

1 Lecture 4: System Modeling

Course: Software Engineering and Systems Analysis

Topic: System Models and Viewpoints

Recommended duration: 90 minutes

1.1 Learning objectives

Students should be able to explain the purpose of system modeling, distinguish context, interaction, structural, and behavioral models, define system boundaries, choose suitable viewpoints, and evaluate a model for clarity and consistency.

1.2 1. What a system model represents

A **system model** is an abstraction of a system and its environment. It describes selected properties so that people can reason about requirements, behavior, structure, risks, or architecture. Because a model is selective, it is always created for a purpose and an audience.

System modeling is not limited to software. A model may describe people, organizations, devices, processes, data stores, external services, and physical constraints. The system boundary determines which elements are treated as part of the system and which are treated as environmental context.

1.3 2. Important viewpoints

A **context model** shows the system in relation to neighboring systems and organizations. It answers: “What is inside the scope, and what interacts with it?” An **interaction model** shows communication between users, systems, or components. A **structural model** shows the organization of entities and relationships. A **behavioral model** shows responses, workflows, or state changes.

These viewpoints should be coordinated. A context model may identify a payment provider that later appears as an actor in a use case and as an external interface in a component model.

1.4 3. Context modeling

Context diagrams are valuable at the beginning of analysis because they make scope visible. They should show the system as one boundary and identify external entities with meaningful labels. A line should communicate a relationship or flow, not merely proximity.

For a hospital appointment system, external entities may include patients, clinic staff, insurance services, notification services, and electronic health record systems. The context model should not prematurely show internal classes or database tables. Those belong to more detailed views.

1.5 4. Interaction modeling

Interaction models focus on messages and responsibilities. A sequence diagram is appropriate when the order of communication matters. It commonly contains actors, lifelines, messages, return values, and fragments for alternatives or repetition.

For “Book Appointment,” a patient may send a request to a web application. The application asks an appointment service for available slots. The service queries a schedule repository and returns choices. After the patient selects a slot, the service creates the appointment and requests a confirmation notification.

The model helps identify missing responsibilities, unnecessary coupling, and failure points. It should not imply that every method call must be documented in one diagram.

1.6 5. Behavioral and state models

A behavioral model describes how the system or a component changes over time. Activity diagrams are useful for workflows with decisions, parallel paths, and responsibilities. State machine diagrams are useful when one entity has a limited set of meaningful states.

An appointment may move through Requested, Confirmed, Completed, Cancelled, or NoShow. Events and rules determine legal transitions. A state model prevents contradictory behavior, such as marking an already cancelled appointment as completed without an explicit business rule.

1.7 6. Architectural viewpoints

Architecture models describe major components, connectors, deployment nodes, and quality concerns. A component model may show a web client, application service, database, and external notification provider. A deployment model may map these artifacts to servers, containers, or devices.

An architecture model should reveal important decisions and trade-offs. For example, separating a notification service may improve scalability and fault isolation, but it may also introduce asynchronous behavior and delivery uncertainty. The model should make such concerns visible rather than hiding them.

1.8 7. Model quality

A model is useful when it is fit for purpose, understandable to its audience, internally consistent, and maintained when important assumptions change. A highly detailed model is not automatically better. Excessive detail can obscure the decision the model is supposed to support.

Check whether boundaries are explicit, names are consistent, relationships have a defined meaning, and important exceptions are represented. Validate the model with examples. Ask different stakeholders to explain what they think the model means; disagreements often reveal ambiguous requirements.

1.9 8. Example model set: library system

A context model places the library management system at the center and connects it to members, librarians, a payment service, and an email service. A use case model shows goals such as search catalog, borrow item, return item, and pay fine. A class model identifies members, items, loans, and fines. A state model describes the lifecycle of a loan.

The model set is more useful than any one diagram because each view answers a different question. The relationships between views also support traceability. For example, the “return item” use case should be explainable through interactions among the relevant classes and should affect the loan state.

1.10 9. Practice activity

Choose an online examination system and produce four views: a context model, one use case description, a sequence model for starting an exam, and a state model for an exam attempt. For each view, state its purpose and audience. Identify one assumption that requires confirmation from a stakeholder.

1.11 Summary

System modeling uses purposeful abstractions to explain context, structure, interaction, behavior, and architecture. Effective modelers control scope, select suitable viewpoints, and keep shared concepts consistent. A model is successful when it reduces uncertainty and supports a concrete engineering decision.

1.12 References