System Modeling:

- System modeling is the process of creating abstract models of a system.
- It involves representing a system using graphical notations, often based on UML.
- Helps analysts understand system functionality and communicate with customers.

Existing and Planned System Models:

- Existing system models clarify the current system's functionality.
- New system models help explain requirements and design proposals.
- Model-driven engineering can generate system implementations from models.

System Perspectives:sibe

- External perspective (modeling the system's context)
- Interaction perspective (modeling interactions)
- Structural perspective (modeling system organization and data structure)
- Behavioral perspective (modeling dynamic system behavior)

UML Diagram Types:cscsa

- Activity diagrams show the activities involved in a process or in data processing .
- Use case diagrams show the interactions between a system and its environment.
- Sequence diagrams show interactions between actors and system and system components
- Class diagrams w the object classes in the system and the associations between these classes
- State diagrams which show how the system reacts to internal and external events.

Use of Graphical Models:

- Facilitating discussion about existing or proposed systems
- Documenting existing systems accurately
- Creating detailed system descriptions for system implementation

Context Models:

- Illustrate the operational context of a system, showing what lies outside the system boundaries.
- Consider social and organizational factors when positioning system boundaries.
- Architectural models depict the system's relationship with other systems.

System Boundaries:

- Define what is inside and outside the system.

- Influence system requirements and can be politically driven.

Interaction Models:

- Model user interactions to identify user requirements.

- Highlight communication issues in system-to-system interactions.

- Help understand system performance and dependability in component interactions.

- Use case diagrams and sequence diagrams are common for interaction modeling.

Use Case Modeling:

- Use cases represent discrete tasks involving external interactions with a system.

- Actors can be people or other systems.

- Represented diagrammatically and in textual form.

Sequence Diagrams:

- Model interactions between actors and objects in a system.

- Show the sequence of interactions during a use case or use case instance.

Structural Models:

- Display the organization and architecture of a system in terms of components and their relationships.

- Static models show system design structure, while dynamic models depict system organization during execution.

- Used in discussing and designing system architecture.

Generalization:

- Manages complexity by placing entities into more general classes.

- In object-oriented modeling, implemented using class inheritance.

- Subclasses inherit attributes and operations from superclasses.

Behavioral Models:

- Model the dynamic behavior of a system during execution.

- Show how a system responds to stimuli from its environment.

- Stimuli can be data or events.

Data-Driven Modeling:

- Depicts the sequence of actions involved in processing input data and generating output.

- Useful for requirements analysis.

Event-Driven Modeling:

- Common in real-time systems, responding to events rather than data.
- Shows how a system responds to external and internal events.

State Machine Models:

- Model system behavior in response to events.
- Represent system states and transitions.
- Useful for modeling real-time systems.

Model-Driven Engineering:

- An approach where models, not programs, are the primary development outputs.
- Programs are generated automatically from models.
    - Pros
        - Allows systems to be considered at higher levels of abstraction
        - Generating code automatically means that it is cheaper to adapt systems to new platforms.
    - Cons Models for abstraction and not necessarily right for implementation.

        Savings from generating code may be outweighed by the costs of developing translators for new platforms.

Types of Model in MDA:

- Computation Independent Model (CIM) domain
- Platform Independent Model (PIM) no reference to implementation
- Platform Specific Model (PSM)

MDA Transformations:

- Transforming models from higher to lower levels of abstraction.

Agile Methods and MDA:

- MDA can be used in an agile development process if transformations are automated.

Adoption of MDA:

- Limited adoption due to specialized tool support, costs, and platform dependence.
- Models may not always be suitable for implementation.