Software Engineering

Lecture 7

System Modeling – Part II

Dr. Obuhuma James

Description

System modeling as a crucial element of system development. The topic continues from where we stopped with part I of system modeling. This topic aims at exploring the UML set of models used in software engineering. The focus will be on use case diagrams, class diagrams, activity diagrams, sequence diagrams, collaboration diagrams, statechart diagrams, and deployment diagrams. In each case emphasis has been put on bringing out the purpose for the model.

Learning Outcomes

By the end of this topic, you will be able to:

- Differentiate the need for main UML set of models as used in software engineering.
- Draw any of the UML's main set of diagrams based on system requirements.
- Comprehend the steps used in UML modeling for software engineering.

Overview of System Modeling

System modeling is the process of developing abstract models of a system. Each model aims at presenting a different view or perspective of the system. System models are developed based on the requirements generated through the requirements engineering process. Thus, the requirements document discussed in topic 5 forms the basis for the development of system models. In most cases, system modeling is anchored over the Unified Modeling Language (UML). UML is an ISO standard that outlines a set of models required to offer graphical representation of systems based on the requirements.

As already mentioned in the previous topic, system modeling helps in communicating requirements in a graphical manner. This helps analysts to understand the functionality of the system as well as facilitate communication with customers. According to Sommerville [1], models of the existing system are used during requirements engineering to clarify what the existing system does. They can be used as a basis for discussing its strengths and weaknesses which in return lead to generation of requirements for the new system. On the other hand, models of the new system are used during requirements engineering to explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation.

The previous topic focused on context diagrams, data flow diagrams, and entity relationship diagrams. This topic continues from that point by focusing on use case diagrams, class diagrams, activity diagrams, sequence diagrams, collaboration diagrams, statechart diagrams, deployment diagrams. These set of models are defined in the UML standard. These diagrams can be classified into two broad categories, namely, structural, and behavioural diagrams.

- Structural diagrams represent the static aspect of the system that entails parts of a diagram that form the main structure of the system. Such parts are represented by classes, interfaces, objects, components, and nodes. Class diagrams, object diagrams, component diagrams, and deployment diagrams are perceived as structural diagrams.
- Behavioral diagrams on the other hand focus on the dynamic aspect of a system that describe the changing or moving parts of a system. Use case diagrams, sequence diagrams, collaboration diagrams, statechart diagrams, and activity diagrams are perceived as behavioural diagrams.

Any given system should be represented using both structural and behavioural diagrams for a full view of its static and dynamic elements.

Use Case Diagram

Use case diagrams portray the dynamic behaviour of a system, which focuses on system behaviour in its operational state. Use case diagrams describe what the system does and not how it does the work [2]. Thus, they are good for the requirements gathering and identification of system actors. Use case diagrams are made up of two main aspects, namely, a use case and an actor. Use cases represent a task that involves external interaction with a system [2]. On the other hand, actors represent people or other systems that perform the tasks specified by use cases. In summary, use case diagrams may be viewed as tools that aid in requirements gathering, getting an outside perspective of the system, identification of external and internal factors that influence the system, and finally as a way of showing the interaction among actors and requirements.

Figure 1 shows the symbols and relationships used in a use case, where:

- A sticky person figure – representing an actor.
- An oval – representing a use case, which is basically a task.
- Connecting lines – acts as links among actors and use cases.

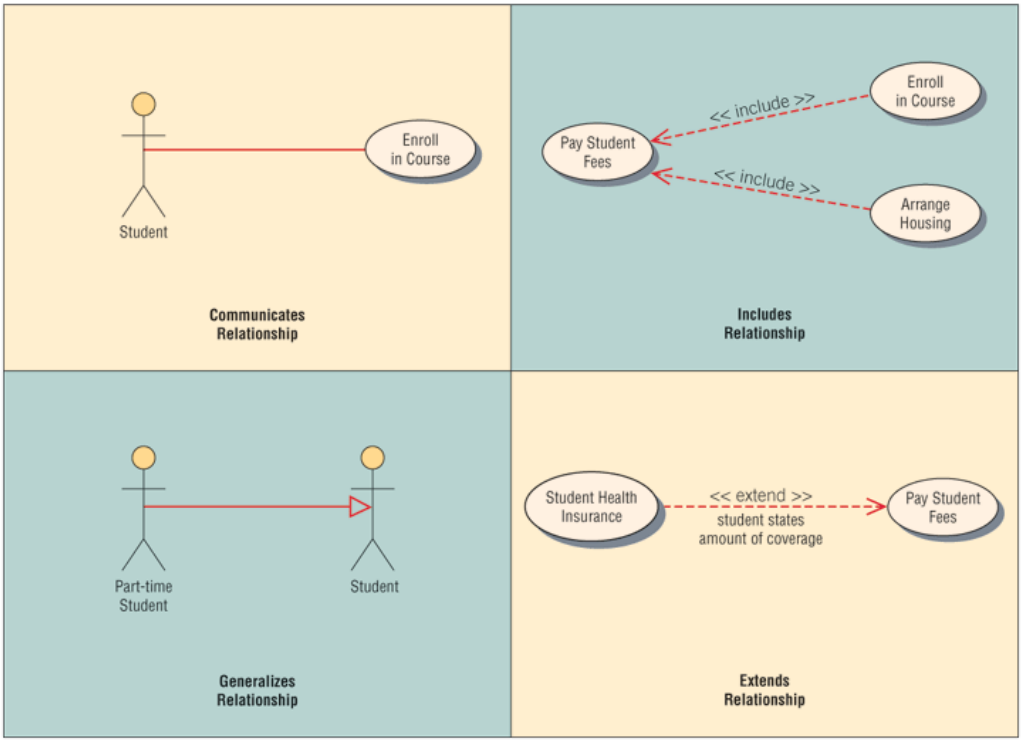

Figure 1. Use Case Diagram Symbols and Relationships [2]

Each actor and use case must have a name describing it. According to Kendall and Kendall [2], to draw a use case diagram, the following steps could be followed

1. Review the business specifications and identify the actors within the problem domain.
2. Identify the high-level events and develop the primary use cases that describe the events and how actors initiate them.
3. Review each primary use case to determine possible variations of flow through the use case.
4. Develop the use case documents for all primary use cases and all important use case scenarios.

Class Diagram

Class diagrams represent a static perspective of a given application. Apart from just visualizing, describing and documenting a system, class diagrams also aid in construction of executable codes of any system. They describe the attributes and operations on classes identified for the system. In addition, they also describe the constraints imposed on the system. According to Kendall and Kendall [2], class diagrams are used during the development of object-oriented system models by showing system classes and their associations, where:

- A class is represented by a compartmented rectangle where the top compartment contains attributes of a class while the lower compartment contains the properties of a class.
- Links between classes may be associations, compositions, or generalization, each of which has a different connection symbol used.

Figure 2 shows an example of a class diagrams founded on association relationships between classes.

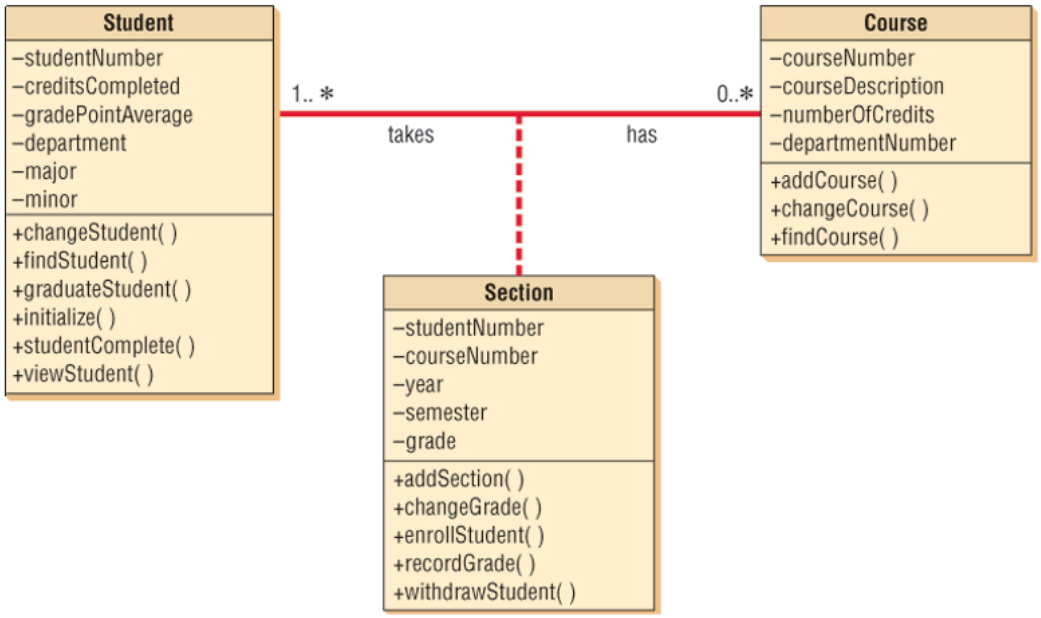

Figure 2. Example of an Association Class Diagram [2]

In summary, class diagrams aid in analysis and design of the static view of an application, communicate responsibilities of a system, act as a basis for component and deployment diagrams, and also aid in forward and reverse engineering.

Activity Diagram

Activity diagrams describe dynamic aspects of a system. They use a flowchart approach as a way of representing flow from activity to activity, where an activity is basically an operation of the system. In addition to providing a visualization aspect of the dynamic system behaviour, activity diagrams aid in construction of the executable system using both forward and reverse engineering techniques. In summary, activity diagrams present the flow of activities of a system, illustrate the sequence from activity to activity, and also portray the parallel, branched and concurrent flow of events. According to Kendall and Kendall [2], activity diagrams show the sequence of both sequential and parallel activities in a process.

Figure 3 shows the symbols used in activity diagrams, where:

- A round cornered rectangle represents and activity.
- A rhombus represents decisions.
- A line with an arrowhead represents events connecting activities.

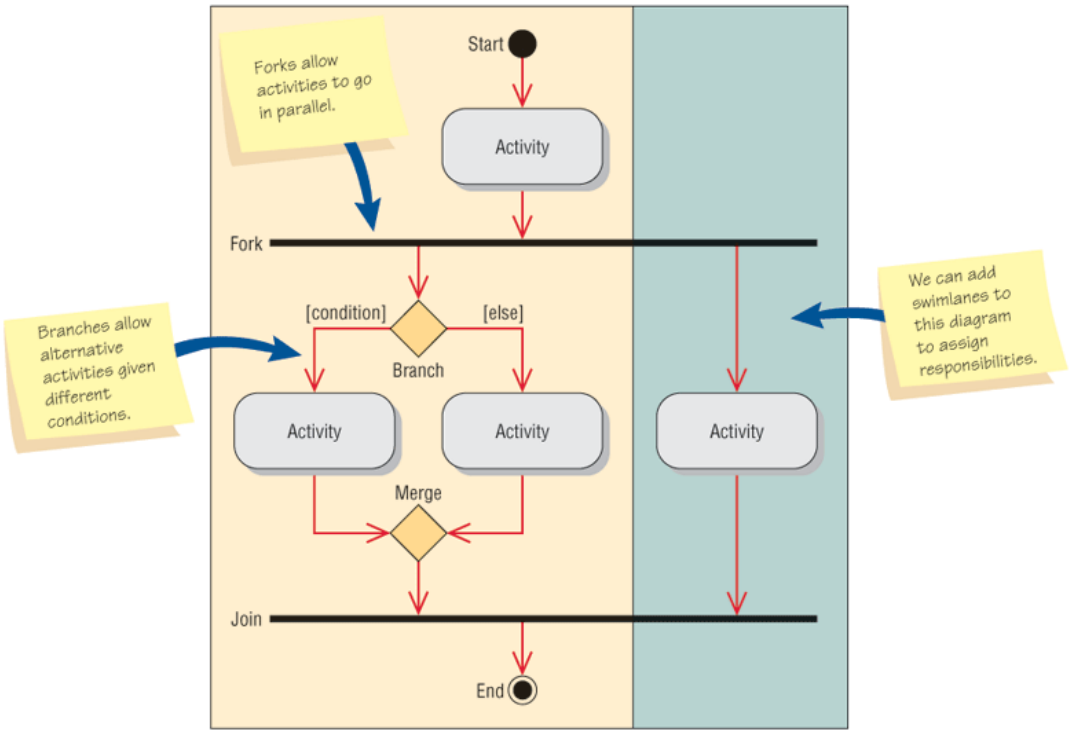

Figure 3. Activity Diagram Symbols [2]

Sequence and Collaboration Diagrams

Sequence and Collaboration diagrams represent a behavioural element of system. A sequence diagram shows a succession of interactions between classes or object instances over time, where [2]:

- The starting point is the leftmost object or class.
- Time sequence follows a top to bottom fashion.
- Messages or signals sent between classes are represented by horizontal arrows such that solid arrowheads represent synchronous calls where the sender class expects and waits for a response while half or open arrowheads represent asynchronous calls that do not expect or wait for a returning signal.

Generally, sequence diagrams emphasize on time ordering of messages or signals as shown in the example in Figure 4 of a student admission process.

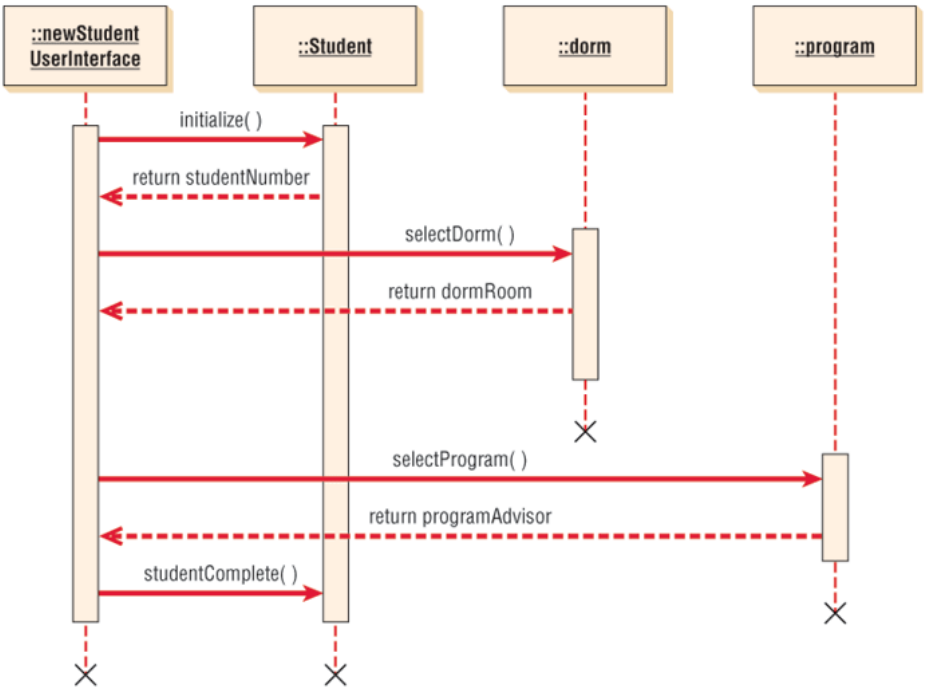

Figure 4. Example of a Student Admission Sequence Diagram [2]

Collaboration diagrams basically convey the same information as a sequence diagram. However, there emphasis is on the organization of the objects or classes which totally varies from that of sequence diagrams. Hence, ordering of events is accomplished by the inclusion of a sequence number per message. Figure 5 conveys similar information to that conveyed in Figure 4.

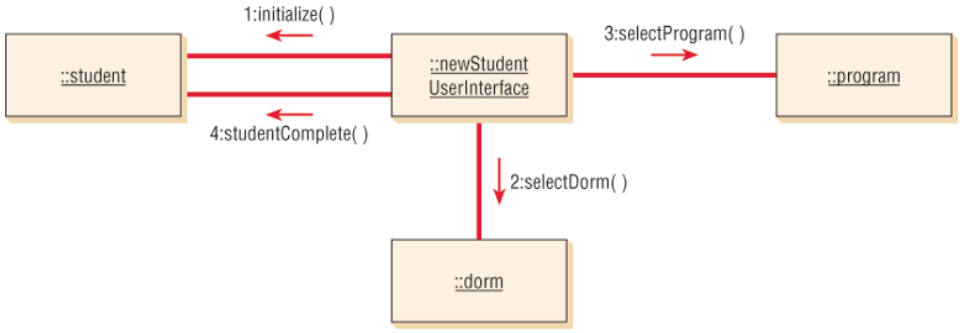

Figure 5. Example of a Student Admission Collaboration Diagram [2]

Thus, it is upon the team involved in system modeling to choose on one of the two models to use for the purpose of showing a succession of interactions between classes or object instances over time. There is no need of drawing both diagrams for one system development case scenario. The two diagrams are sometimes referred to as interaction diagrams which represent a dynamic behaviour of a system, describe the message flow, structural organization of objects, and the interaction among objects in a given system.

Statechart Diagram

A statechart diagram also referred to as a state transition diagram models the dynamic behaviour of a system by defining the different states of a given object or class during its lifetime. Such states are changed through execution of events that can be either internal or external and that happen at a specific time and place. By so doing, statechart diagrams describe the flow of control from state to state where a state is perceived from a given point in time. Statechart diagrams model the lifetime of an object or class from its inception all the way to its termination.

Figure 6 shows an example of a statechart diagram for a student progression process where states are shown in round cornered circles, while labeled lines with arrowheads depict events leading to state changes.

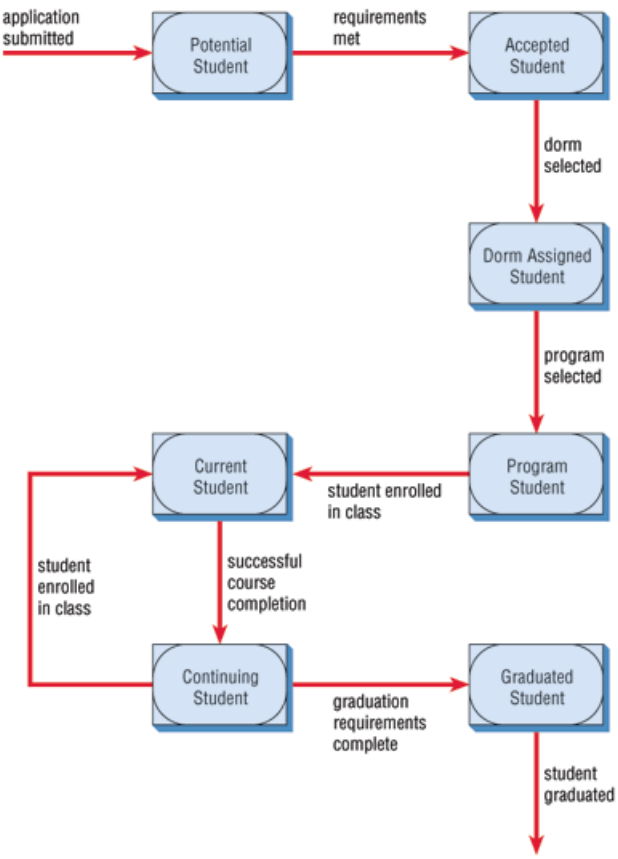

Figure 6. Example of a Student Progression Statechart Diagram [2]

In summary, statechart diagrams model dynamic elements of a system by covering its lifetime from individual object or class perspective. In return, they end up as state machines modeling states of system objects or classes. Thus, according to Kendall and Kendall, statechart diagrams show class states and the events that cause them to transition between states.

Statechart diagrams are applicable in cases where a class has an operational and/or complex life cycle, an instance of a class may update its attributes in various ways during its lifecycle, two classes depend on each other, and an object's current behaviour depends on its previous happenings.

Deployment Diagram

A deployment diagram targets at providing a visualization of the topology of the hardware making up a system. Ideally, hardware provides an operational environment for software. They provide an outline of nodes and their relationships in a network. Such nodes includes workstations, servers, switches, routers, among others as shown in the example in Figure 7.

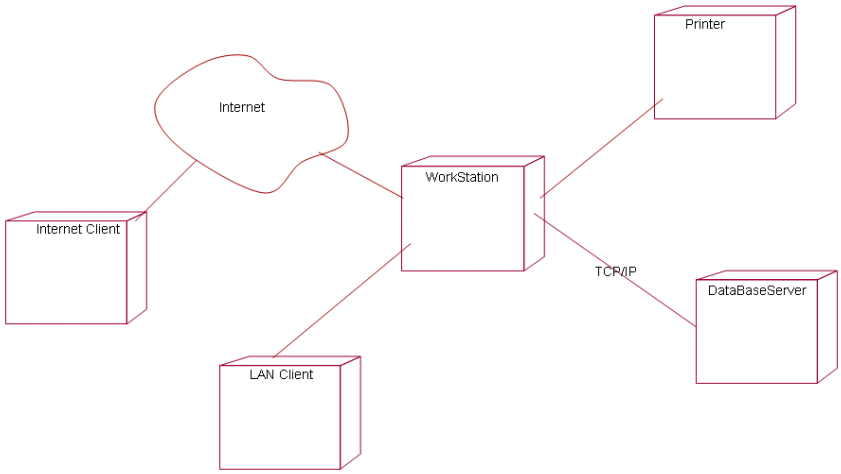

Figure 7. Example of a Deployment Diagram
(

Steps Used in UML Modeling

According to Kendall and Kendall [2], the following are guiding steps used in UML modeling:

1. Definition of the use case model.
2. Proceed with UML modeling for the system during the analysis phase.
3. Create class diagrams.
4. Create statechart diagrams.
5. Commence system design by UML models refinement.
6. Perform a detailed system design.

Summary

The topic finalized system modeling by exploring UML set of models as an enhancement of the non-UML models covered in the previous topic. The UML models covered in this topic are use case diagram, class diagram, activity diagram, sequence diagram, collaboration diagram, statechart diagram, and deployment diagram. In each case the discussion emphasized on the purpose of the model, as a way to help software engineers in choosing appropriate models to develop. The next topic will cover architectural design as another perspective of system modeling.

Check Points

1. Discuss any three reasons why UML modeling is essential.
2. Differentiate between a sequence diagram and a collaboration diagram.
3. Discuss the benefits of use case diagrams in the system development process.
4. Discuss the benefits of activity diagrams in the system development process.
5. Discuss the benefits of class diagrams in the system development process.
6. Describe the relationship between statechart diagrams and deployment diagrams.

Learning Resources

Core Textbooks

1. Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

Other Resources

2. Kendall, K. E. and Kendall, J. E., Systems Analysis and Design, 8th Edition, Pearson, 2011.

SReferences

[1] Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

[2] Kendall, K. E. and Kendall, J. E., Systems Analysis and Design, 8th Edition, Pearson, 2011.

[3] Elmasri, R., & Navathe, S. B. (2021). Fundamentals of Database System.

[4] Pries, K. H., & Quigley, J. M. (2010). Scrum project management. CRC press.

[5] Schwaber, K. (2004). Agile project management with Scrum. Microsoft press.

[6] Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. Communications of the ACM, 40(11), 110-118.

[7] Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.