Agile Modeling:
No, It's Not An Oxymoron

IBM Rational Software Development Conference 2008

WHERE TEAMS ARE R-HEROES

IBM

Rational. software

Who is Terry Quatrani?

tquatrani@us.ibm.com

The Importance of Modeling

What is a Model?

- A model is an abstraction of a physical system
- Typically, you will create different models for a physical system to visualize different points of view
  - Users
  - Developers
  - Graphic Artists
  - Database developers
  - Testers
  - Documenters
  - And on and on ……

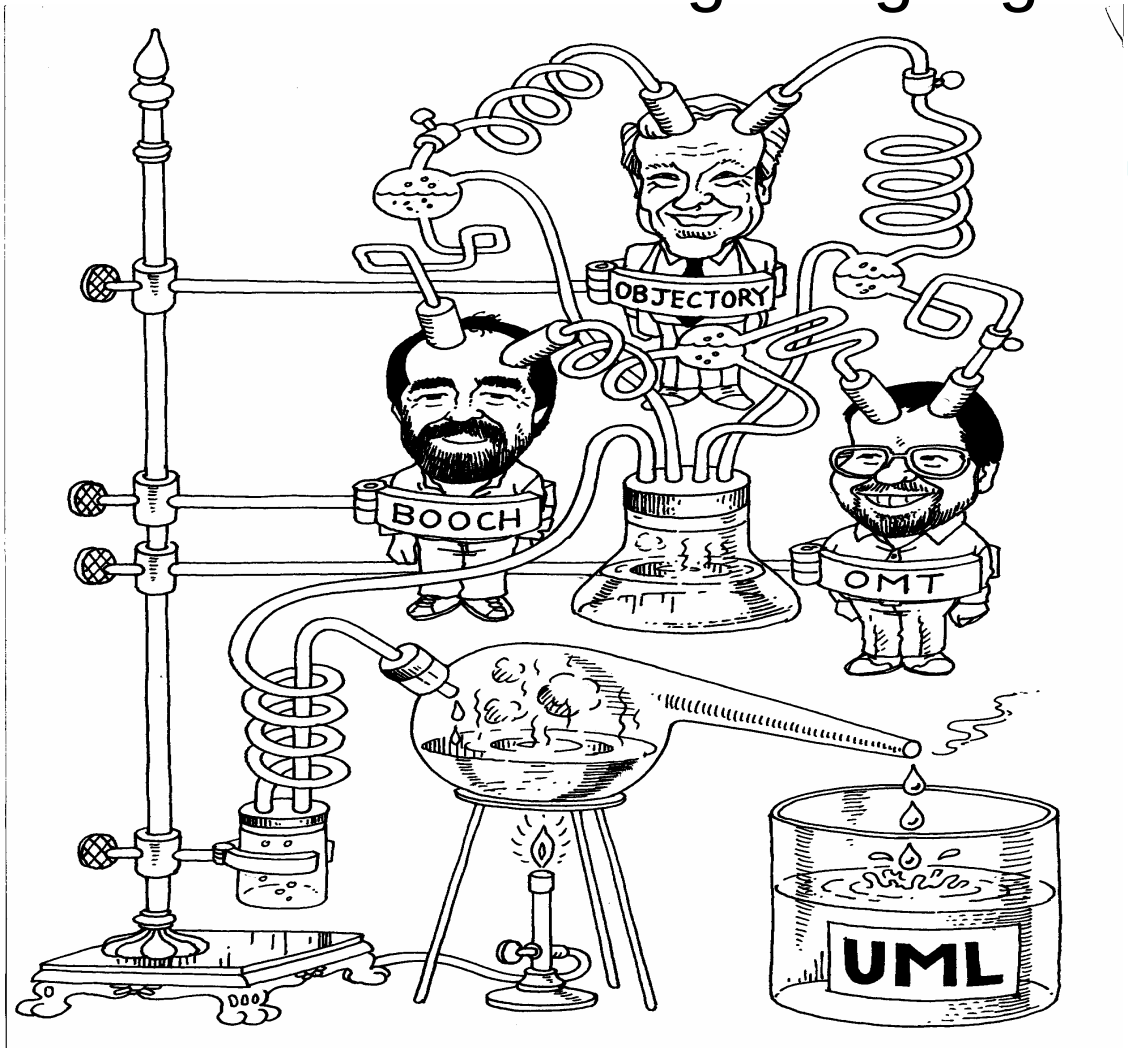
Why do we model?

- To manage complexity
- To detect errors and omissions early in the lifecycle
- To communicate with stakeholders
- To understand requirements
- To drive implementation
- To understand the impact of change
- To ensure that resources are deployed efficiently

Why do we model?

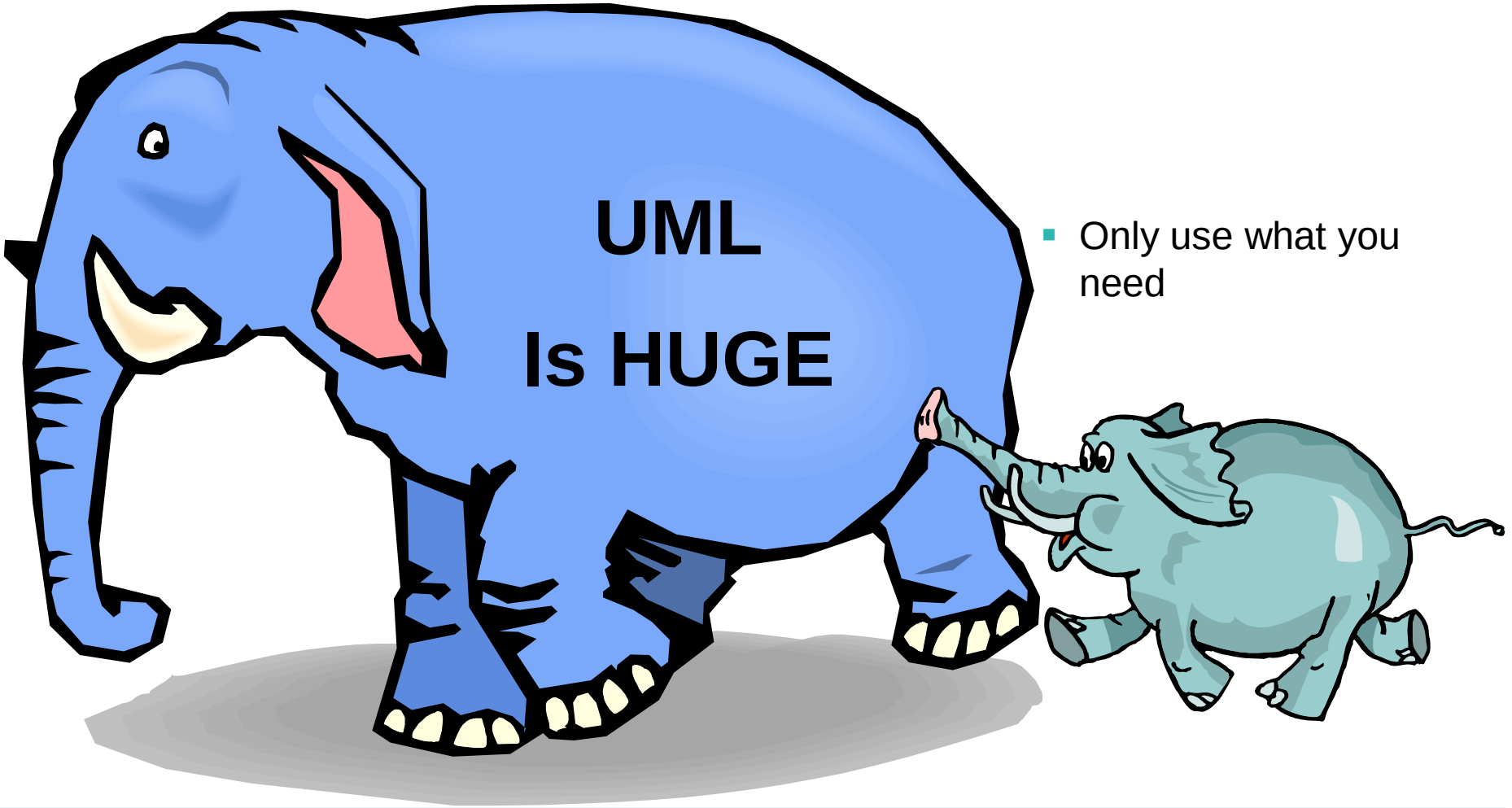
- To manage complexity
- To detect errors and omissions early in the lifecycle
- To communicate with stakeholders
- To understand requirements
- To drive implementation
- To understand the impact of change
- To ensure that resources are deployed efficiently

The Unified Modeling Language



- The UML is the standard language for visualizing, specifying, constructing, and documenting software and systems

TQ's Golden Rule



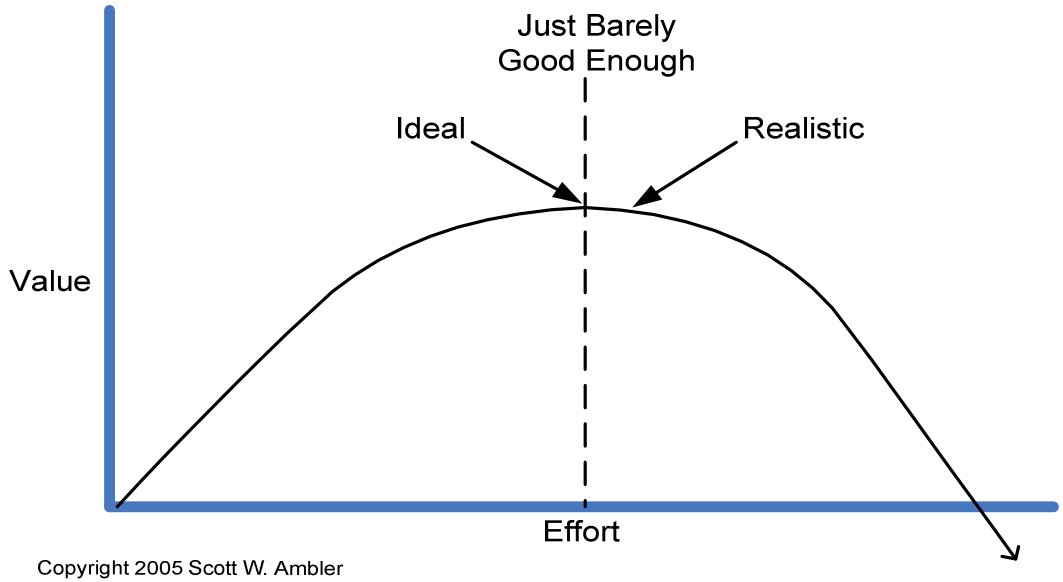
- Only use what you need

Scott Ambler's Critical Concepts of Agile Modeling

- Apply the right artifact(s)
- Maximize stakeholder return on investment (ROI)
- Active stakeholder participation
- Iterate to another artifact
- Model in small increments
- Create several models in parallel
- Collective ownership
- Single source information
- Prefer executable specifications

What Are Agile Models?

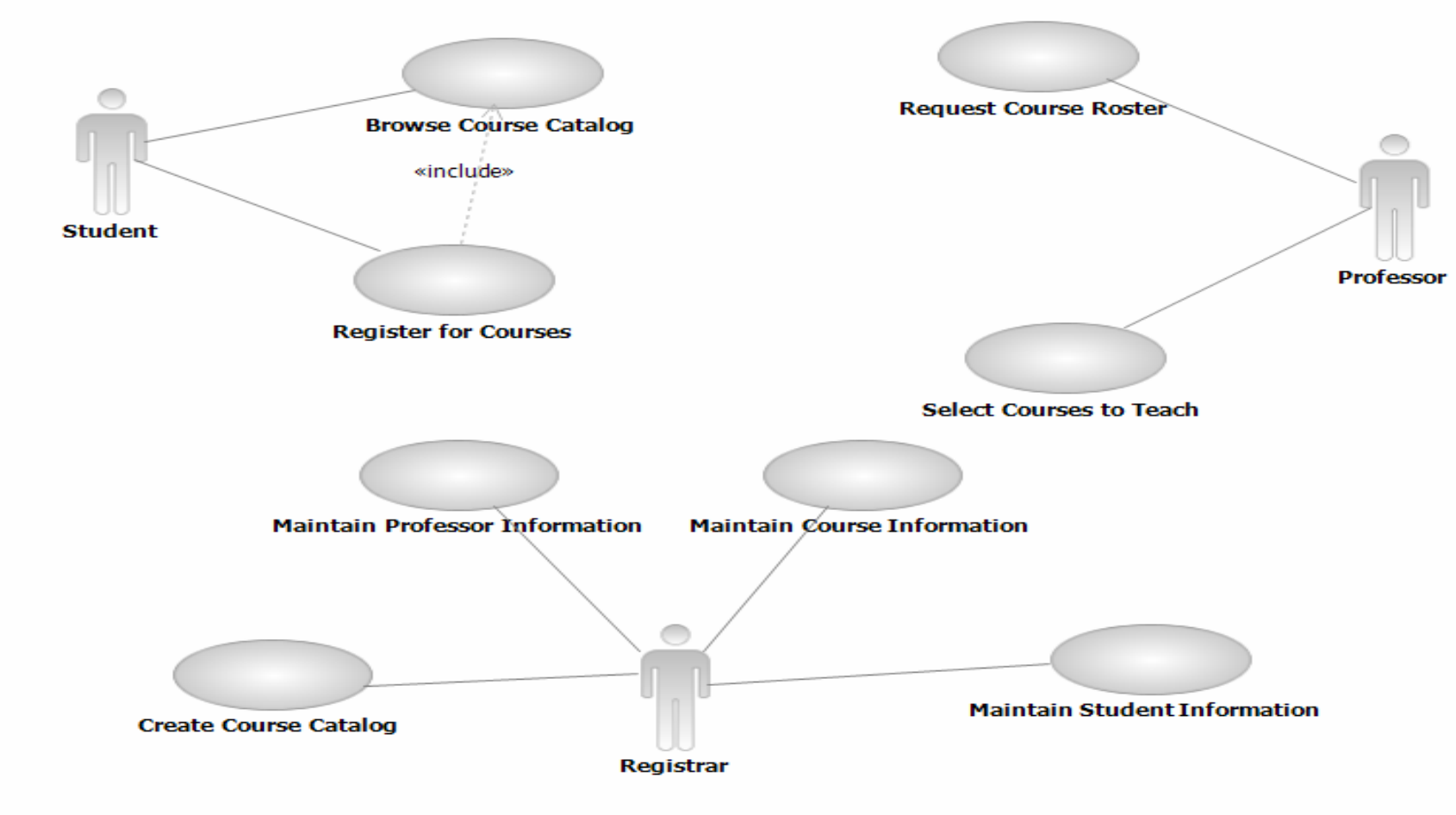
- Agile models:
  - Fulfill their purpose
  - Are understandable
  - Are sufficiently accurate
  - Are sufficiently consistent
  - Are sufficiently detailed
  - Provide positive value
  - Are as simple as possible
- Agile models are just barely enough!



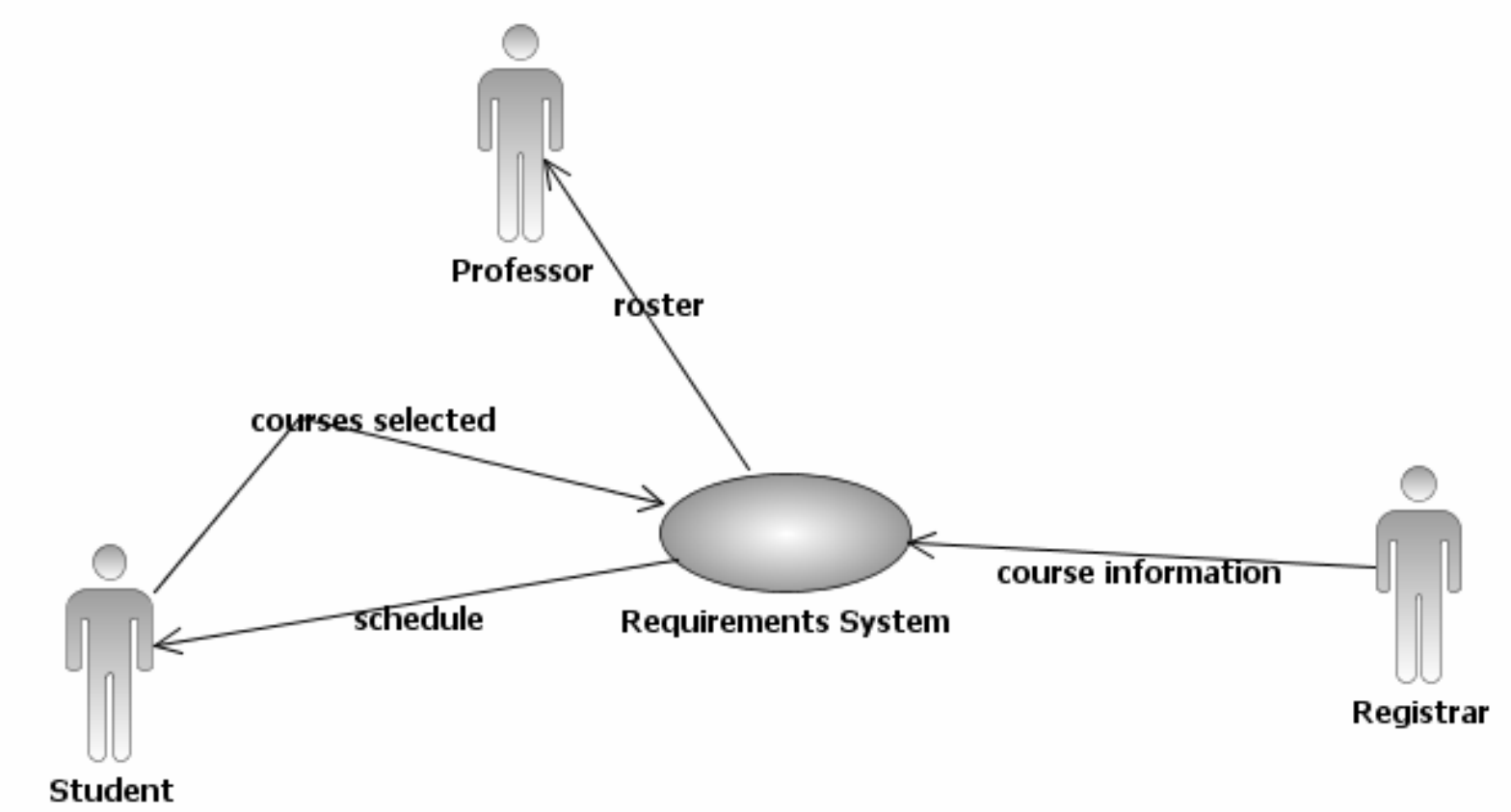
Requirements – System Context

- Actors are people or systems that need to communicate with the system being built
- Use cases are a way to capture functional requirements of a system
- In theory, you should be able to determine system context by looking at a use case diagram
  - Works well if there are a limited number of use cases
  - Does not work as well for larger systems

Use Case Diagram

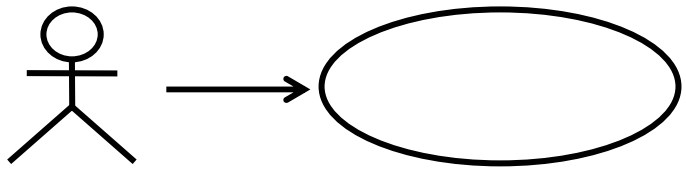


Not Exactly UML but….



Documenting Use Cases (Functional Requirements)

- Use cases are shown in diagrams



Student ***Register for Courses***

- Use cases are described in text

Use-Case Specification – Register for Courses

Brief Description

This use case allows a Student to register for course offerings in the current semester. The Student can also modify or delete course selections if changes are made within the add/drop period at the beginning of the semester. The Course Catalog System provides a list of all the course offerings for the current semester.

Actors

1. ***Primary Actor – Student***
2. ***Secondary Actor - Course Catalog System***

Flow of Events

1. ***Basic Flow***

1.1. LOG ON.
This use case starts when a student accesses the Course Registration System. The student enters a student ID and password and the system validates the student.

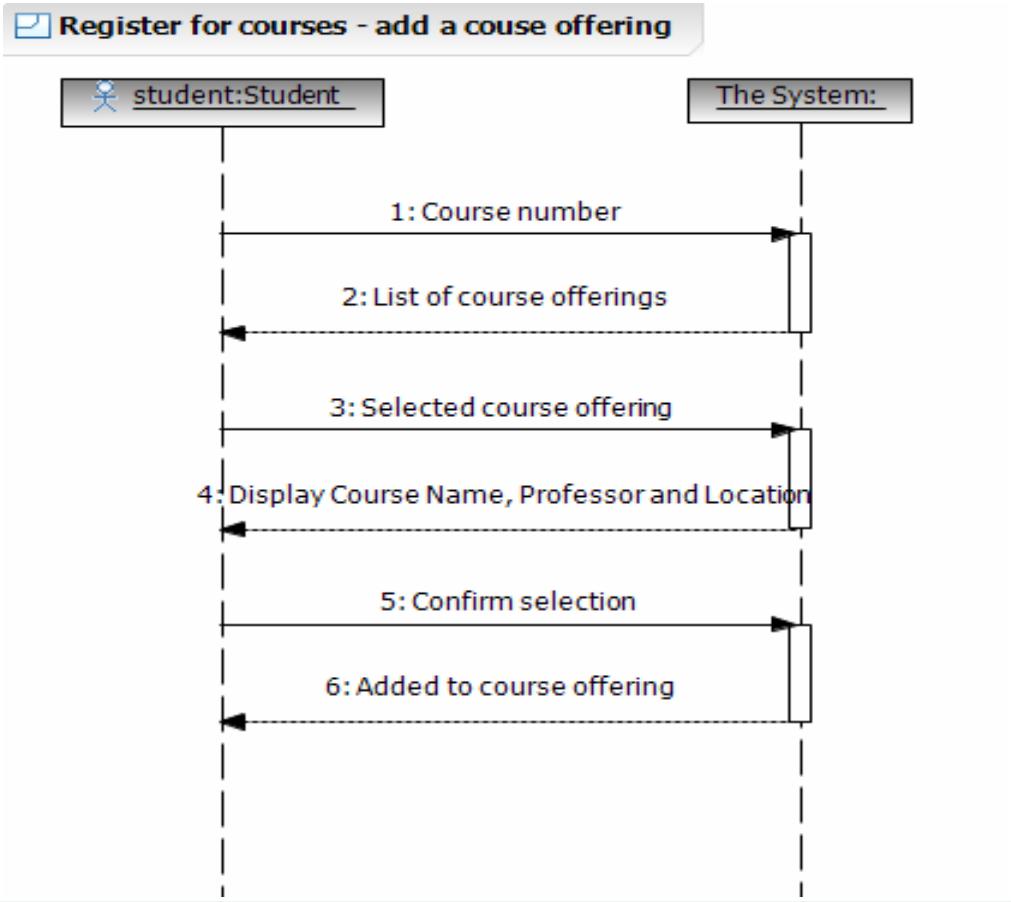
1.2. CREATE SCHEDULE.
The system displays the functions available to the student. These functions are: Create A Schedule, Modify a Schedule and Delete a Schedule. The student selects 'Create a Schedule'.

1.3. SELECT COURSES
The system retrieves a list of available course offerings from the Course Catalog System and displays the list to the student. The Student selects up to 4 primary course offerings and 2 alternate course offerings from the list of available offerings. The student can add and delete courses as desired until choosing to submit the schedule.

1.4. SUBMIT SCHEDULE.
The student indicates that the schedule is complete. The system validates the courses selected and displays the schedule to the student. The system displays the confirmation number for the schedule. The systems saves the student's schedule information. The use case ends.

But…. The UML does not specify how a use case should be described

Simple Sequence Diagram is Worth Hundreds of Words



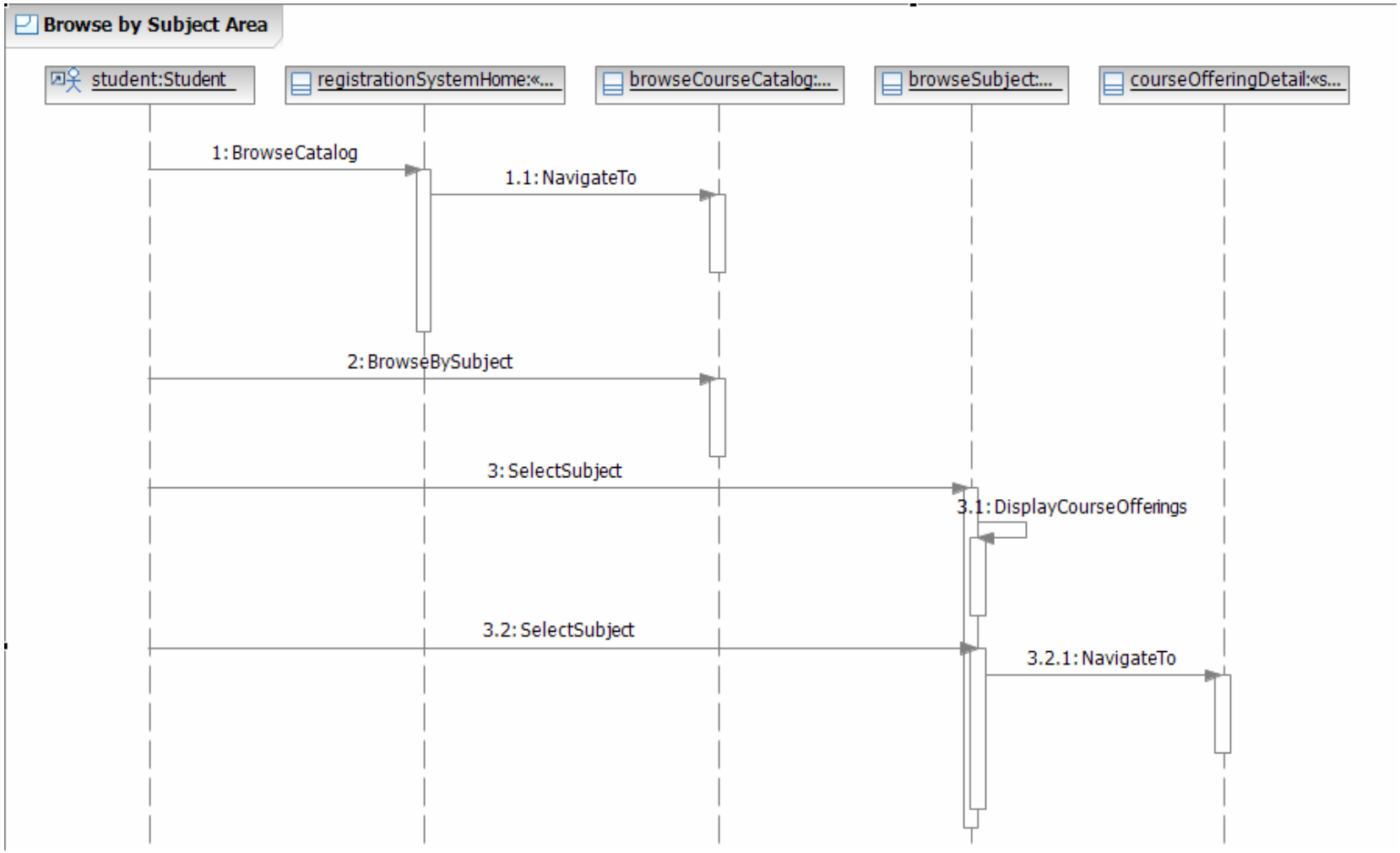
Bulleted Outline is Usually Sufficient

- Time ordered outline of the steps in the use case
  - Typically just simple sentences
- Concentrate on the steps in the basic flow
- List major alternate flows and exceptions
- This is just a first cut at the use case flow of events
- Benefits
  - Allow you to get a handle on the complexity of the use case (more steps typically more complexity)
  - Allow you to get early buy in from the customer that you are building the right system
  - Provide basis for prototyping

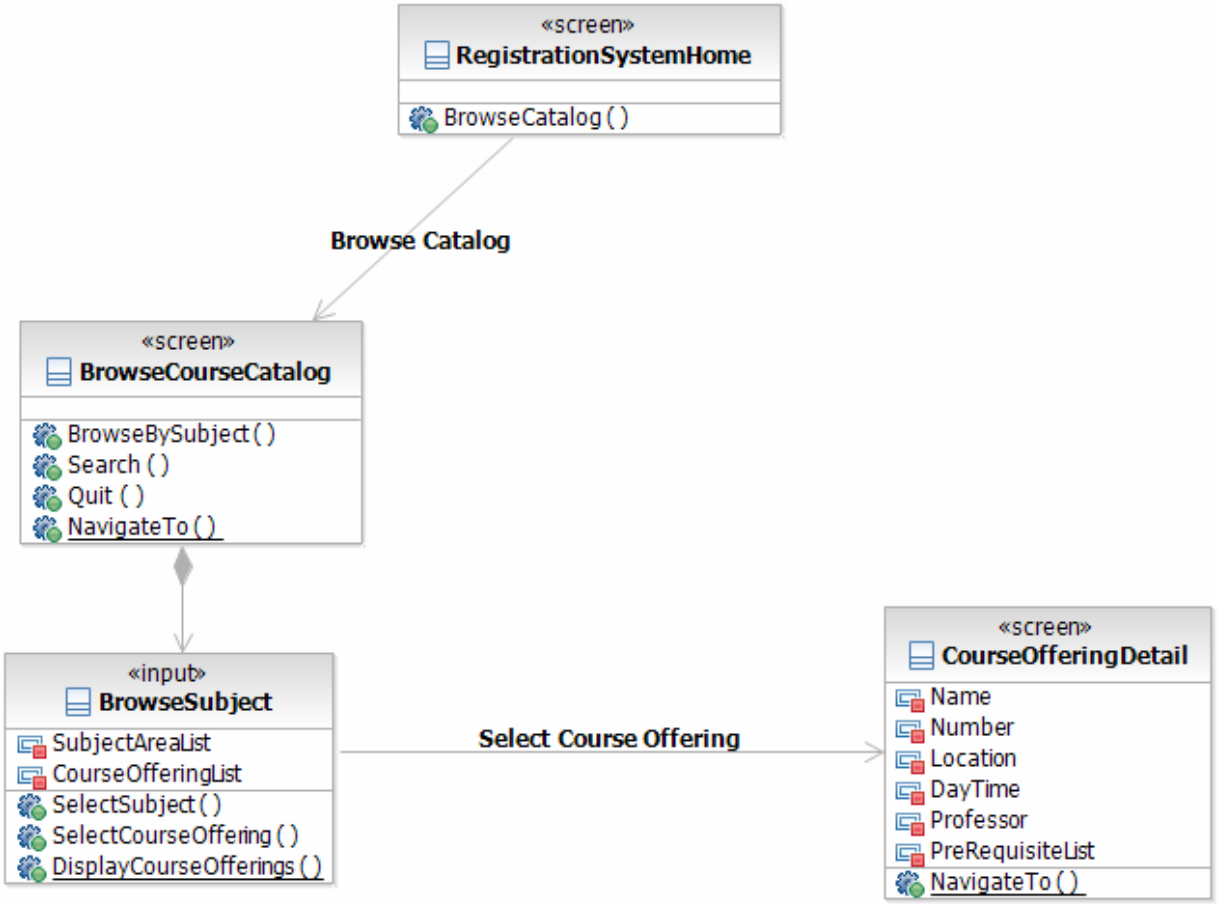
Add a Course Bulleted Outline

- Basic Flow
  - Student logs onto system
  - Student opts to add a course to a schedule
  - Student enters a course number
  - System verifies that student has satisfied pre-requisites for the course
  - System displays a list of open course offerings
  - Student selects an offering
  - Student is registered for the course
- Alternate Flows
  - Pre-requisites not satisfied
  - No course offerings available

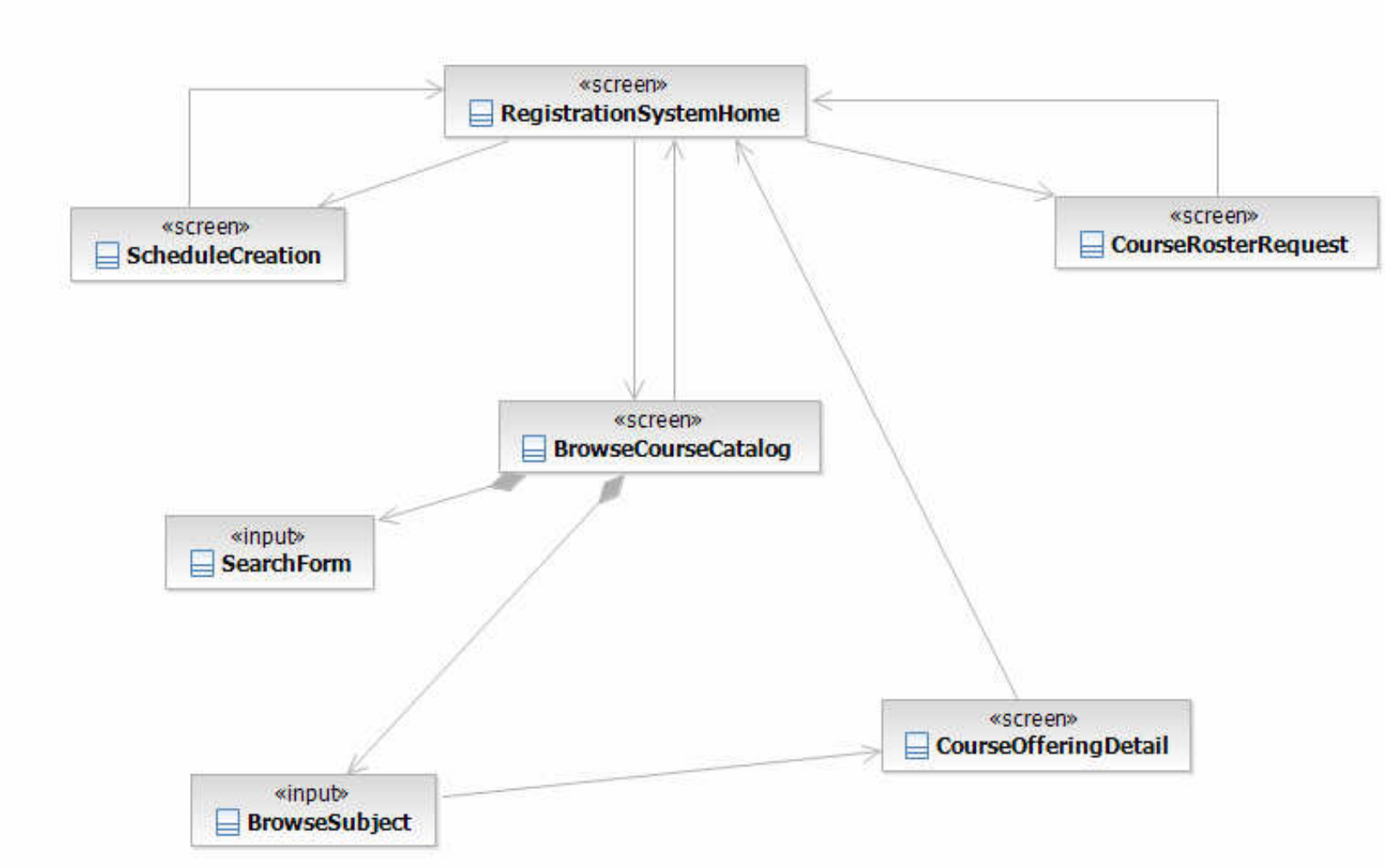
User Interface Storyboard



User Interface Screens



User Interface Navigation Map



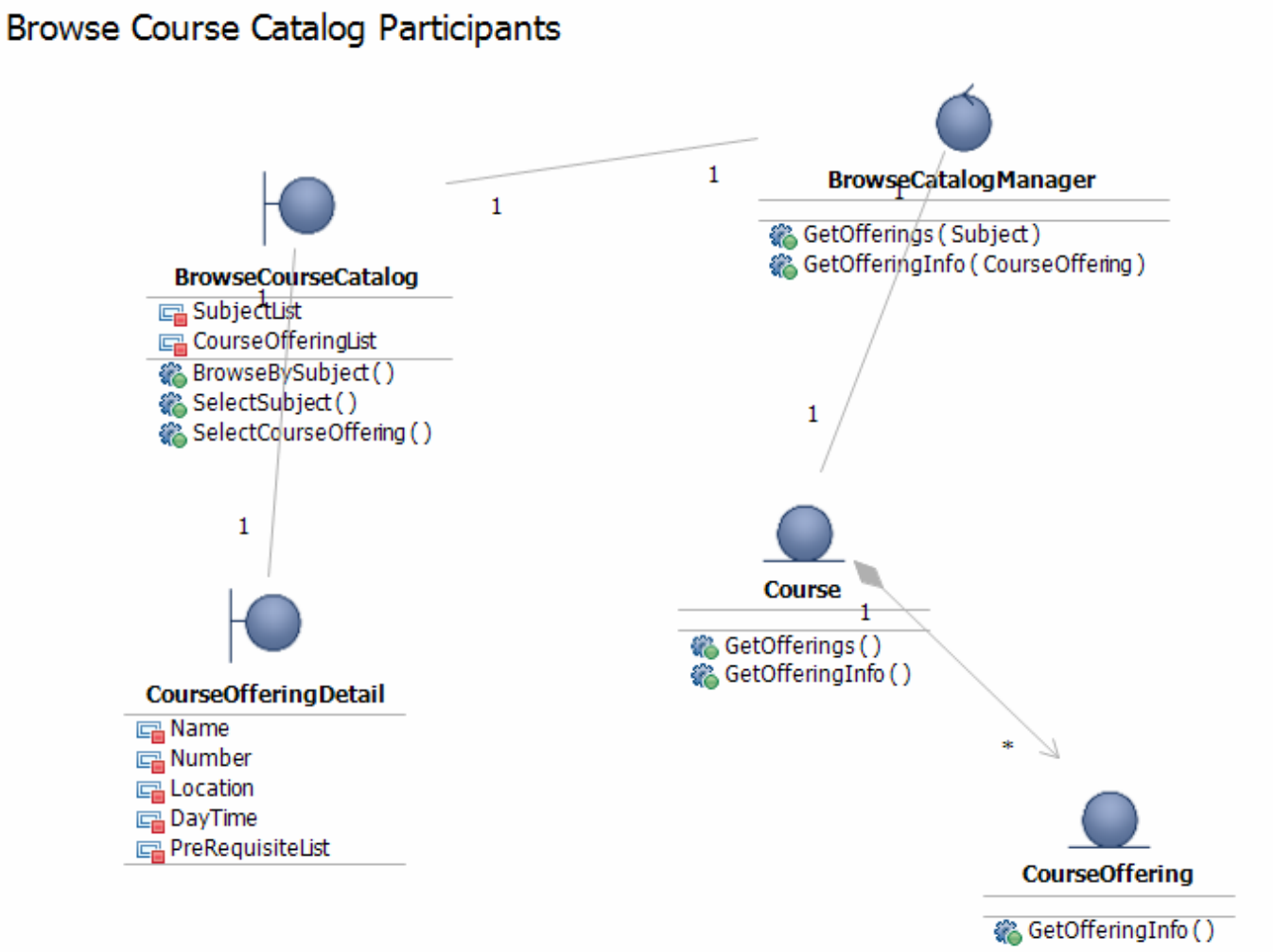
Analysis Classes

- Boundary Class
  - A class used to model interaction between the system's surroundings and its inner workings

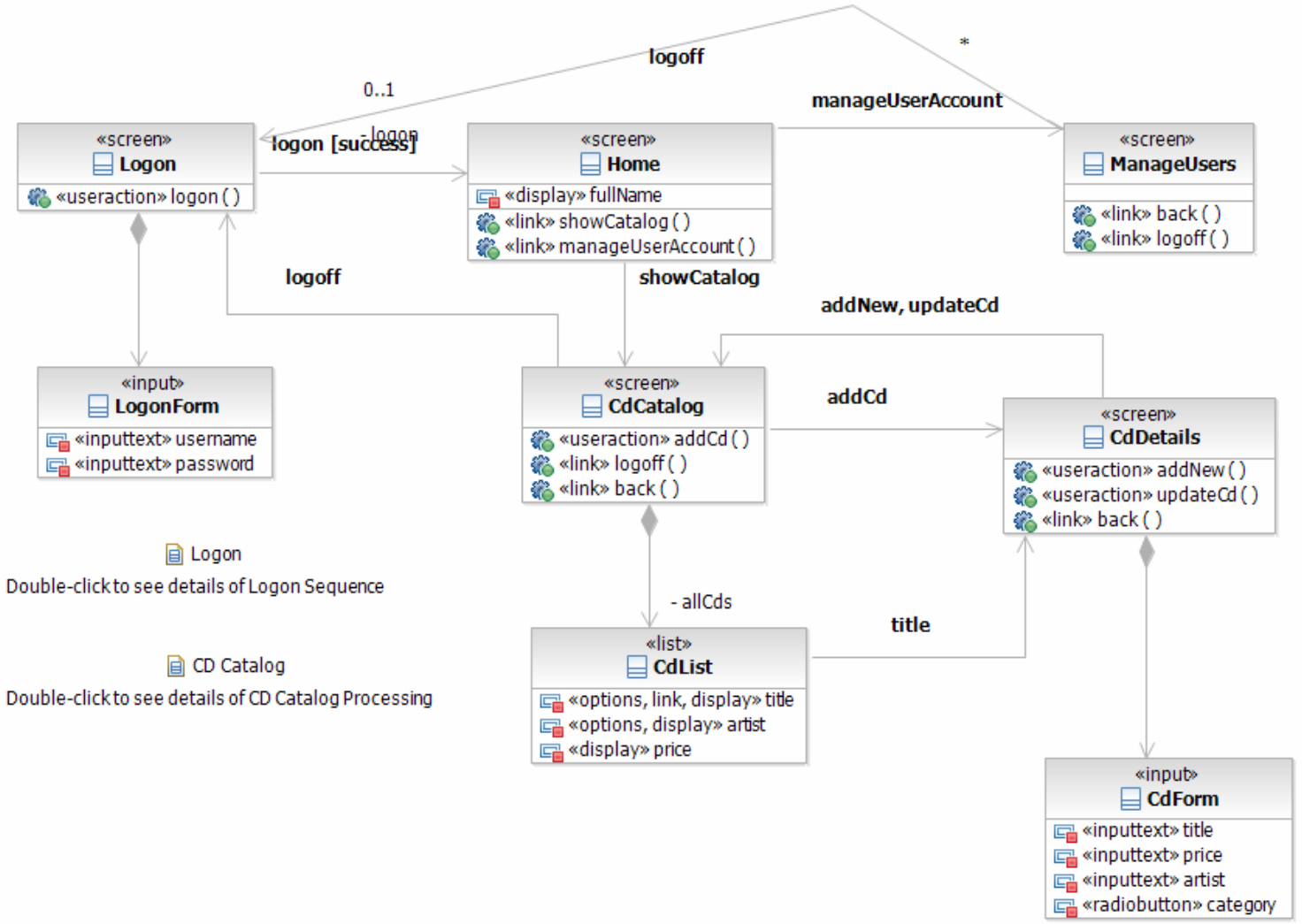
- Control Class
  - A class used to model control behavior specific to one or a few use cases

- Entity Class
  - A class used to model information and associated behavior that must be stored

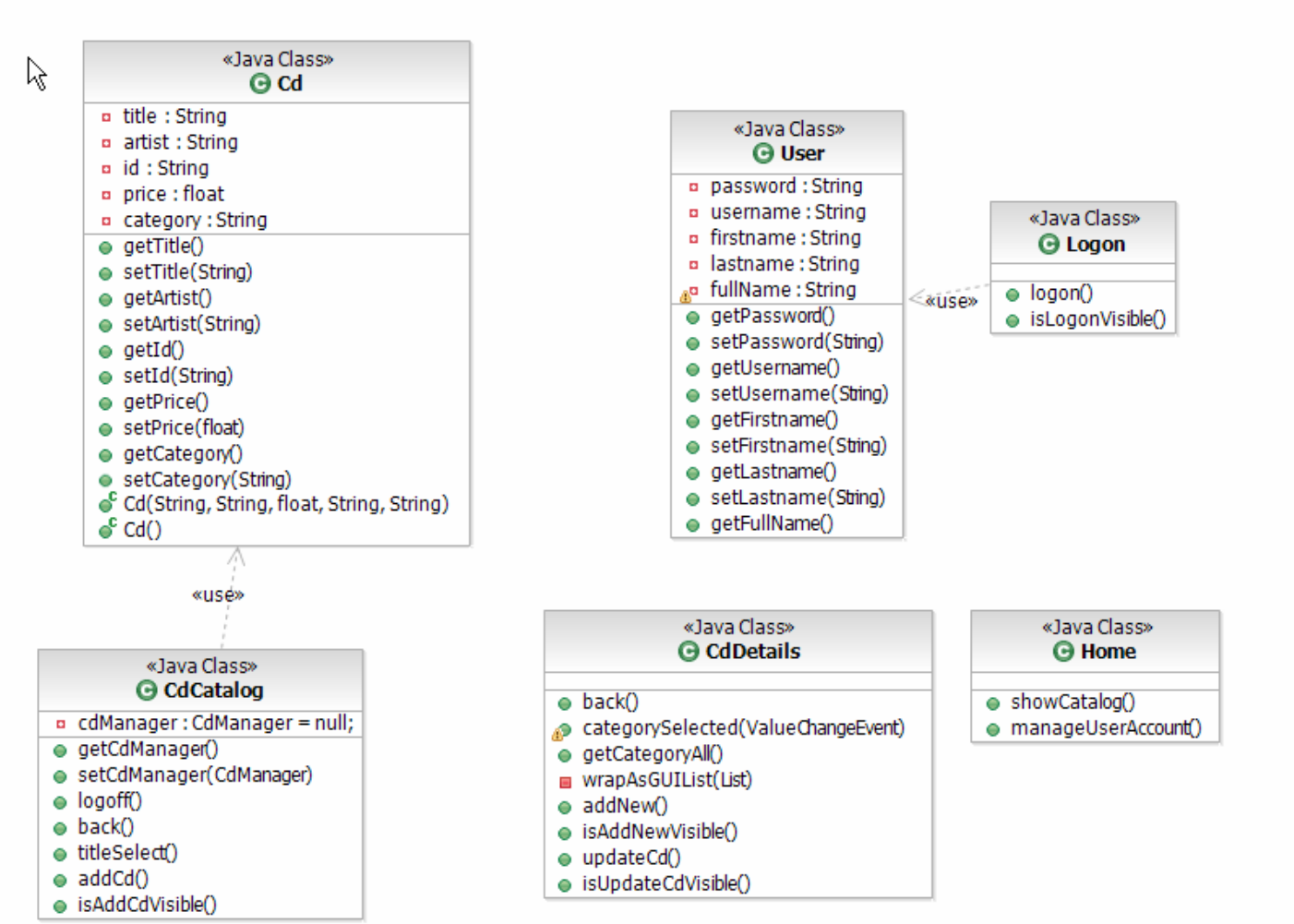
Analysis Level Diagram



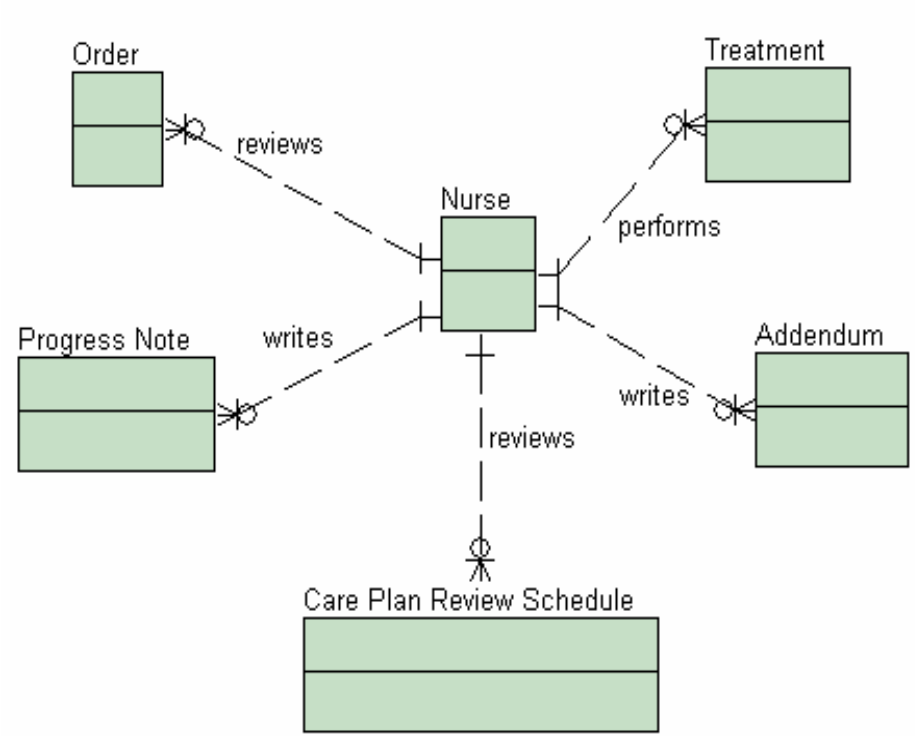
User Interface Design Diagram



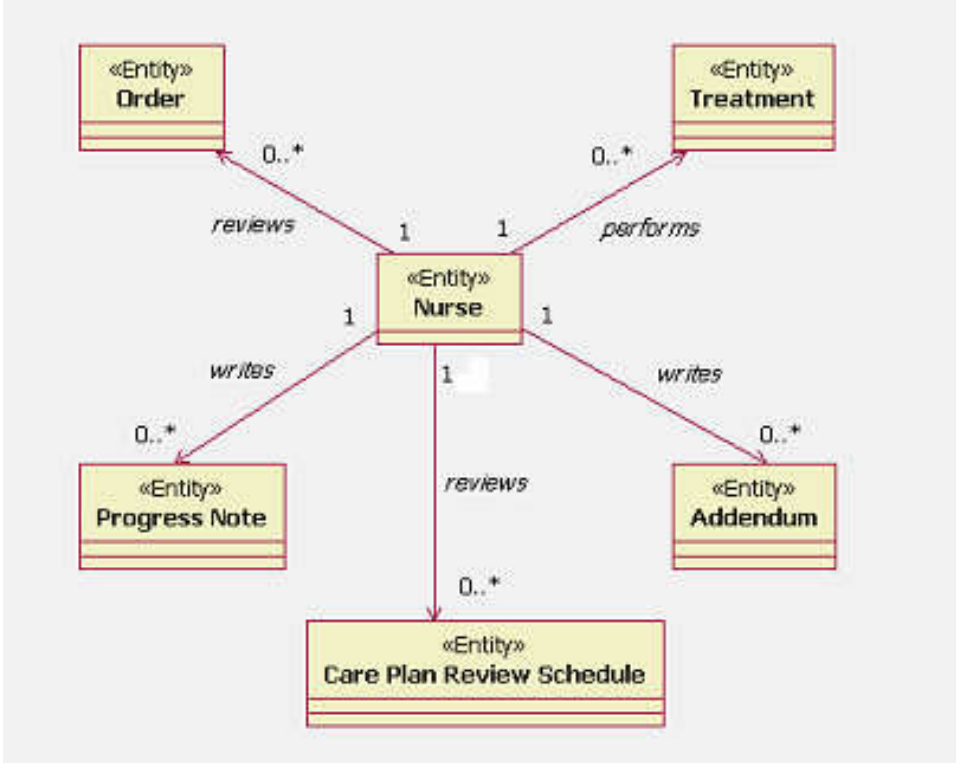
Java Class Design Diagram



Entity-Relationship Modeling

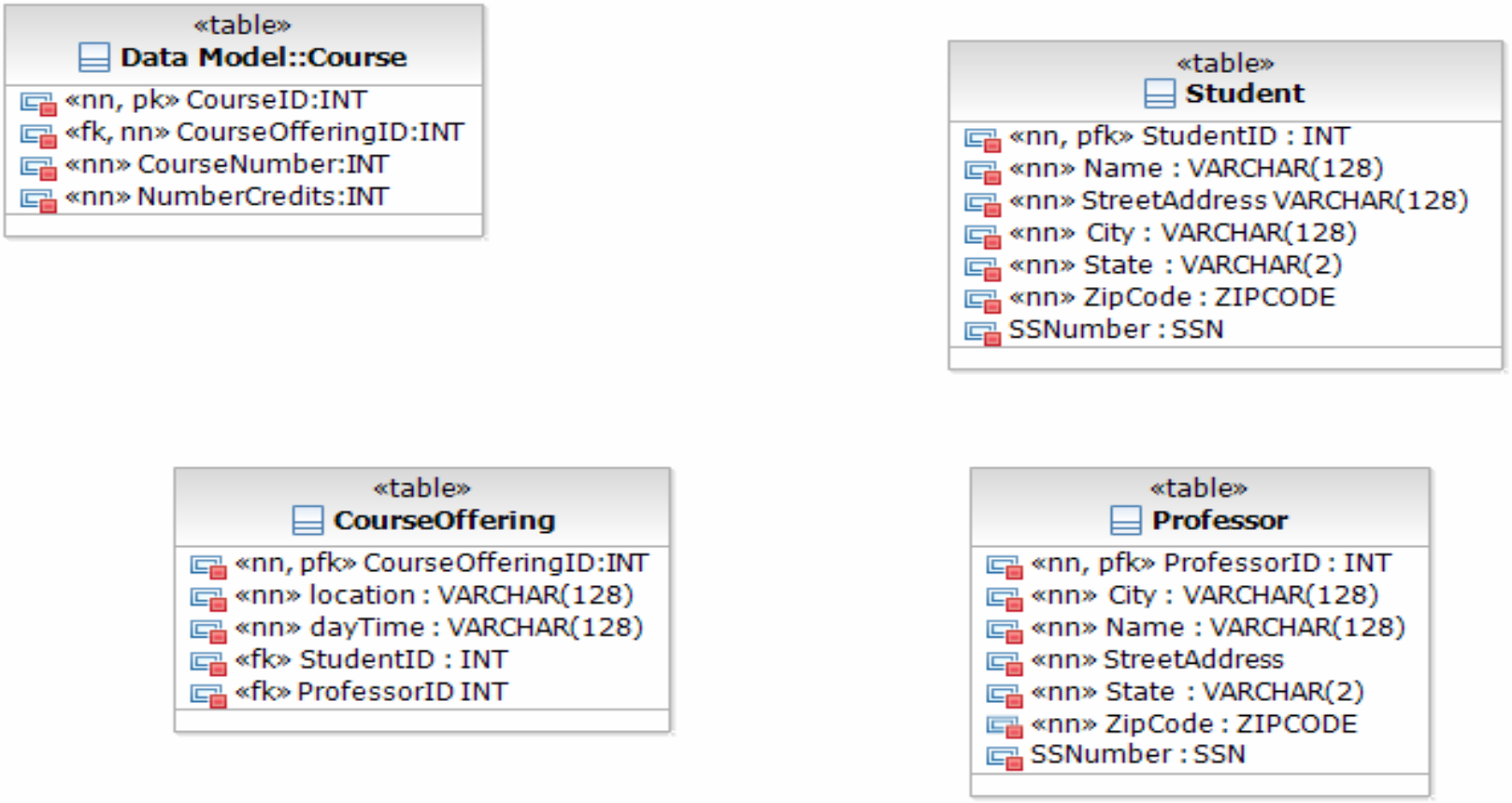


Crow's Feet Notation

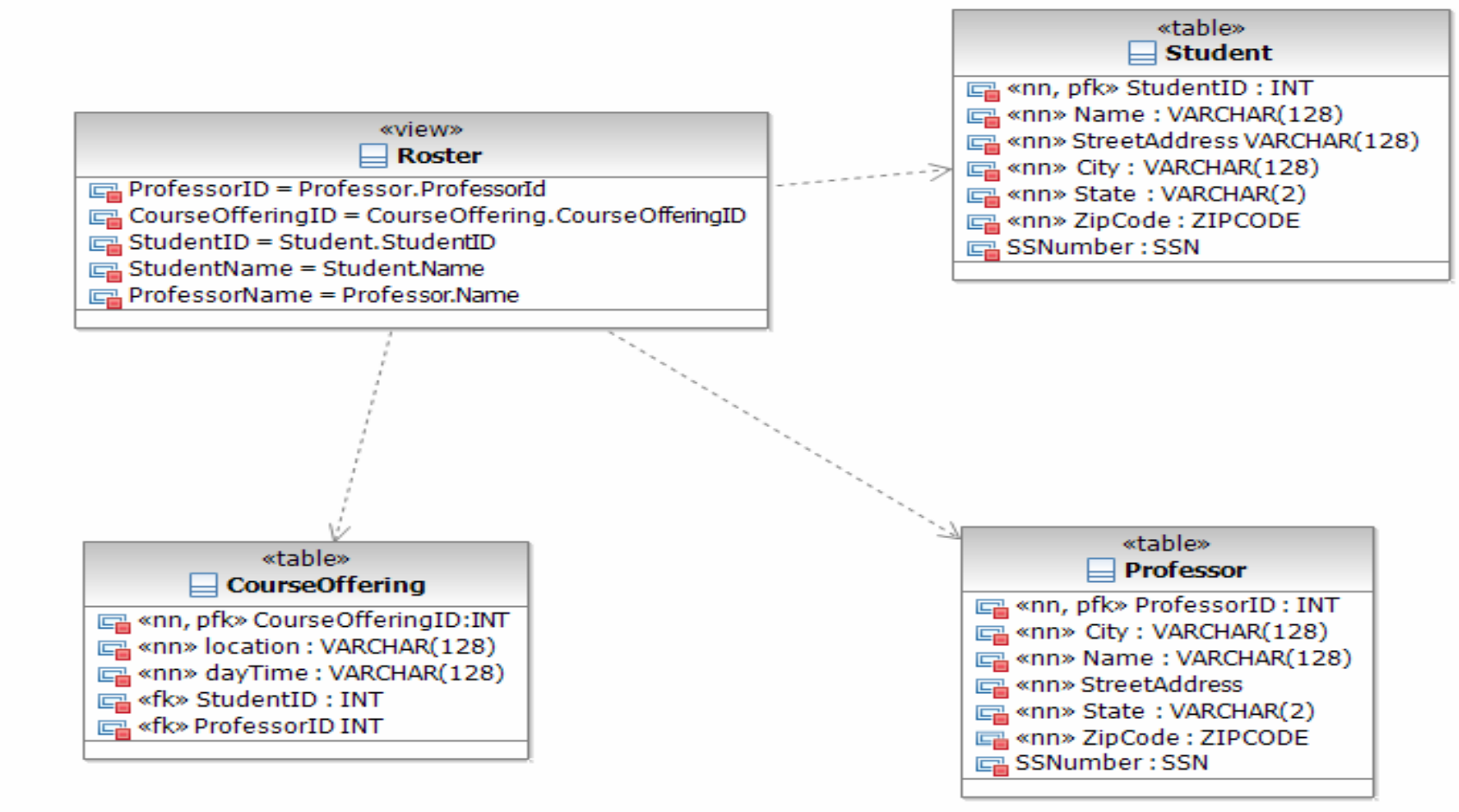


UML Notation

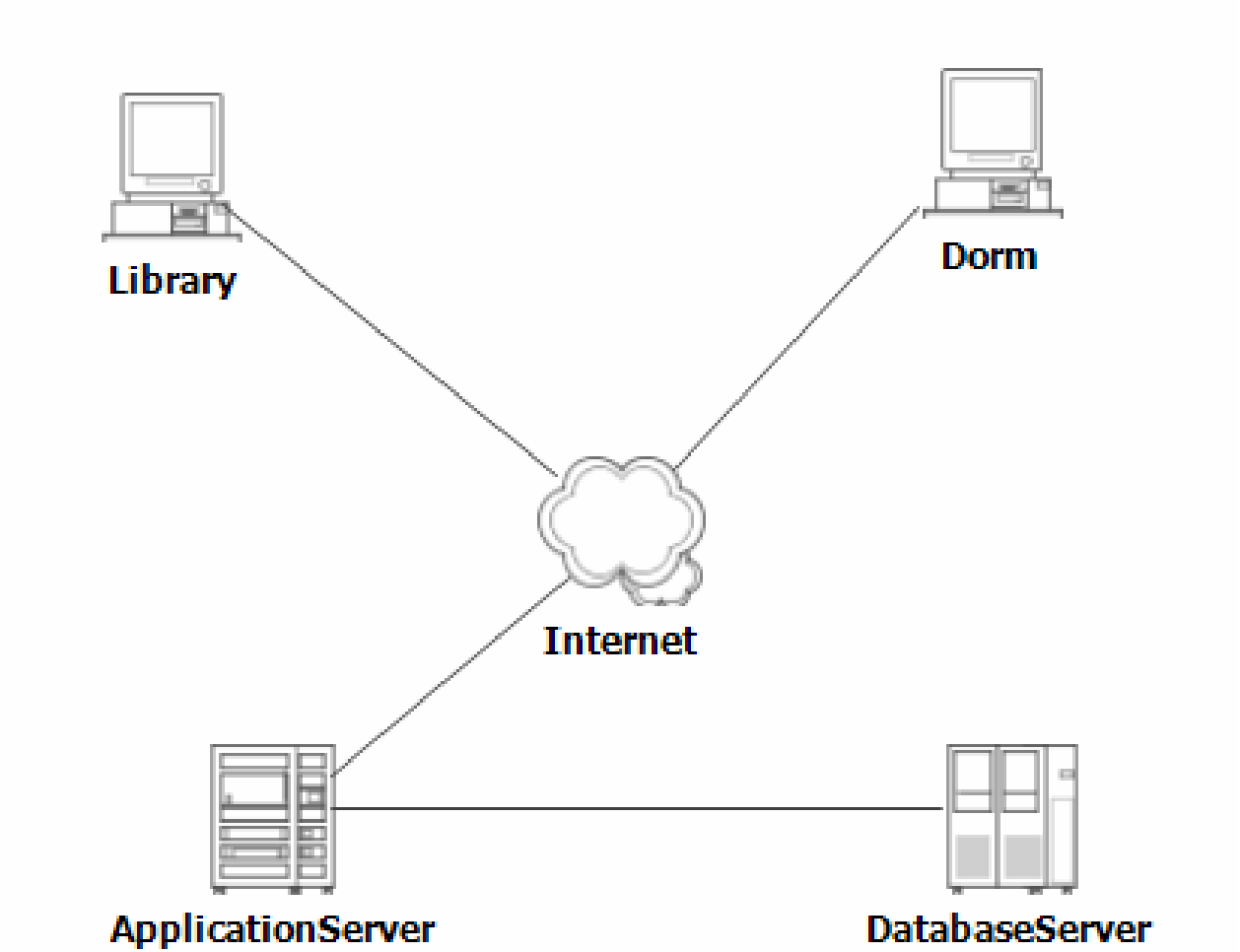
Course Registration Physical Data Model



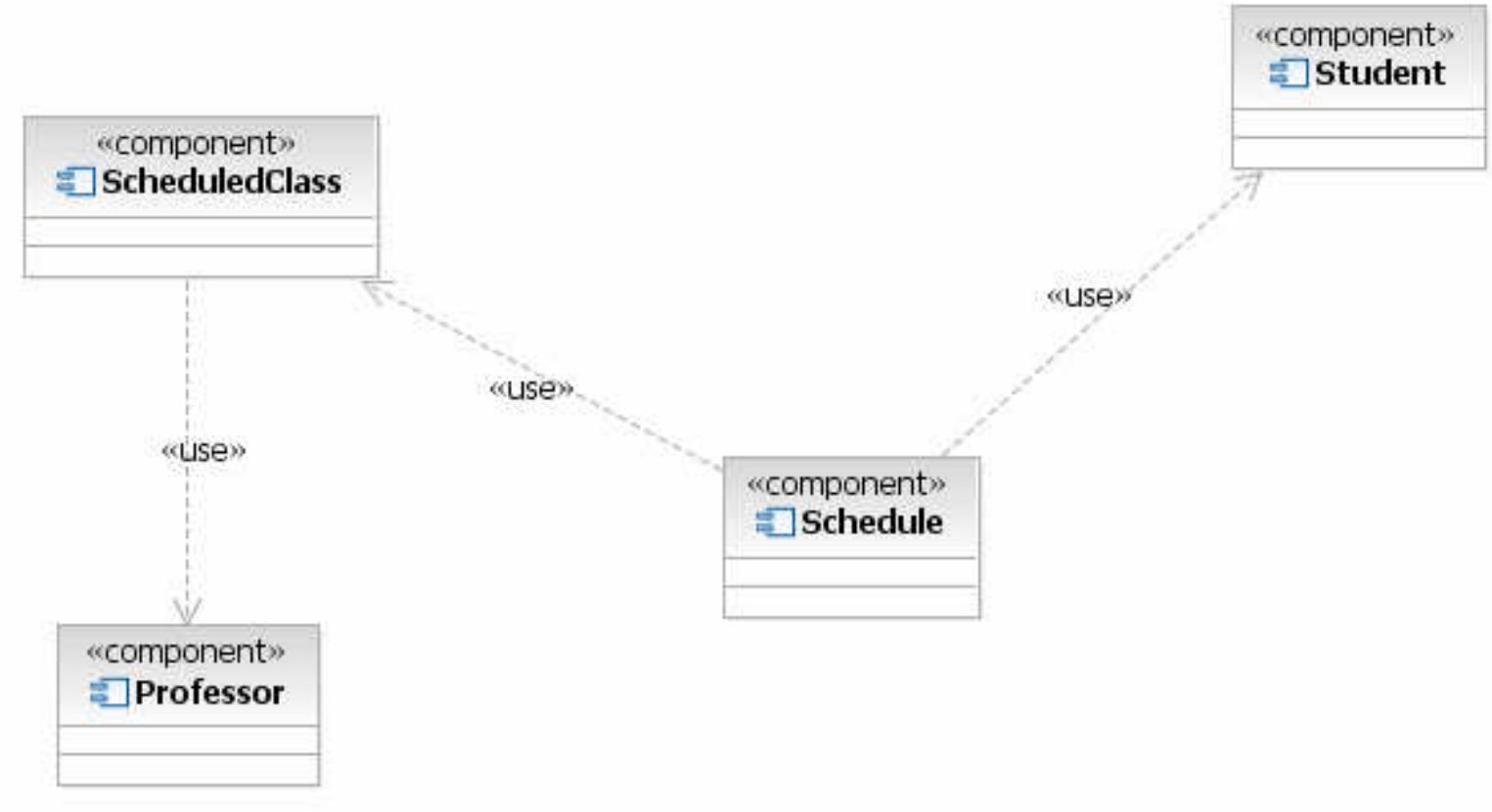
Roster Database View



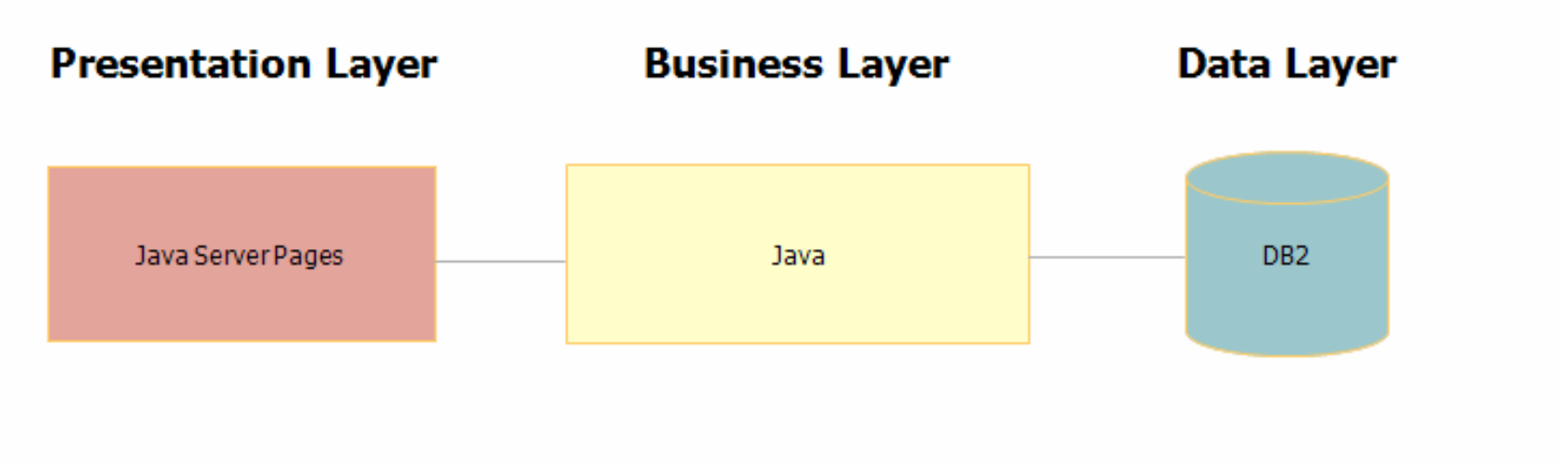
Deployment Diagram



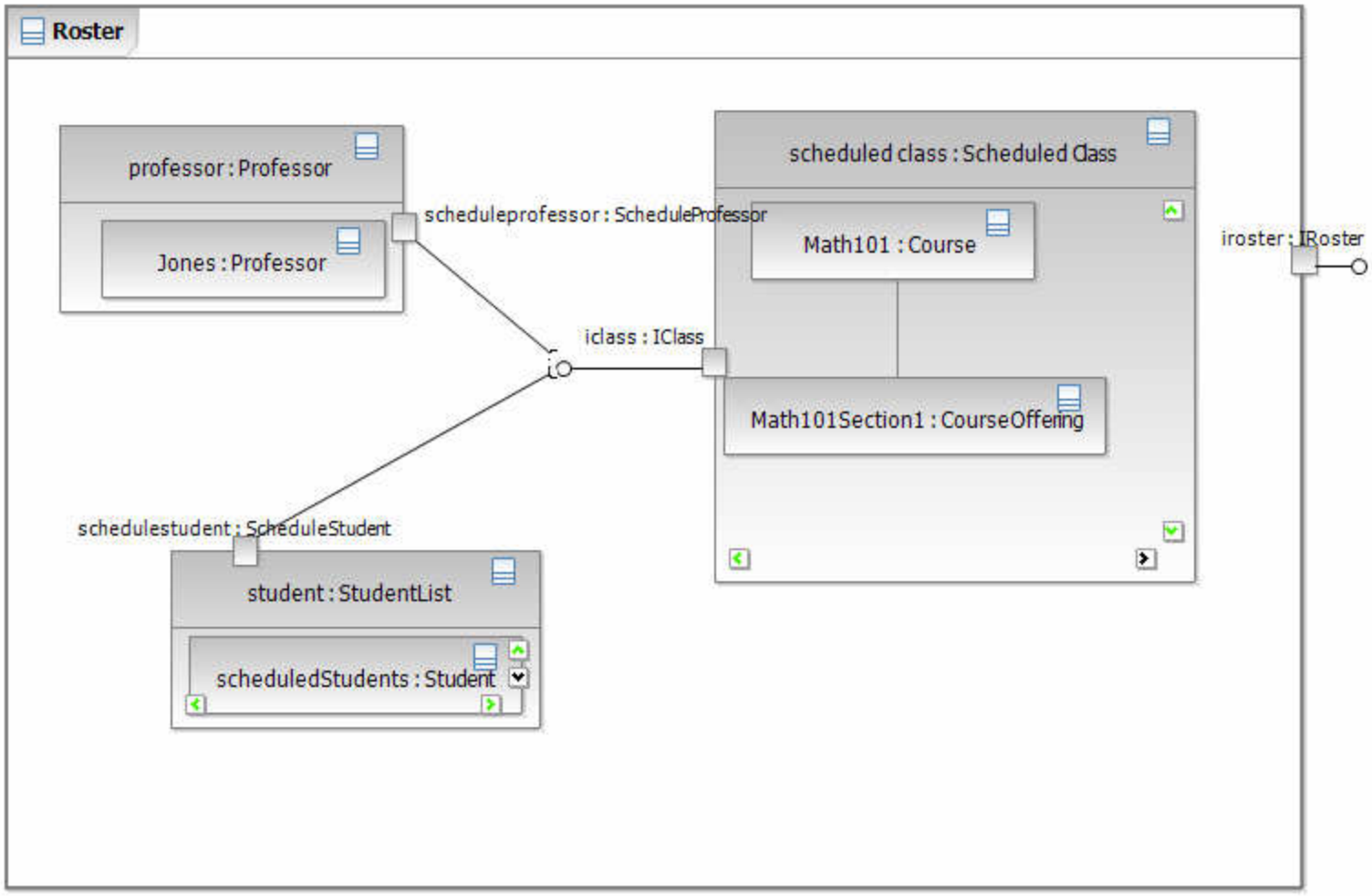
Component Diagram



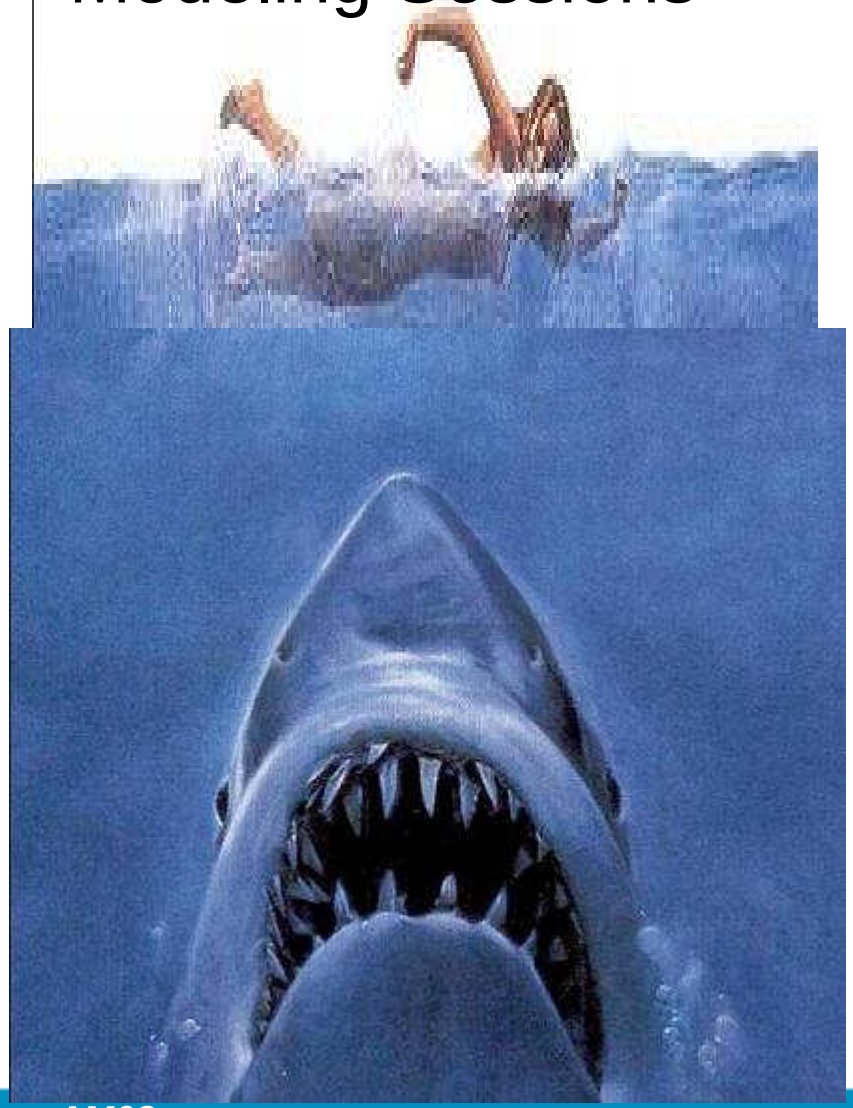
Architecture



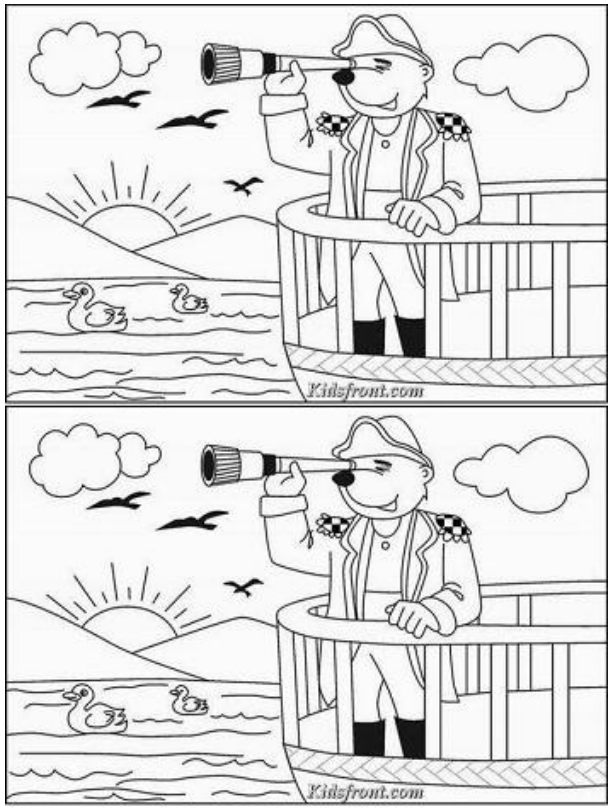
Complex Architecture

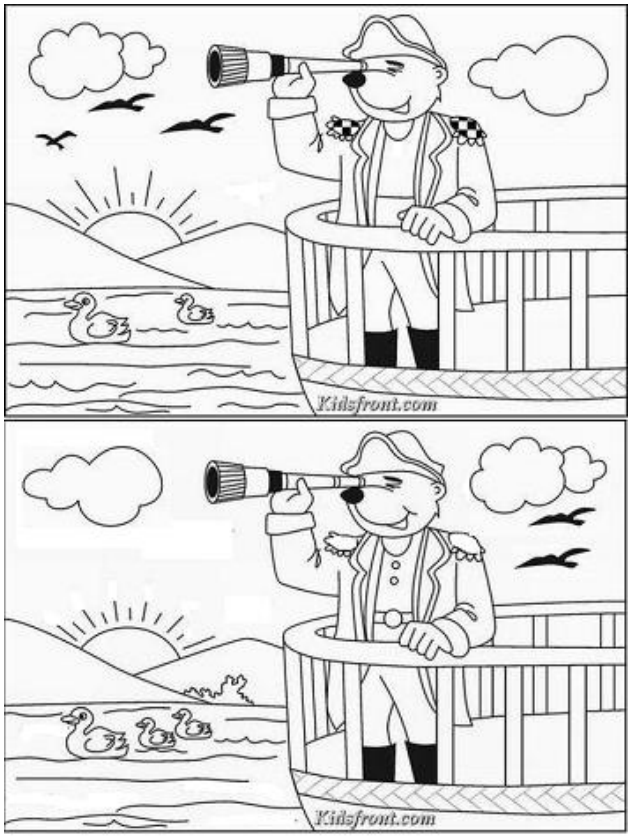


Modeling Sessions



Synchronizing Models and Code





Update ONLY when it hurts

What About Tools?

- Use the simplest tool for the job at hand

get

Drive how you do the job

Common Misconceptions About CASE Tools

- Agile modelers don't use CASE tools
  - Agile modelers use the simplest tool, and if the simplest tool for the job is a CASE tool, then that is what will be used
- UML requires CASE tools
  - Not true. UML drawings are often done by hand
- You start modeling with CASE tools
  - Typically modeling is started with a simple tool (e.g. flip charts) and then you migrate to a CASE tool if needed (e.g. to create persistent models)
- The CASE tool is the master
  - Not true. Once code is either generated from the model or written by hand, the code is the master. One tough decision that has to be made is should the model be updated to reflect the code?

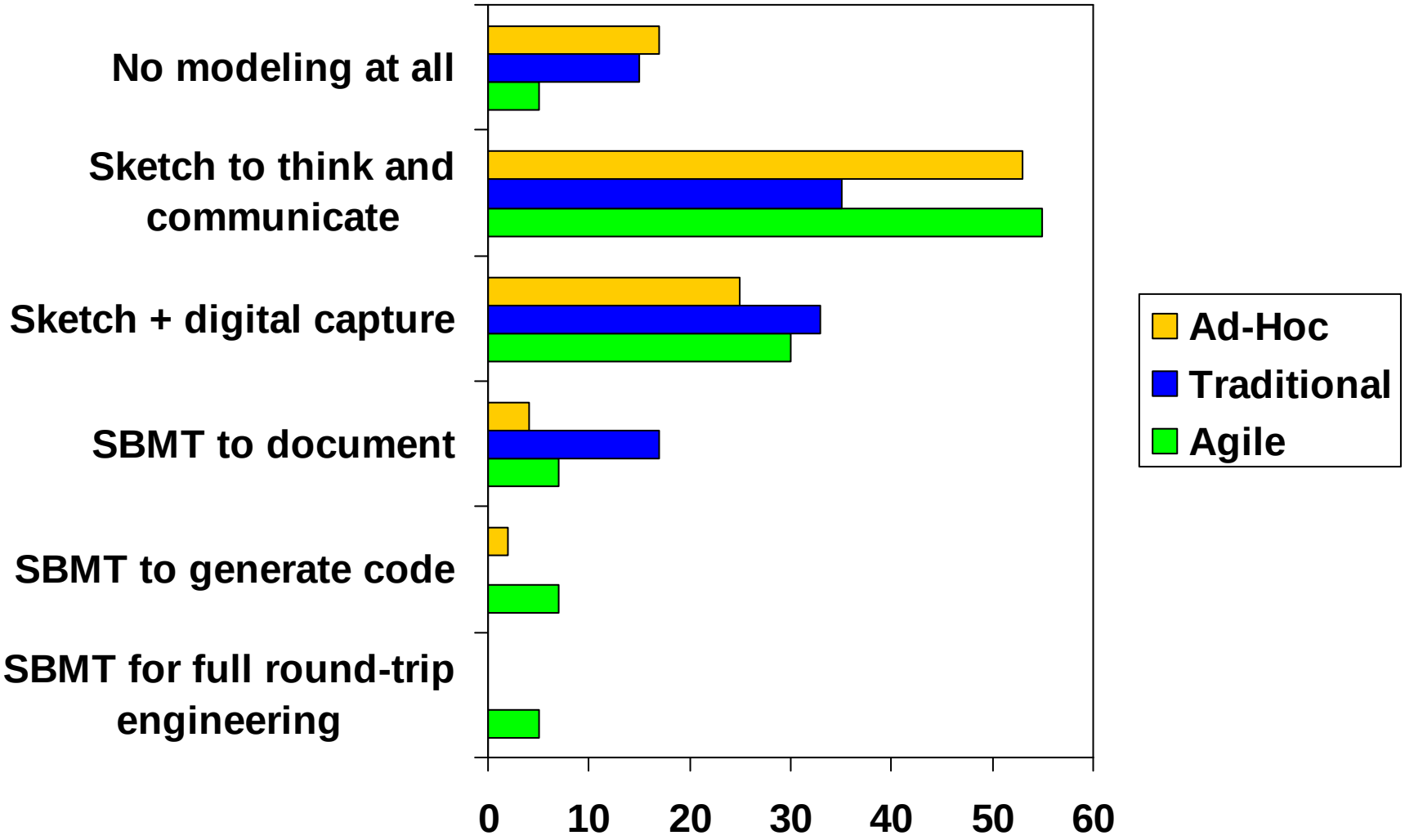
Survey Says….

- DDJ Modeling and Documentation survey
  - Currently in progress
  - 274 respondents so far
- Topics:
  - Top 5 modeling tools
  - Why people model
  - Primary application of tools
  - What documents we produce

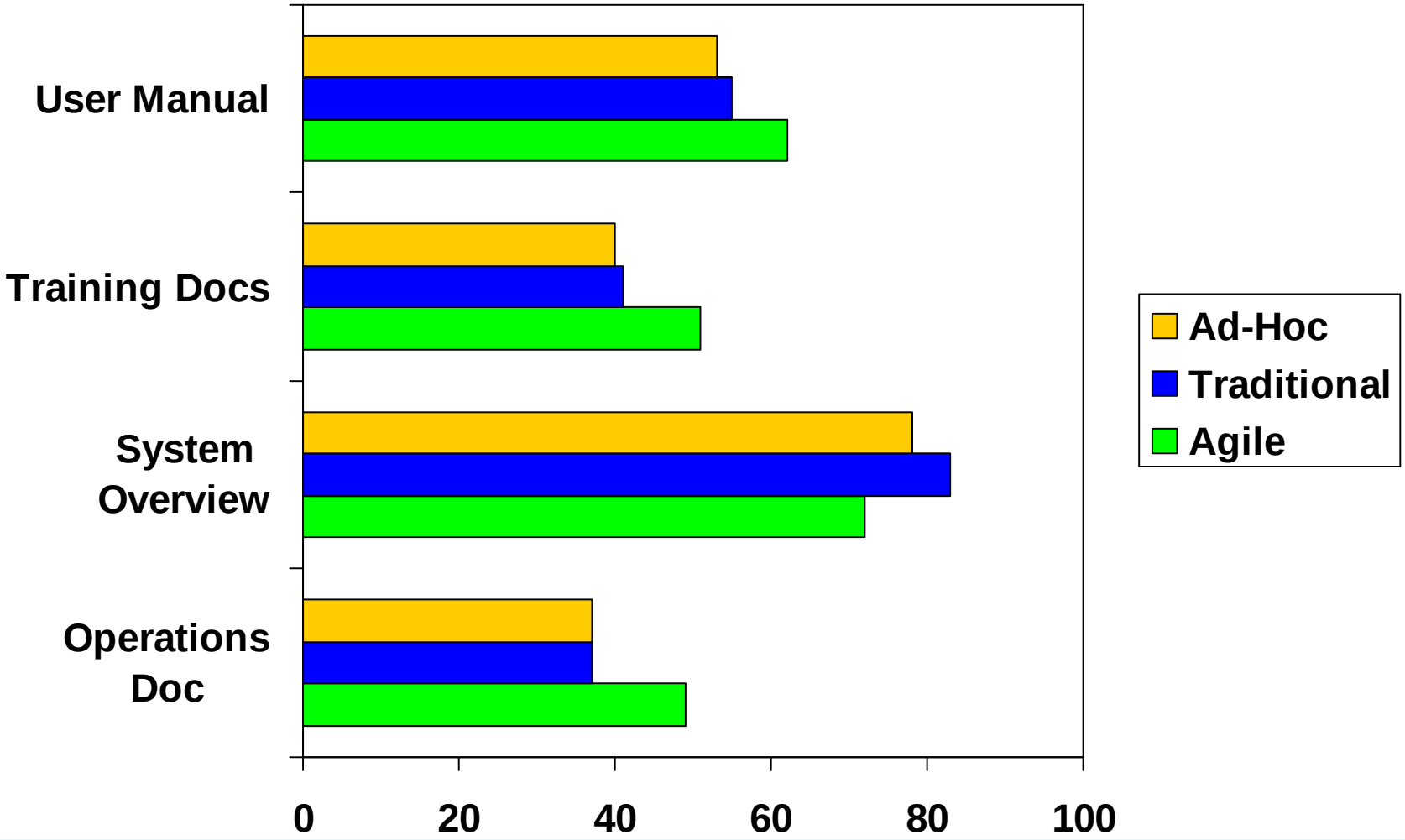
Top 5 Tools

Agile	Traditional	Ad-Hoc
Word Processors (89%)	Drawing Tools (89%)	Word Processors (83%)
Drawing Tools (83%)	Word Processors (89%)	Drawing Tools (74%)
Individual Whiteboards (54%)	Shared Whiteboards (54%)	Post-It Notes (46%)
Wikis (48%)	Individual Whiteboards (52%)	Shared Whiteboards (44%)
Post-It Notes (48%)	Software-based modeling tools (37%)	Individual Whiteboards (43%)
	Post-It Notes (37%)	

Primary Approach to Modeling (%)



Percentage of Teams Creating Deliverable Documentation



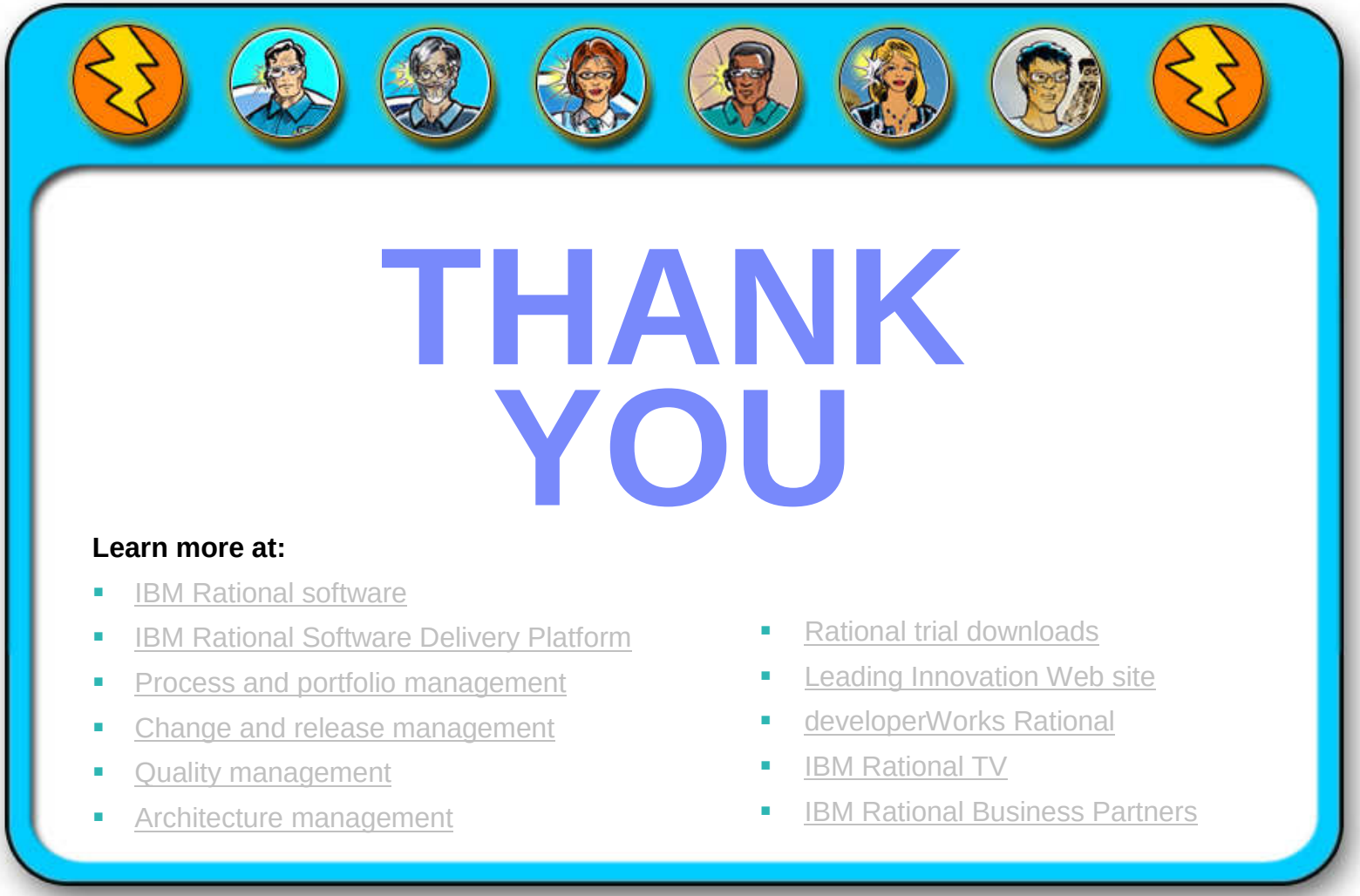
You are Engaged in Agile Modeling if…

- Your customers/users are active participants
- Changing requirements are welcomed and acted upon
- You work on the highest priority requirements first
- You take an iterative and incremental approach to modeling
- Your primary focus is the development of software, not documentation or models themselves
- You model as a team where everyone's input is welcome
- You actively try to keep things as simple as possible
- You discard most of your models as development progresses
- Customers/business owners make business decisions; developers make techical decisions
- The model's content is recognized as being significantly more important thatn the format/representation of that content
- How you test what you describe with your model(s) is a critical issue continually considered as you model

References and Recommended Reading

- www.agilealliance.com
- www.agilemodeling.com
- www.agiledata.org
- www.databaserefactoring.com
- www.enterpriseunifiedprocess.com
- Ambler, S.W. (2002). *Agile Modeling: Effective Practices for XP and the UP*. New York: John Wiley & Sons.
- Ambler, S.W. (2003). *Agile Database Techniques*. New York: John Wiley & Sons.
- Ambler, S.W. (2004). *The Object Primer 3rd Edition: AMDD with UML 2*. New York: Cambridge University Press.
- Ambler, S.W. and Sadalage, P.J. (2006). *Refactoring Databases: Evolutionary Database Design*. Reading, MA: Addison Wesley Longman, Inc.
- Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide*. Reading, MA: Addison Wesley


QUESTIONS



THANK YOU

Learn more at:

- IBM Rational software
- IBM Rational Software Delivery Platform
- Process and portfolio management
- Change and release management
- Quality management
- Architecture management
- Rational trial downloads
- Leading Innovation Web site
- developerWorks Rational
- IBM Rational TV
- IBM Rational Business Partners