Unified Modeling Language (UML)

Topics

- UML
- Difference in Terminologies
- Model
- Four General Model Elements of UML
- Changes in the Model

Topics

- Baseline Diagrams
  - Use Case Diagram
  - Class Diagram
  - Activity Diagram
  - Package Diagram
  - State-Transition Diagram
  - Sequence Diagram
  - Collaboration Diagram
  - Component Diagram
  - Deployment Diagram

Unified Modeling Language

- The Unified Modeling Language (UML) is the standard language for specifying, visualizing, constructing, and documenting all the work products or artifacts of a software system.
- It unifies the notation of Booch, Rambaugh, and Jacobson, and augmented with other contributors once submitted to OMG.
- It proposes a standard for technical exchange of models and designs.

UML is NOT

- It is not a method or methodology.
- It does not indicate a particular process.
- It is not a programming language.

Difference of Terminology

UML	*Class*	*Association*	*Generalization*	*Aggregation*
Booch	Class	Uses	Inherits	Containing
Coad	Class & Object	Instance Connnection	Gen-Spec	Part-Whole
Jacobson	Object	Acquaintance Association	Inherits	Consists of
Odell	Object Type	Relationship	Subtype	Composition
Rambaugh	Class	Association	Generalizationn	Aggregation
Shlaer/Mellor	Object	Relationship	Subtype	n/a

Model

- A model is a pattern of something to be made.
- It is a representation of something in the real world.
  - They are built quicker and easier that the objects they represent.
  - They are used to simulate to better understand the objects they represent.
  - They are modified to evolve as one learns about a task or problem.
  - They are used to represent details of the models that one chooses to see, and others ignored.
  - They are representation of real or imaginary objects in any domain.

Four General Elements

- Icons
- Two-dimensional Symbols
- Paths
- Strings

Changes in the Models

- Level of Abstraction
- Degree of Formality
- Level of Detail

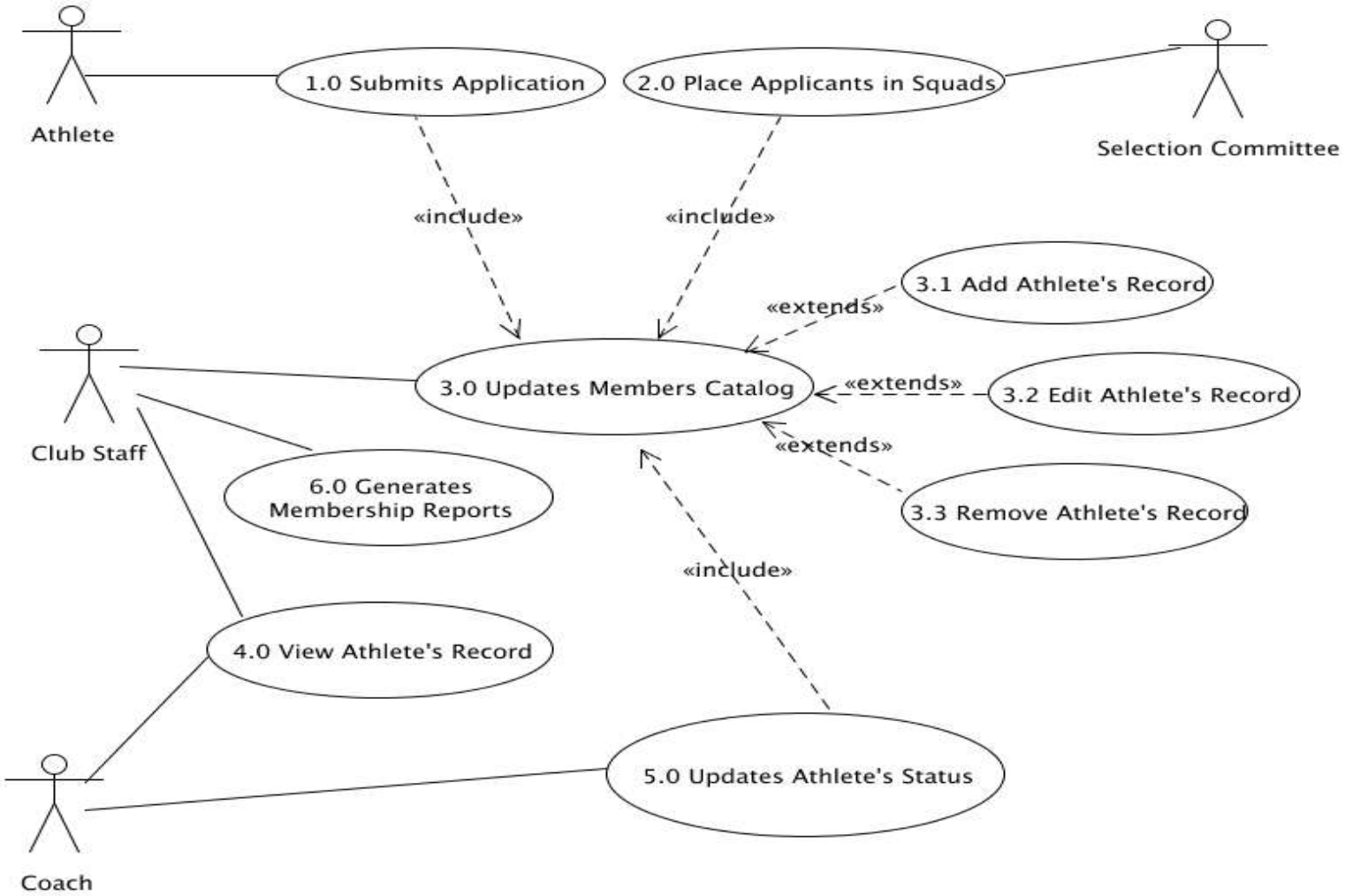
UML Baseline Diagrams

- Use Case Diagrams*
- Class Diagrams*
- Package Diagrams
- Activity Diagrams
- State-Transition Diagrams
- Sequence Diagrams
- Collaboration Diagrams
- Deployment Diagrams

Use Case Diagram

- Provides a basis of communication between end-users, stakeholders and developers in the planning of the software project.
- Attempts to model the system environment by showing the external actors and their connection to the functionality of the system.

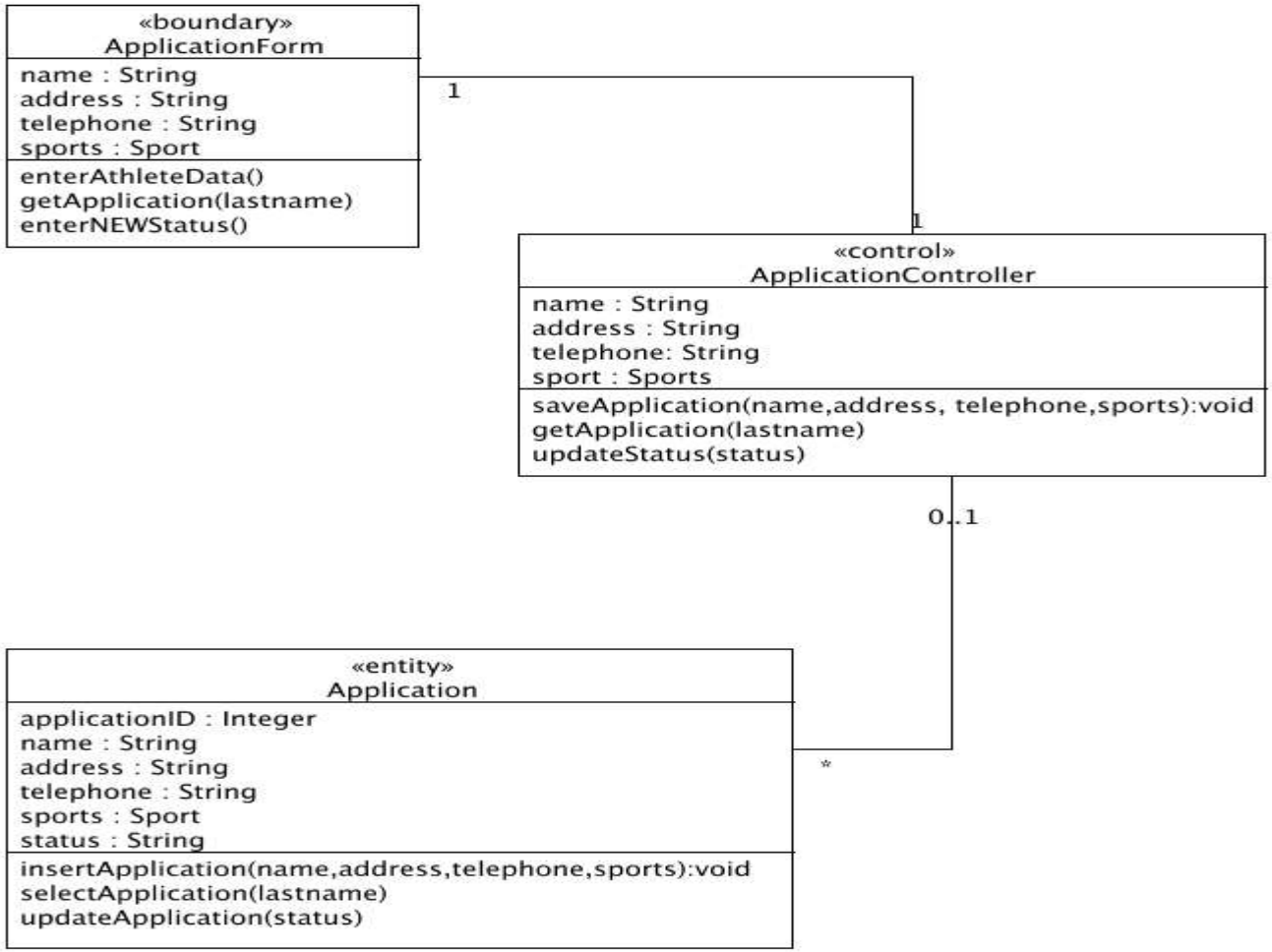
Sample Use Case Diagram



Class Diagrams

- Shows the static structure of the domain abstractions of the system.
- Describes the types of objects in the system and the various kinds of static relationships that exists among them.
- Show the attributes and operations of a class and constraints for the way objects collaborate

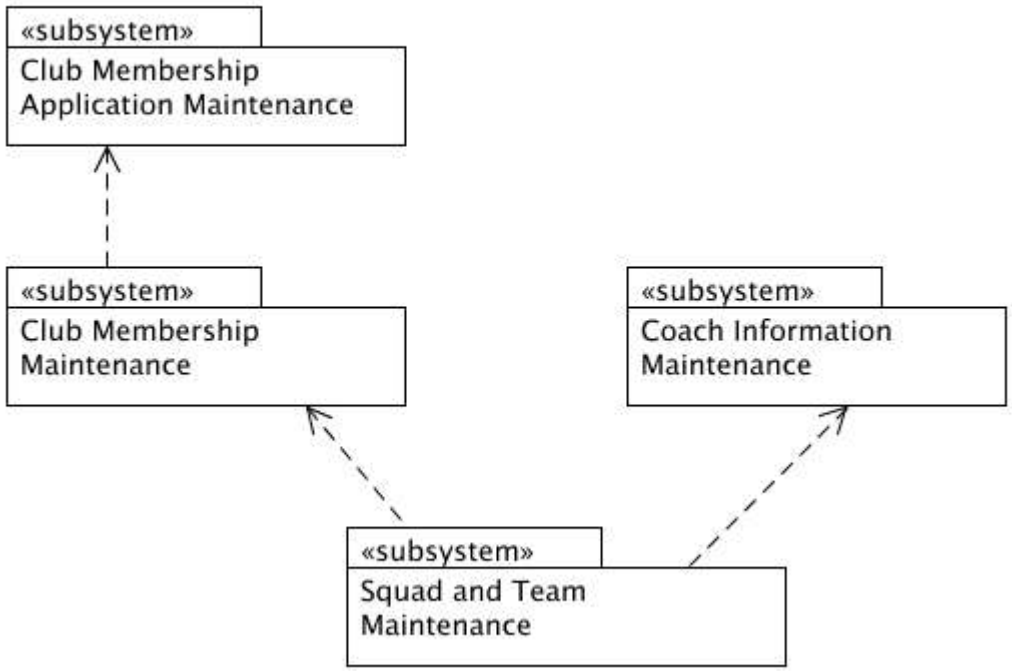
Sample Class Diagrams



Package Diagrams

- Shows the breakdown of larger systems into a logical grouping of smaller subsystems.
- Shows groupings of classes and dependencies among them
- A dependency exists between two elements if changes to the definition of one element may cause changes to the other.

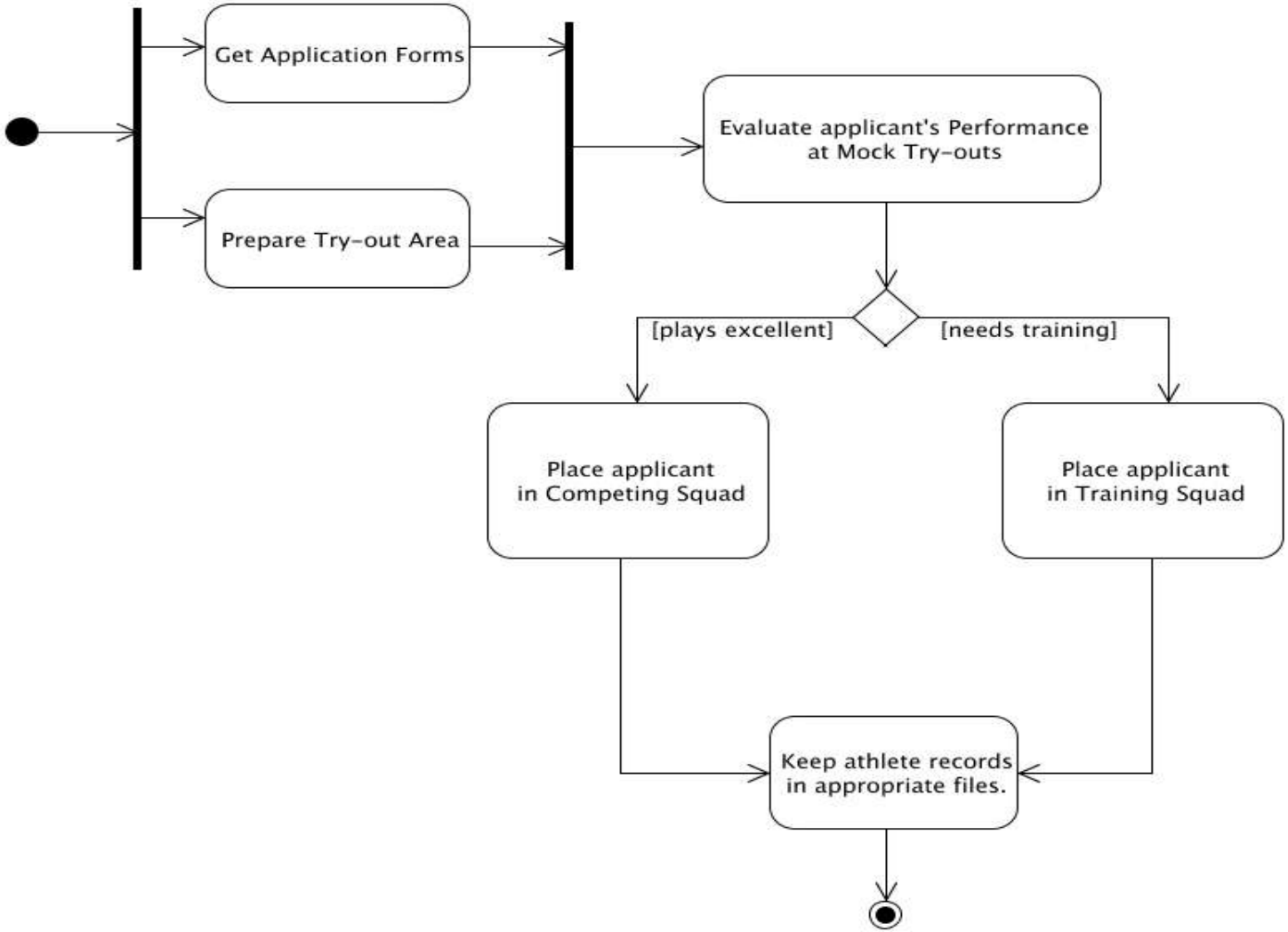
Sample Package Diagram



Activity Diagram

- Show the sequential flow of activities
  - Typically, in an operation
  - Also in a use case or event trace
- Complement the class diagram by showing the workflow of the business (a.k.a Flowchart)
- Encourage discovery of parallel processes which helps eliminate unnecessary sequences in business processes

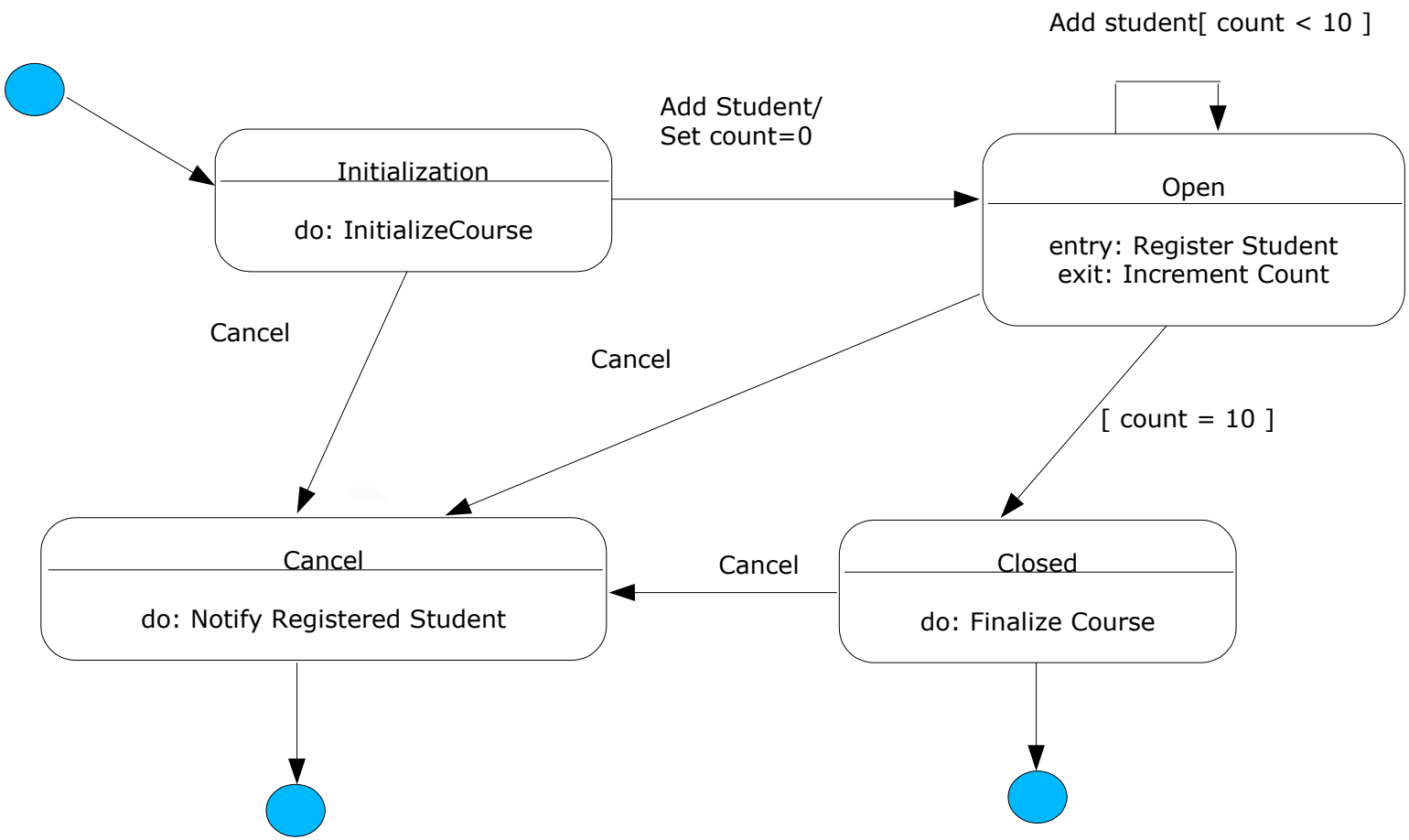
Sample Activity Diagram



State-Transition Diagrams

- Show all the possible states that objects of the class can have and which events cause them to change
- Show how the object's state changes as a result of events that are handled by the object
- Good to use when a class has complex lifecycle behavior

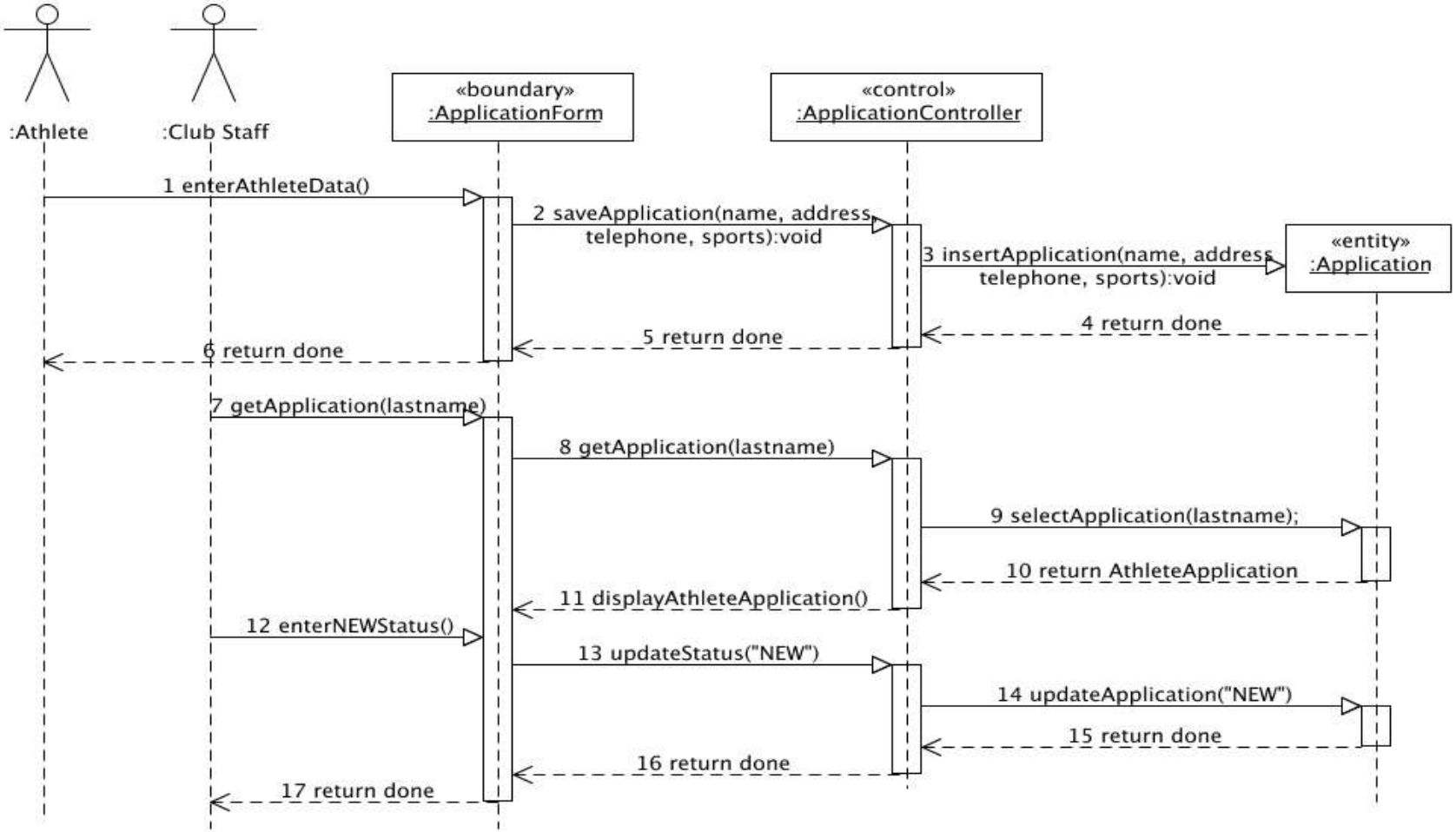
Sample State-Transition Diagram



Sequence Diagrams

- Show the dynamic collaboration between objects for a sequence of messages send between them in a sequence of time
- Time sequence is easier to see in the sequence diagram read from top to bottom
- Choose sequence diagram when only the sequence of operations needs to be shown

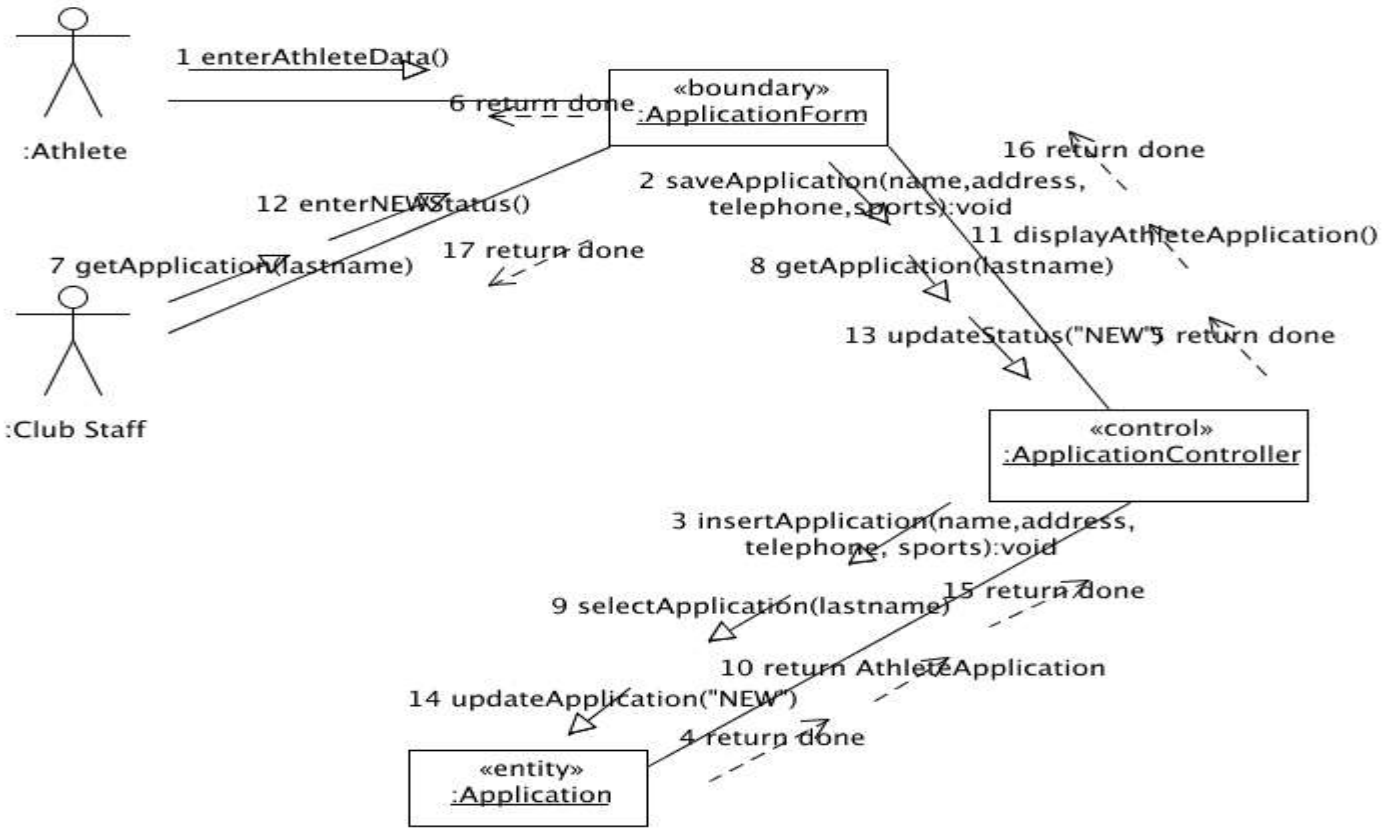
Sample Sequence Diagram



Collaboration Diagram

- Show the actual objects and their links, the “network of objects” that are collaborating
- Time sequence is shown by numbering the message label of the links between objects
- Choose collaboration diagram when the objects and their links facilitate understanding the interaction, and sequence of time is not as important

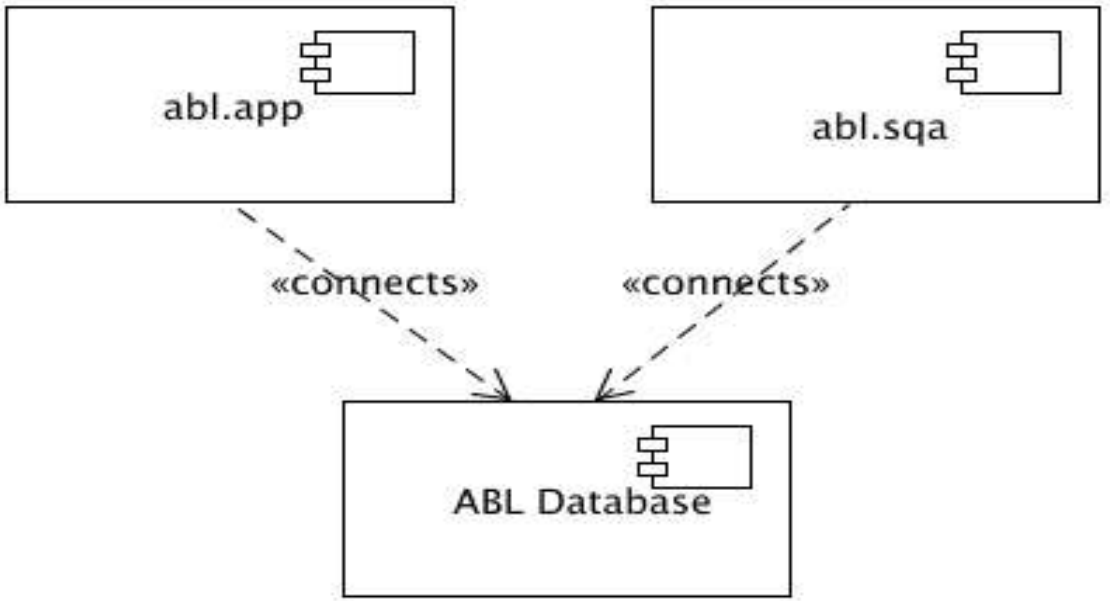
Sample Collaboration Diagram



Component Diagram

- It is used to model the components of the software.
- Components may be a source code, binary code, executable code or dynamically linked libraries.
- Components encapsulates implementation and shows a set of interfaces.

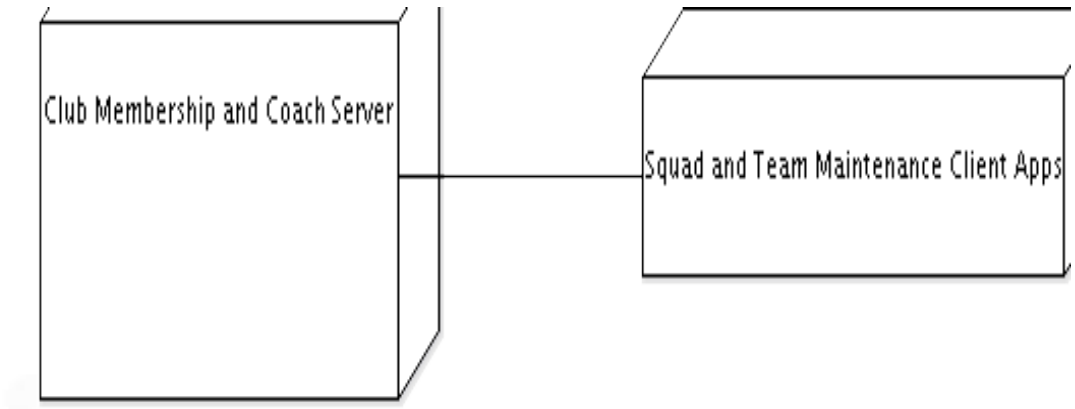
Sample Component Diagram



Deployment Diagram

- Show the physical architecture of the hardware and software of the system
- Highlight the physical relationship among software and hardware components in the delivered system
- Components on the diagram typically represent physical modules of code and correspond exactly to the package diagram

Sample Deployment



Summary

- UML
- Difference in Terminologies
- Model
- Four General Model Elements of UML
- Changes in the Model

Summary

- Baseline Diagrams
  - Use Case Diagram
  - Class Diagram
  - Activity Diagram
  - Package Diagram
  - State-Transition Diagram
  - Sequence Diagram
  - Collaboration Diagram
  - Component Diagram
  - Deployment Diagram