System Modelling and UML

Shihong Huang
shihong@fau.edu
Department of Computer & Electrical Engineering
and Computer Science
Florida Atlantic University
Boca Raton FL 33431

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

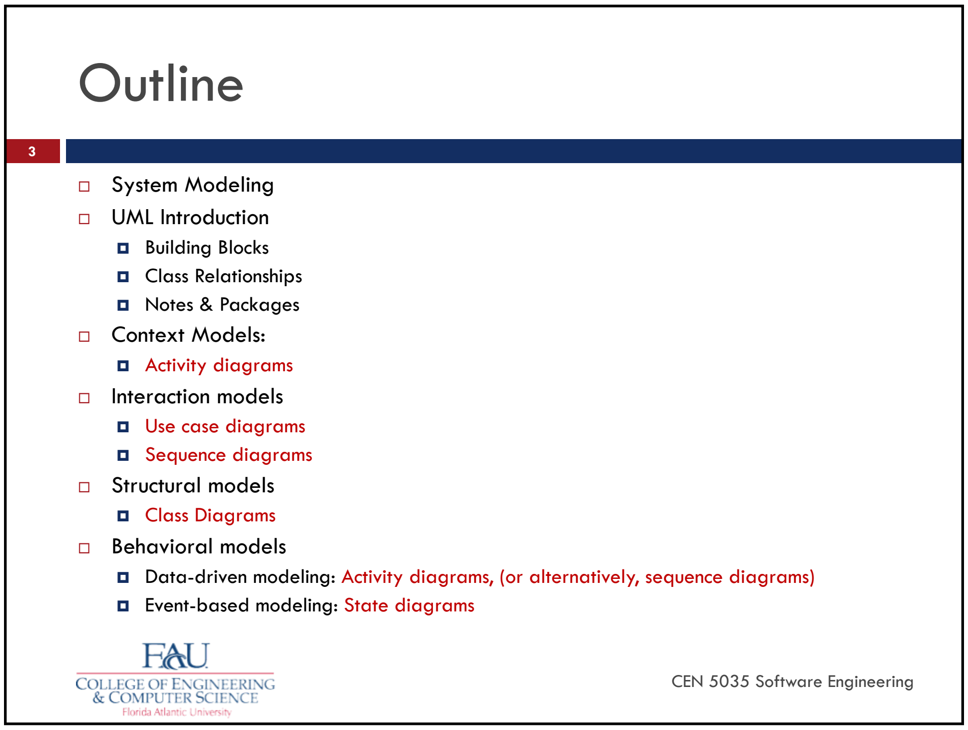
CEN 5035 Software Engineering

Motivation

2

- The UML is a large visual modeling language with many details to remember
- The UML plays an increasingly important role in representing system modeling and software architecture

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University



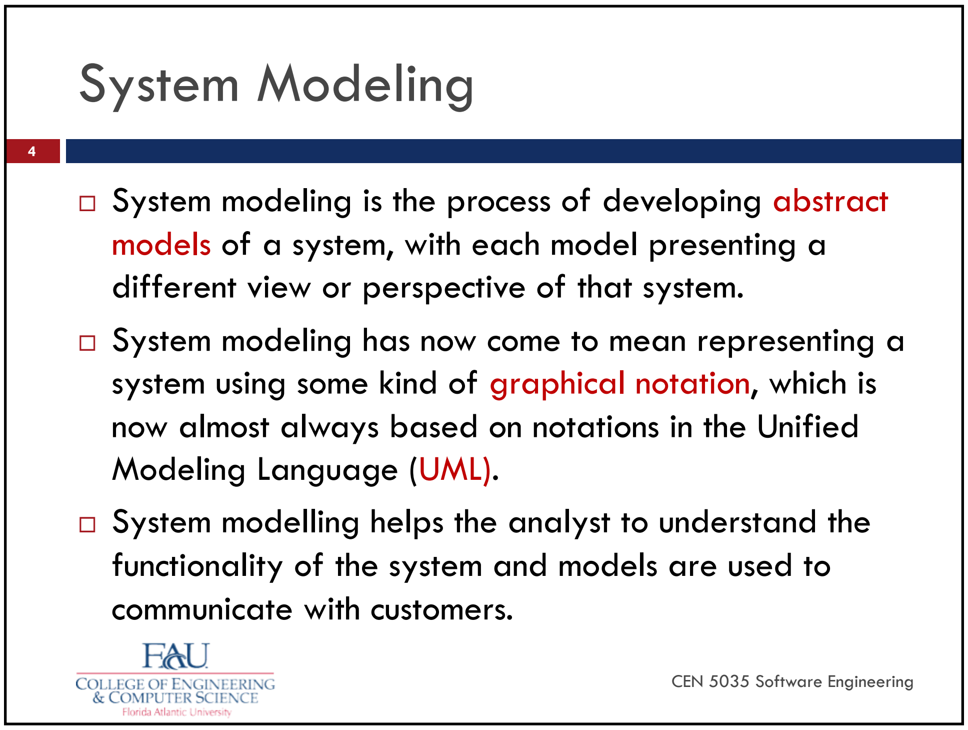
Outline

3

- System Modeling
- UML Introduction
  - Building Blocks
  - Class Relationships
  - Notes & Packages
- Context Models:
  - Activity diagrams
- Interaction models
  - Use case diagrams
  - Sequence diagrams
- Structural models
  - Class Diagrams
- Behavioral models
  - Data-driven modeling: Activity diagrams, (or alternatively, sequence diagrams)
  - Event-based modeling: State diagrams

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering



System Modeling

4

- 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 has now come to mean representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

System Perspectives

- An external perspective, where you model the context or environment of the system.
- An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
- A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.
- A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

UML Diagram Types

- Activity diagrams, which show the activities involved in a process or in data processing .
- Use case diagrams, which show the interactions between a system and its environment.
- Sequence diagrams, which show interactions between actors and the system and between system components.
- Class diagrams, which show 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.

UML Introduction –1

7

Modeling Software with the UML

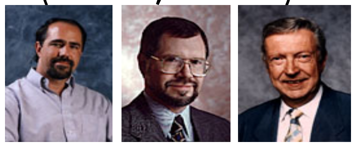
- Specifying
- Constructing
- Visualizing
- Documenting

CEN 5035 Software Engineering

UML Introduction –2

8

- A successor to the OOA&D methods of the late 1980s and the early 1990s
- The ***de facto*** industry standard
- Unifies the "three amigos":
  - Grady Booch (Rational; Ada work)
  - Jim Rumbaugh (GE; OMT)
  - Ivar Jacobson (Ericcson; switches)



CEN 5035 Software Engineering

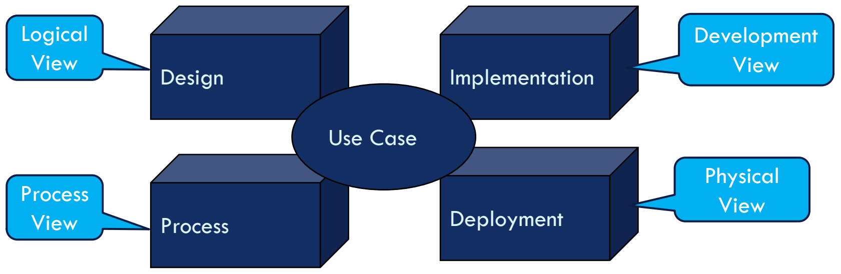
UML Introduction –3

- The mixing and matching of the three approaches became Unified Method
  - January 1997, UML V1.0
  - November 1997, UML V1.1, standard by Object Management Group (OMG)
  - Current version UML 2.5 (released October 2012)

UML Introduction –4

- Primarily a graphical notation technique
- Not a design method or a process
- Related to 4+1 architecture views:

Philippe Kruchten for "describing the architecture of software intensive systems, based on the use of multiple, concurrent views



UML Introduction –5

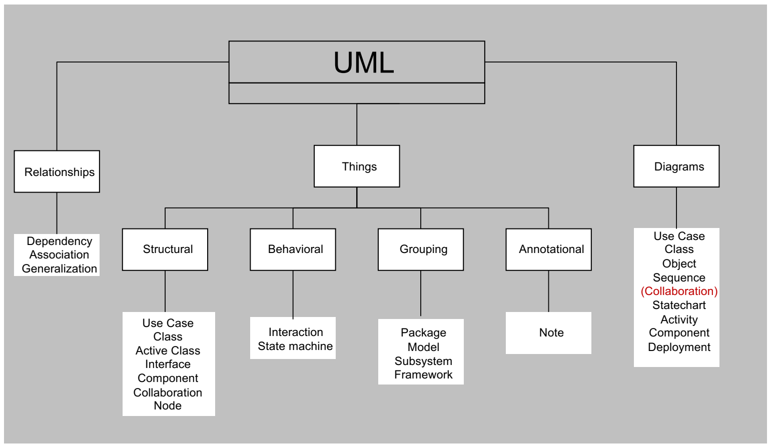
- The ***Rational Unified Process*** is a merged software process based on UML:
  - Use-case driven
  - Architecture-centric
  - Iterative and incremental
- Four Phases:
  - Inception
  - Elaboration
  - Construction
  - Transition
- Five work flows:
  - Requirements
  - Analysis
  - Design
  - Implementation
  - Testing

Building Blocks –1

- Things are first-class abstractions:
  - Structural
  - Behavioral
  - Grouping
  - Annotational
- Relationships tie things together
- Diagrams group collections of interesting things

Building Blocks –2

13

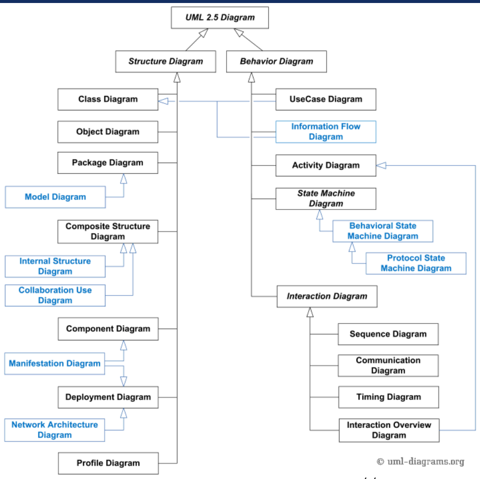


COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

UML 2.5 Diagrams

14



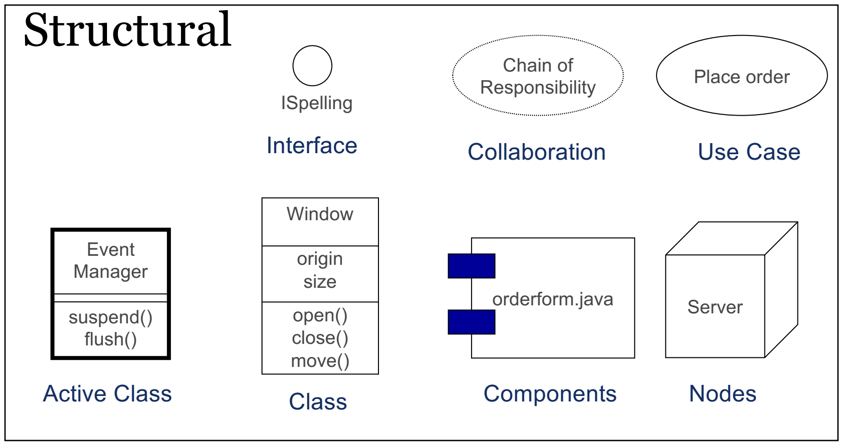
Note: items in blue are not part of official taxonomy of UML 2.5 diagrams.

https://www.uml-diagrams.org/uml-25-diagrams.html

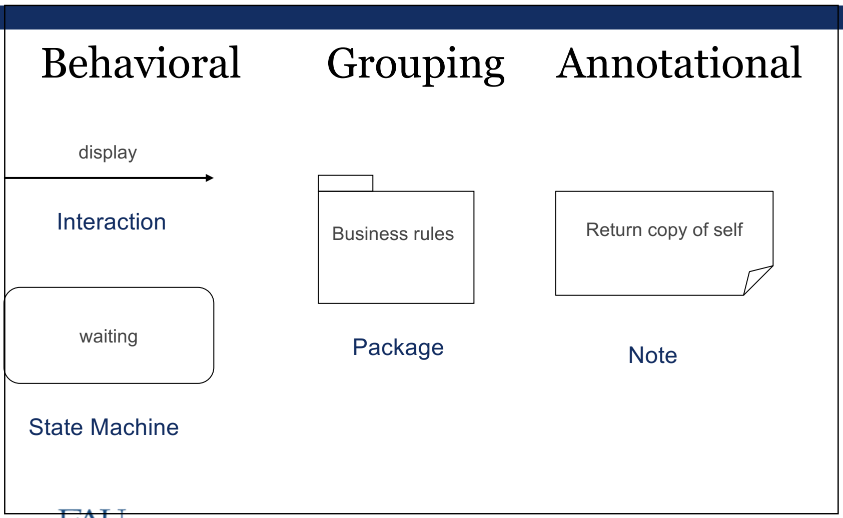
COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

Building Blocks –3: Things



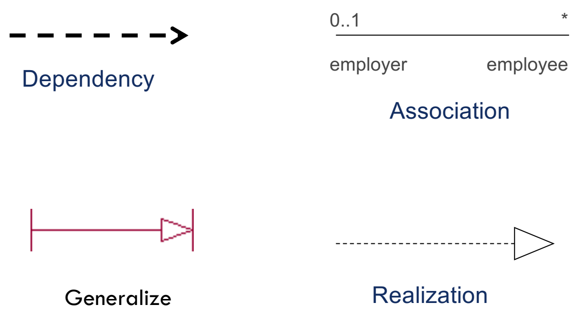
Building Blocks –4: Things



Building Blocks –5

17

Relationships



Dependency

Association

Generalize

Realization

Building Blocks –6

18

Diagrams

- Class
- Object
- Use Case
- Sequence
- Collaboration (removed from UML 2.x)
- Statechart
- Activity
- Component
- Deployment

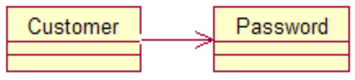
Class Relationships –1

Associations

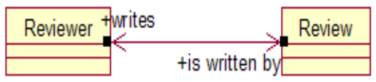
- A structural connection between classes
- An associations can contain *roles*, which are the faces class represented to other
- *Multiplicity* indicates numbers of objects associated with each class
  - fixed value (e.g. 1 or 3)
  - * (many)
  - range of values (0..1, 3..*)
  - Set of values (1,3,5,7)

Class Relationships –2

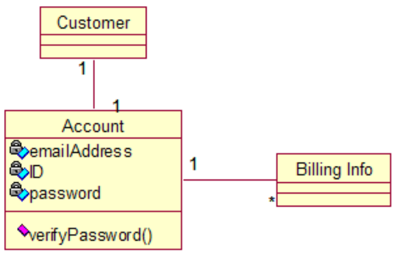
Association Examples:



One-way navigation between classes



Associate Roles

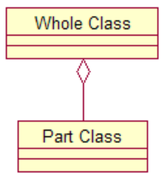


Association Multiplicity

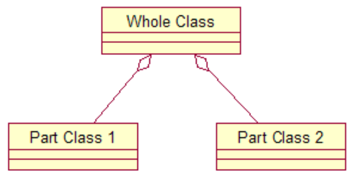
Class Relationships –3

Aggregation

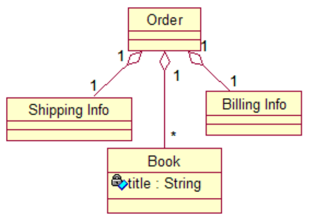
- A special kind of association
- A "whole/part" relationship
- Example of Aggregation:



Notation



Aggregating multiple classes

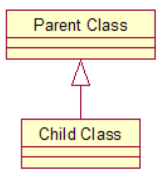


Sample aggregation

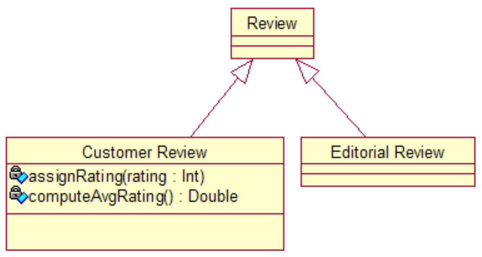
Class Relationships –4

Generalization

- A relationship between a general class (super/parent class) and a more specific version of that class (the sub/child class)
- Single inheritance vs. multiple inheritance

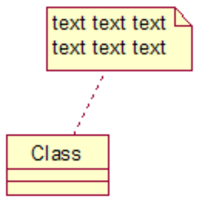


Notation

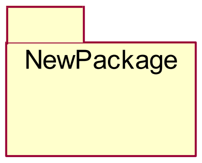


Notes and Packages

- Notes: adornments
- Packages: a group of pieces of a model



Note notation



Package notation

Context Models

Context Models

25

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.

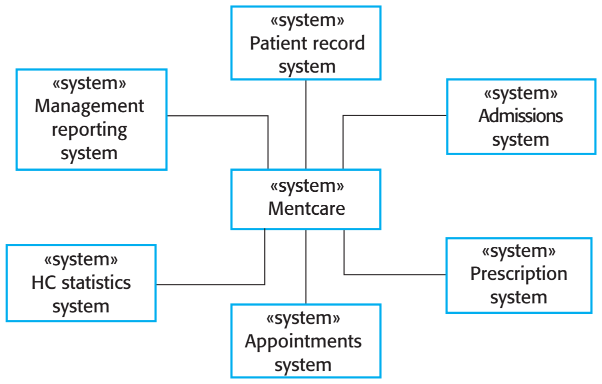
System Boundaries

26

- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

The Context of the Mentcare System

27


«system» Patient record system
«system» Management reporting system
«system» Admissions system
«system» Mentcare
«system» HC statistics system
«system» Prescription system
«system» Appointments system

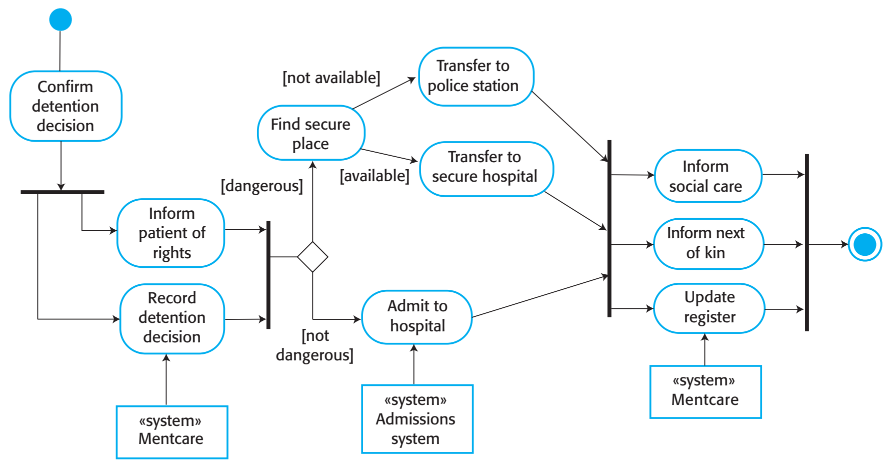
Process Perspective

28

- Context models simply show the other systems in the environment, not how the system being developed is used in that environment.
- Process models reveal how the system being developed is used in broader business processes.
- UML activity diagrams may be used to define business process models.

Process Model of Involuntary Detention

29



FAU College of Engineering & Computer Science Florida Atlantic University

30

Interaction models

FAU College of Engineering & Computer Science Florida Atlantic University

Interaction Models

31

- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and sequence diagrams may be used for interaction modeling.

Use Case Diagrams –1

32

- UML use-case driven nature
- Use case model allow the stakeholders to agree on what the system should do
- Severs as the foundation for all other development
- Actor:
  - A role use play with regard to a system
  - An entity, e.g. another system, DB that is outside the system

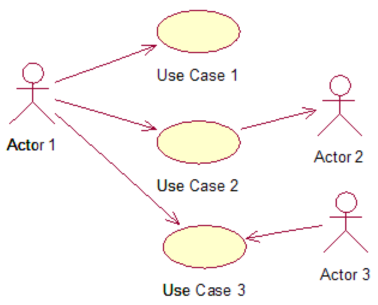
Use Case Diagrams –2

- Use case:
  - A sequence of actions that an actor performs within a system to achieve a particular goal
  - Describe *what* the system need to do, without specifying *how* the system will perform
- Total set of actors with a use case model should reflect everything that needs to exchange info with the system
- Total set of use cases should capture all the functional requirements that system stakeholders have put forth

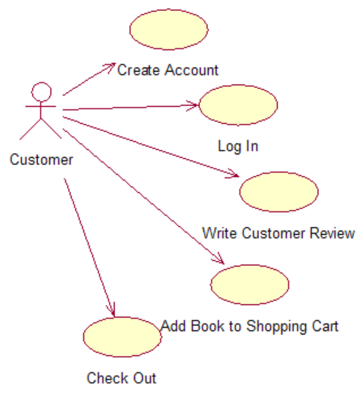
NewUseCase

Use Case Diagrams –3

Examples



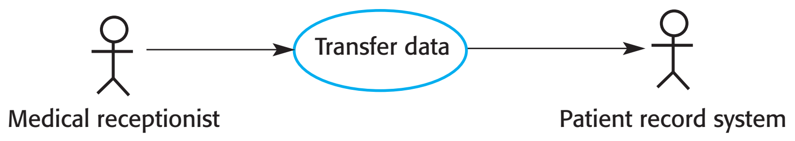
Use Case diagram



Sample use case diagram

Transfer-data Use Case

- A use case in the Mentcare system



Tabular Description of the 'Transfer data' Use-case

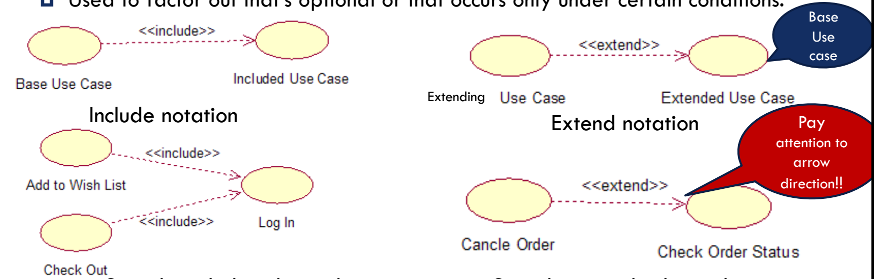
MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Use Case Diagrams –4

37

Include & Extend

- *Include* – one use case explicitly includes the behavior of another use case at a specific point within a course of action. Common features
- *Extend: (extending use case is optional)*
  - a base use case implicitly includes the behavior of another use case at one or more specific points.
  - Used to factor out that's optional or that occurs only under certain conditions.



Include notation

Extend notation

Sample include relationship

Sample extend relationship

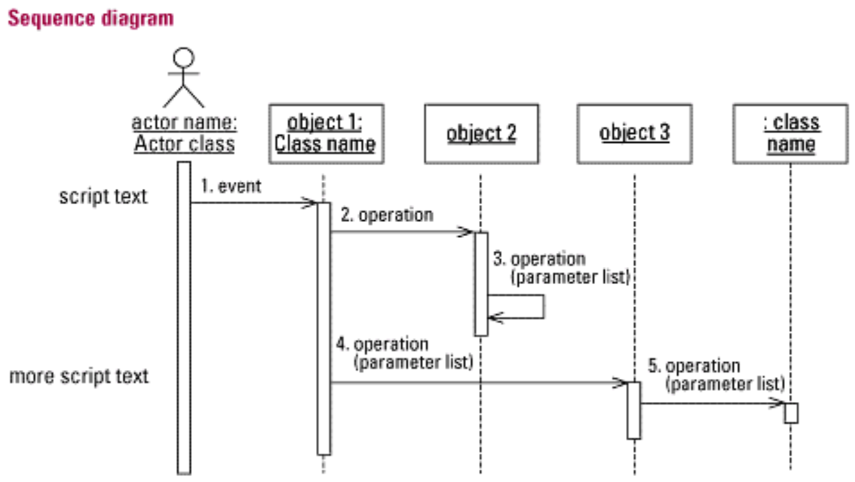
Sequence Diagrams –1

38

- Focus on the time ordering of the message that go back and forth between objects
- Are associated with the design workflow within the RUP
- The development team uses sequence diagrams in deciding where to assign operations on classes, based on the methods that they assign to objects on the diagram

Sequence Diagrams –2

39



FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

Collaboration Diagrams –1

40

Note: Collaboration diagram is removed from UML 2.x

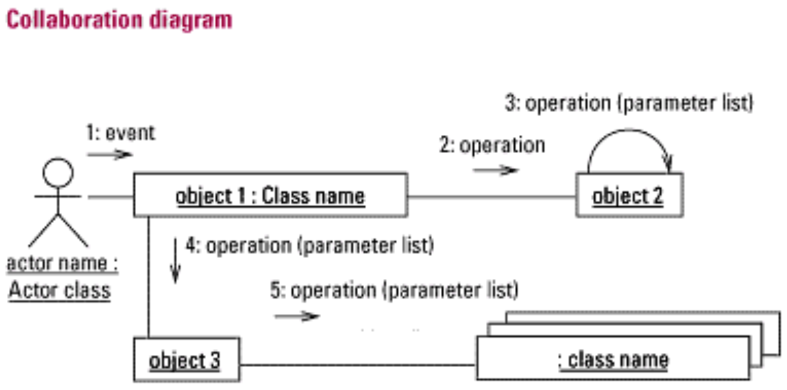
- Focuses on the organization of the objects that participates in a given set of messages
- Collaboration diagrams and sequence diagrams show the same information
- In Rational Rose, toggle switch F5
- Collaboration diagrams focus on structural information, whereas the sequence diagrams focus on time ordering
- Show objects and messages, but no lifeline or focus-of-control rectangles

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

Collaboration Diagrams –2

- Example of collaboration diagram



Structural models

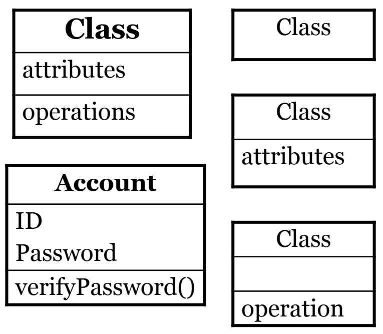
Structural Models

- 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 structure of the system design, or dynamic models, which show the organization of the system when it is executing.
- You create structural models of a system when you are discussing and designing the system architecture.
- UML class diagrams are used for modeling static structure of the object classes in a software system

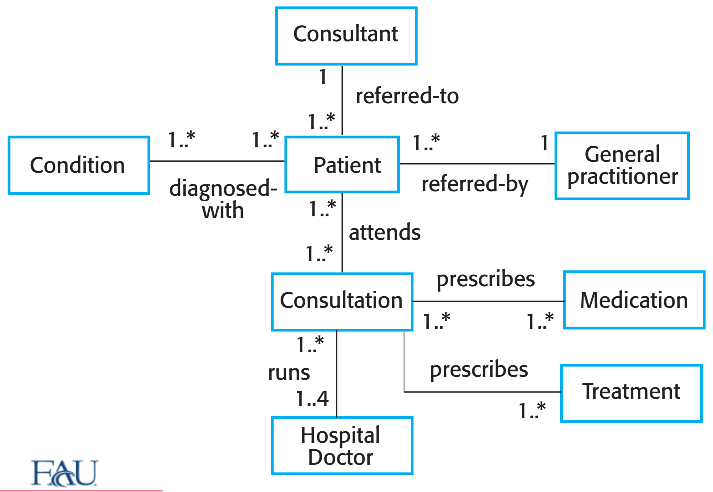
Class Diagrams

- Classes
  - A collection of objects with same characteristics
  - Identity – a name
  - Defines attributes of objects of that class
  - Operation – a service that called by an object
  - A *method* is an implementation of the service

- UML notation



Example: Classes and associations in the MHC-PMS



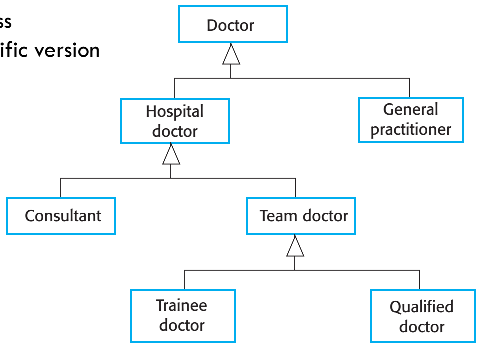
Generalization

- In modeling systems, it is often useful to examine the classes in a system to see if there is scope for generalization. If changes are proposed, then you do not have to look at all classes in the system to see if they are affected by the change.
- In object-oriented languages, such as Java, generalization is implemented using the class inheritance mechanisms built into the language.
- In a generalization, the attributes and operations associated with higher-level classes are also associated with the lower-level classes.
- The lower-level classes are subclasses inherit the attributes and operations from their superclasses. These lower-level classes then add more specific attributes and operations.

Example: A Generalization Hierarchy

47

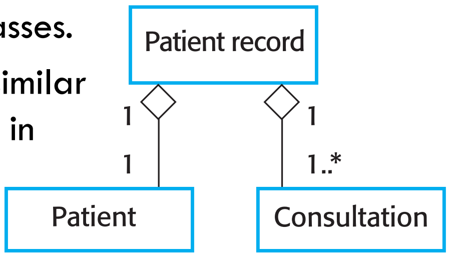
A relationship between a general class (super/parent class) and a more specific version of that class (the sub/child class)



Aggregation Association

48

- An aggregation model shows how classes that are collections are composed of other classes.
- Aggregation models are similar to the part-of relationship in semantic data models.



49

Behavioral models

Behavioral Models

50

- 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.
- You can think of these stimuli as being of two types:
  - 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, although this is not always the case.

Data-Driven Modeling

51

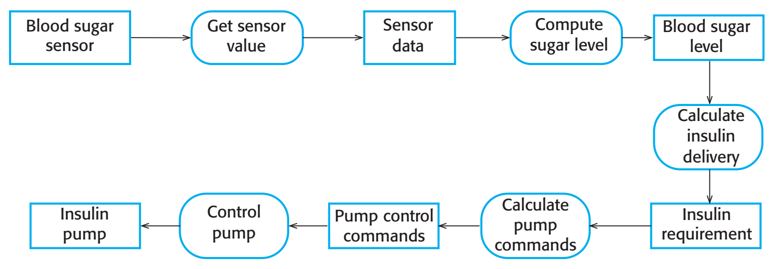
- Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing.
- Data-driven models show the sequence of actions involved in processing input data and generating an associated output.
- They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system.

COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

An Activity Model of the Insulin Pump's Operation

52

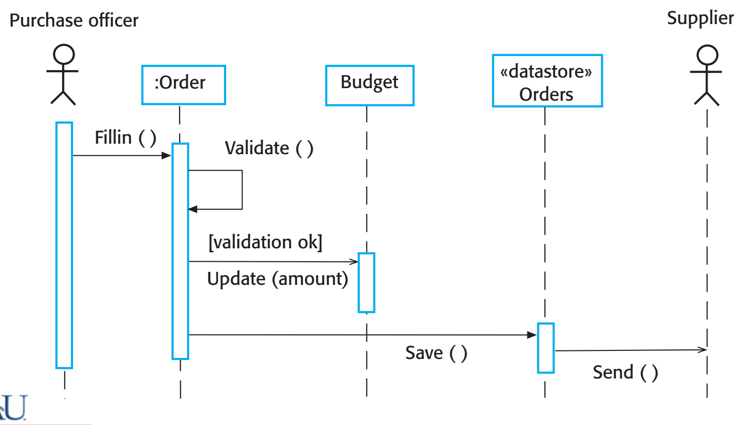


COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

Order Processing

Alternatively, use sequence diagram showing sequence of processing



Event-Driven Modeling

- Real-time systems are often event-driven, with minimal data processing. For example, a landline phone switching system responds to events such as ‘receiver off hook’ by generating a dial tone.
- 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.
- >> Finite state automata

State Machine Models

55

- These model the behaviour of the system in response to external and internal events. Trigger
- They show the system's responses to stimuli so are often used for modelling real-time systems.
- State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another.
- Statecharts are an integral part of the UML and are used to represent state machine models.

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

State Diagrams –1

56

- A *state* is a condition in which an object can be at some point during its life-time
- An object can do any or all of the following while it's particular state:
  - Perform an activity
  - Wait for an event
  - Satisfy one or more conditions
- A *transition* is a change of an object from one state to another

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

State Diagrams –2

57

- even/action:
  - action executes unconditionally
- Guard conditions
  - A Boolean expression that must evaluated to True before a given transition can fire
  - Use form: *eventName [guard condition]*
  - *eventName [guard condition]/action*: if Boolean is False, no state change occurs

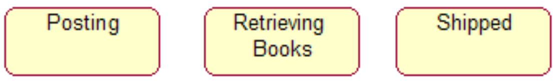
FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

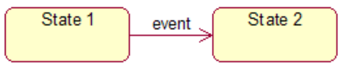
State Diagrams –3

58

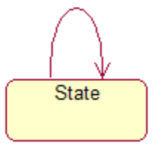
- State and transition notations



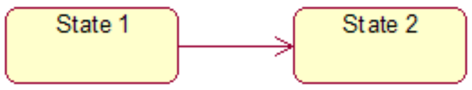
Sample states



Event "triggers" the transition



Self-transition

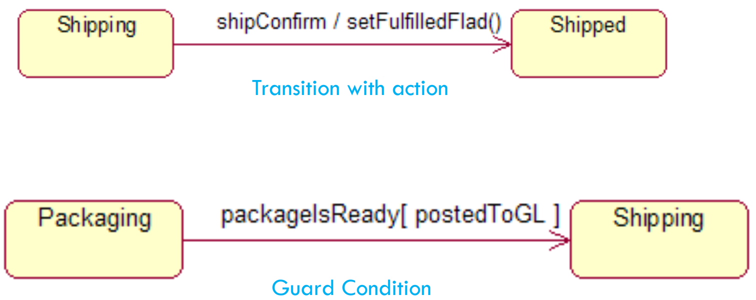


Triggerless transition

FAU COLLEGE OF ENGINEERING & COMPUTER SCIENCE Florida Atlantic University

CEN 5035 Software Engineering

State Diagrams –4

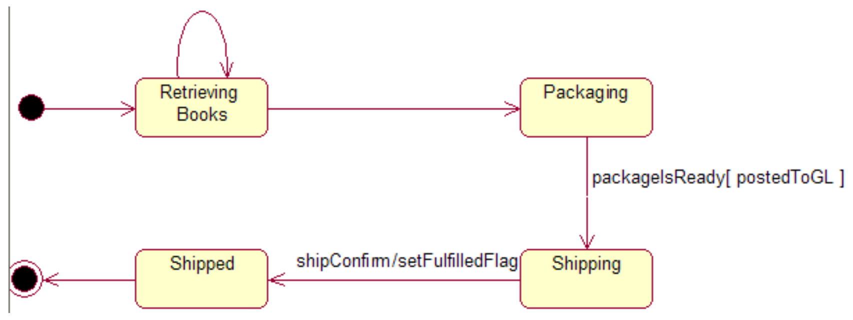

Shipping
shipConfirm / setFulfilledFlad()
Shipped
Packaging
packageIsReady[postedToGL]
Shipping

Transition with action

Guard Condition

Boolean Condition: Guard conditions affect the behavior of a state machine by enabling actions or transitions only when they evaluate to TRUE and disabling them when they evaluate to FALSE

State Diagrams –5



State diagram example

References

61

- www.uml.org
- www.omg.org
- https://www.uml-diagrams.org
- “UML Explained” by Kendall Scott (Addison-Wesley, 2001)
- Plus *many* other books
- Ian Sommerville, Software Engineering (10th Edition), 2015

COLLEGE OF ENGINEERING & COMPUTER SCIENCE
Florida Atlantic University

CEN 5035 Software Engineering

Contact

62

Dr. Shihong Huang
shihong@fau.edu
Dept. of Computer & Electrical Engineering and Computer Science

Engineering East (EE96) Room 434
Florida Atlantic University

Boca Raton FL 33431

COLLEGE OF ENGINEERING & COMPUTER SCIENCE
Florida Atlantic University

CEN 5035 Software Engineering