Chapter 5

System Modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.

Models are used during the Requirements Engineering process to help derive the requirements for the system, during the design process to describe the system to engineers implementing the system and after implementation to document the system's structure and operation.

Models of Existing Systems and System to-be Developed

- Models of the existing system are used during requirements engineering. They help clarify what the existing system does and can be used as basis for discussingits strengths and weaknesses. These then lead to requirements for the new system
- Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation. In a model-driven engineering process, it is possible to generate a complete or partial system implementation from the system model.

The most important aspect of a system model is that it leaves out detail. A model is an abstraction of the system being studied rather than an alternative representation of that system. An abstraction deliberately simplifies and picks out the most salient characteristics.

Different Perspectives from which to create different models to represent the system:

1. External Perspective
    a. You model the context or environment of the system
2. Interaction Perspective
    a. You model the interactions between a system and its environment or between the components of a system
3. Structural Perspective
    a. You model the organization of a system or the structure of the data that is processed by the system
4. Behavioral Perspective
    a. Where you model the dynamic behavior of the system and how it responds to events

5 Diagrams from UML that could represent the essentials of a system

1. Activity Diagrams: show the activities involved in a process or in data processing
2. Use Case Diagrams: show the interactions between a system and its environment
3. Sequence Diagrams: Show interactions between actors and the system and between system component
4. Class Diagrams: show the object classes in the system and the associations between these classes.
5. State Diagrams: Show how the system reacts to internal and external events.

3 Ways Graphical Models are used:

1. As a means of facilitating discussion about an existing or proposed system
    a. Purpose of the models is to stimulate the discussion among SW engineers
2. As a way of documenting an existing system
    a. These do not have to be complete as you only develop models for some parts of the system
3. As a detailed system description that can be used to generate a system implementation.
    a. These models have to be both complete and correct because they are used as a basis for generating the source code of the system.

Context Models (Section 5.1)

- Context models normally show that the environment includes several other automated systems
- Context models do not show the types of relationships between the systems in the environment and the system that is being specified
- These models are often used with other models such as Business Process Models (which are Activity Diagrams.
    - Activity Diagrams are intended to show that activities that make up a system process and the flow of control from one activity to another. These Diagrams also show which systems in the system and/or environment take care of certain activities

Interaction Models (Section 5.2)

- Models User Interaction and System-to-System Interaction and Component Interaction.
- 2 Approaches to Interaction Modeling
    - 1) Use Case Modeling
        - Used to model interactions between a system and external actors (users or other systems)
    - 2) Sequence Diagrams
        - Used to model interactions between system components, although external agents may also be included
- Use Case Modeling (Section 5.2.1)
    - Used to support Requirements Elicitation
    - They give a fairly simple overview of an interaction so you have to provide more detail to understand what is involved. This detail can either be a simple textual description, a structured description in a table, or a sequence diagram.
- Sequence Diagrams (Section 5.2.2)
    - Used to model the interactions between the actors and objects in a system and the interactions between the object themselves.
    - Shows the sequence of interactions that take place during a particular use case or use case instance.
    - The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these. Interactions between objects are indicated by annotated arrows. The rectangles on the dotted lines indicate the lifeline of the objects concerned. The annotations of the arrows indicate the calls to the objects, their parameters, and their return values.

    - Unless you are using sequence diagrams for code generation or detailed documentation, you don't have to include every interaction in these diagrams.
    - If you develop system models early in the development process to support requirements engineering and high-level design, there will be many interactions which depend on implementation decisions.

Structural Models (Section 5.3)

- These display the organization of a system in terms of the components that make up that system and their relationships.
- Class Diagrams (Section 5.3.1)
    - Used when developing an Object-Orient system model to show that classes in the system and the associations between these classes.
    - An association is a link between classes that indicates that there is a relationship between these classes. These are shown in diagrams with simple lines and number that show 1-1 or 1-n or m-n relationships.
    - Can also be used to describe databases
    - Sometimes these diagrams can omit class details such as attributes and methods
        - If they are included, the name of the object class is at the top of the box, the attributes in the middle (this must include attribute names and optionally their types), and operations in the lower section
- Generalization (section 5.3.2)
    - Rather than learn the detailed characteristics of every entity that we experience, we place these entities in more general classes (animals, cars, houses, etc.) and learn the characteristics of these classes
        - In UML, generalization (in Java, called Inheritance), is shown through the use of a line with a triangle on one side denoting the parent
- Aggregation (Section 5.3.3)
    - UML provides a special type of association between classes called Aggregation where one object (the whole) is composed of other objects (the parts).
    - To show this, we use a diamond shaped next to the class that represent the whole.

Behavioral Models (Section 5.4)

- Are models of the dynamic behavior of the system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- 2 Types of Stimuli
    - Data
        - Some data arrives that has to be processed by the system
    - Events
        - Some event happens that triggers system processing. Events may have associated data but this is not always the case.
- Data Driven Modeling (Section 5.4.1)
    - Shows the sequence of actions involved in processing input data and generating an associate output. They show the entire sequence of actions that take place from

an input being processed to the corresponding output, which is the system's response.
    - UML Sequence Diagrams can also be used here but you must structure them so that messages are only sent from the left to the right.
- Event Driven Modeling (Section 5.4.2)
    - Shows how a system responds to external and internal events
    - Based on the assumption that a system has a finite number of states and that events may cause a transition from one state to another.
    - State Diagrams show system states and events that cause transitions from one state to another. They do NOT show the flow of data within the system.
    - Each state in the diagram must include a brief description of the actions taken in that state.
    - In addition to the state diagram, you have to provide more detail about both the stimuli and the system states. (You can use tabular descriptions)

Model Driven Engineering (Section 5.5)

- Is an approach to SW Development where models rather than program are the principal outputs of the development process. The programs that execute on hardware/software platform are then generated automatically from the models.
- Model Driven Architecture focuses on the design and implementation stages of software development whereas Model Driven Engineering is concerned with all aspects of the SW Engineering Process.
- Arguments for and against Model Driven Engineering
    - For
        - Allows engineers to think about system at a high level of abstraction, without concern for the details of their implementation.
        - This reduces the likelihood of errors, speeds up the design and implementation process, and allows for the creation of reusable, platform-independent application models
    - Against
        - Models are a good way of facilitating discussions about SW design…It does NOT always follow that the abstractions that supported by the model are the right abstractions for the implementation.
        - Arguments for platform independence are only valid for large long-lifetime systems where the platforms become obsolete during a system's lifetime. For this class of system, we know that implementation is not the major problem-requirements engineering, security and dependability, integration with legacy systems, and testing are more significant
- Model Driven Architecture (Section 5.5.1)
    - Model-focused approach of SW Design and Implementation that uses a subset of UML models to describe a system.
    - 3 Types of Abstract System Model should be produced
        - 1) Computation independent model (CIM)
            - Models the important domain abstractions used in the system
        - 2) Platform Independent Model (PIM)

        - Models the operation of the system without reference to its implementation
    - 3) Platform Specific Models (PSM)
        - Transformations of the PIM with a separate PSM for each application platform. There may be layers of PSM, with each layer adding some platform-specific detail.
  - Transformations between these models may be defined and pplied automatically by software tools
  - CIM to PIM translation is till at the research prototype stage
  - PIM to PSM is mature and several communication tools are available that provide translators from PIMs to common platforms such as Java and J2EE.
- Executable UML (Section 5.5.2)
  - Completely automated transformation of models to code
  - To create an executable subset of UML, the number of model types has therefore been dramatically reduced to 3 key model types:
    - Domain models identify the principal concerns in the system. These are defined using UML class diagrams that include objects, attributes, and associations.
    - Class models, in which classes are define, along with their attributes and operations
    - State models, in which a state diagram is associated with each class and is used to describe the lifecycle of the class

Key Points

- A model is an abstract view of a system that ignores some system details. Complementary system models can be developed to show the system's context, interactions, structure, and behavior.
- Context models show how a system that is being modeled is positioned in an environment with other systems and processes. They help define the boundaries of the system to be developed
- Use case diagrams and Sequence diagrams are used to describe the interactions between user and the system being designed and users/other systems. Use Cases describe interactions between a system and external actors; Sequence diagrams add more information to these by showing interactions between system objects.
- Structural models show the organization and architecture of a system. Class diagrams are used to define the static structure of classes in a system and their associations
- Behavioral Models are used to describe the dynamic behavior of an executing system. This can be modeled from the perspective of the data processed by the system or by the events that stimulate responses from a system.
- Activity diagrams may be used to model a system's behavior in response to internal or external events
- Model Driven Engineering is an approach to SW development in which a system is represented as a set of models that can be automatically transformed to executable code.