Importance of Rational ROSE in Software Development Process Models

Dr. Ahmad Al-Rababah

Rational ROSE

An introduction

The advantages of visual modeling

Modeling is a way of thinking about the problems using models organized around the real world ideas.

- understanding of various interrelationships of a system
- fastest way to delineate the complex relationships
- easier for developers, software architects and customers to communicate on a common platform

Why UML ?

- Large enterprise applications
- a way that enables
  - Scalability
  - security
  - robust execution under stressful conditions
- *code reuse*

- ROSE = Rational Object Oriented Software Engineering
- Rational Rose is a set of visual modeling tools for development of object oriented software.
- Rose uses the UML to provide graphical methods for non-programmers wanting to model business processes as well as programmers modeling application logic.
- facilitates use of the Unified Modeling Language (UML), Component Object Modeling (COM), Object Modeling Technique (OMT), and Booch ‘93 method for visual modeling.

When to use Rational ROSE

- Modeling can be useful at any point in the application development process.

- Initial Design Work (Requirement Analysis and Definition)
  - Use Cases
  - Class Diagrams
  - Sequence Diagram

  - Generality is Good in early design.

When to use Rational ROSE

- Refinement of Early Models (System & Software Design)
- Introduced in Middle of Project
  - Rational Rose includes tools for reverse engineering as well as forward engineering of classes and component architectures.
  - You can gain valuable insights to your actual constructed architecture and pinpoint deviations from the original design.
  - Rose offers a fast way for clients and new employees to become familiar with system internals

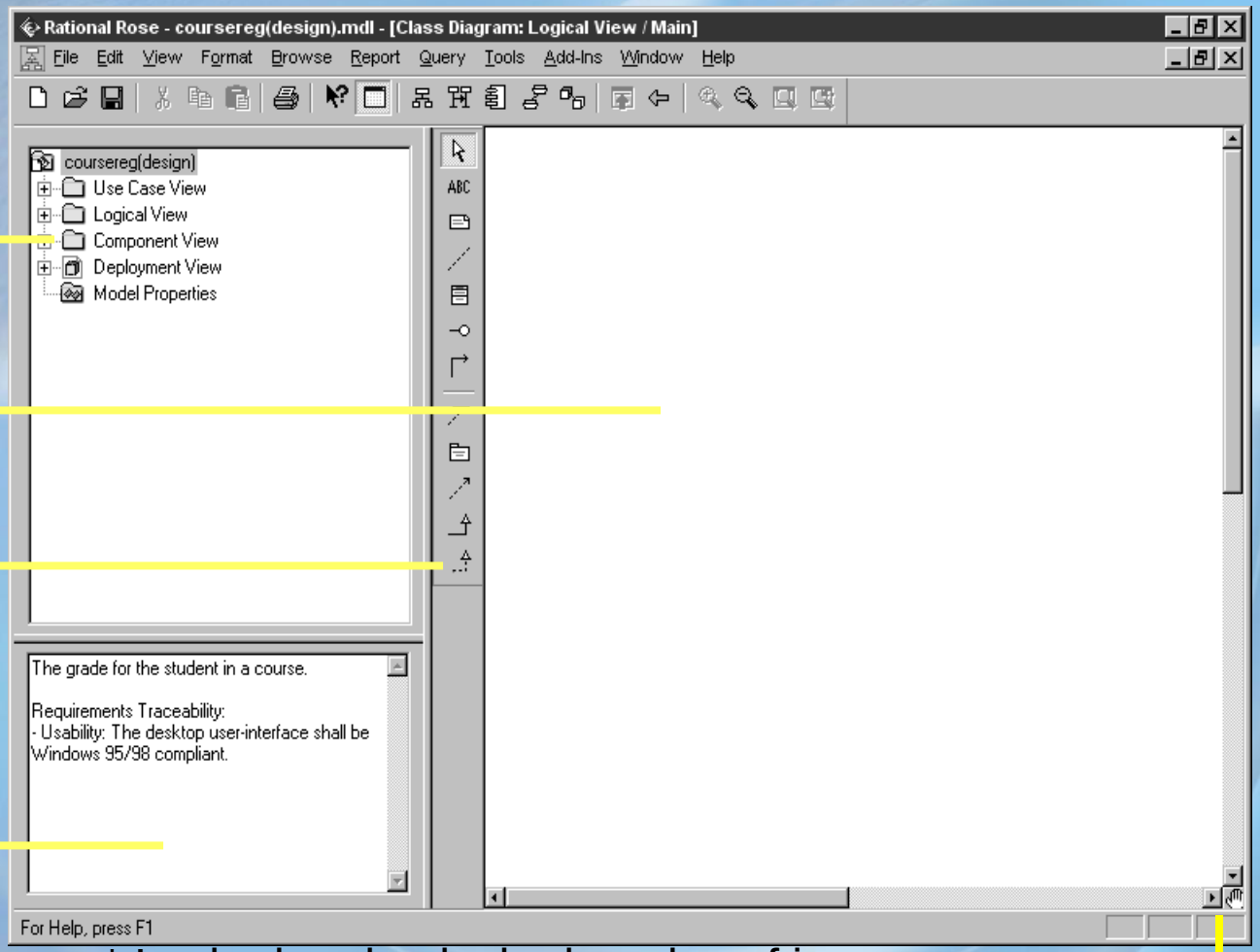
Rational ROSE INTERFACE

Browser

Diagram window

Diagram toolbar

Documentation Window docked/floating



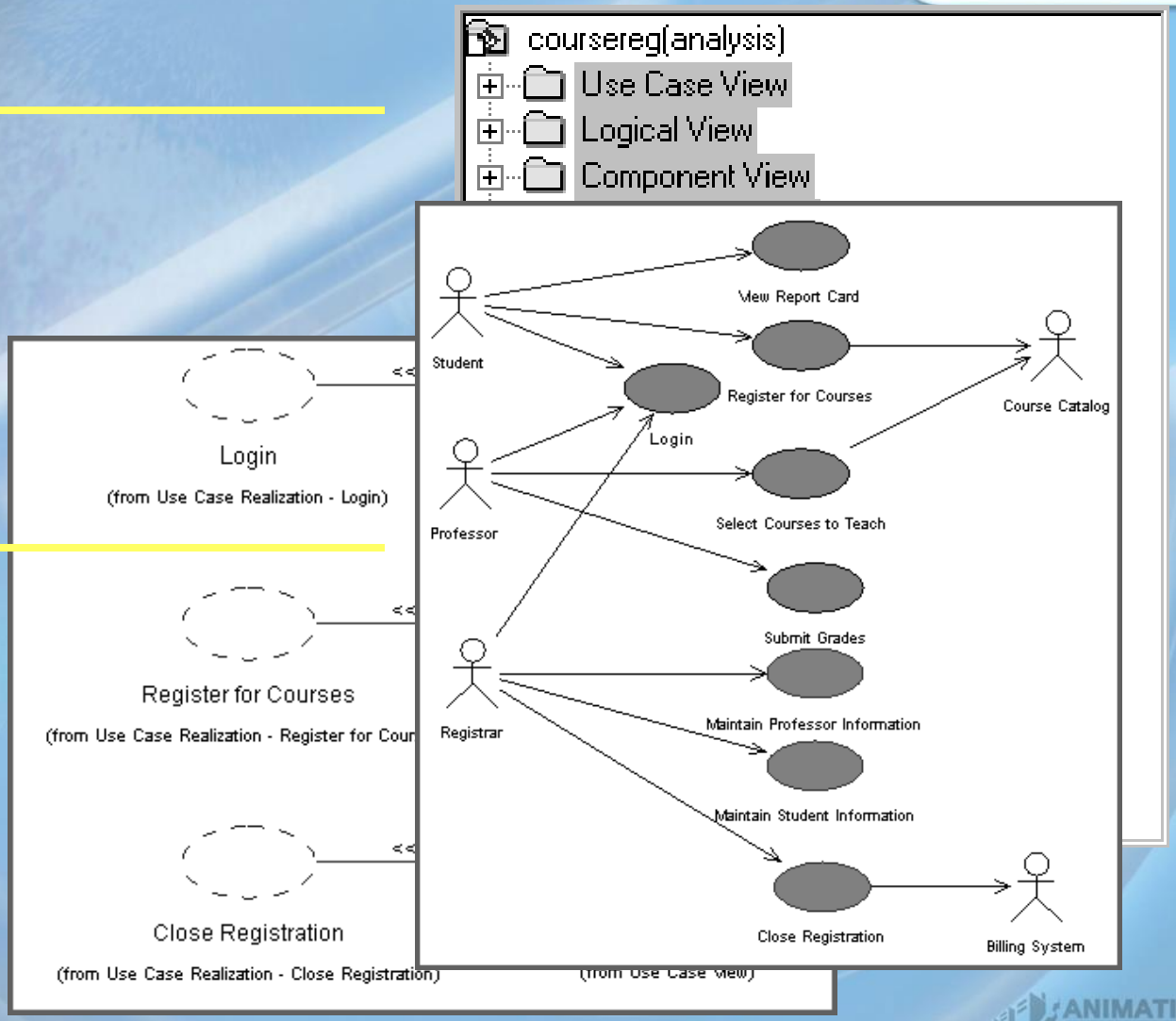
* Locked and unlocked modes of icons

** Customizing the tool bar

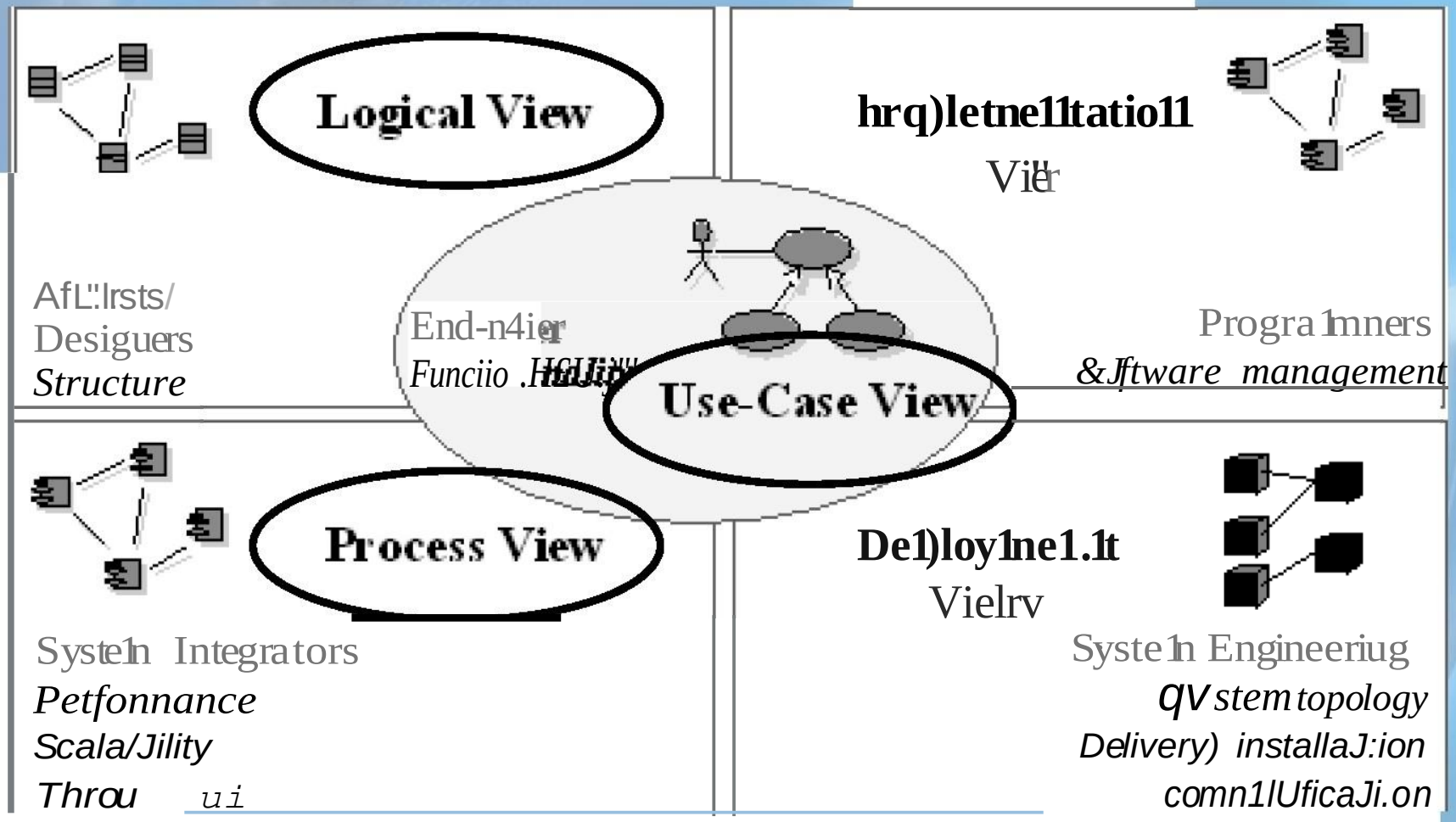
Views and Diagrams ???

Views

Diagrams



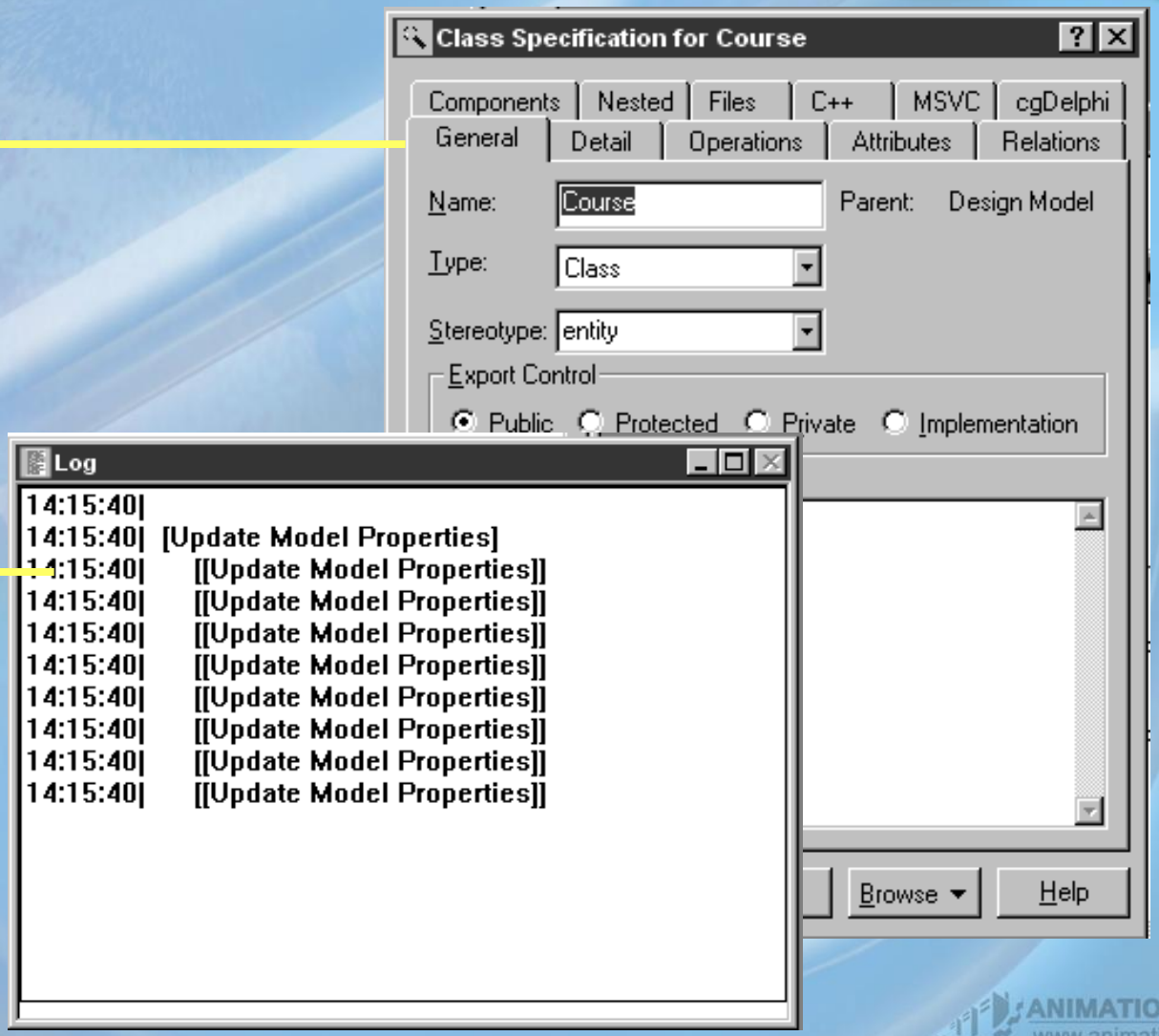
The different Views



Rational Rose Interface

Specification window

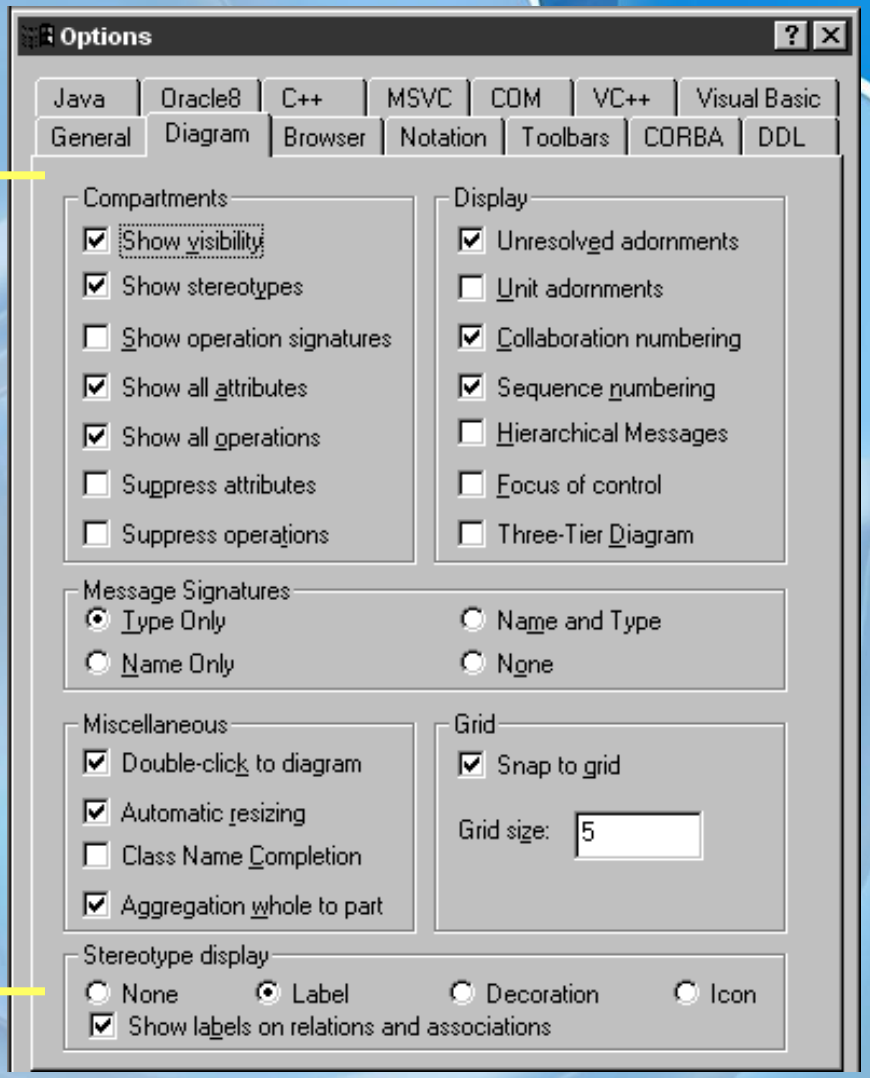
Log window



Options window

Options window

Stereotype display



Using the browser

- Hiding and Displaying the Browser
- Positioning the Browser
  - » Docked and floating
- Expanding and Collapsing the Browser Tree
- Selecting Multiple Elements in the Browser
- Navigating a Model
- Creating and Editing Model Elements
- Naming an Element in the Browser

Rational ROSE DIAGRAMS

- Use Case
- Collaboration
- Sequence
- Class
- Statechart
- Activity
- Component
- Deployment

Deleting in Rational ROSE

- **Shallow Delete**
  - Click **Edit** > **Delete**
  - Press **CTRL + X**
  - Press the **DELETE** key

  Note: *If you perform a shallow delete on an element without a name, Rational Rose will delete the model element completely out of the model.*

- **Deep Delete**
  - Click **Edit** > **Delete** from Model
  - Press **CTRL + D**
  - Right-click on an element in the browser and then select **Delete**
  - from the shortcut menu

How to use Rational ROSE

- Selecting a diagram
- Right-clicking as short cut
- Adding diagram elements from toolbar and browser
- Setting up default stereotypes
- Idea about the Reverse engineering
- Deleting from a diagram and the browser

Use Case Diagram

Use Case Diagrams describe the functionality of a system and users of the system. These diagrams contain the following elements:

- Actors, which represent users of a system, including human users and other systems.
- Use Cases, which represent functionality or services provided by a system to users.

Class diagrams

Class Diagrams describe the static structure of a system, or how it is structured rather than how it behaves. These diagrams contain the following elements.

- Classes, which represent entities with common characteristics or features. These features include attributes, operations and associations.
- Associations, which represent relationships that relate two or more other classes where the relationships have common characteristics or features. These attributes and operations.

Class Specification

- **Export Control** field.
  - »Public, private, protected, implementation
- **The Cardinality Concurrency**
- A class concurrency defines its semantics in the presence of multiple threads of control.

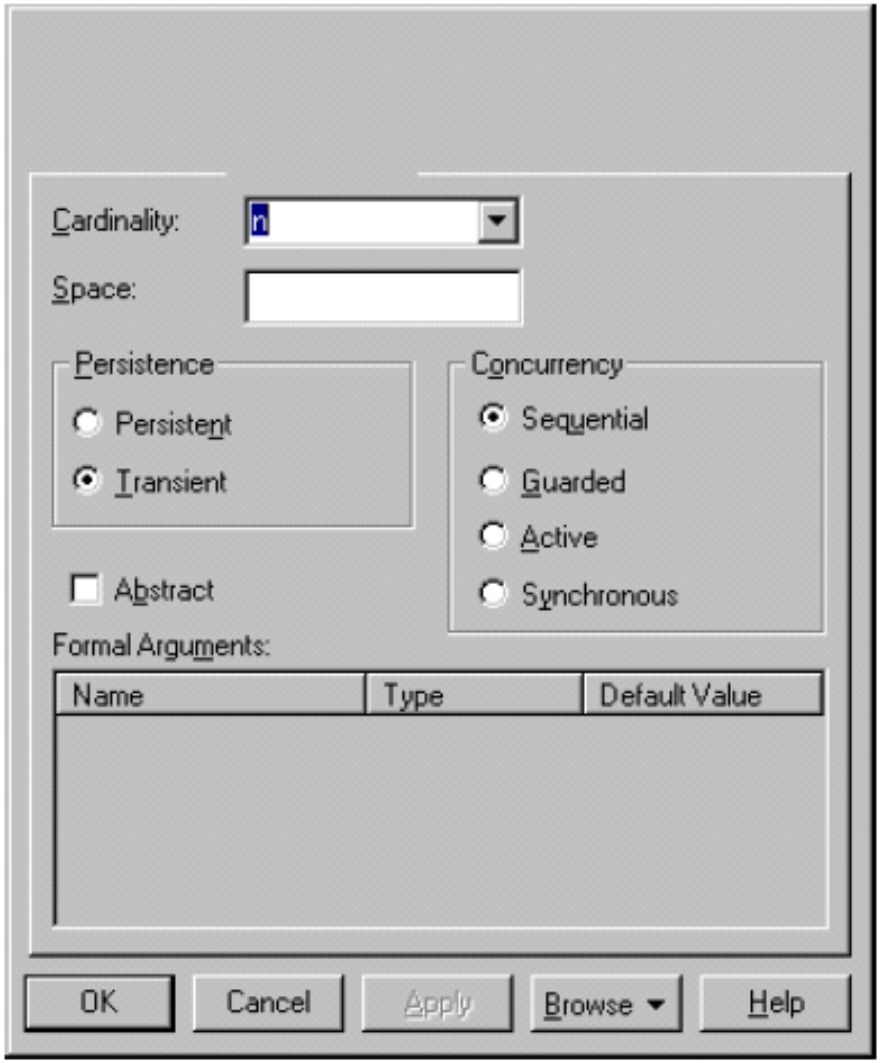


Figure 18 Class Specification—Detail Tab

Object Diagram

Object Diagrams describe the static structure of a system at a particular time. Whereas a class model describes all possible situations, an object model describes a particular situation. Object diagrams contain the following elements:

- Objects, which represent particular entities. These are instances of classes.
- Links, which represent particular relationships between objects. These are instances of associations.

Sequence Diagram

Sequence Diagrams describe interactions among classes. These interactions are modeled as exchange of messages. These diagrams focus on classes and the messages they exchange to accomplish some desired behavior. Sequence diagrams are a type of interaction diagrams. Sequence diagrams contain the following elements:

- Class roles, which represent roles that objects may play within the interaction.
- Lifelines, which represent the existence of an object over a period of time.
- Activations, which represent the time during which an object is performing an operation.
- Messages, which represent communication between objects.

Collaboration Diagrams

Collaboration Diagrams describe interactions among classes and associations. These interactions are modeled as exchanges of messages between classes through their associations. Collaboration diagrams are a type of interaction diagram. Collaboration diagrams contain the following elements.

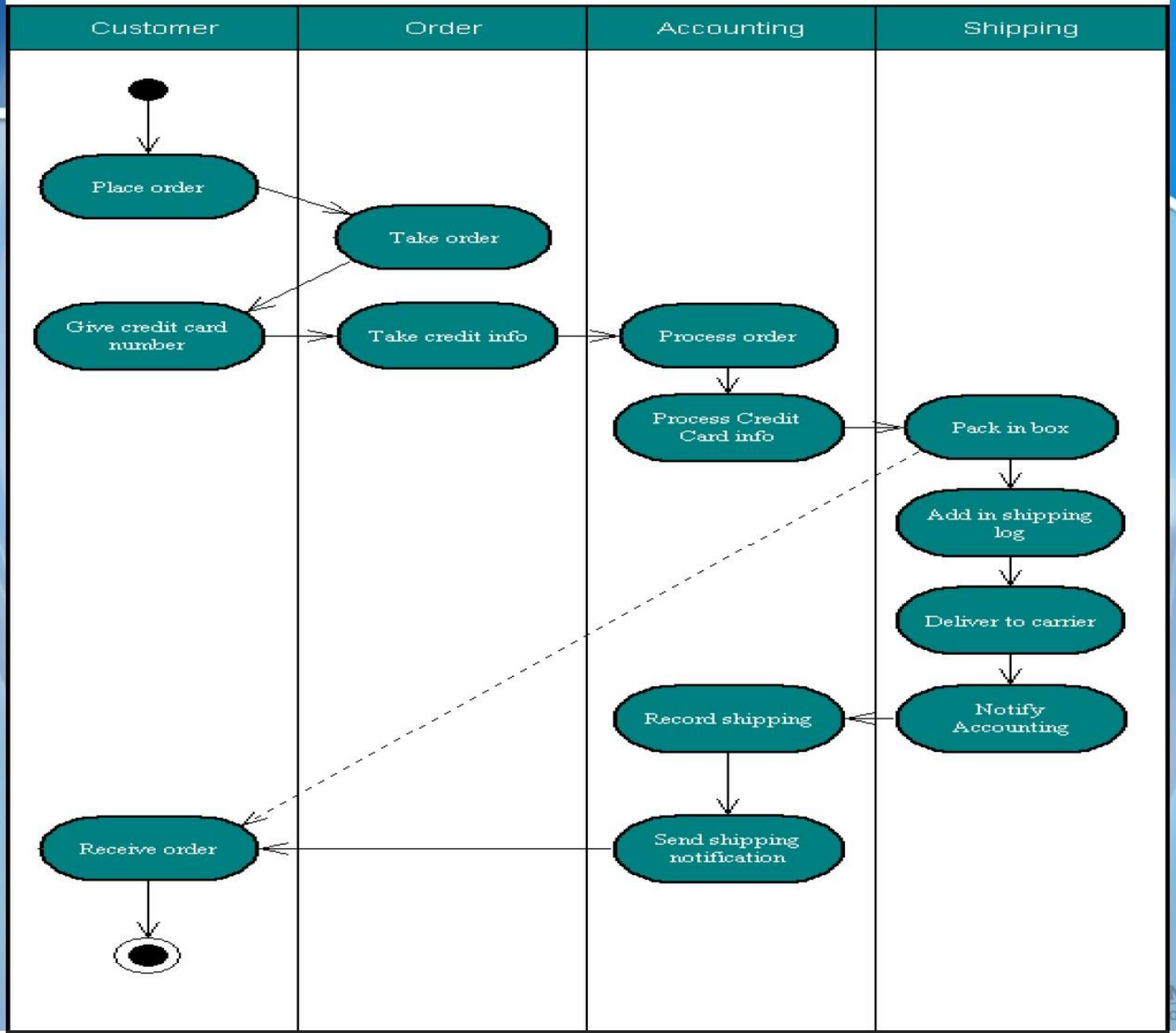
- Class roles, which represent roles that objects may play within the interaction.
- Association roles, which represent roles that links may play within the interaction.
- Message flows, which represent messages sent between objects via links. Links transport or implement the delivery of the message.

Activity Diagrams

Activity diagrams describe the activities of a class. These diagrams are similar to statechart diagrams and use similar conventions, but activity diagrams describe the behavior of a class in response to internal processing rather than external events as in statechart diagram.

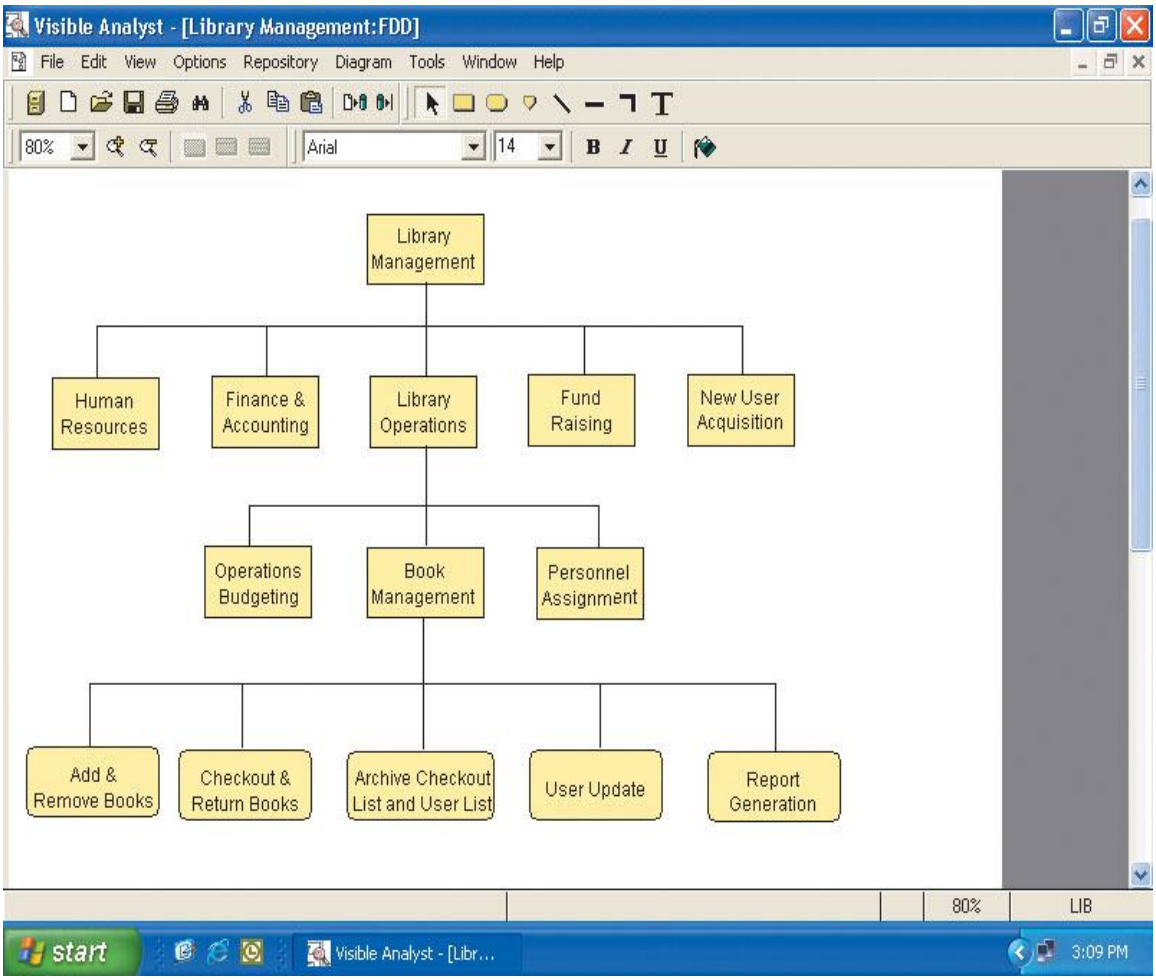
- Swimlanes, which represent responsibilities of one or more objects for actions within an overall activity; that is, they divide the activity states into groups and assign these groups to objects that must perform the activities.
- Action States, which represent atomic, or noninterruptible, actions of entities or steps in the execution of an algorithm.
- Action flows, which represent relationships between the different action states of an entity
- Object flows, which represent the utilization of objects by action states and the influence of action states on objects.

UML Activity Diagram: Order Processing

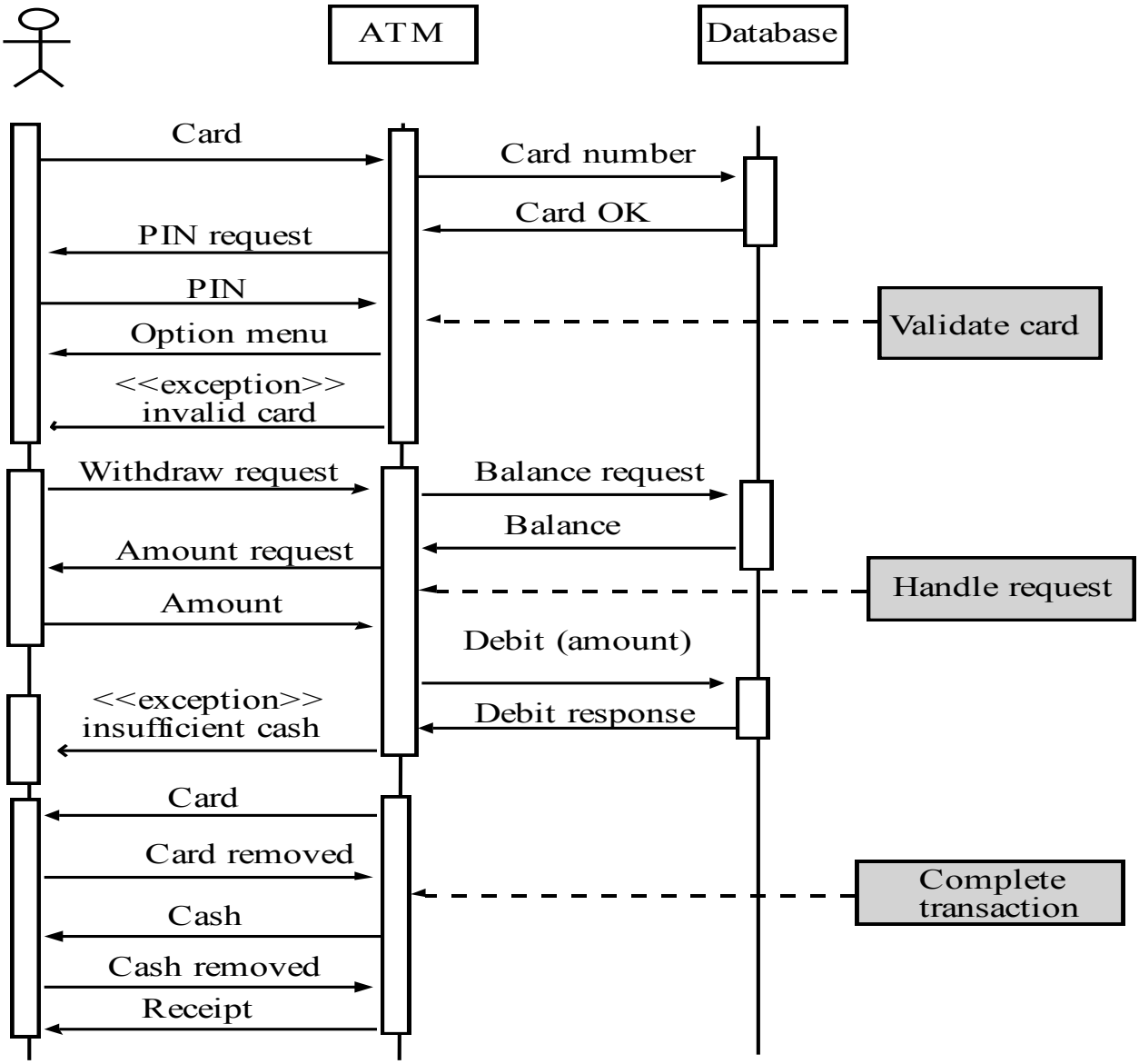


Examples:

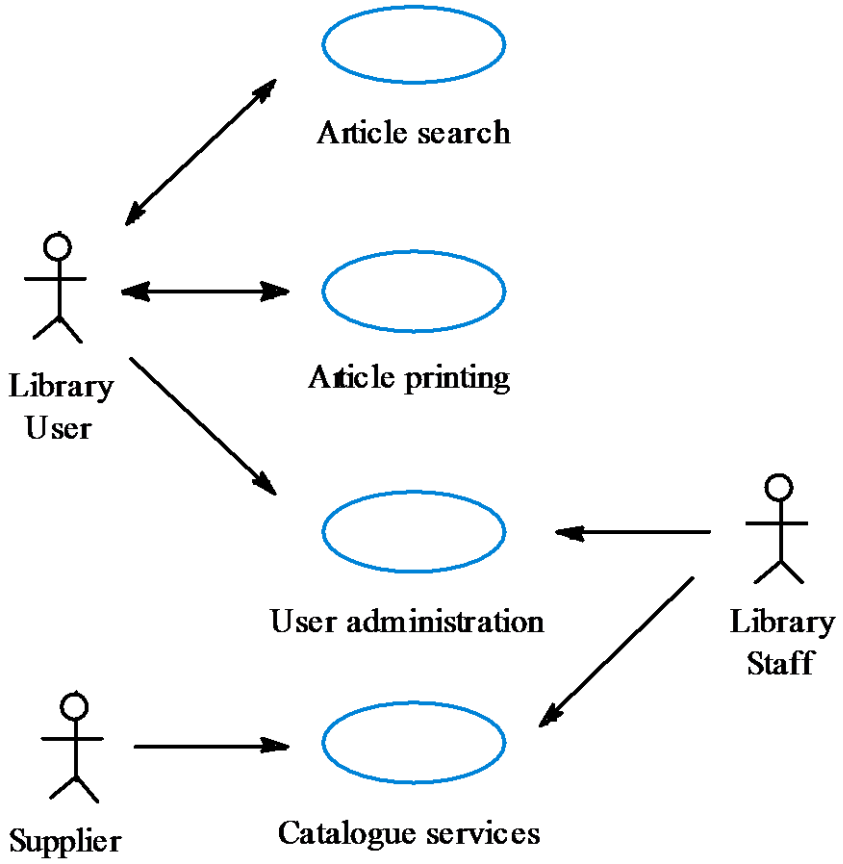
- *Functional Decomposition Diagrams*



- *Sequence diagram of ATM withdrawal*



• *LIBSYS use cases*



References

- UML Home Page - http://www.platinum.com/corp/uml/uml.htm

- Online Tutorials for Rational Rose - http://www.rational.com/products/rose/gstart/online.jtmpl

- Rose Whitepapers http://www.rational.com/products/rose/prodinfo/whitepapers/index.jtmpl

- Rose Architect E-Magazine http://www.rosearchitect.com/mag/index.shtml

- **Visual modeling with Rational Rose and UML**
  Source Addison Wesley Object Technology Series Year of Publication: 1998 ISBN:0-201-31016-3
  Author Terry Quatrani
  Publisher Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA