

# CS 4320 / 7320

# Software Engineering

Module 2 - Models and Methods:

## MODELS

# Modeling...

... is an **organized** and **systematic** approach for representing **significant aspects** of the software under consideration.

...**facilitates decision-making** about the software

...**communicates** those decisions to *various stakeholders*

# Modeling Principles: Abstraction

Model the **essentials**...

# Modeling Principles: Restricted Views

Provide specific (rule-based) views

**Views:** Structural view, behavioral view, temporal view, organizational view, etc...

**Rules:** notation, vocabulary, methods, tools

# Modeling Principles: Communication

Modeling enables effective communication

- Application domain vocabulary

- Modeling language

- Semantic expression (meaning within context)

# Caution...

## False confidence

Be aware a model or models do not yield complete understanding  
Models are abstractions (stuff is missing)

## Syntax

Understand and adhere to the precise meanings of syntax

## Changes

Be aware of changing context and assumptions (more on that later)

# Beware Assumptions

“Abstraction leads to a set of assumptions  
about the context in which the model is placed  
that should also be captured in the model.”

SWEBOK 9-3

Preconditions

Postconditions

Invariants

# Beware Assumptions

“Abstraction leads to a set of assumptions about the context in which the model is placed that should also be captured in the model.”

SWEBOK 9-3



# Unified Modeling Language (UML)

1994-1995

Developed by Grady Booch, Ivar Jacobson, and James Rumbaugh at Rational Software to **standardize notation** in Software Engineering

1997

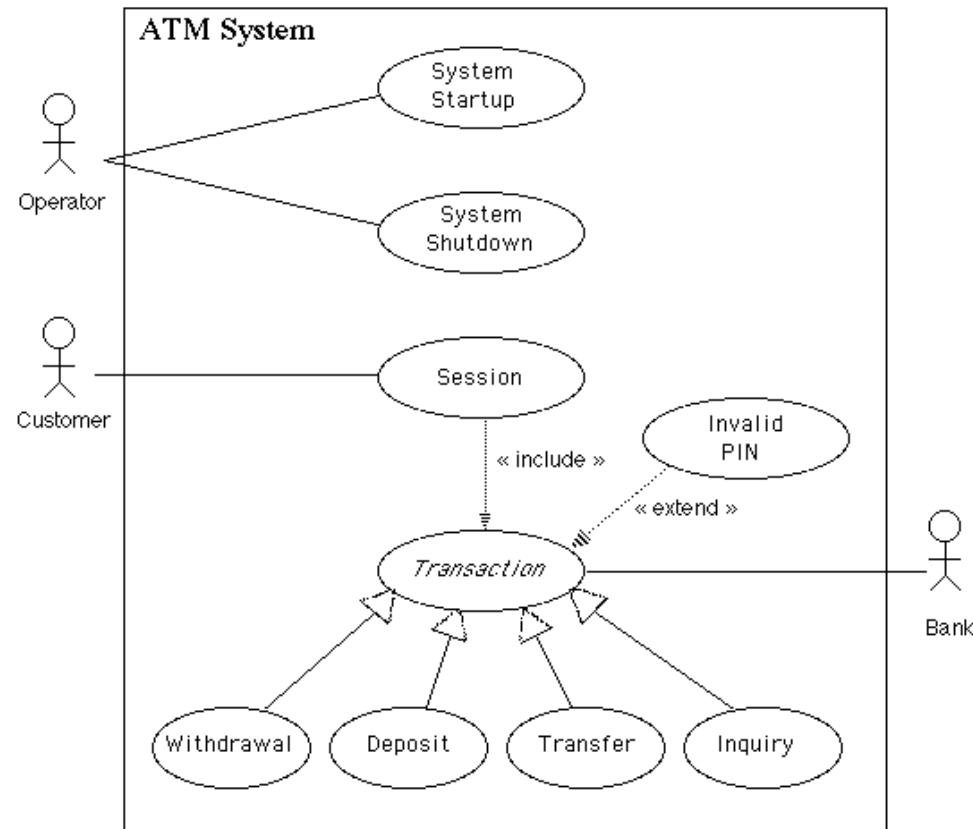
Adopted by Object Management Group (OMG)

2005

Adopted by International Organization for Standardization (ISO)

# A Special Case: Use Cases

For Modeling Requirements



# Types of Models

## Information Modeling

*Conceptual, logical, and physical data models*

## Behavioral Modeling

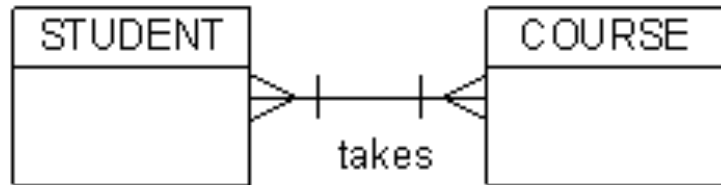
*State machine, control-flow, and data-flow models*

## Structure Modeling

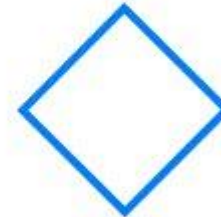
*Class, component, object, deployment, and packaging models*

# Informational Modeling

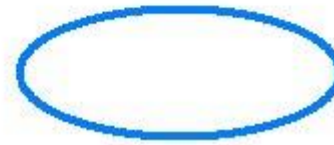
- Entity
- Relationship
- Attribute



Entity



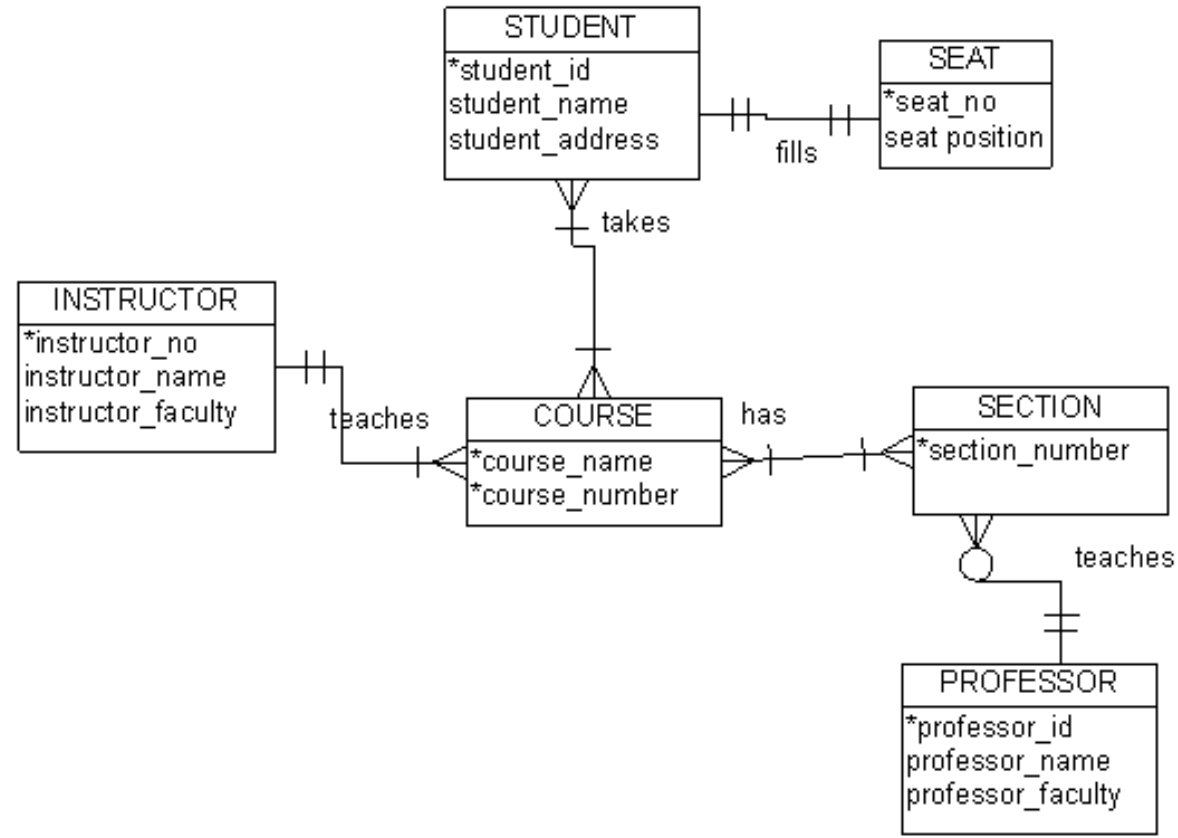
Relationship



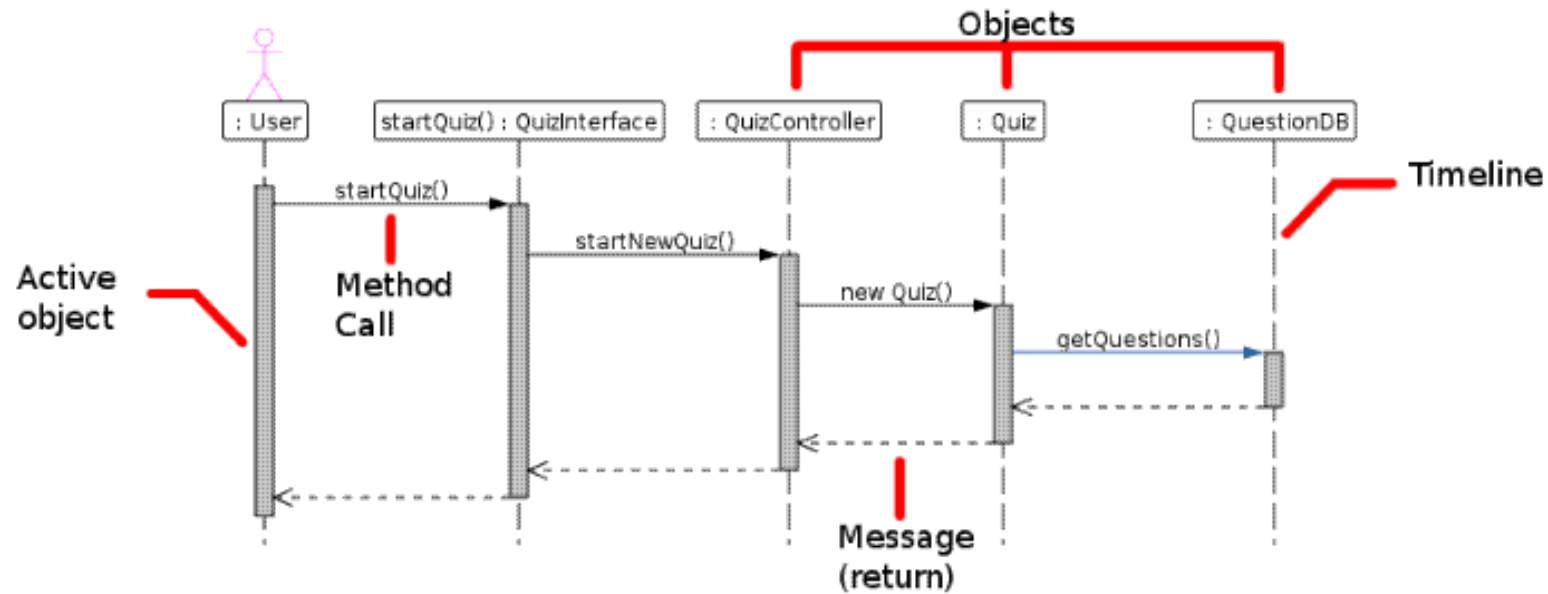
Attribute

# Informational Modeling

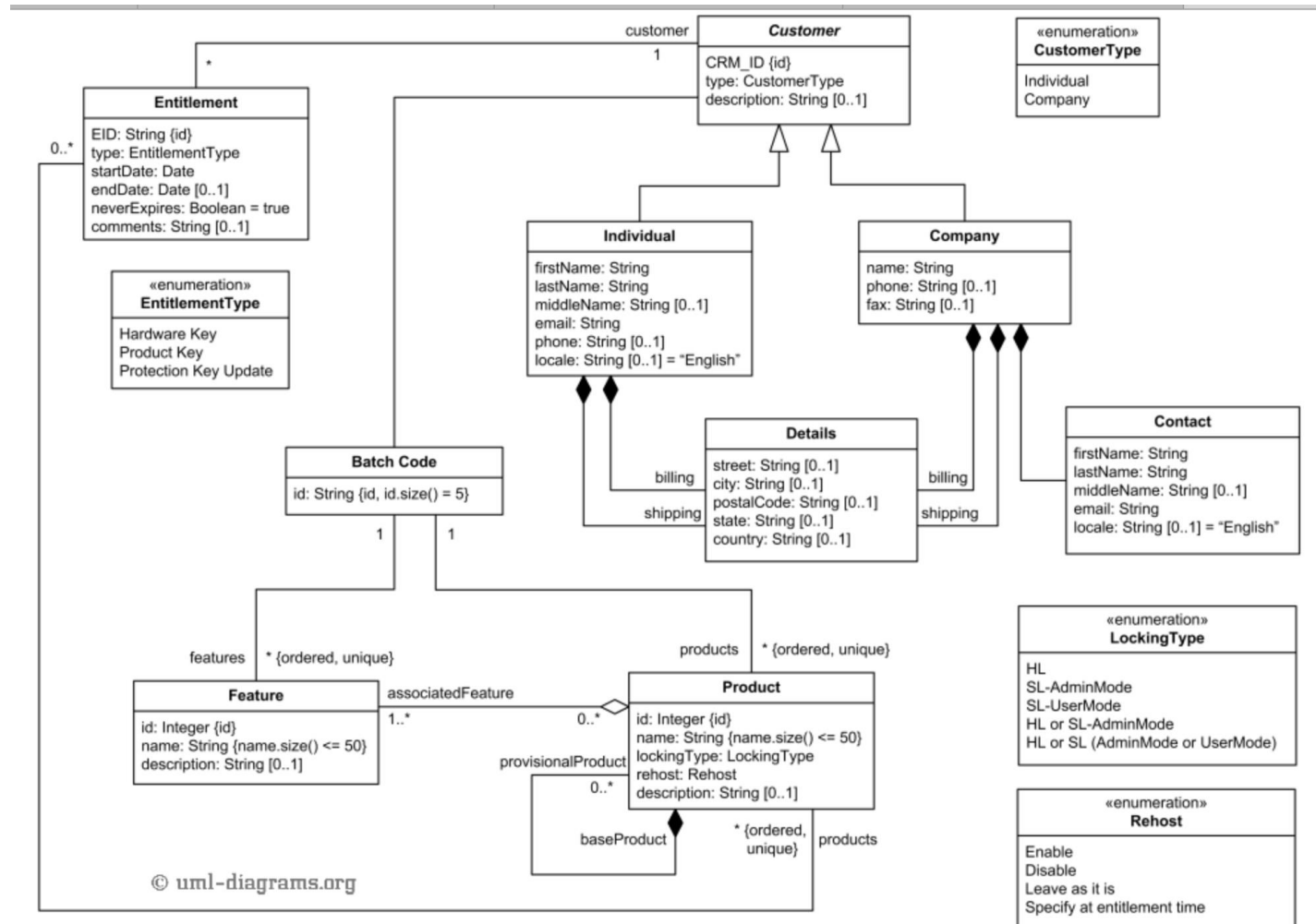
- Entity
- Relationship
- Attribute



# Behavioral Modeling



# Structure Modeling



# Analysis of Models

## Analyzing for **Completeness**

Are all requirements implemented and verified?

## Analyzing for **Consistency**

Do the models conflict?

## Analyzing for **Correctness**

Is the model syntactically and semantically correct?



# Analysis of Models

## Traceability

Can the requirements, models, and code be connected up?

Can changes be traced?

## Interaction Analysis

Does the control flow between parts of the system work as intended?