

# **(8-2) More with UML**

Instructor - Andrew O'Fallon

CptS 122

Washington State University



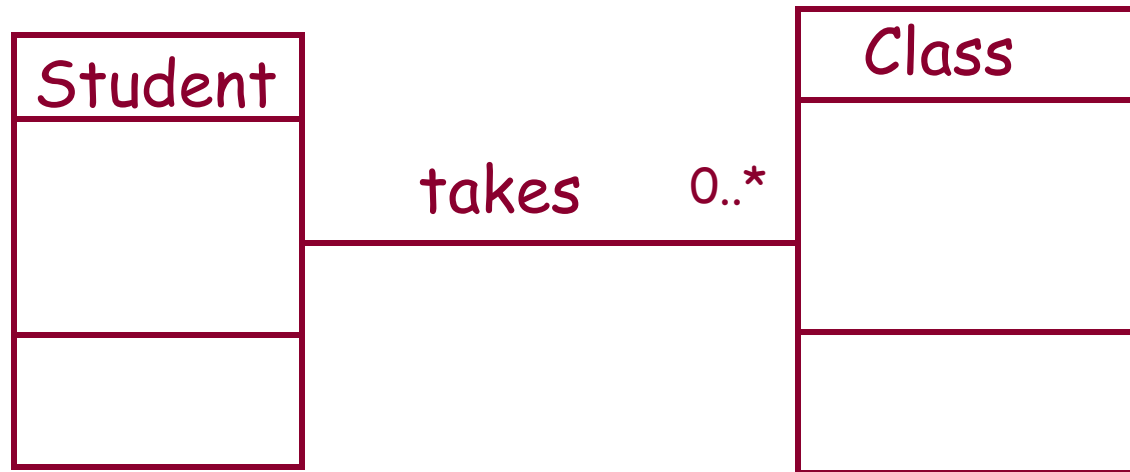
# UML as a Model

- UML is a notational syntax for expressing Object Oriented models
- Merges Booch, Rumbaugh, and Jacobson
- Not a methodology (although the Unified Process is)
- UML Models can (should be) an important source for test



# Relationships in UML Models

- Relationships in models can show a dependency between two instances
- The example shows a relationship such that a student takes 0 to many classes; We might question the many (limiting it to some max value) but we can definitely look for tests about this relationship



Built in relationships have a corresponding generic test requirements that can be identified by applying a relational test strategy to each UML diagram



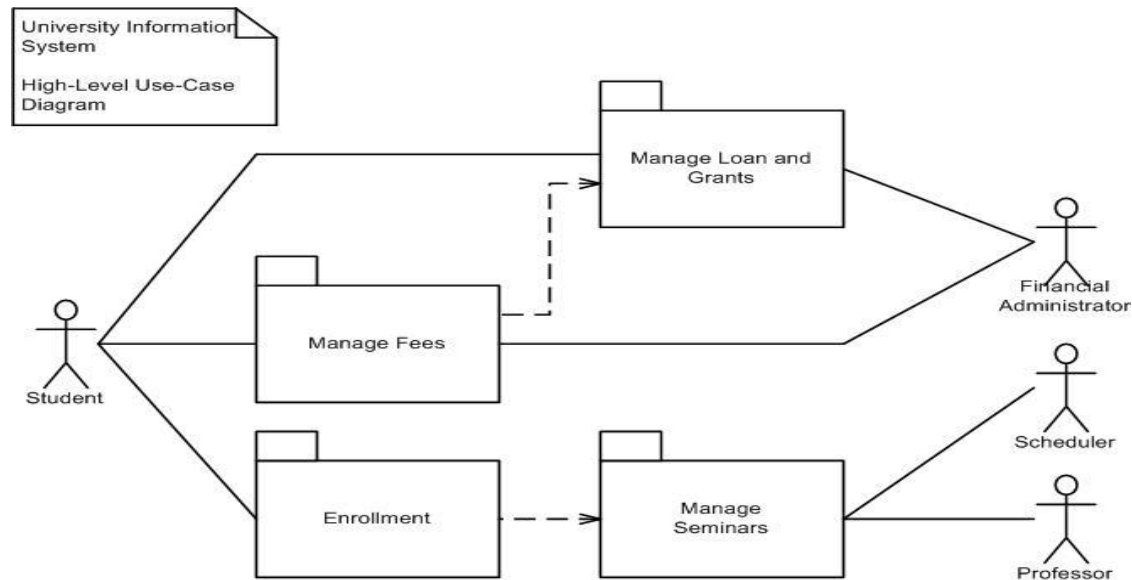
# General Purpose Elements of UML

- Organize diagrams
- Express details



# Packages and Package Diagrams

- Package
  - A group of UML diagrams and diagram elements of any kind, including other packages
- A package diagram shows the organization of packages



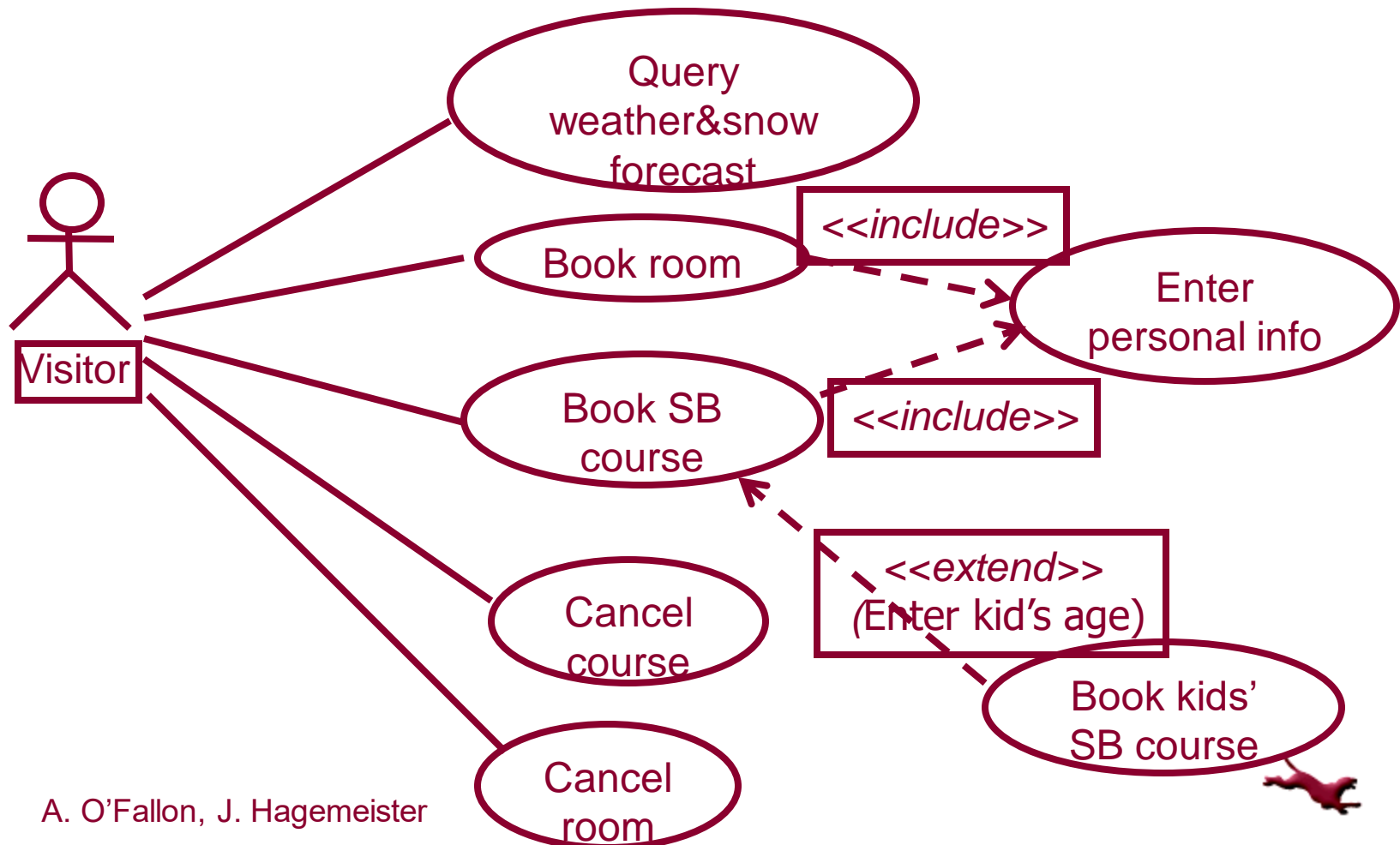
# Expressions, Constraints, Comments, and NOTES

- Expression → a string from an executable language that can be evaluated to produce a result
- Constraints → a predicate expression on an element
- Comments → a natural language constraint

A *note* is a box with a dog eared corner. It may or may not be connected to a diagram element. It contains a textual description or explanation.



# Use Case Models (1)



# Use Case Models (2)

- Use Case: Book SB course
- Precond: -
- Main flow:
  1. Visitor enters date
  2. Include (Enter personal info)
  3. (Enter kid's age)
  4. Store reservation
  5. Confirm reservation to Visitor
- Exceptional flow:
  - If number of course participants for specified date > 8, then tell visitor so and let him choose another date





# Use Case Models (3)

- Use Case: Book kids' SB course
- Precond: SB course is for a kid
- Main flow:
  1. Enter kid's age
  2. Store reservation
  3. Confirm reservation to Visitor
- Exceptional flow:
  - If course for specified date is adult course, then tell visitor so and let him choose another date
- Exceptional flow:
  - If course for specified date is kids' course, and the specified age is outside the course's age range, then tell visitor so and let him choose another date





# Use Case

- An abstraction of the system to model behavior to external interaction
- Accomplish important tasks from the user's point of view
- Represent system requirements
  - Functional
  - Allocation to classes
  - Object interaction and interfacing
  - User interfaces
  - User documentation



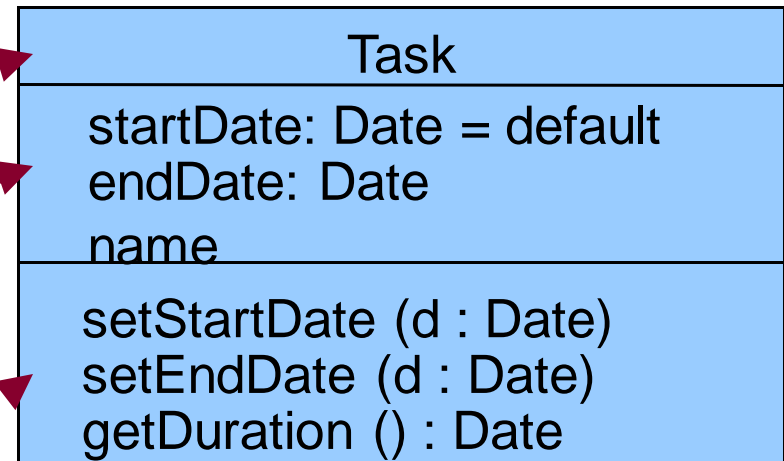
# Class Diagram

- Central for OO modeling
- Shows **static structure** of the system
  - Types of objects
  - Static relationships
    - Association (e.g.: a company has many employees) 
    - Generalization (subtypes) (e.g.: an employee **is a kind of** person) 
    - Dependencies (Aggregation) (e.g.: a company is using trucks to ship products)

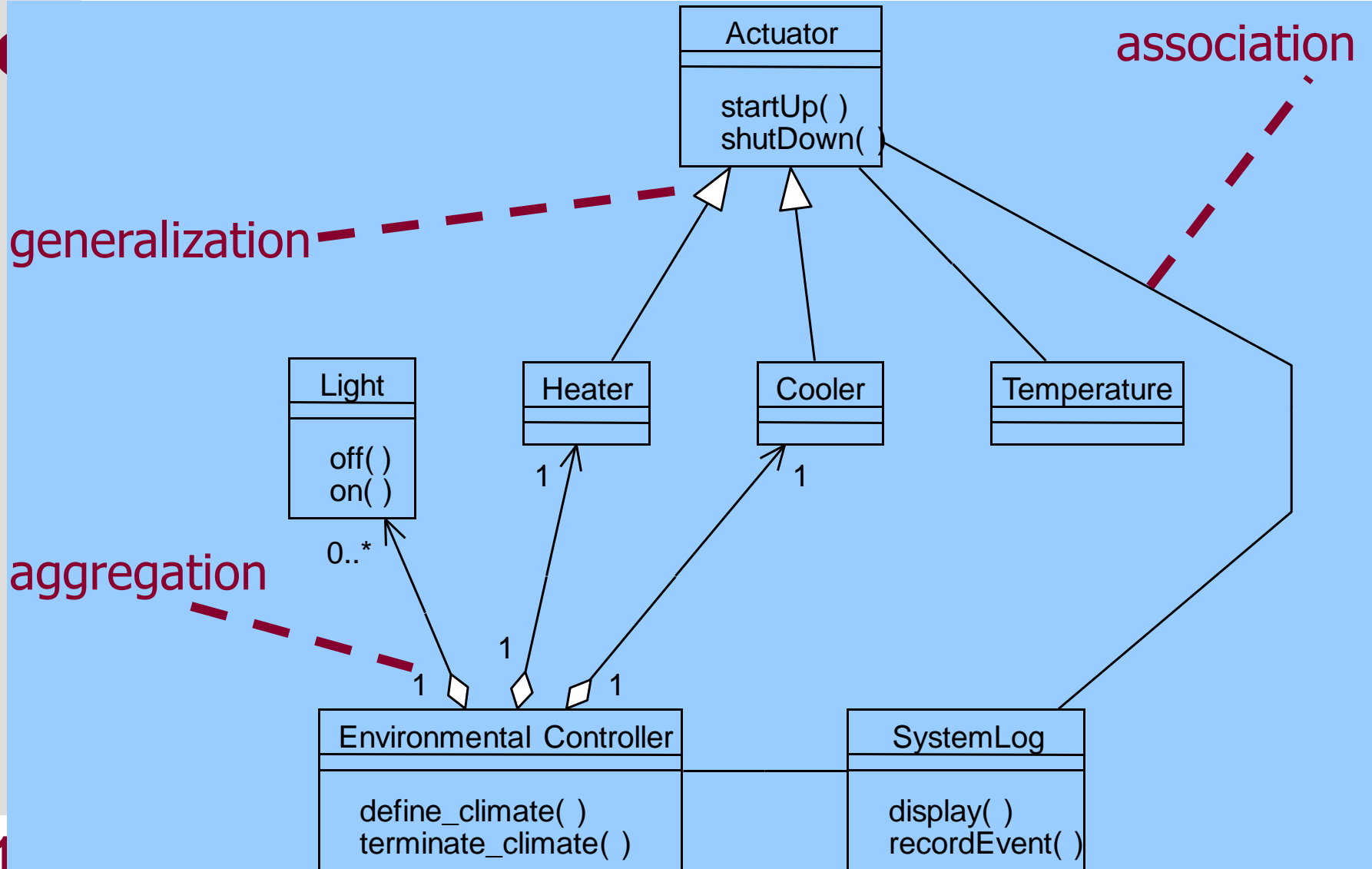


# Class

- Set of objects
- Defines
  - Name
  - Attributes  
(optional: type  
optional: initial value)
  - Operations



# Class diagram example



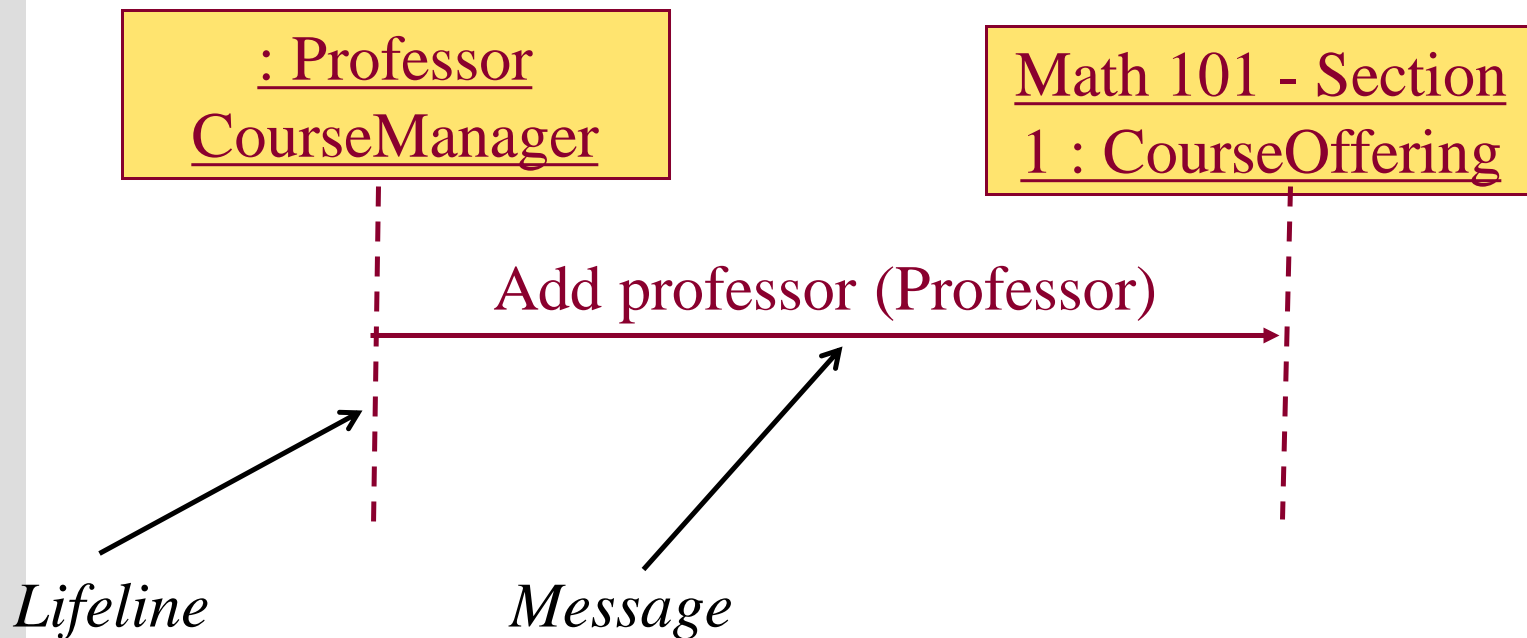
# Sequence Diagrams

- Shows object interactions arranged in time sequence
- It focuses on
  - Objects (and classes)
  - Message exchange to carry out the scenarios functionality
- The objects are organized in an horizontal line and the events in a vertical time line

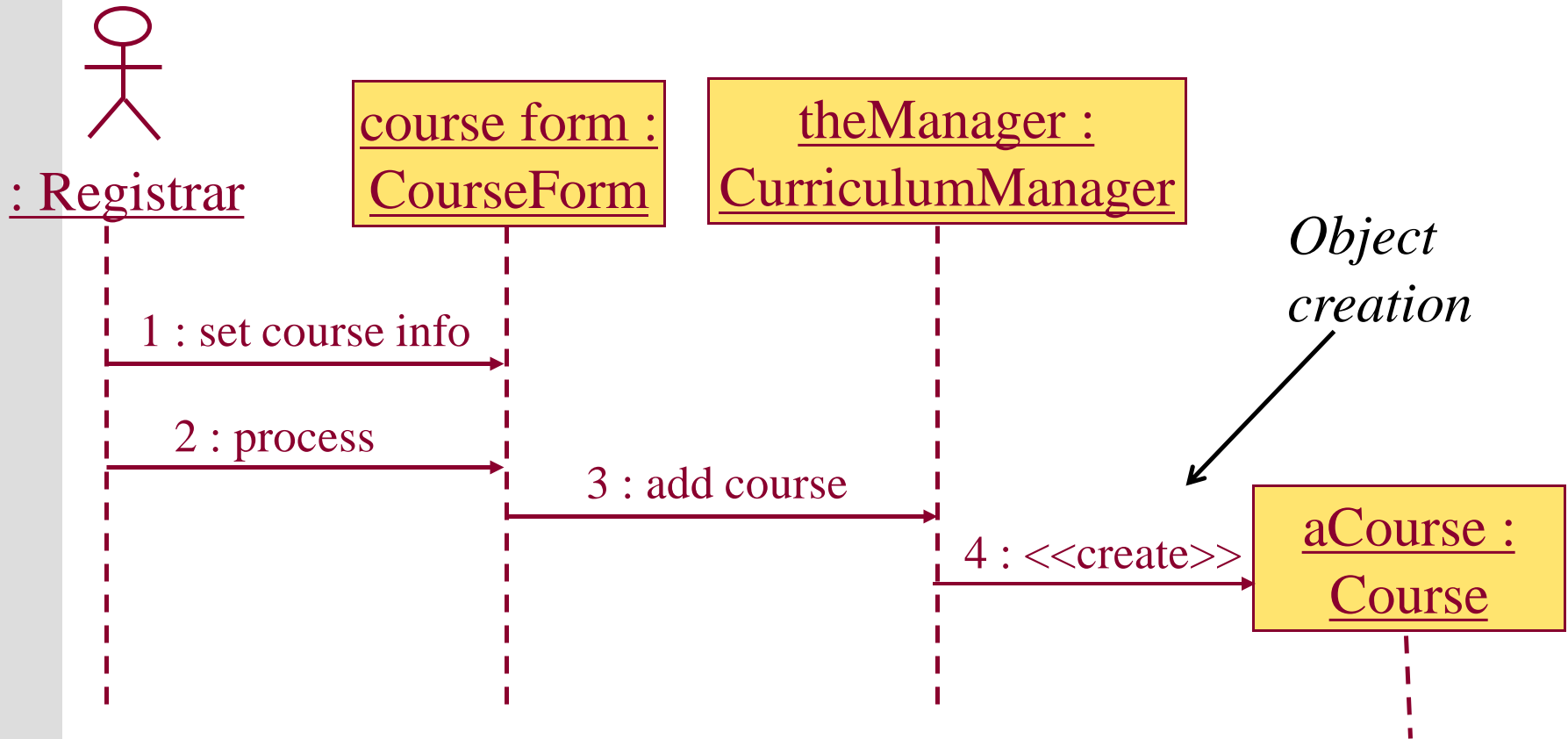


# Notation Example (simple version)

- Messages point from client to supplier

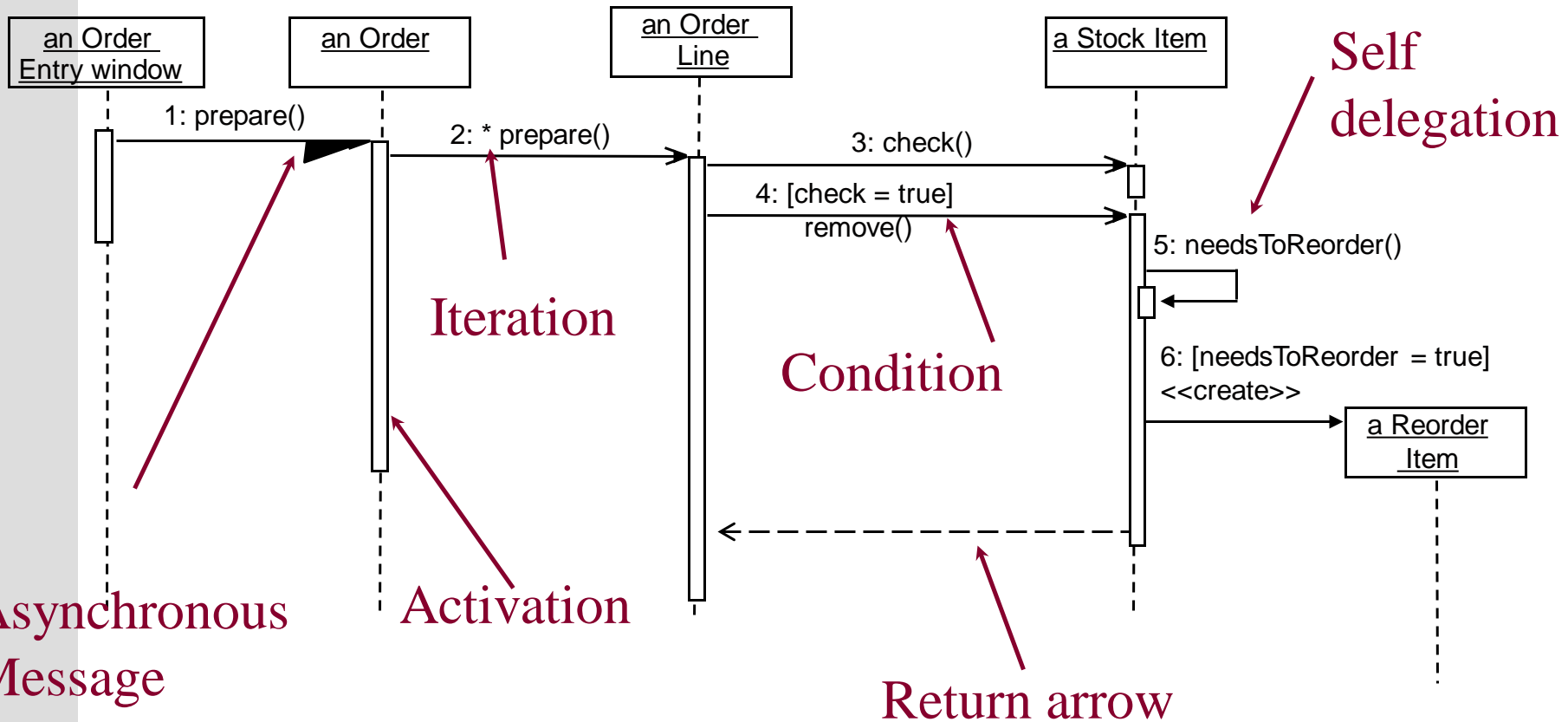


# Sequence Diagram: Larger Example

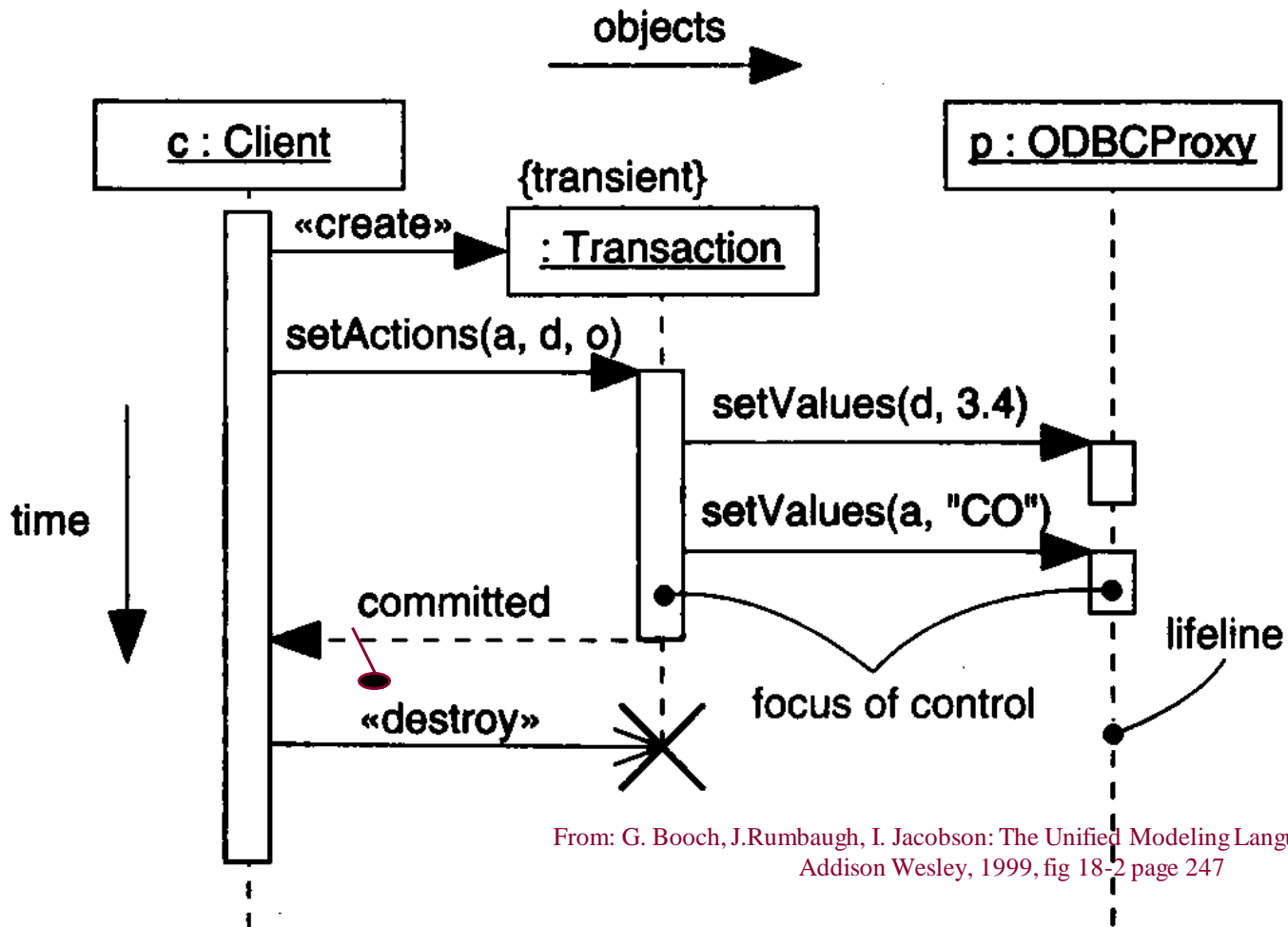




# Sequence Diagrams: More Details



# Example of a Transaction



From: G. Booch, J. Rumbaugh, I. Jacobson: The Unified Modeling Language User Guide. Addison Wesley, 1999, fig 18-2 page 247



# Content of Sequence Diagrams

- Objects
  - They exchange messages among each other
- Messages
  - **Synchronous**: “call events,” denoted by the full arrow; Duration of synchronization should be indicated by activation bar or return arrow
  - **Asynchronous**: “signals,” denoted by a half arrow
  - There are also «create» and «destroy» messages



# Asynchronous messages

- Do not **block** the caller
- Can do 3 things:
  - Create a new thread
  - Create a new object
  - Communicate with a thread that is already running



# References

- Robert V. Binder, *Testing Object-Oriented Systems: Models, Patterns, and Tools*, Addison-Wesley, 2000.



# Collaborators

- Jack Hagemeister

