- **Wireframe** (https://wireframe.cc/)
- **InVision** (https://www.invisionapp.com/studio)
- **Lunacy** (https://icons8.com/lunacy)
- **Fluid UI** (https://www.fluidui.com/)
- **Sketch UX Kit** (https://www.sketchappsources.com/free-source/3244-wireframing-ux-kit-sketch-freebie-resource.html)