McMaster University

Inspiring Innovation and Discovery

Rational Rose Tutorial

Anis Yousefi

Teaching Assistant

SE3KO4 / SE3MO4

Software Development for Computer and Electrical Engineering

Email: yousea2@mcmaster.ca

Objectives

- Get to know Rational Rose
- Get Familiar with general functions of Rational Rose for Modeling
- Create UML Diagrams with Rational Rose

Assumption

- You are familiar with Unified Modeling Language (UML)
  - Either
    - Read a book on UML
    - Been trained in UML
    - Used UML on work project
- You are familiar with object oriented software

Access To Rational Rose

- Available at student lab
- Can download full version from Rational
  - 15-day trial license

What is Rational Rose?

- ROSE = Rational Object Oriented Software Engineering
- Rational Rose is a set of visual modeling tools for development of object oriented software.
- Visual Modeling is the process of graphically depicting the system to be developed
  - Presenting essential details
  - Filtering out non-essential details
  - Viewing the system from different perspectives

Why Model?

- The UML models act as an architectural blueprint for software development.
- Good models:
    - Identify requirements and communicate information
    - Allows focus on how system components interact, without get bogged out in specific details
    - Allows you to see relationships among design components
    - Improves communication across your team through the use of common graphical language

Visual Modeling Tools May Help Mitigate these Problems

- Software that poorly fits user needs
- Inability to deal with changing requirements
- Software integration problems
- Discovery of serious flaws too late in the project
- Software that is hard to maintain and extend

When Should ROSE be Used?

- 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

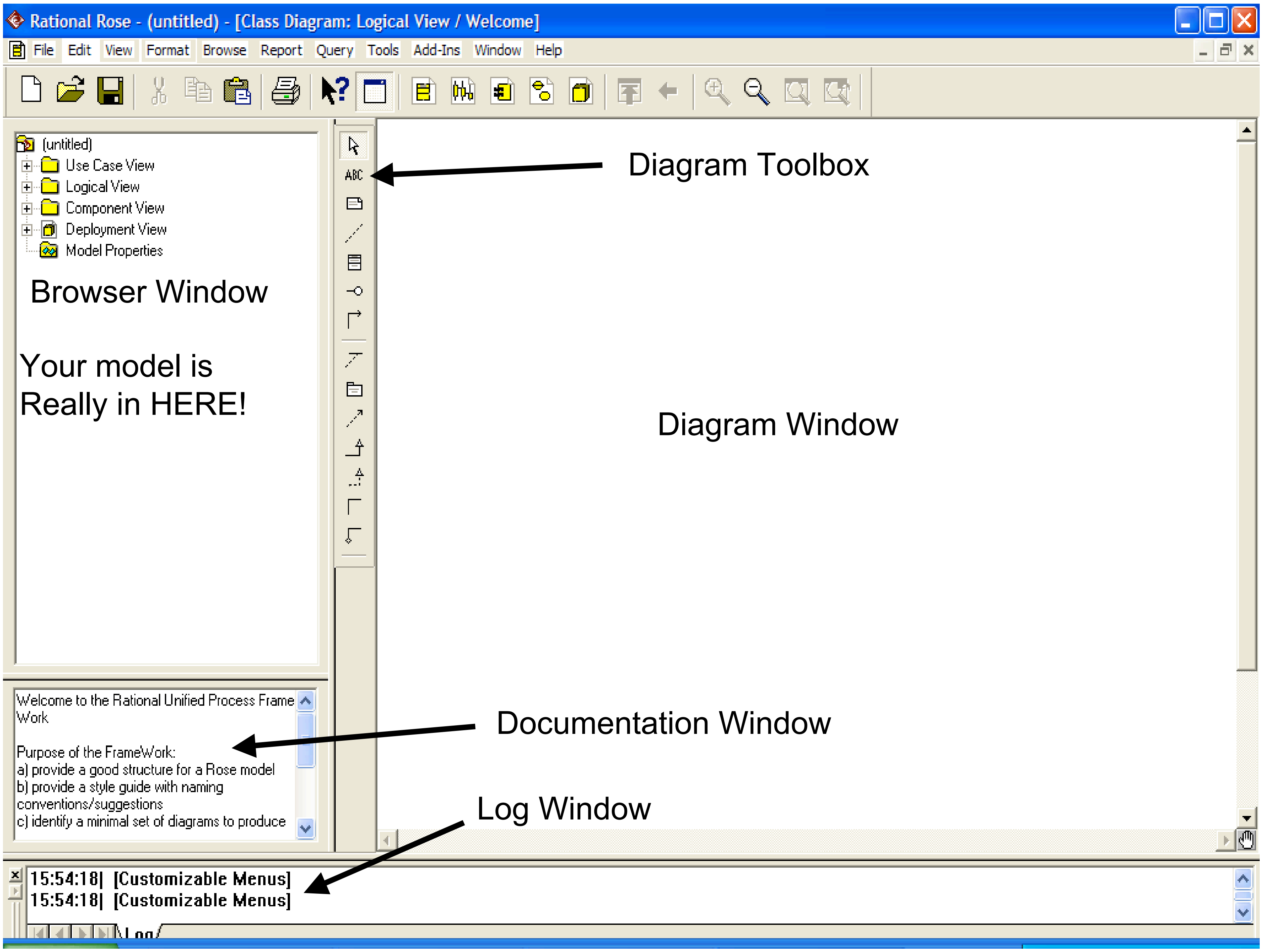
Spend your time dealing with issues such as the **planned uses** of software system and **how** you will implement a programming environment to address these issues--not where and how you are going to place the buttons on your first screen.

When Should ROSE be Used?

- 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

Rose

- Rose Enterprise:
  - Supports multiple languages, including VC++, VB, Java, CORBA
- Rose GUI:
  - Standard ToolBar
  - Diagram ToolBox
  - Browser
  - Diagram Window
  - Documentation Window
  - Specifications
  - Log Window



Welcome to the Rational Unified Process Frame Work

Purpose of the FrameWork:
a) provide a good structure for a Rose model
b) provide a style guide with naming conventions/suggestions
c) identify a minimal set of diagrams to produce

```
15:54:18|  [Customizable Menus]
15:54:18|  [Customizable Menus]
```

Rational Rose Interface

- The Rose standard toolbar (near the top of the window)
  - is always displayed - independent of the current diagram type.
  - While in Rose, place your cursor over the toolbar to display a tooltip for each icon.

- The browser
  - a hierarchical navigational tool allowing you to view the names and icons representing diagrams and model elements.
  - The plus (+) sign next to an icon indicates the item is collapsed and additional information is located under the entry. Click on the + sign and the tree is expanded.
  - Conversely, a minus (-) sign indicates the entry is fully expanded.
  - If the browser is not displayed, select Browser from the View menu.

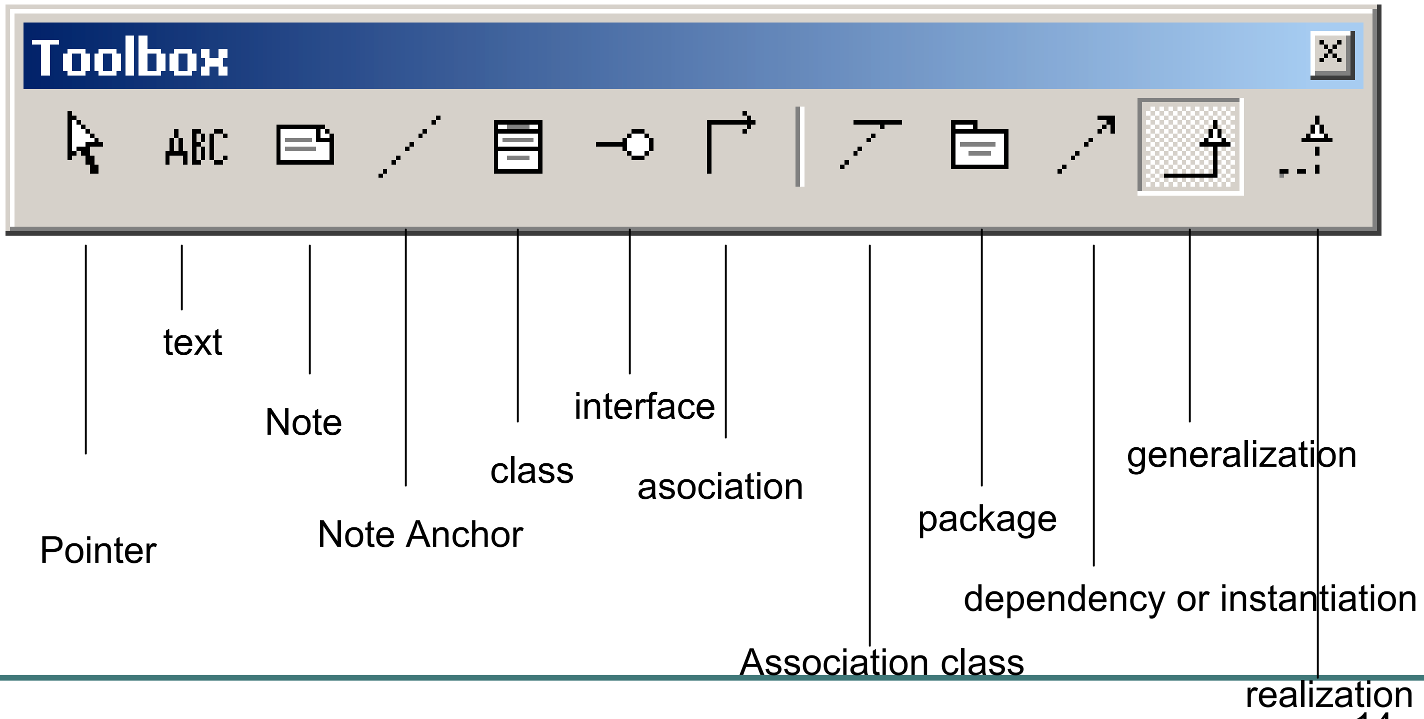
Rational Rose Interface

- Diagram window
  - Allows you to create, update, and model different diagrams, that are, graphical views of the model

- Diagram toolbar
  - Is unique to each diagram type and can be customized.
  - Is active only when a diagram is displayed.
  - May be visible or hidden; docked or floating.
  - As with the standard toolbar, placing your cursor on an icon displays the tooltip for that icon.

Toolbar for Class Diagrams

- Any element of a diagram can be created by
  - placing the mouse pointer over a Tool in the Toolbar
  - Drag&Drop over the diagram canvas



Rational Rose Interface

- The Specification window – (right click on Use Case View Package; Open Specification…)
  - Is a textual representation of a model element that allows you to view and manipulate the element's properties.
  - Note that information added to the documentation window is automatically added to the documentation field in the specification window.
- The Log window – (down at very bottom)
  - Reports progress, results, and errors
  - Right-click on Log window to see available actions

Models, Views, Diagrams

- Models themselves are constructed using different views and diagrams to accurately depict different stakeholder perspectives and the system's building blocks, respectively.
- **Models** are complete representations of the system.
- **Views** allow different stakeholders to see the system from their own perspectives
  - Views contain Models…
    - E.g. Logical View contains analysis model, business object model, design model (Sometimes models can contain 'views' too…)
  - Models generally contain a number of diagrams – some of these terms are 'used' interchangeably…
    - E.g. Design model contains class diagrams, sequence diagrams, and a number of others….
- **Diagrams:** means by which we view of the system.
  - Different building blocks (model elements) for different types.
  - E.g.: classes, interfaces, collaborations, components, nodes, dependencies, generalizations, and associations.

Views

- Just as there are many views of a house under construction - the floor plan, the wiring diagram, the elevation plan, there are many views of a software project under development.

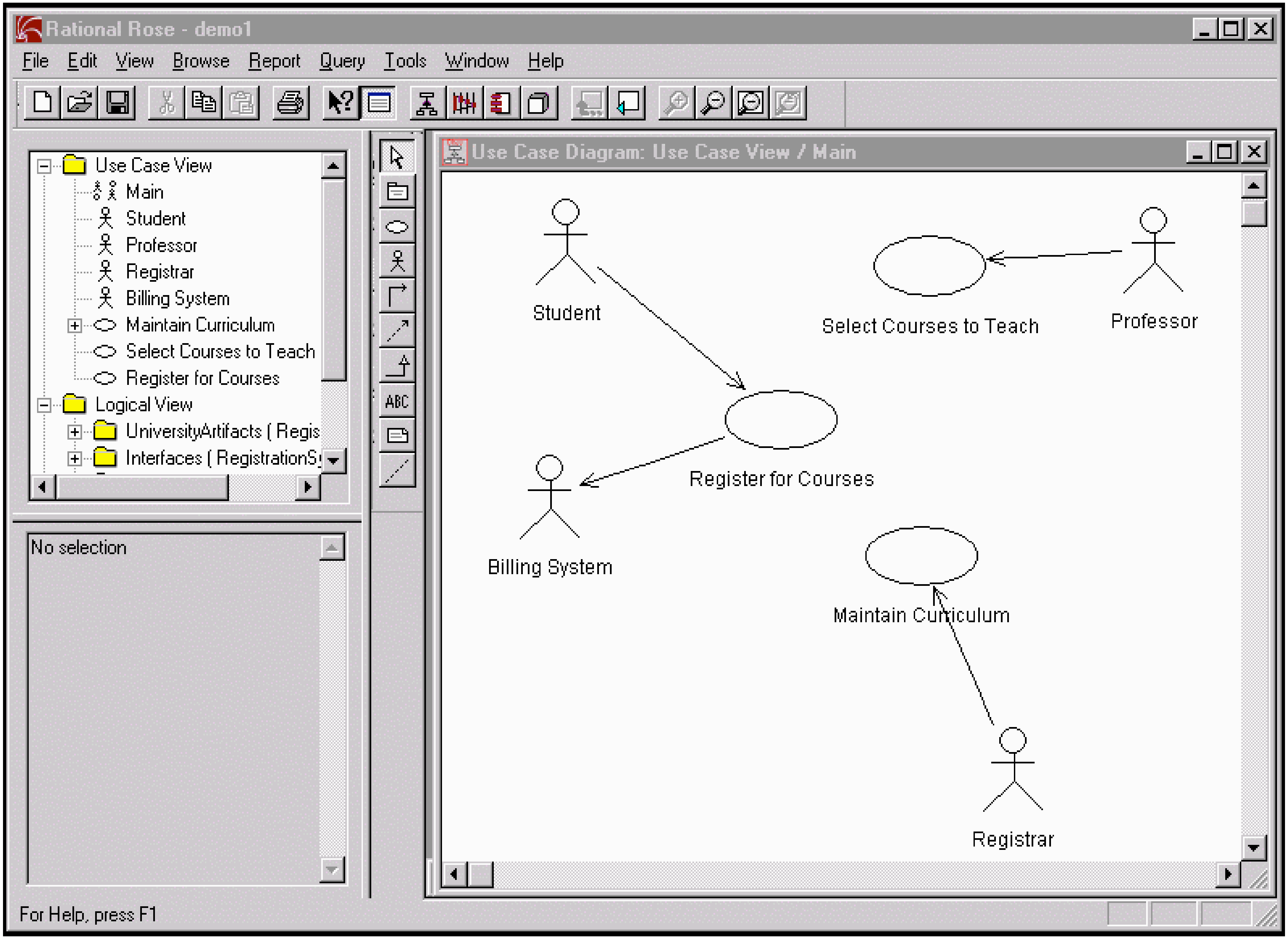
- Rational Rose is organized around the following views of a software project:
  - Use Case
  - Logical
  - Component
  - Deployment

  Each of these views presents a different aspect of the model and is explained in subsequent slides.

The use-case view

- The use-case view helps you to understand and use the system. This view looks at how actors and use cases interact.
- The diagrams in this view are:
  - Use-case diagrams
  - Sequence diagrams
  - Collaboration diagrams
  - Activity diagrams

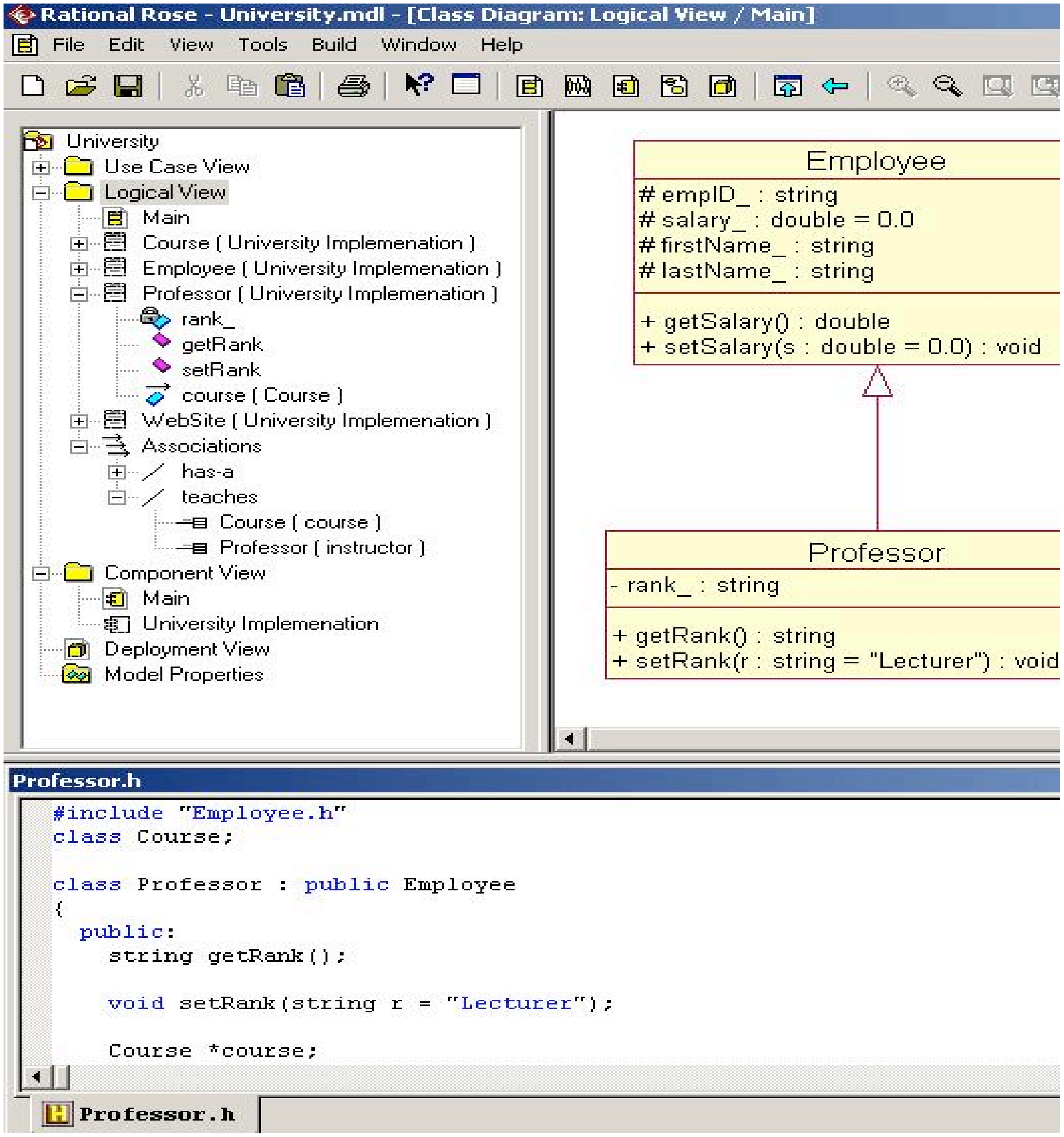
This view contains a Main diagram by default. Additional diagrams can be added throughout the analysis and design process.


Rational Rose - demo1
File Edit View Browse Report Query Tools Window Help
Use Case View
Main
Student
Professor
Registrar
Billing System
Maintain Curriculum
Select Courses to Teach
Register for Courses
Logical View
UniversityArtifacts (Regis
Interfaces (RegistrationS
No selection
ABC
Use Case Diagram: Use Case View / Main
Student
Select Courses to Teach
Professor
Register for Courses
Billing System
Maintain Curriculum
Registrar
For Help, press F1

The logical view

- The logical view addresses the functional requirements of the system.
- This view looks at classes and their relationships.
- The diagrams in this view are:
  - Class diagrams
  - Statechart diagrams

This view contains a Main diagram by default. Additional diagrams can be added throughout the analysis and design process.

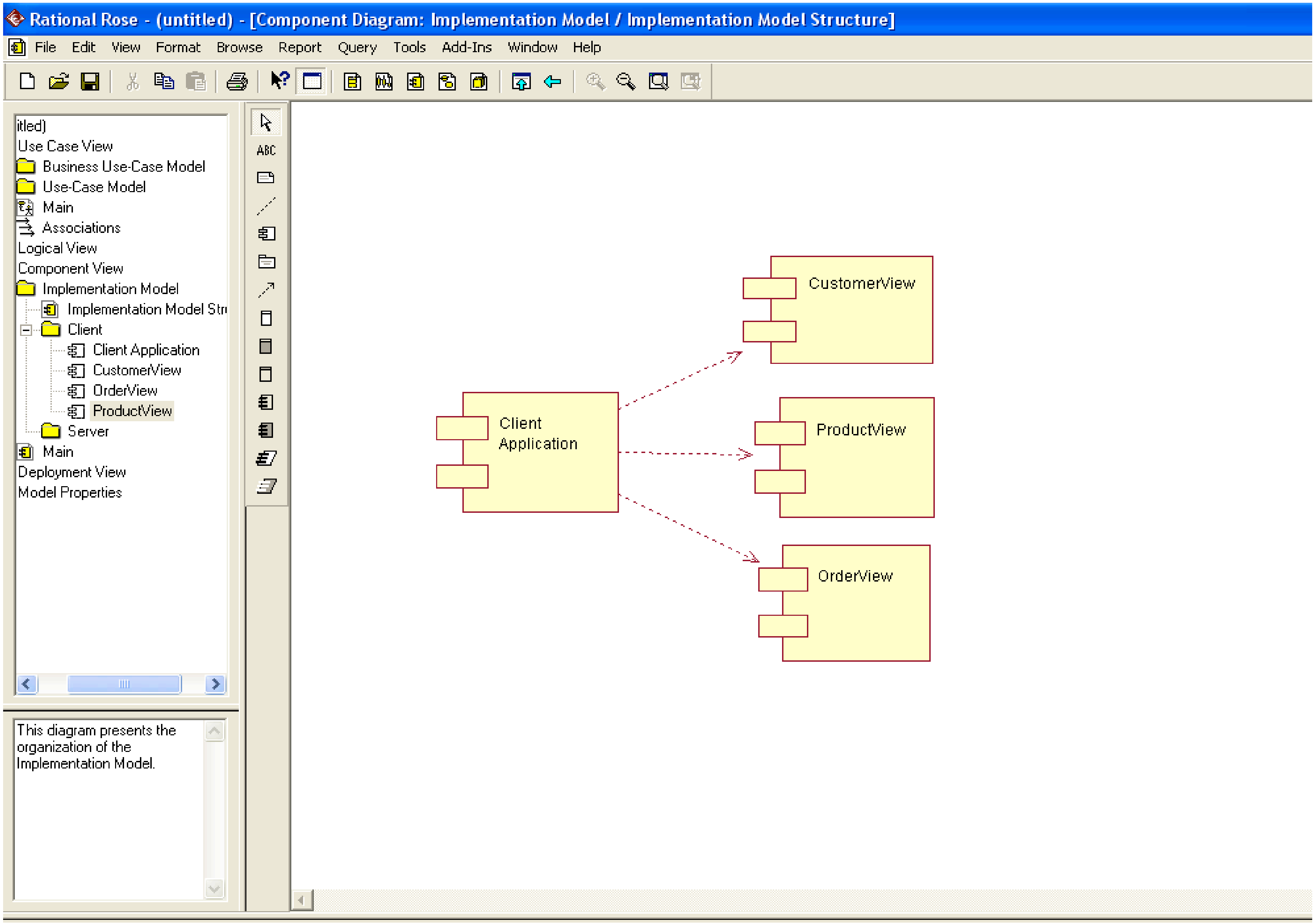


```
#include "Employee.h"
class Course;

class Professor : public Employee
{
  public:
    string getRank();

    void setRank(string r = "Lecturer");

    Course *course;
```

The component view

- The component view addresses the software organization of the system.
- This view contains information about the software, executable and library components for the system.
- This view contains only component diagrams.

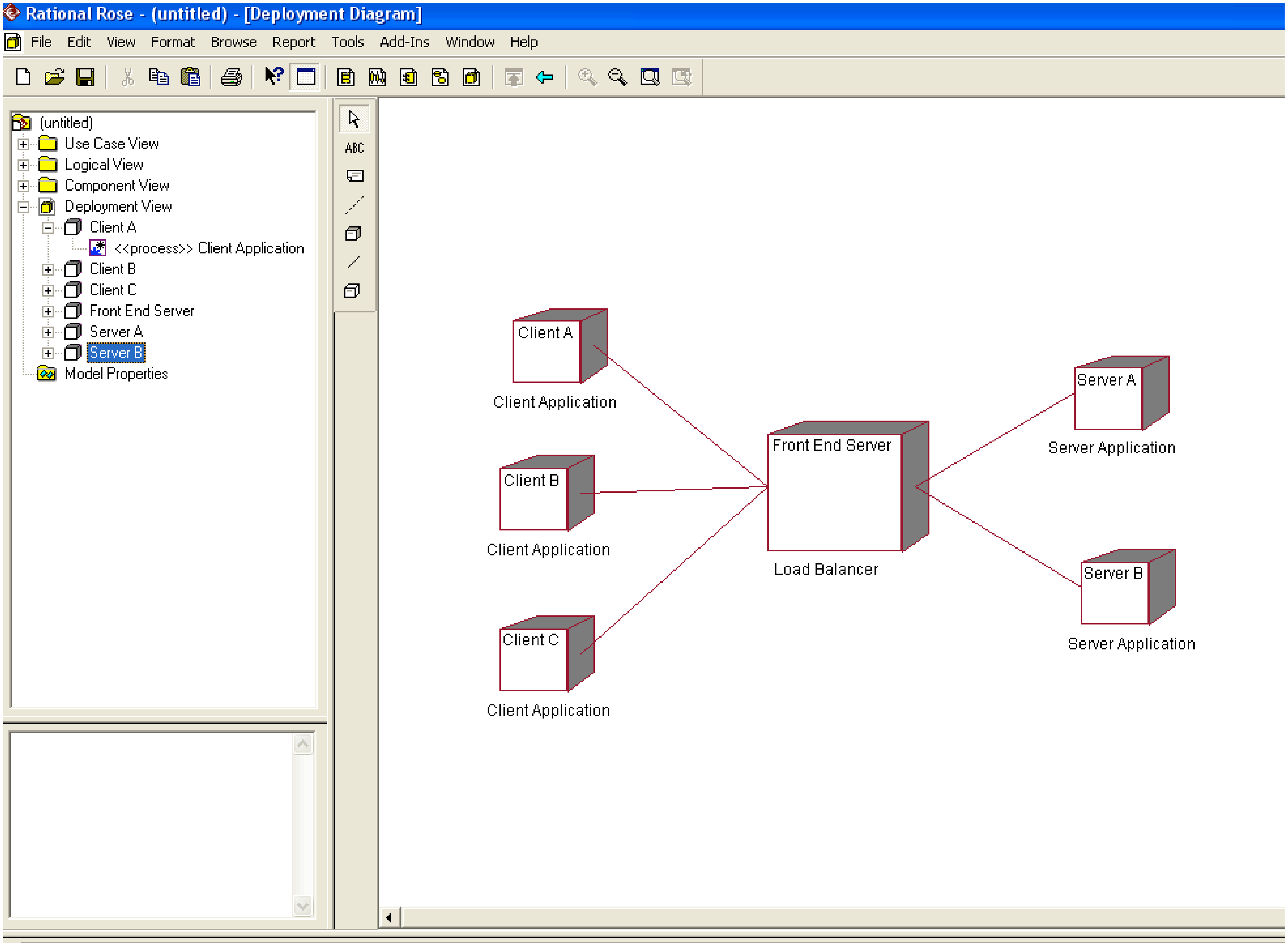
The component view contains a Main diagram by default. Additional diagrams can be added to this view throughout the analysis and design process.



This diagram presents the organization of the Implementation Model.

The deployment view

- The deployment view shows the mapping of processes to hardware.
- This type of diagram is most useful in a distributed architecture environment where you might have applications and servers at different locations.
- This view contains only one diagram -the deployment diagram.


Rational Rose - (untitled) - [Deployment Diagram]
File Edit View Format Browse Report Tools Add-Ins Window Help
(untitled)
Use Case View
Logical View
Component View
Deployment View
Client A
<<process>> Client Application
Client B
Client C
Front End Server
Server A
Server B
Model Properties
ABC
Client A
Client Application
Client B
Client Application
Client C
Client Application
Front End Server
Load Balancer
Server A
Server Application
Server B
Server Application

Diagrams

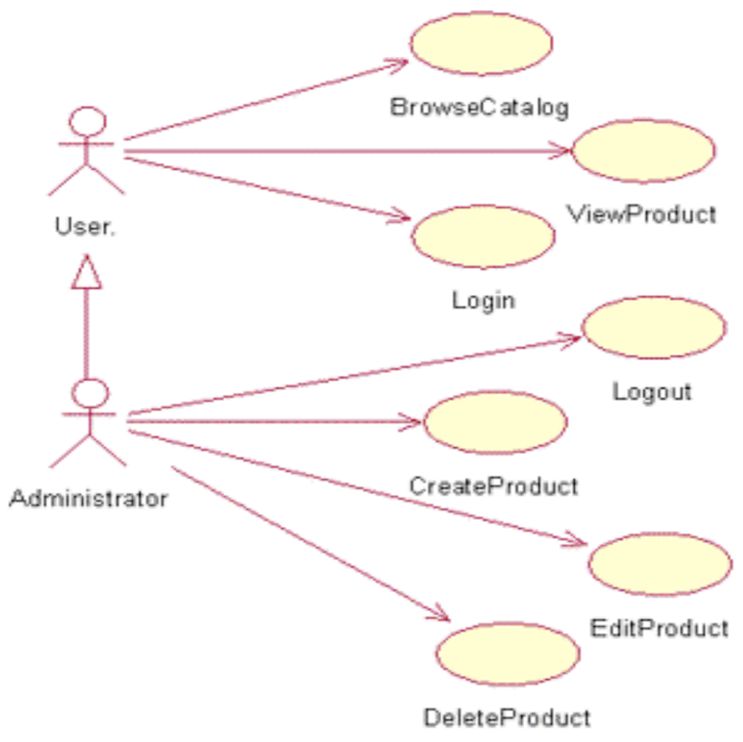
- Simply put, a diagram is a graphical representation of the elements of your system.
- Different diagram types allow you to view your system from multiple perspectives.
- You can create various types of diagrams in Rational Rose. The diagram types include:
  - Use-Case
  - Class
  - Activity
  - Statechart
  - Component
  - Deployment

  Each of these diagram types is explained in subsequent slides.

Use-case diagrams

- Use-case diagrams present a high-level view of system usage as viewed from an outsider's (actor's) perspective.
- These diagrams show the functionality of a system or a class and how the system interacts with the outside world.
- Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work.
- During the design phase, use-case diagrams specify the behavior of the system as implemented.
- Rose automatically creates a Main use-case diagram in the use-case view. There are typically many use-case diagrams in a single model.

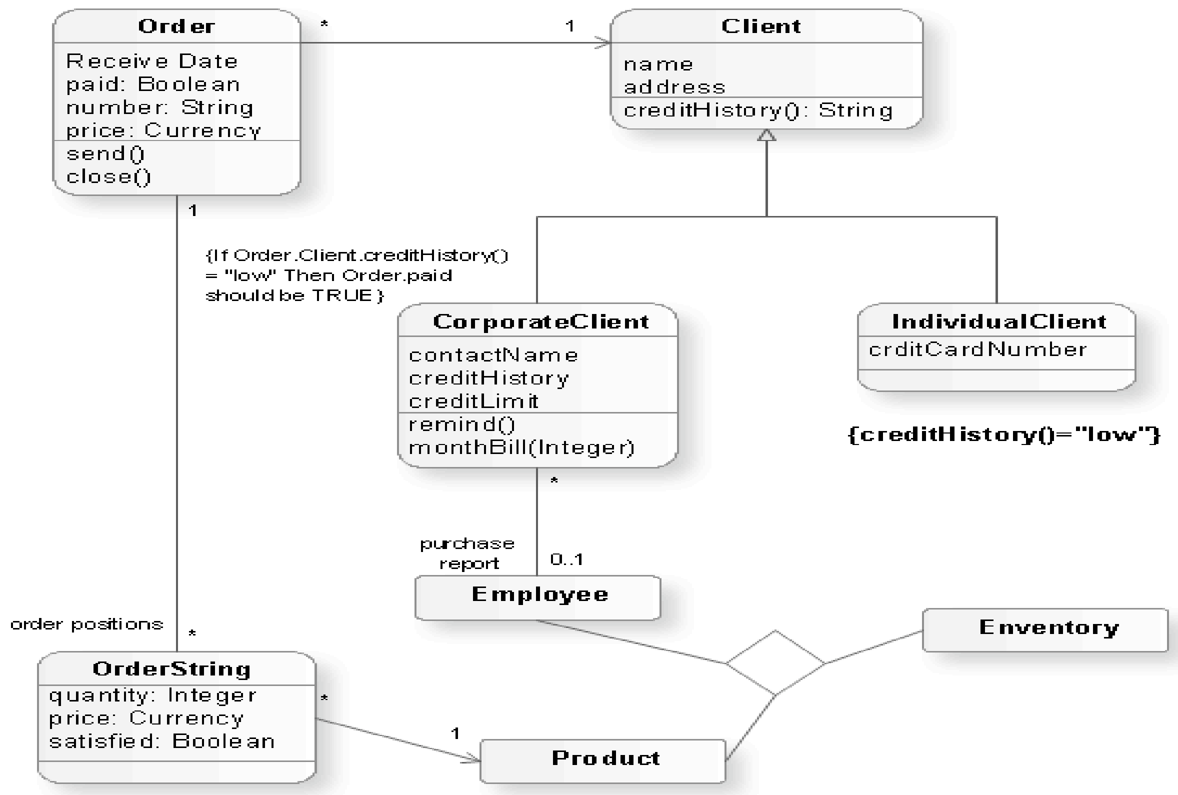
UC Diagram example



Class diagrams

- A class diagram helps you visualize the structural or static view of a system and is one of the most common diagram types.
- Class diagrams show the relationships among and details about each class.
- Class diagrams are also the foundation for component and deployment diagrams.
- Rose automatically creates a Main class diagram in the logical view. There are typically many class diagrams in a single model.

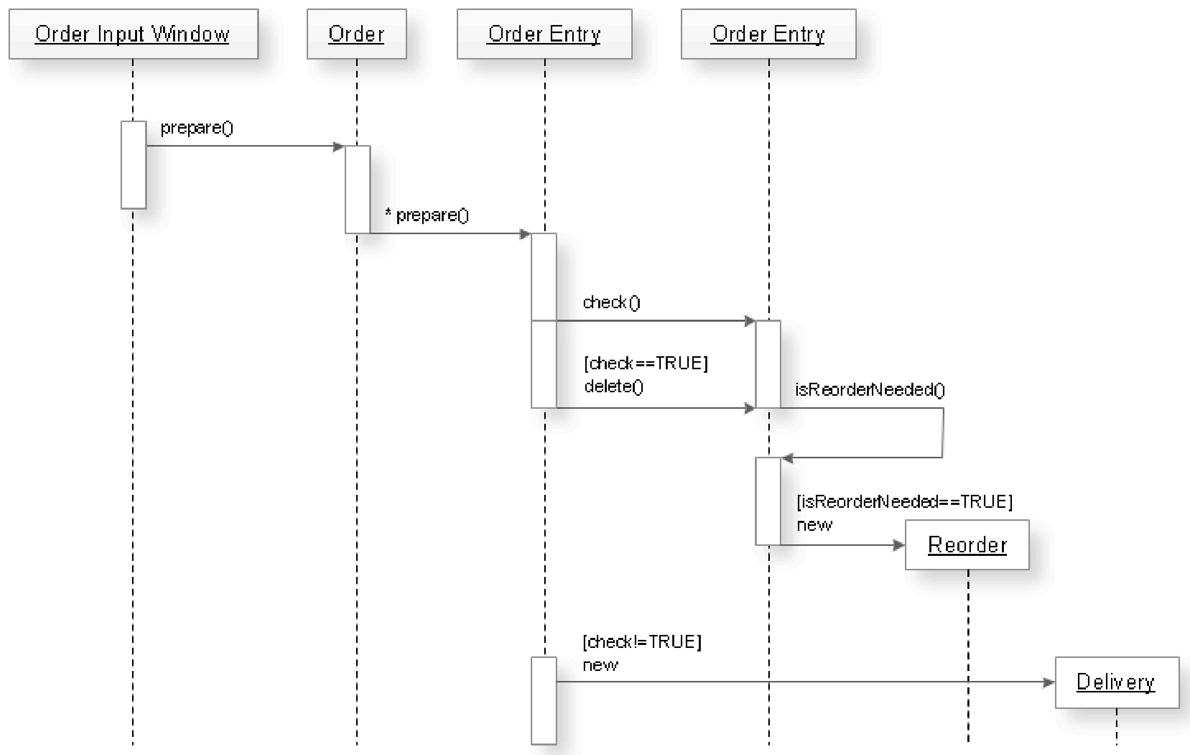
Class Diagram Sample



Sequence diagrams

- A sequence diagram illustrates object interactions arranged in a time sequence.
- These diagrams are typically associated with use cases.
- Sequence diagrams show you step-by-step what has to happen to accomplish something in the use case.
- This type of diagram emphasizes the sequence of events, whereas collaboration diagrams (an alternative view of the same information) emphasize the relationship.
- This type of diagram is best used early in the design or analysis phase because it is simple and easy to comprehend.

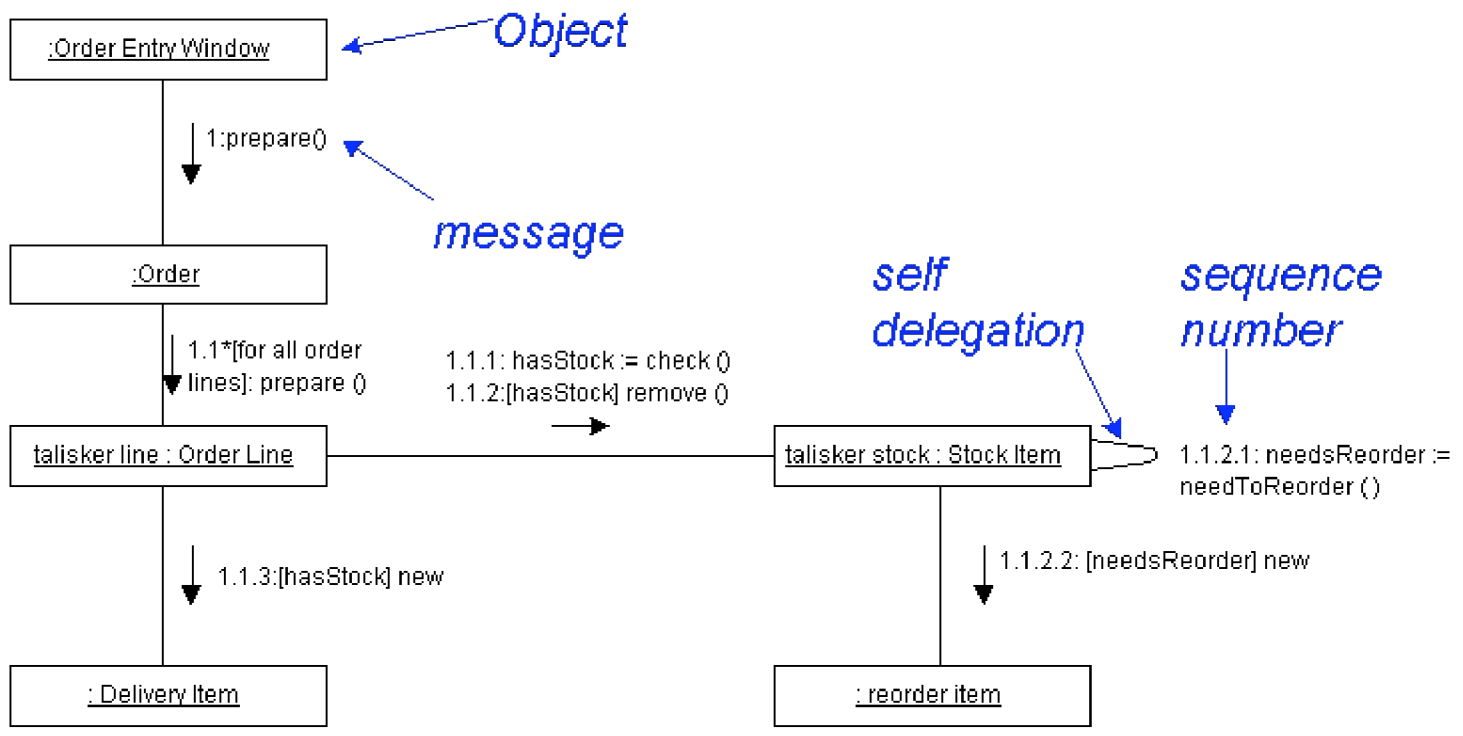
Sequence Diagram Example



Collaboration diagrams

- Collaboration diagrams provide a view of the interactions or structural relationships between objects in the current model.
- This type of diagram emphasizes the relationship between objects whereas sequence diagrams emphasize the sequence of events.
- Collaboration diagrams contain objects, links, and messages.
- Use collaboration diagrams as the primary vehicle to describe interactions that express decisions about system behavior.

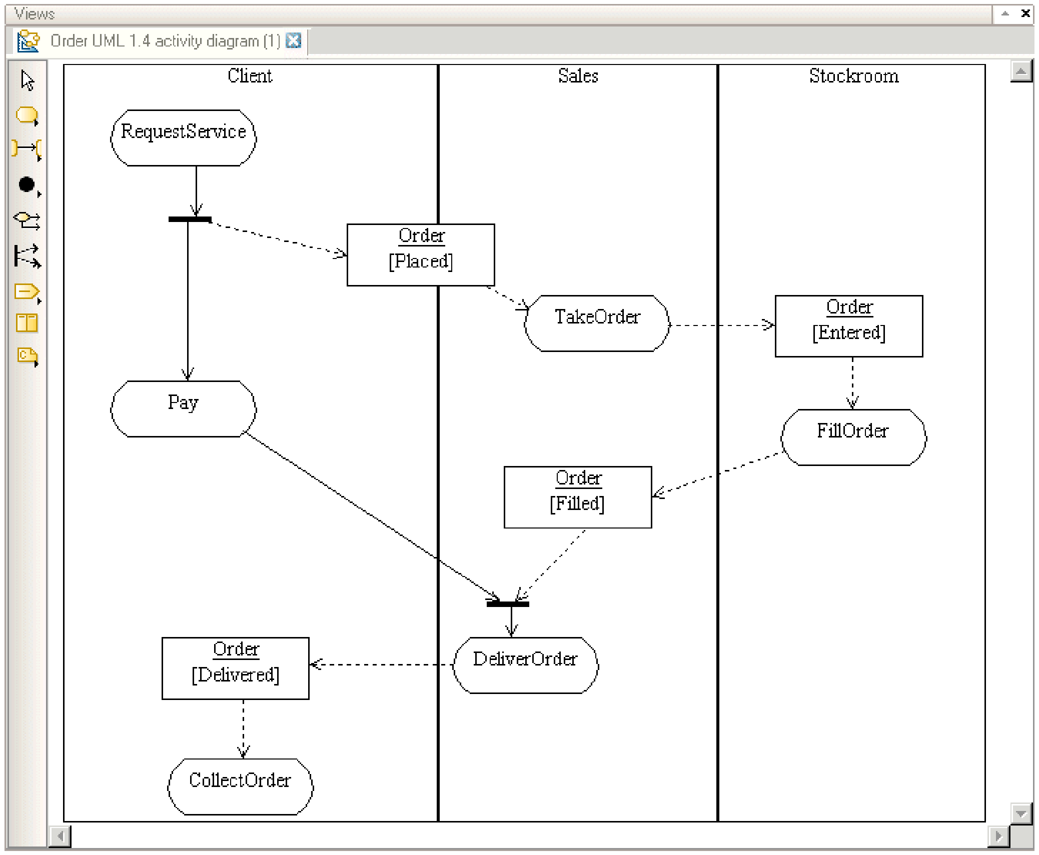
Collaboration Diagram Example


:Order Entry Window
Object
1:prepare()
message
:Order
self delegation
sequence number
1.1*[for all order lines]: prepare ()
1.1.1: hasStock := check ()
1.1.2:[hasStock] remove ()
talisker line : Order Line
talisker stock : Stock Item
1.1.2.1: needsReorder := needToReorder ( )
1.1.3:[hasStock] new
1.1.2.2: [needsReorder] new
: Delivery Item
: reorder item

Activity diagrams

- Activity diagrams model the workflow of a business process and the sequence of activities in a process.
- These diagrams are very similar to a flowchart because you can model a workflow from activity to activity or from activity to state.
- It is often beneficial to create an activity diagram early in the modeling of a process to help you understand the overall process.
- Activity diagrams are also useful when you want to describe parallel behavior or illustrate how behaviors in several use cases interact.

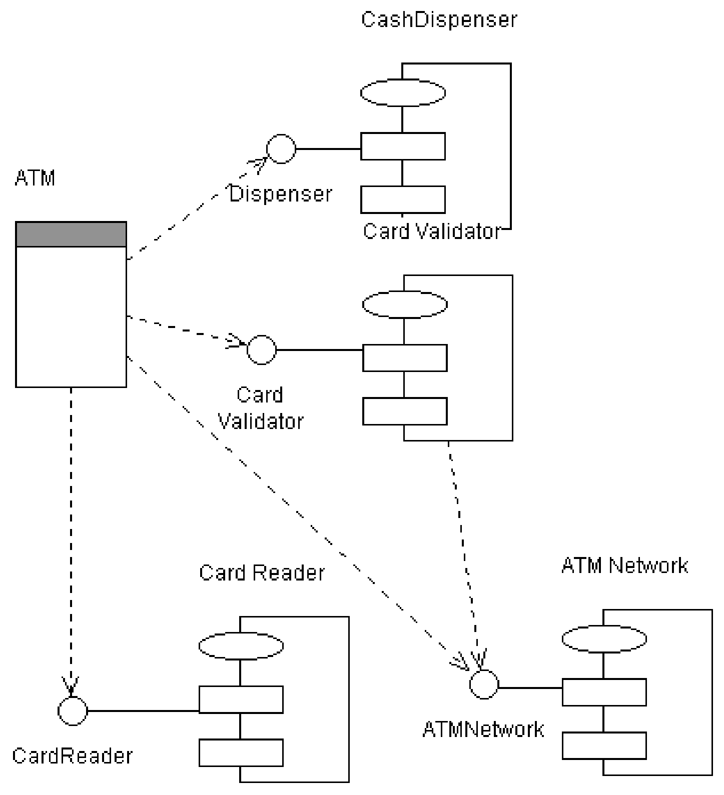
Activity Diagram Example



Component diagrams

- Component diagrams provide a physical view of the current model.
- They show the organization and dependencies among software components, including source code, binary code, and executable components.
- You can create one or more component diagrams to depict components and packages or to represent the contents of each component package.

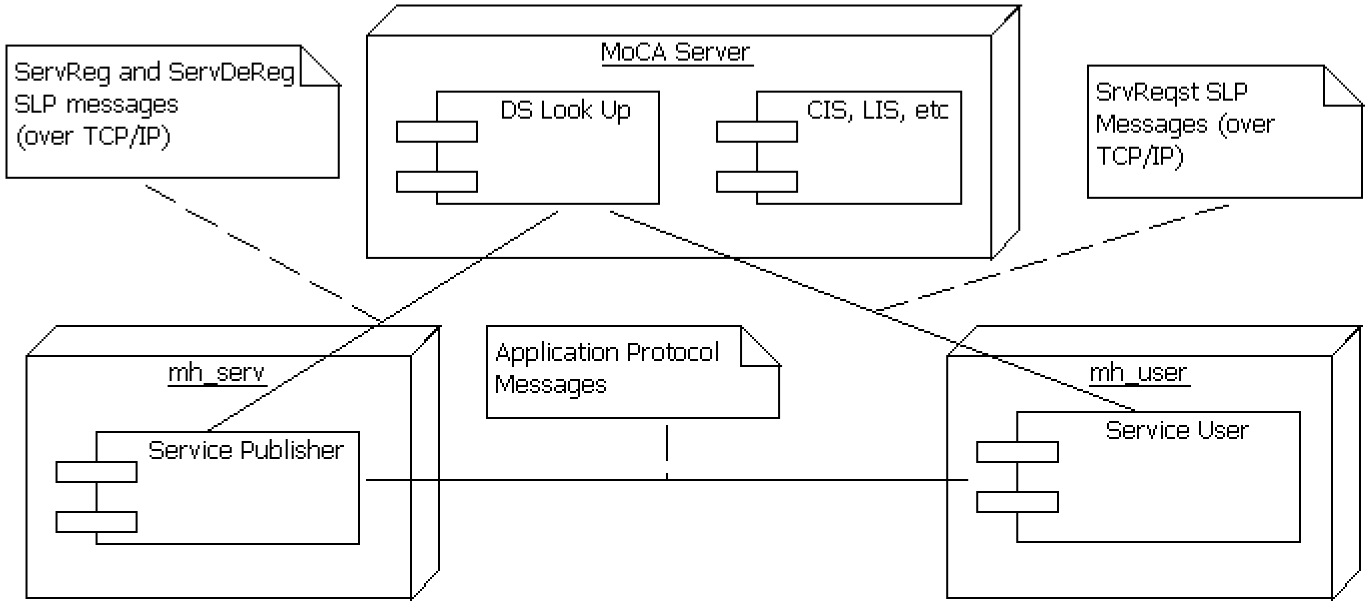
Component Diagram Example



Deployment diagrams

- Each model contains a single deployment diagram that shows the mapping of processes to hardware.

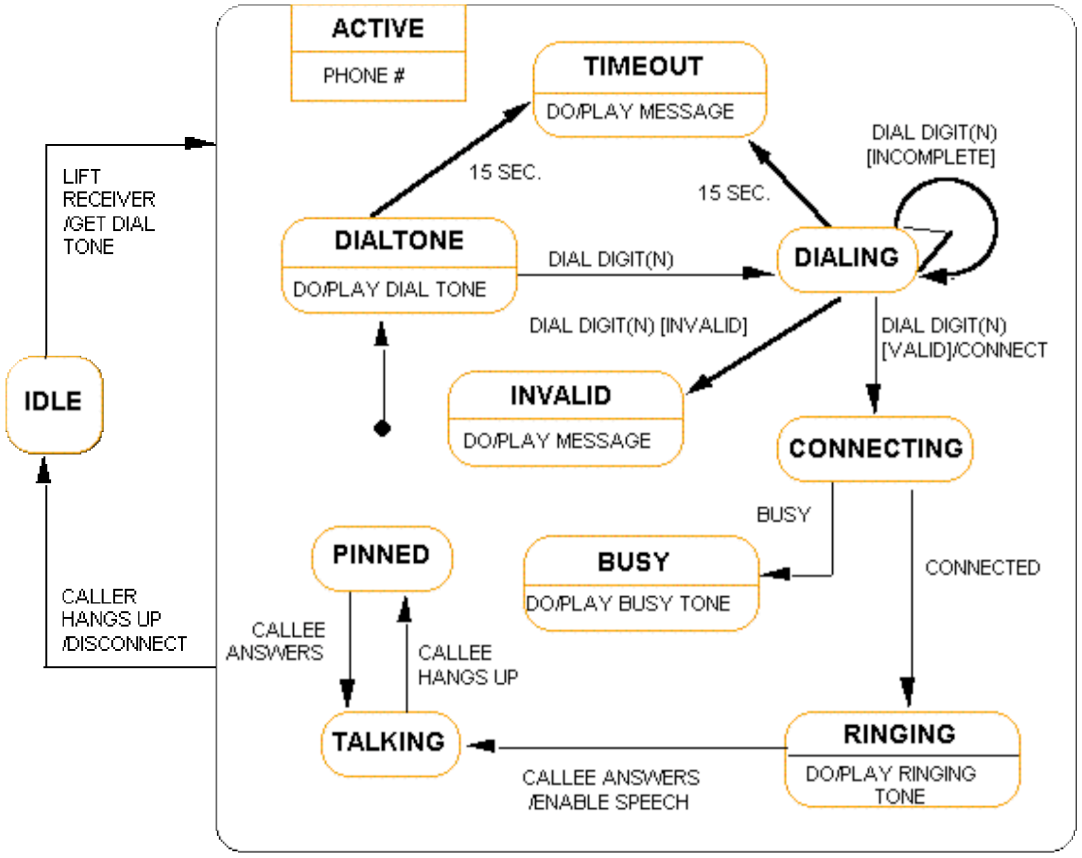
Deployment Diagram Example


ServReg and ServDeReg SLP messages (over TCP/IP)
MoCA Server
DS Look Up
CIS, LIS, etc
SrvReqst SLP Messages (over TCP/IP)
mh_serv
Service Publisher
Application Protocol Messages
mh_user
Service User

Statechart diagrams

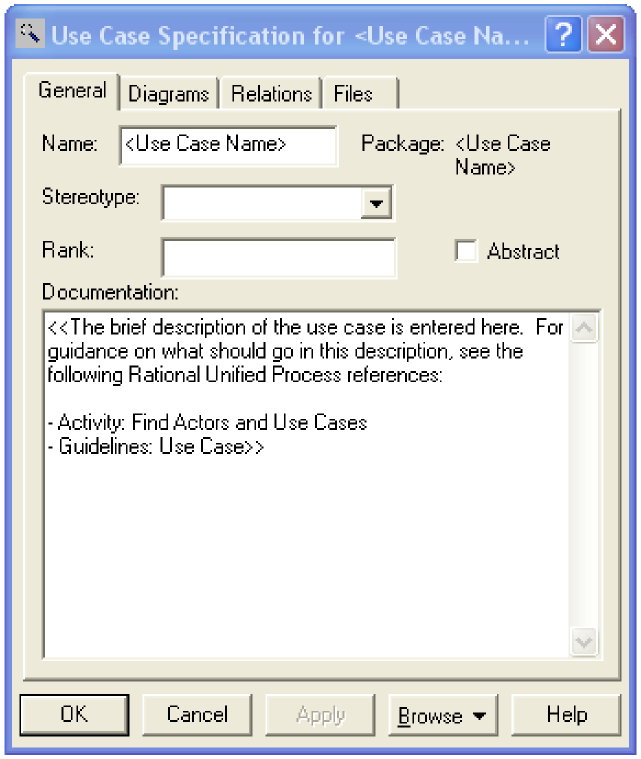
- You can use statechart diagrams to model the dynamic behavior of individual classes or objects.
- Statechart diagrams show the sequences of states that an object goes through, the events that cause a transition from one state or activity to another, and the actions that result from a state or activity change.
- A statechart diagram is typically used to model the discrete stages of an object's lifetime, whereas an activity diagram is better suited to model the sequence of activities in a process.

Statechart Diagram Example



Specifications

- Specifications are dialog boxes that allow you to set or change model element properties.
- Changes made to a model element either through the specification or directly on the icon are automatically updated throughout the model.


Use Case Specification for <Use Case Na...

General | Diagrams | Relations | Files

Name: <Use Case Name>
Package: <Use Case Name>
Stereotype:
Rank:
Abstract
Documentation:
<<The brief description of the use case is entered here. For guidance on what should go in this description, see the following Rational Unified Process references:

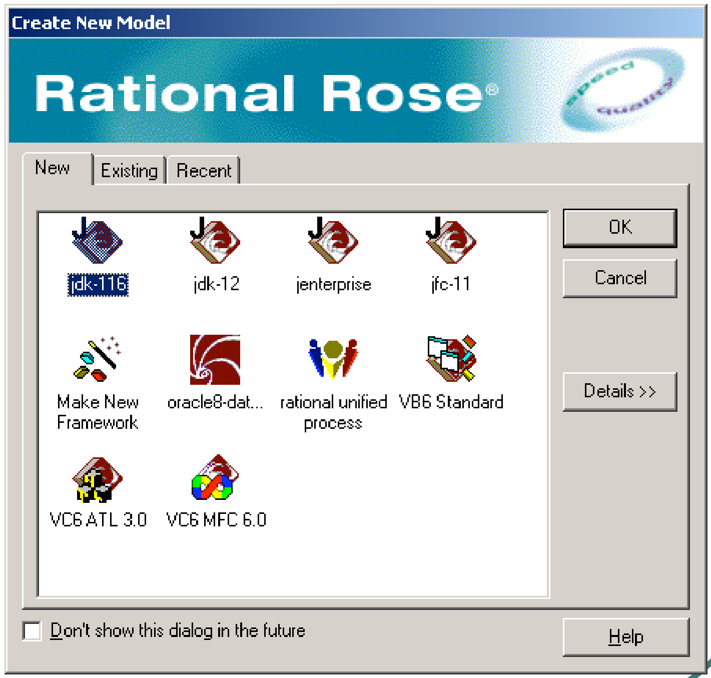
- Activity: Find Actors and Use Cases
- Guidelines: Use Case>>

Start Rational Rose

- Start Rose
  - Start → Programs → Rational Rose

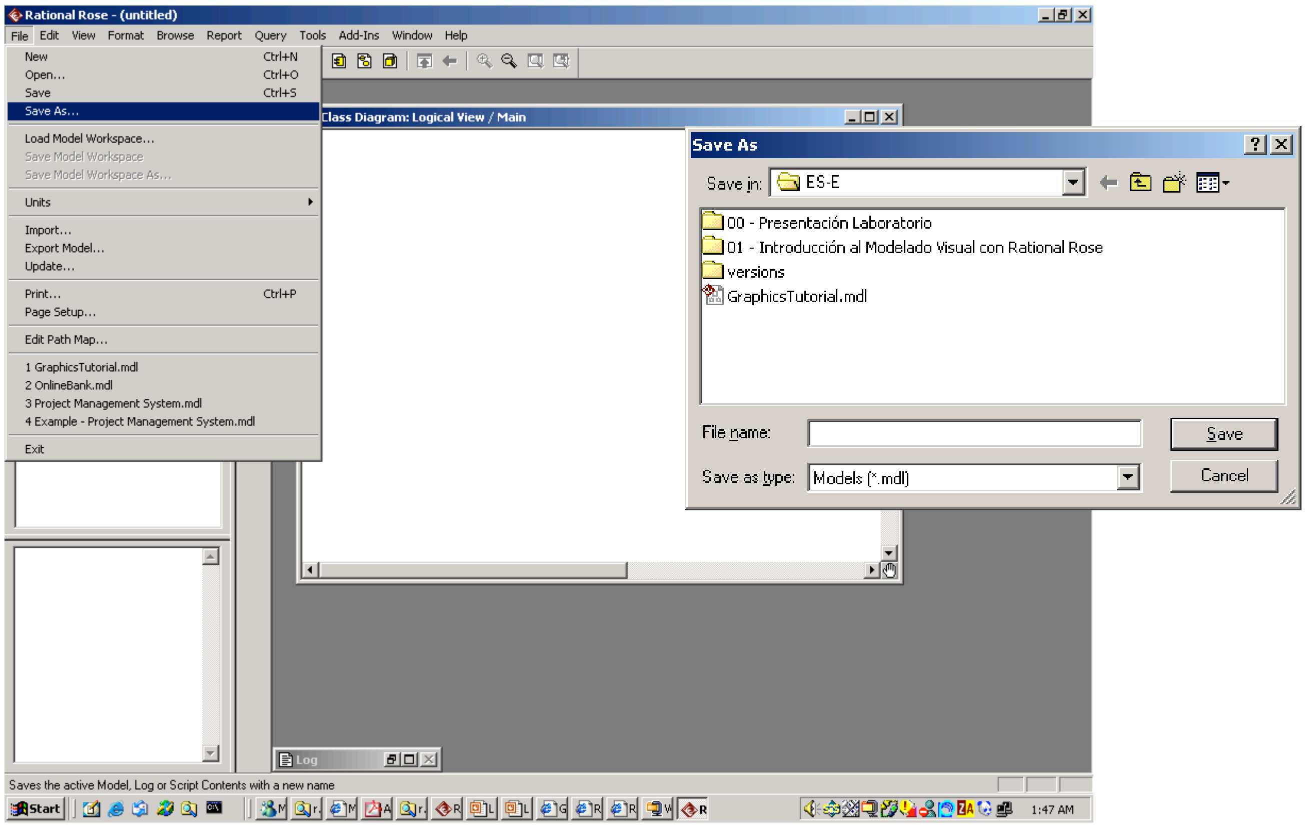
Create a New Model

- When Rose is started
- When Rose has been started: File → New
- From Scratch: New
- From File System: Existing OR Recent



To Save a Model

- File → Save o Save As



Rational Rose - (untitled)
File Edit View Format Browse Report Query Tools Add-Ins Window Help
New Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As...
Load Model Workspace...
Save Model Workspace
Save Model Workspace As...
Units
Import...
Export Model...
Update...
Print... Ctrl+P
Page Setup...
Edit Path Map...
1 GraphicsTutorial.mdl
2 OnlineBank.mdl
3 Project Management System.mdl
4 Example - Project Management System.mdl
Exit
Class Diagram: Logical View / Main
Save As
Save in: ES-E
00 - Presentación Laboratorio
01 - Introducción al Modelado Visual con Rational Rose
versions
GraphicsTutorial.mdl
File name:
Save as type: Models (*.mdl)
Save
Cancel
Log
Saves the active Model, Log or Script Contents with a new name
Start
1:47 AM

Delete an Element

- Shallow Delete
  - Edit Delete
  - Select element in diagram → key DEL

It is not deleted from the MODEL!! (only from the diagram, not from Browser)

- Deep Delete
  - Select element in Browser → click right button → Delete
  - Select element in diagram → Click CTRL+D

It is deleted from the MODEL!! (It will disapear from the diagram and from the Browser)

Move Elements across Packages

- Drag&Drop in each Package Browser from one to another
  - One to One
  - Pay attention: by moving the classes it does not mean that associations will move as well!! (Rational Rose 2000>)
    - They are elements with Identity

More Information

- 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,** by Terry Quatrani