Software Engineering

Lesson #04 - Lecture

Lesson #04 - Lecture

Your KBTU 202309 Software Engineering class information is updating …

Lesson #04 update is in progress

This will take around 2 hours to complete

Please, don’t turn off your head

System modeling

System modeling

Agenda: Lesson #04 - Software Engineering - Lecture

1 Introduction & Context models

2 Interaction models & Structural models

3 Behavioral models

4 Model-driven engineering

Agenda: Lesson #04 - Software Engineering - Lecture

1 **Introduction & Context models**

2 Interaction models & Structural models

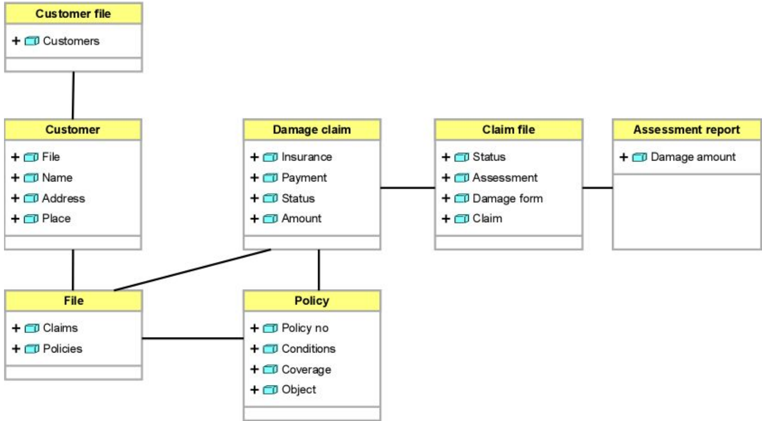
3 Behavioral models

4 Model-driven engineering

Introduction

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

System modeling now usually means representing a system using some kind of graphical notation based on diagram types in the Unified Modeling Language (UML)



Introduction

Models are used during the requirements engineering process to help derive the detailed requirements for a 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

Introduction

You may develop models of both the existing system and the system to be developed:

- Models of the existing system are used during requirements engineering
- They help clarify what the existing system does, and they can be used to focus a stakeholder discussion on its strengths and weaknesses

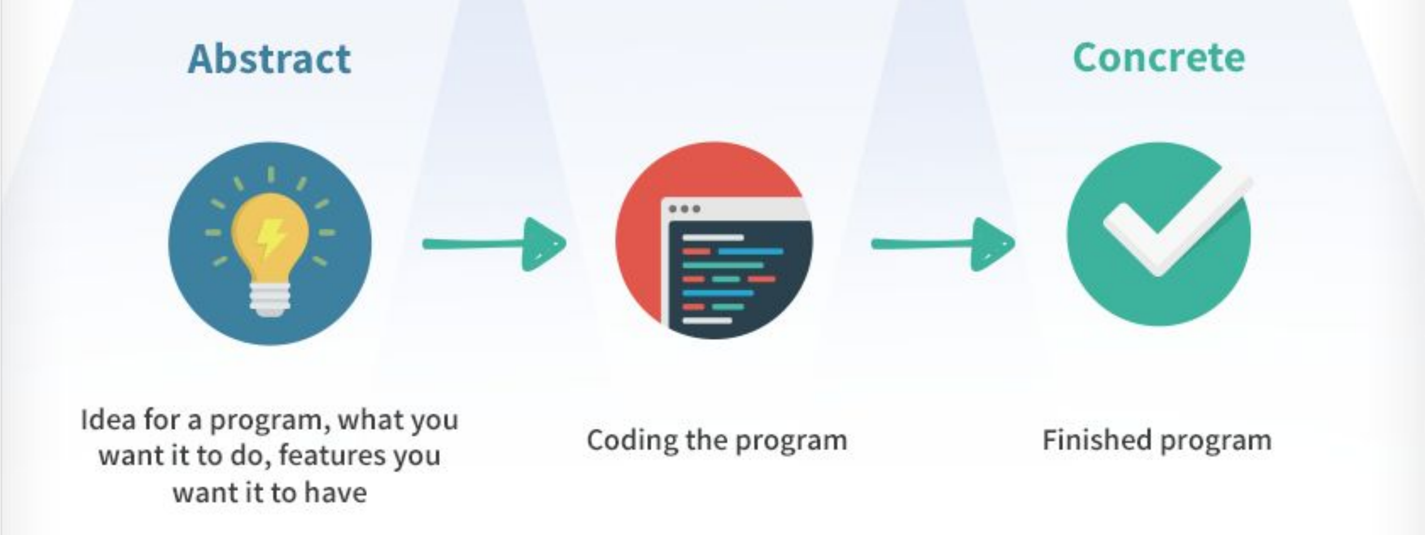
Introduction

You may develop models of both the existing system and the system to be developed:

- 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

Introduction

It is important to understand that a system model is not a complete representation of system


Abstract
Concrete
Idea for a program, what you want it to do, features you want it to have
Coding the program
Finished program

Introduction

The detail and rigor of a model depend on how you intend to use it

There are three ways in which graphical models are commonly used:

- As a way to stimulate and focus discussion about an existing or proposed system

Introduction

The detail and rigor of a model depend on how you intend to use it

There are three ways in which graphical models are commonly used:

- As a way of documenting an existing system

Introduction

The detail and rigor of a model depend on how you intend to use it

There are three ways in which graphical models are commonly used:

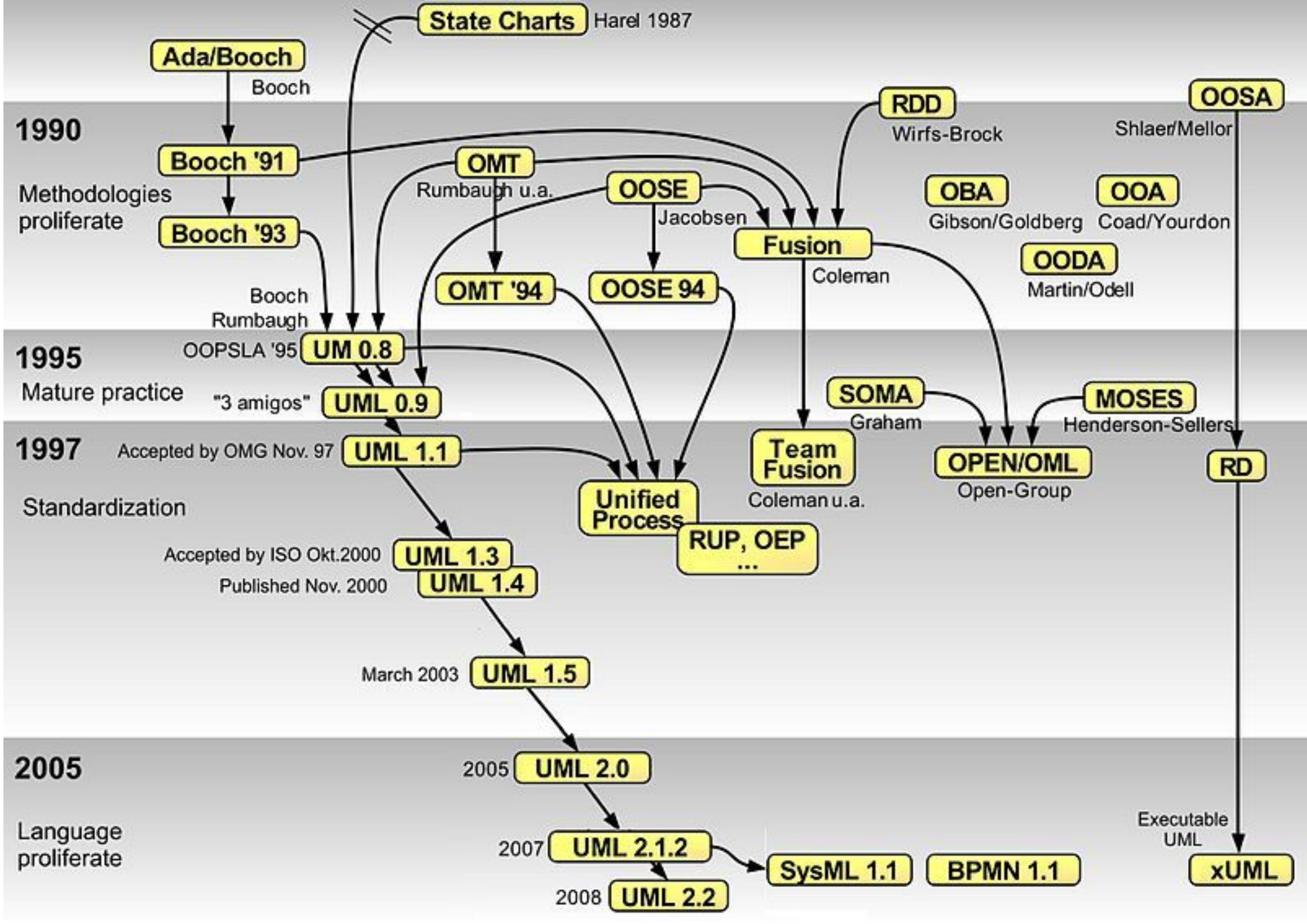
- As a detailed system description that can be used to generate a system implementation

Introduction

UML (Unified Modelling Language) -> 14 diagram types

Widely used 5 diagram types

- Activity diagrams
- Use case diagrams
- Sequence diagrams
- Class diagrams
- State diagrams



State Charts Harel 1987

Ada/Booch Booch

1990

Booch '91

Methodologies proliferate

Booch '93

OMT Rumbaugh u.a.

OOSE Jacobsen

Fusion Coleman

RDD Wirfs-Brock

OOSA Shlaer/Mellor

OBA Gibson/Goldberg

OOA Coad/Yourdon

OODA Martin/Odell

OMT '94

OOSE 94

Booch Rumbaugh

1995

Mature practice

OOPSLA '95 UM 0.8

"3 amigos" UML 0.9

SOMA Graham

MOSES Henderson-Sellers

1997

Accepted by OMG Nov. 97 UML 1.1

Standardization

Unified Process

RUP, OEP ...

Team Fusion Coleman u.a.

OPEN/OML Open-Group

RD

Accepted by ISO Okt.2000 UML 1.3

Published Nov. 2000 UML 1.4

March 2003 UML 1.5

2005

Language proliferate

2005 UML 2.0

2007 UML 2.1.2

2008 UML 2.2

SysML 1.1

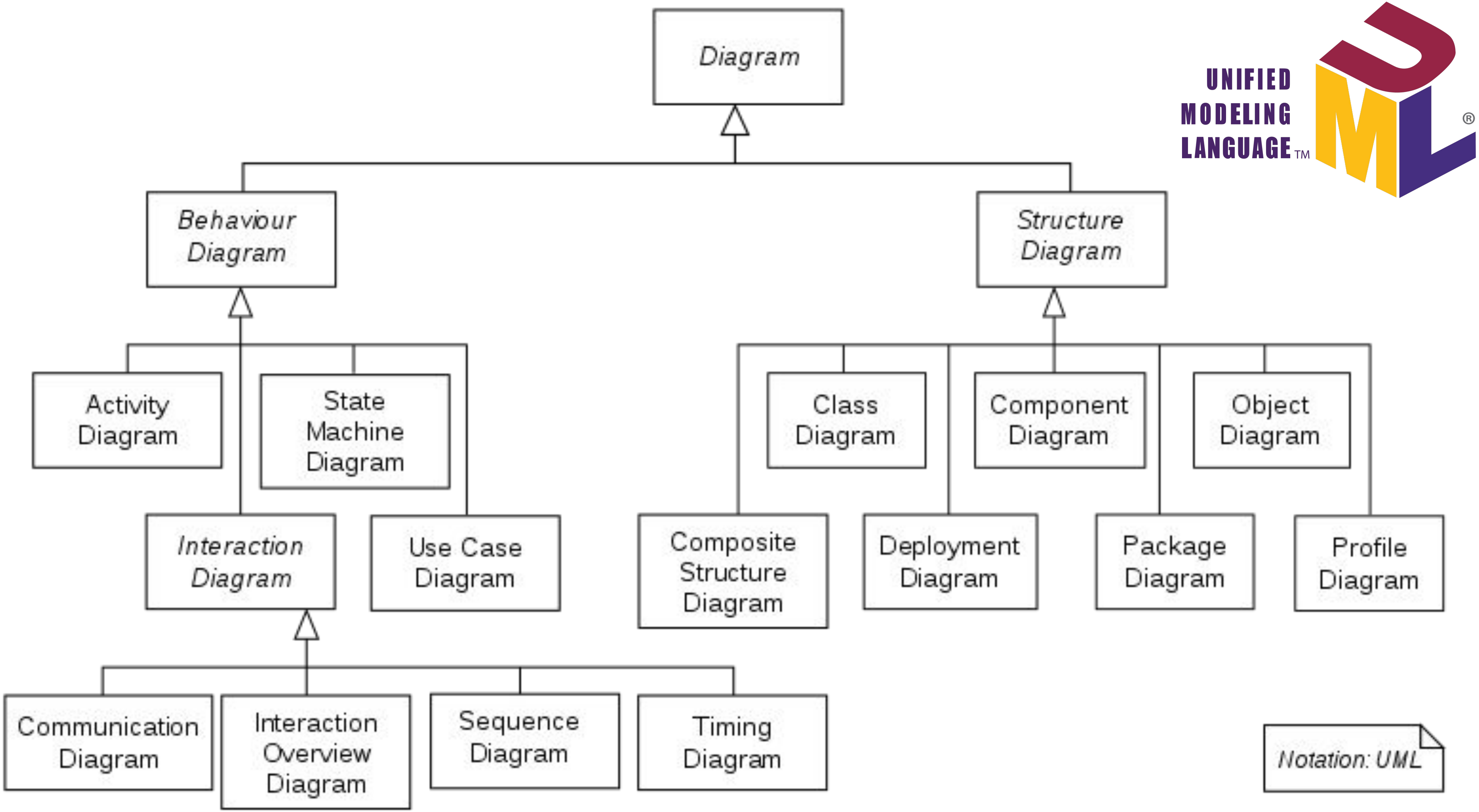
BPMN 1.1

Executable UML

xUML

UNIFIED
MODELING
LANGUAGE TM

UNIFIED MODELING LANGUAGE™



Diagram

Behaviour Diagram

Structure Diagram

Activity Diagram

State Machine Diagram

Interaction Diagram

Use Case Diagram

Communication Diagram

Interaction Overview Diagram

Sequence Diagram

Timing Diagram

Class Diagram

Component Diagram

Object Diagram

Composite Structure Diagram

Deployment Diagram

Package Diagram

Profile Diagram

Notation: UML

Context models

At an early stage in the specification of a system, you should decide on the system boundaries, that is, on what is and is not part of the system being developed

This involves working with system stakeholders to decide what functionality should be included in the system and what processing and operations should be carried out in the system’s operational environment

Context models

Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries

The definition of a system boundary is not a value-free judgment

Social and organizational concerns may mean that the position of a system boundary may be determined by nontechnical factors

Context models

Once some decisions on the boundaries of the system have been made, part of the analysis activity is the definition of that context and the dependencies that a system has on its environment

Normally, producing a simple architectural model is the first step in this activity

Agenda: Lesson #04 - Software Engineering - Lecture

1. Introduction & Context models
2. **Interaction models & Structural models**
3. Behavioral models
4. Model-driven engineering

Interaction models

All systems involve interaction of some kind. This can be user interaction, which involves user inputs and outputs; interaction between the software being developed and other systems in its environment; or interaction between the components of a software system

User interaction modeling is important as it helps to identify user requirements

Interaction models & Structural models

Interaction models

This section discusses two related approaches to interaction modeling:

- Use case modeling, which is mostly used to model interactions between a system and external agents (human users or other systems)
- Sequence diagrams, which are used to model interactions between system components, although external agents may also be included

Interaction models

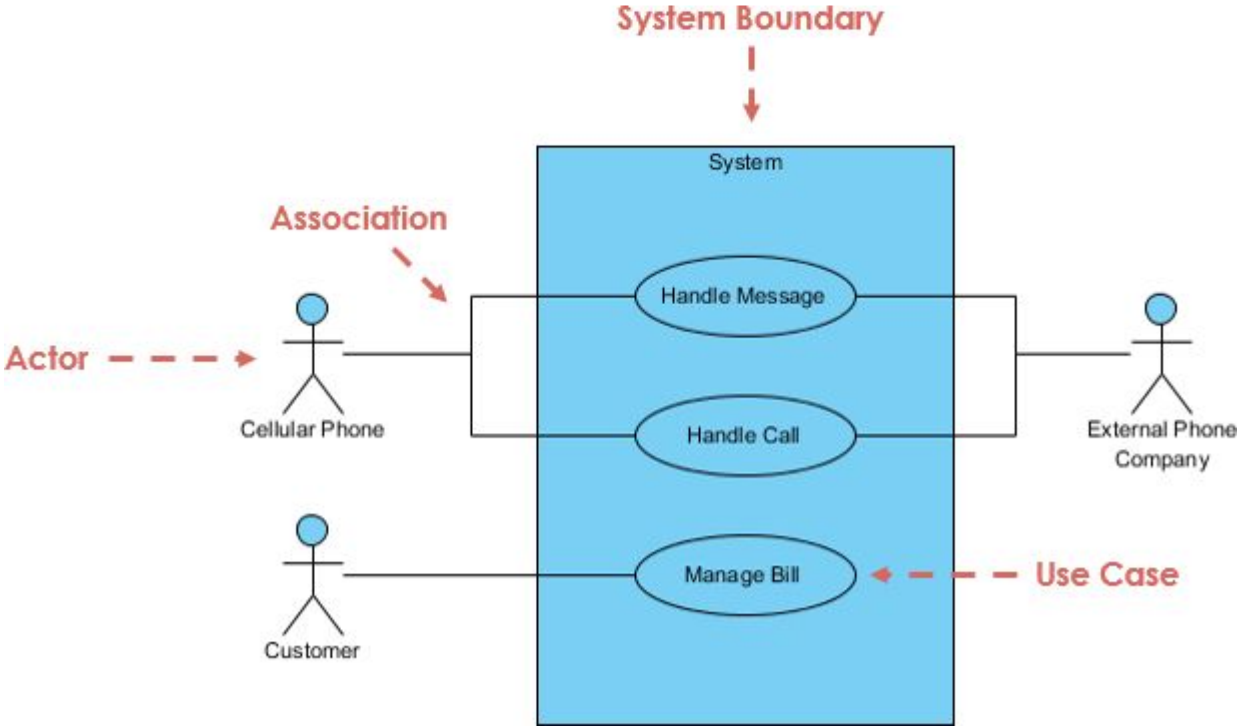
Use case modeling

- A use case can be taken as a simple description of what a user expects from a system in that interaction

- Each use case represents a discrete task that involves external interaction with a system

Interaction models

Use case modeling



Interaction models

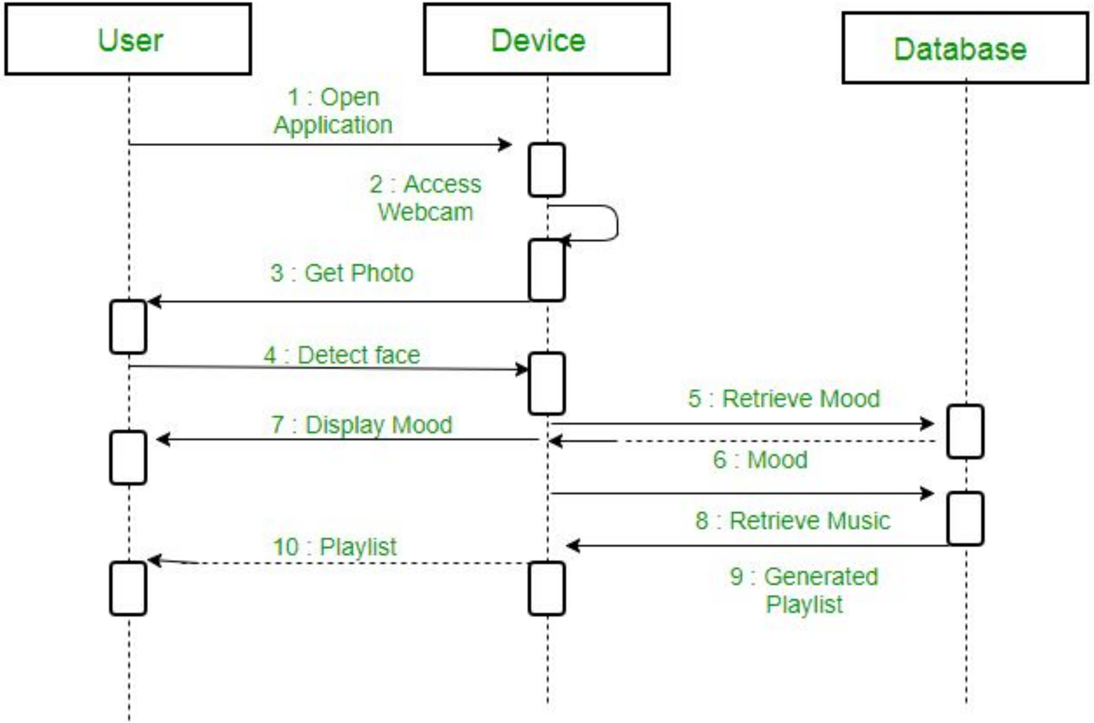
Sequence diagrams

- Sequence diagrams in the UML are primarily used to model the interactions between the actors and the objects in a system and the interactions between the objects themselves

- The UML has a rich syntax for sequence diagrams, which allows many different kinds of interaction to be modeled

Interaction models

Sequence diagrams



Structural models

Structural models of software display the organization of a system in terms of the components that make up that system and their relationships

Structural models may be

- static models, which show the organization of the system design, or
- dynamic models, which show the organization of the system when it is executing

Interaction models & Structural models

Structural models

Class diagrams

- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes

- Loosely, an object class can be thought of as a general definition of one kind of system object

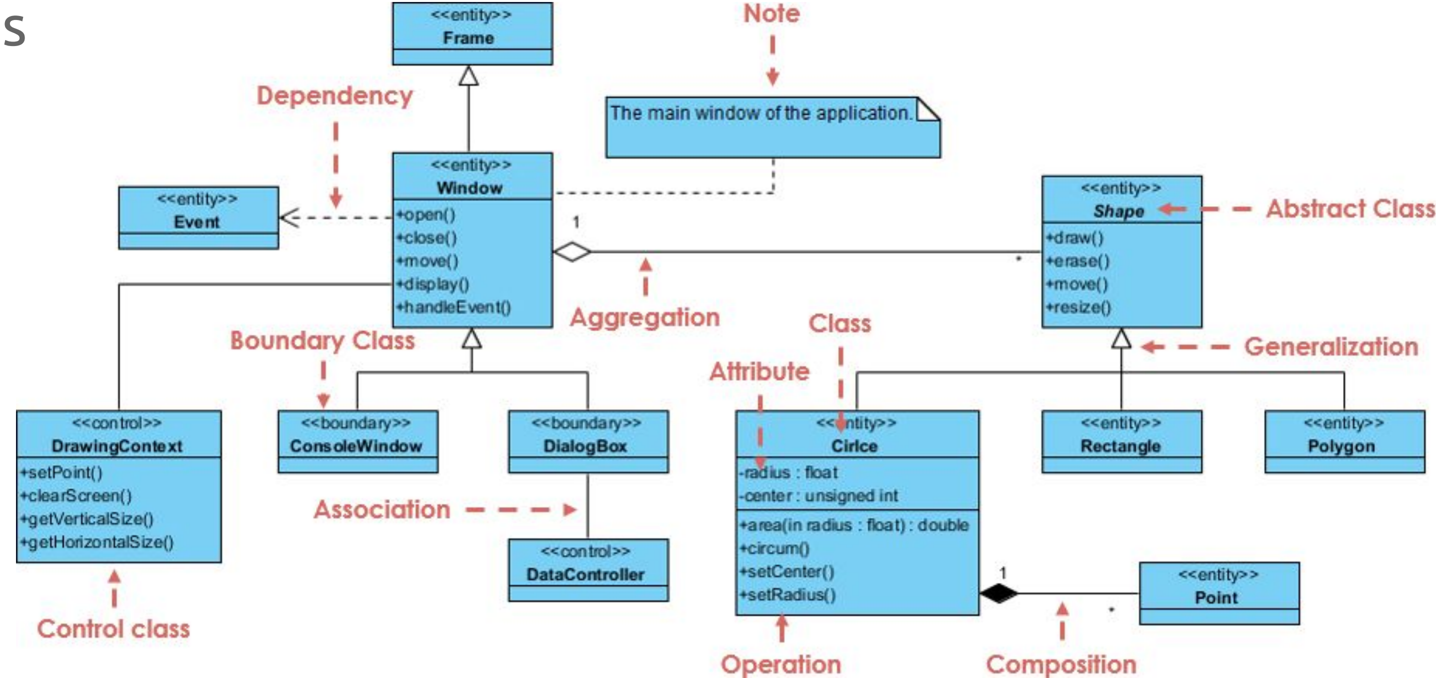
Structural models

Class diagrams

- An association is a link between classes indicating that some relationship exists between these classes
- Consequently, each class may have to have some knowledge of its associated class

Structural models

Class diagrams



Agenda: Lesson #04 - Software Engineering - Lecture

1 Introduction & Context models

2 Interaction models & Structural models

3 Behavioral models

4 Model-driven engineering

Behavioral models

Behavioral models are models of the dynamic behavior of a 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

Behavioral models

These stimuli may be either data or events:

- Data becomes available that has to be processed by the system. The availability of the data triggers the processing

- An event happens that triggers system processing. Events may have associated data, although this is not always the case

Behavioral models

Data-driven modeling

- Data-driven models show the sequence of actions involved in processing input data and generating an associated output

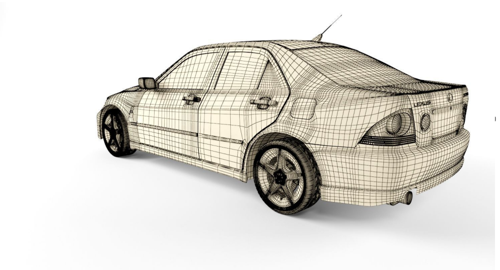
- They can be used during the analysis of requirements as they show end-to-end processing in a system

Behavioral models

Event-driven modeling

- Event-driven modeling shows how a system responds to external and internal events

- It is based on the assumption that a system has a finite number of states and that events (stimuli) may cause a transition from one state to another



Behavioral models

Model-driven engineering

- Model-driven engineering (MDE) is an approach to software development whereby models rather than programs are the principal outputs of the development process (Brambilla, Cabot, and Wimmer 2012)

- The programs that execute on a hardware/software platform are generated automatically from the models

Agenda: Lesson #04 - Software Engineering - Lecture

1 Introduction & Context models

2 Interaction models & Structural models

3 Behavioral models

4 Model-driven engineering

Model-driven engineering

Model-driven architecture (Mellor, Scott, and Weise 2004; Stahl and Voelter 2006) is a model-focused approach to software design and implementation that uses a subset of UML models to describe a system

Model-driven engineering

The MDA method recommends that three types of abstract system model should be produced:

- A computation independent model (CIM) CIMs model the important domain abstractions used in a system and so are sometimes called domain models. You may develop several different CIMs, reflecting different views of the system.

Model-driven engineering

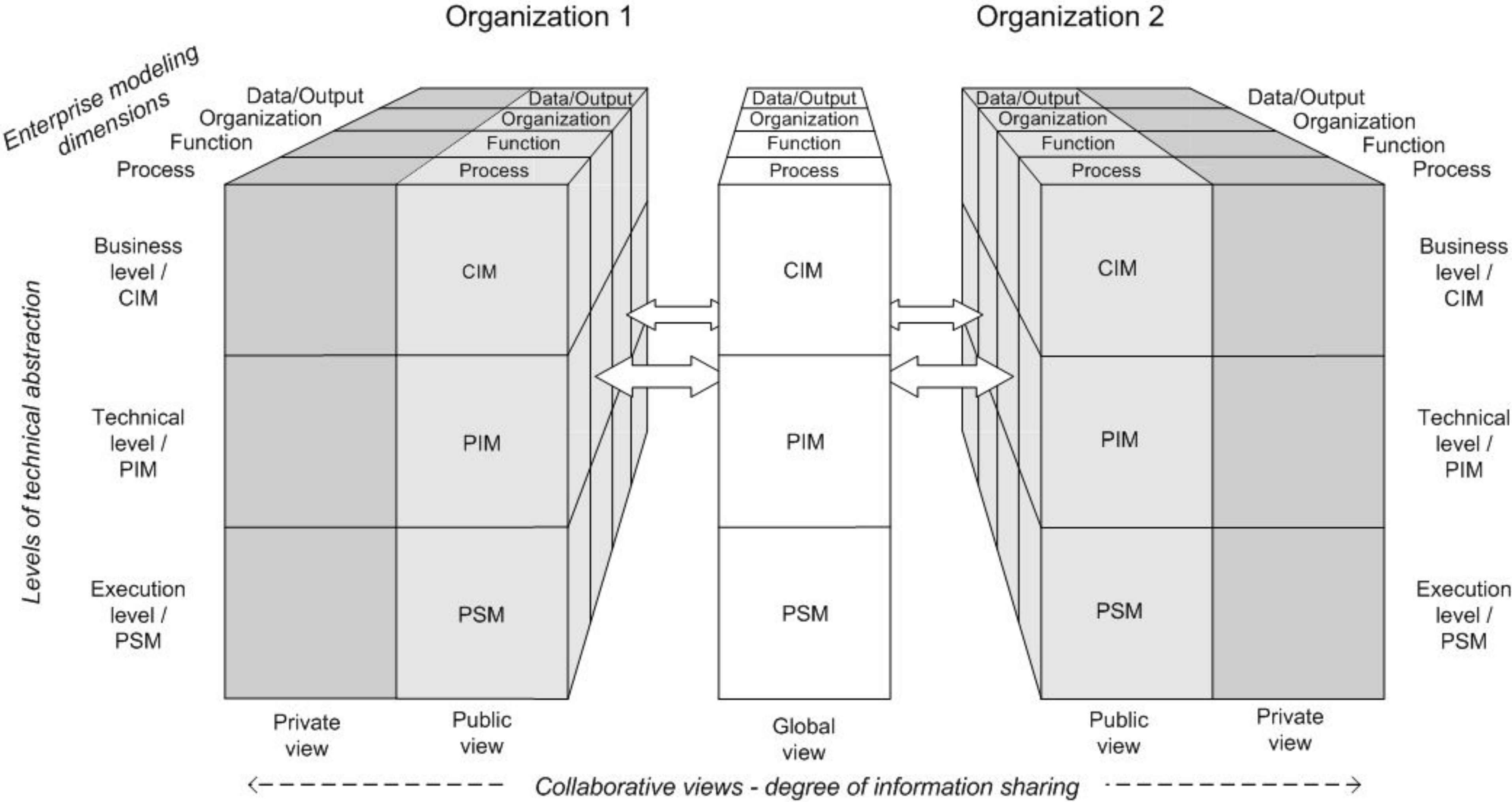
The MDA method recommends that three types of abstract system model should be produced:

- A platform-independent model (PIM) PIMs model the operation of the system without reference to its implementation. A PIM is usually described using UML models that show the static system structure and how it responds to external and internal events

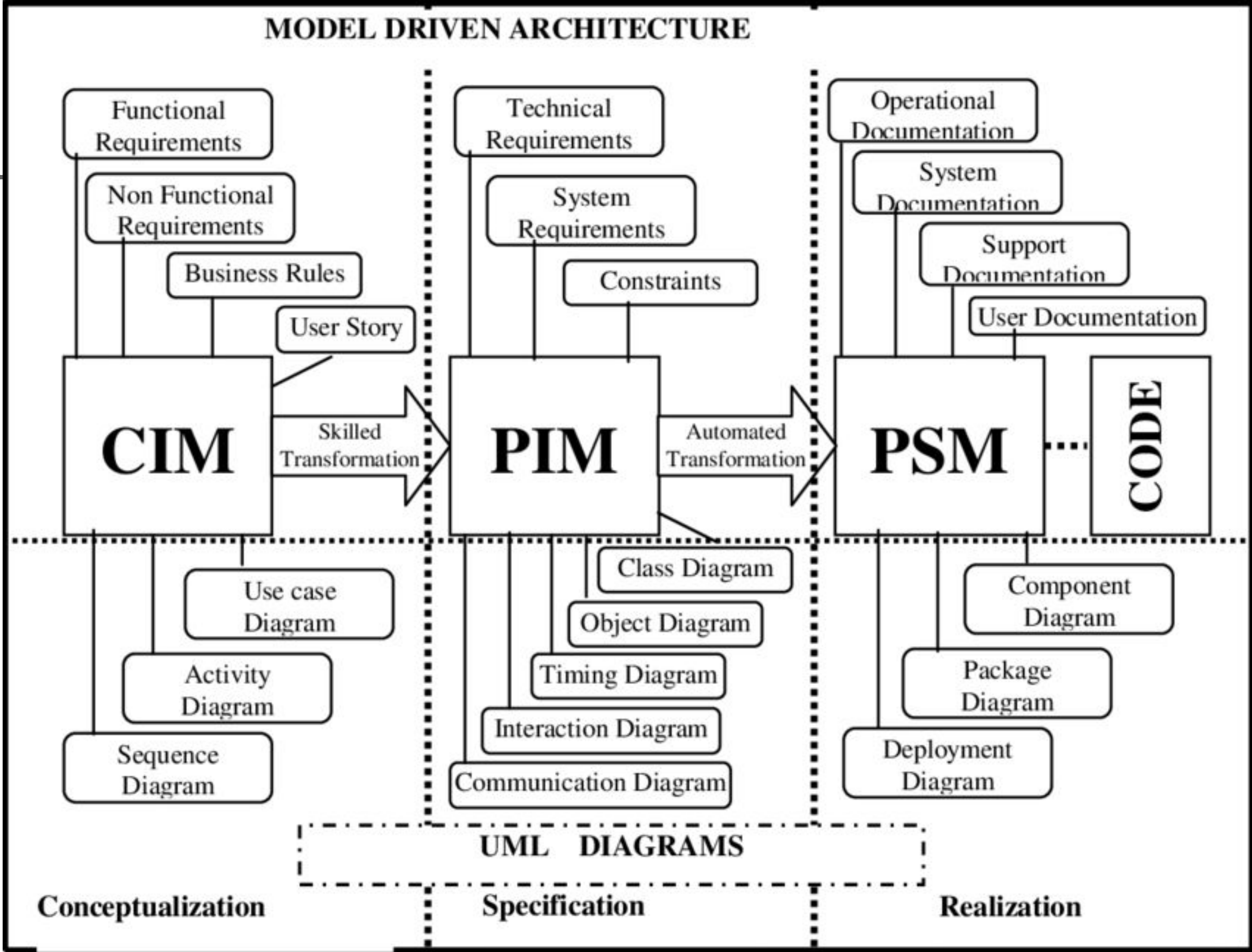
Model-driven engineering

The MDA method recommends that three types of abstract system model should be produced:

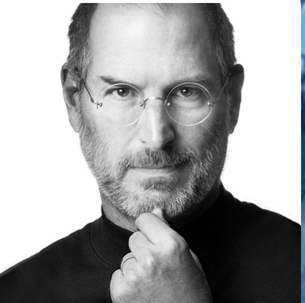
- Platform-specific models (PSM) PSMs are transformations of the platform-independent model with a separate PSM for each application platform. In principle, there may be layers of PSM, with each layer adding some platform-specific detail


Organization 1
Organization 2
Enterprise modeling dimensions
Data/Output
Organization
Function
Process
Data/Output
Organization
Function
Process
Data/Output
Organization
Function
Process
Data/Output
Organization
Function
Process
Data/Output
Organization
Function
Process
Levels of technical abstraction
Business level / CIM
Technical level / PIM
Execution level / PSM
CIM
PIM
PSM
CIM
PIM
PSM
CIM
PIM
PSM
Business level / CIM
Technical level / PIM
Execution level / PSM
Private view
Public view
Global view
Public view
Private view
Collaborative views - degree of information sharing

MD engineering



10 to
3 to 1
or
Pixel
Perfect
Prototype



Agenda: Lesson #04 - Software Engineering - Lecture

Q & A