SOFTWARE ENGINEEERING LAB

CSE 625

Practical File

Submitted in partial fulfilment of the
requirements for the award of degree
of

Bachelor of Technology

In

Computer Science and Engineering

By

SUSEEM VIKRAM
Enrolment No: A60205220003, 6th Semester (A)

Submitted To

Dr. Arvind Kumar Upadhyay

Department of Computer Science and Engineering
Amity School of Engineering and Technology
Amity University Madhya Pradesh, Gwalior

LIST OF EXPERIMENTS

S. No.	Experiment		Signature
1	Learning the concepts of Feasibility Study	2-3	
2	Understanding the concepts of Software Documentation	4-8	
3	Learning Project Management	9-13	
4	Getting familiarized with the Unified Modeling Language Environment	14-22	
5	Working with the Use – Case View of UML	23-28	
6	Working with the Class Diagram of UML	29-32	
7	Working with the State Transition Diagram of UML	33-35	
8	Identifying the requirements from Problem Statements	36-37	

1. Learning the concepts of Feasibility Study.

OBJECT

Learning the concepts of Feasibility Study.

THEORY

The development of software begins with requirement gathering and their analysis. The **Software Requirements** are the descriptions of the system services and constraints that are generated during the requirements engineering process. Before the software requirements are gathered and analysed, it is necessary to do some research to study the impact of the software, which is to be developed, over the client‟s business. This research is known as Feasibility study.

Software Requirements Engineering Process

It is the process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.

The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements. However, there are a number of generic activities common to all processes.

1. Feasibility Studies
2. Requirements elicitation and analysis
3. Requirements validation
4. Requirements management

The Feasibility Study

The Feasibility Study is a combination of a market study and an economic analysis that provides an investor with knowledge of both the environment where a project exists and the expected return on investment to be derived from it.

Feasibility Study may also be defined as:

1. A preliminary study undertaken before the real work of a project starts to ascertain the likelihood of the projects success.
2. A small-scale investigation of a problem to ascertain whether a proposed research approach is likely to provide useful data.
3. An analysis of the practicability of a proposal.
4. A study that usually recommends selection of a cost-effective alternative.
5. A study that decides whether or not the proposed system is worthwhile.

Feasibility Study is a study undertaken:

1. To assess in broad terms of technology, cost and time the feasibility of meeting a Staff Target.
2. To identify alternative solutions, their relative advantages and disadvantages, and key problem areas for further study.
3. To provide detailed costed proposals for project definition studies and information in the detail necessary to draft a Staff Requirement.
4. To check:
5. If the system contributes to organisational objectives
6. If the system can be engineered using current technology and within budget
7. If the system can be integrated with other systems that are used

Feasibility Study Report

1. The study addresses issues including the project's benefits, costs, effectiveness, alternatives considered, analysis of alternative selection, environmental effects, public opinions, and other factors.
2. A written report includes the study findings, recommendations, and (when the goal is feasible) a campaign plan, timetable, and budget.

Feasibility Study Implementation

Feasibility Study is carried out through information assessment (i.e. what is required), information collection and report writing. Certain areas may be targeted by asking the following questions:

1. What if the system wasn‟t implemented?
2. What are current process problems?
3. How will the proposed system help?
4. What will be the integration problems?
5. Is new technology needed? What skills?
6. What facilities must be supported by the proposed system?

Case Studies of Feasibility Studies

Consider the following Real-world case studies of Feasibility Studies carried out by various organizations:

Case 1: BankSoft Feasibility Study - World Bank ATM Pilot Project

This document outlines a feasibility study done by BankSoft for World Bank. World Bank has proposed a pilot project where they would like to implement several ATM machines to measure the popularity of such machines. In the future, they would use the data that they collect on these machines to implement more machines worldwide.

World Bank is renowned for its attention to security. Herein, three alternatives have been suggested for designing these test machines. Each alternative is quite secure, but has its own advantages and disadvantages over the other. The alternatives are as follows:

1. An embedded system. This would require a lot of time and custom hardware, but would be extremely fast and BankSoft would provide an excellent warranty for the software.
2. Custom software on Microsoft Windows. This system would require minimal time and cost, but would not require any special hardware. Interoperability with the bank's central computer would be maximized, and warranty coverage would also be at peak.
3. Custom software on a free Unix OS. This system would also not require special hardware, but making the software interoperable with the bank's central computer would take additional time and effort. Such a system would be extremely stable and inexpensive.

The recommendation has been made that the second alternative be designed, with custom software designed on top of the Microsoft Windows platform. It is strongly believed that its advantages outweigh its disadvantages more than any other alternative.

2. Understanding the concepts of Software Documentation.

OBJECT

Understanding the concepts of Software Documentation.

THEORY

All large software development projects, irrespective of application, generate a large amount of associated documentation. For moderately sized systems, the documentation will probably fill several filing cabinets; for large systems, it may fill several rooms. A high proportion of software process costs is incurred in producing this documentation. Furthermore, documentation errors and omissions can lead to errors by end-users and consequent system failures with their associated costs and disruption. Therefore, managers and software engineers should pay as much attention to documentation and its associated costs as to the development of the software itself.
The documents associated with a software project and the system being developed, have a number of associated requirements:

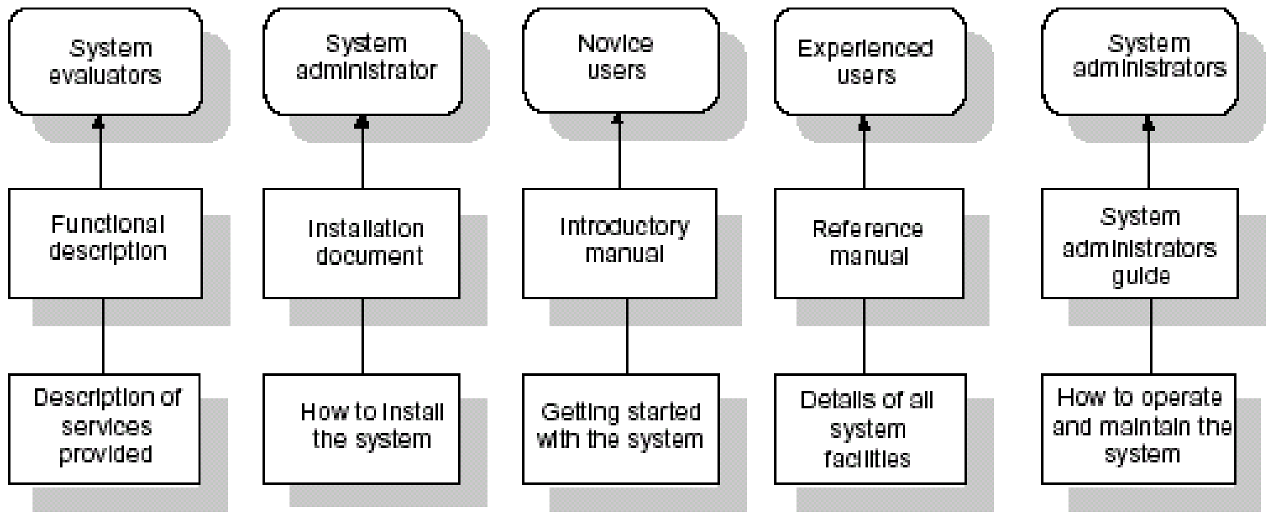
1. They should act as a communication medium between members of the development team.
2. They should be a system information repository to be used by maintenance engineers.
3. They should provide information for management to help them plan, budget and schedule the software development process.
4. Some of the documents should tell users how to use and administer the system.

Satisfying these requirements requires different types of document from informal working documents through to professionally produced user manuals. Software engineers are usually responsible for producing most of this documentation although professional technical writers may assist with the final polishing of externally released information.

Process and Product Documentation
Generally, the documentation produced falls into two classes:

1. *Process documentation:* These documents record the process of development and maintenance. Plans, schedules, process quality documents and organizational and project standards are process documentation. The major characteristic of process documentation is that most of it becomes outdated. Plans may be drawn up on a weekly, fortnightly or monthly basis. Progress will normally be reported weekly. Memos record thoughts, ideas and intentions, which change. Although of interest to software historians, much of this process information is of little real use after it has gone out of date and there is not normally a need to preserve it after the system has been delivered. However, there are some process documents that can be useful as the software evolves in response to new requirements. For example, test schedules are of value during software evolution as they act as a basis for re-planning the validation of system changes. Working papers which explain the reasons behind design decisions (design rationale) are also potentially valuable as they discuss design options and choices made.

2. *Product documentation* This documentation describes the product that is being developed. System documentation describes the product from the point of view of the engineers developing and maintaining the system; user documentation provides a product description that is oriented towards system users. Unlike most process documentation, it has a relatively long life. It must evolve in step with the product, which it describes. Product documentation includes user documentation, which tells users how to use the software product and system documentation, which is principally intended for maintenance engineers. See Figure 2.1 for different types of User Documents.



System evaluators
System administrator
Novice users
Experienced users
System administrators
Functional description
Installation document
Introductory manual
Reference manual
System administrators guide
Description of services provided
How to install the system
Getting started with the system
Details of all system facilities
How to operate and maintain the system

Figure: Different Types of User Documents

In short, Process documentation is produced so that the development of the system can be managed. Product documentation is used after the system is operational but is also essential for management of the system development. The creation of a document, such as a system specification, may represent an important milestone in the software development process.

Document Quality

Document quality is as important as program quality. Without information on how to use a system or how to understand it, the utility of that system is degraded. Achieving document quality requires management commitment to document design, standards, and quality assurance processes. Producing good documents is neither easy nor cheap and many software engineers find it more difficult that producing good quality programs.

Document structure

The document structure is the way in which the material in the document is organized into chapters and, within these chapters, into sections and subsections. Document structure has a major impact on readability and usability, and it is important to design this carefully when creating documentation. As with software systems, you should design document structures sothat the different parts are as independent as possible. This allows each part to be read as a single item and reduces problems of cross-referencing when changes have to be made.

2.2. **Component**	**Description**
Identification data	Data such as a title and identifier that uniquely identifies the document.
Table of contents	Chapter/section names and page numbers.
List of illustrations	Figure numbers and titles
Introduction	Defines the purpose of the document and a brief summary of the contents
Information for use of the documentation	Suggestions for different readers on how to use the documentation effectively.
Concept of operations	An explanation of the conceptual background to the use of the software.
Procedures	Directions on how to use the software to complete the tasks that it is designed to support.
Information on software commands	A description of each of the commands supported by the software.
Error messages and problem resolution	A description of the errors that can be reported and how to recover from these errors.
Glossary	Definitions of specialized terms used.
Related information sources	References or links to other documents that provide additional information
Navigational features	Features that allow readers to find their current location and move around the document.
Index	A list of key terms and the pages where these terms are referenced.
Search capability	In electronic documentation, a way of finding specific terms in the document.

Figure: Suggested components in a software user document

Documentation Standards

Documentation standards act as a basis for document quality assurance. Documents produced according to appropriate standards have a consistent appearance, structure and quality. The standards that may be used in the documentation process are:

1. ***Process standards*** These standards define the process, which should be followed for high-quality document production.

2. ***Product standards*** These are standards, which govern the documents themselves.

3. ***Interchange standards*** It is increasingly important to exchange copies of documents via electronic mail and to store documents in databases. Interchange standards ensure that all electronic copies of documents are compatible.

Writing style
Writing documents well is neither easy nor is it a single stage process. Written work must be written, read, criticized, and then rewritten until a satisfactory document is produced. Technical writing is a craft rather than a science but some broad guidelines about how to write well are:
1. Use active rather than passive tenses
2. Use grammatically correct constructs and correct spelling
3. Do not use long sentences which present several different facts
4. Keep paragraphs short
5. Don"t be verbose
6. Be precise and define the terms which you use
7. If a description is complex, repeat yourself
8. Make use of headings and sub-headings.
9. Itemize facts wherever possible
10. Do not refer to information by reference number alone

Documents should be inspected in the same way as programs. During a document inspection, the text is criticized, omissions pointed out and suggestions made on how to improve the document. As well as personal criticism, grammar checkers can also be used, which are incorporated in word processors.

On-line documentation
It is now normal to provide some on-line documentation with delivered software systems. This can range from simple „read me" files that provide very limited information about the software through interactive hypertext-based help systems to a complete on-line suite of system documentation. Most commonly, however, hypertext-based help systems are provided. These may be based on a specialized hypertext system or may be HTML-based and rely on web browsers for access. The main advantage with on-line documentation is its accessibility.

Document Preparation
Document preparation is the process of creating a document and formatting it for publication. Figure 2.3 shows the document preparation process as being split into 3 stages namely document creation, polishing and production. Modern word processing systems are now integrated packages of software tools that support all parts of this process. However, it is still the case that for the highest-quality documents, it is best to use separate tools for some preparation processes rather than the built-in word processor functions. The three phases of preparation and associated support facilities are:

1. ***Document creation*** the initial input of the information in the document. This is supported by word processors and text formatters, table and equation processors, drawing and art packages.

2. ***Document polishing*** This process involves improving the writing and presentation of the document to make it more understandable and readable. This involves finding and removing spelling, punctuation and grammatical errors, detecting clumsy phrases and removing redundancy in the text. Tools such as on-line dictionaries, spelling checkers, grammar and style checkers and style checkers may support the process.

3. ***Document production*** This is the process of preparing the document for professional printing. It is supported by desktop-publishing packages, artwork packages and type styling programs.

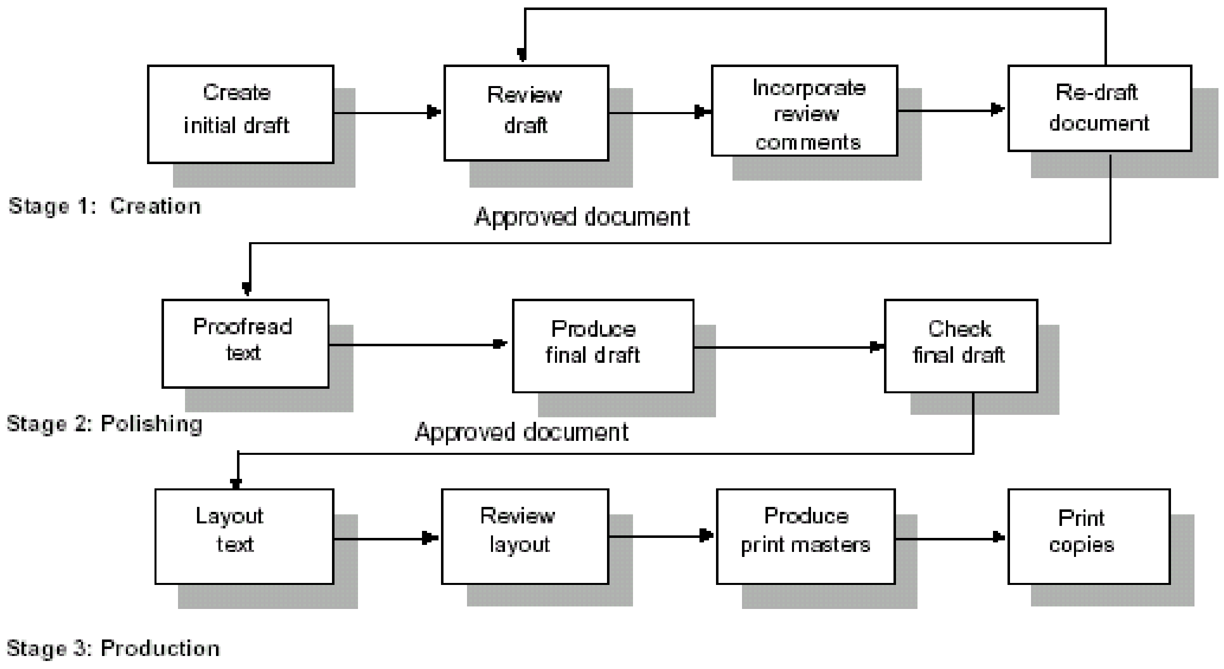


Figure: Stages of document preparation

3. Learning Project Management

OBJECT

Learning Project Management

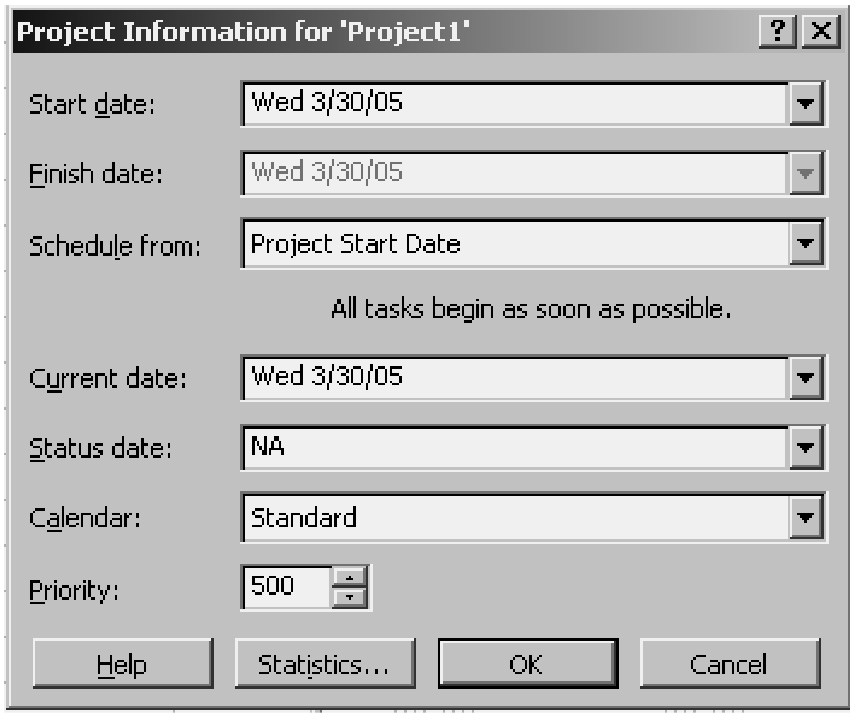
THEORY

Project management is the process of planning, organizing, and managing tasks and resources to accomplish a defined objective, usually within constraints on time, resources, or cost. Sound planning is one key to project success, but sticking to the plan and keeping the project on track is no less critical. Microsoft Project features can help in monitoring progress and keep small slips from turning into landslides.
First let us understand some basic terms and definitions that will be used quite often:

Tasks**:** They are a division of all the work that needs to be completed in order to accomplish the project goals.
Scope: of any project is a combination of all individual tasks and their goals.
Resources: can be people, equipment, materials or services that are needed to complete various tasks. The amount of resources affects the scope and time of any project.

STARTING A NEW PROJECT

You can create a Project from a Template File by choosing File > New from the menu. In the New File dialog box that opens, select the Project Templates tab and select the template that suits your project best and click OK. (You may choose a Blank Project Template and customize it)
Once a new Project page is opened, the Project Information dialog box opens.


Project Information for 'Project1'

Start date: Wed 3/30/05
Finish date: Wed 3/30/05
Schedule from: Project Start Date
All tasks begin as soon as possible.
Current date: Wed 3/30/05
Status date: NA
Calendar: Standard
Priority: 500

Figure: Project Information dialog box

Enter the start date or select an appropriate date by scrolling down the list. Click OK. Project automatically enters a default start time and stores it as part of the dates entered and the application window is displayed.

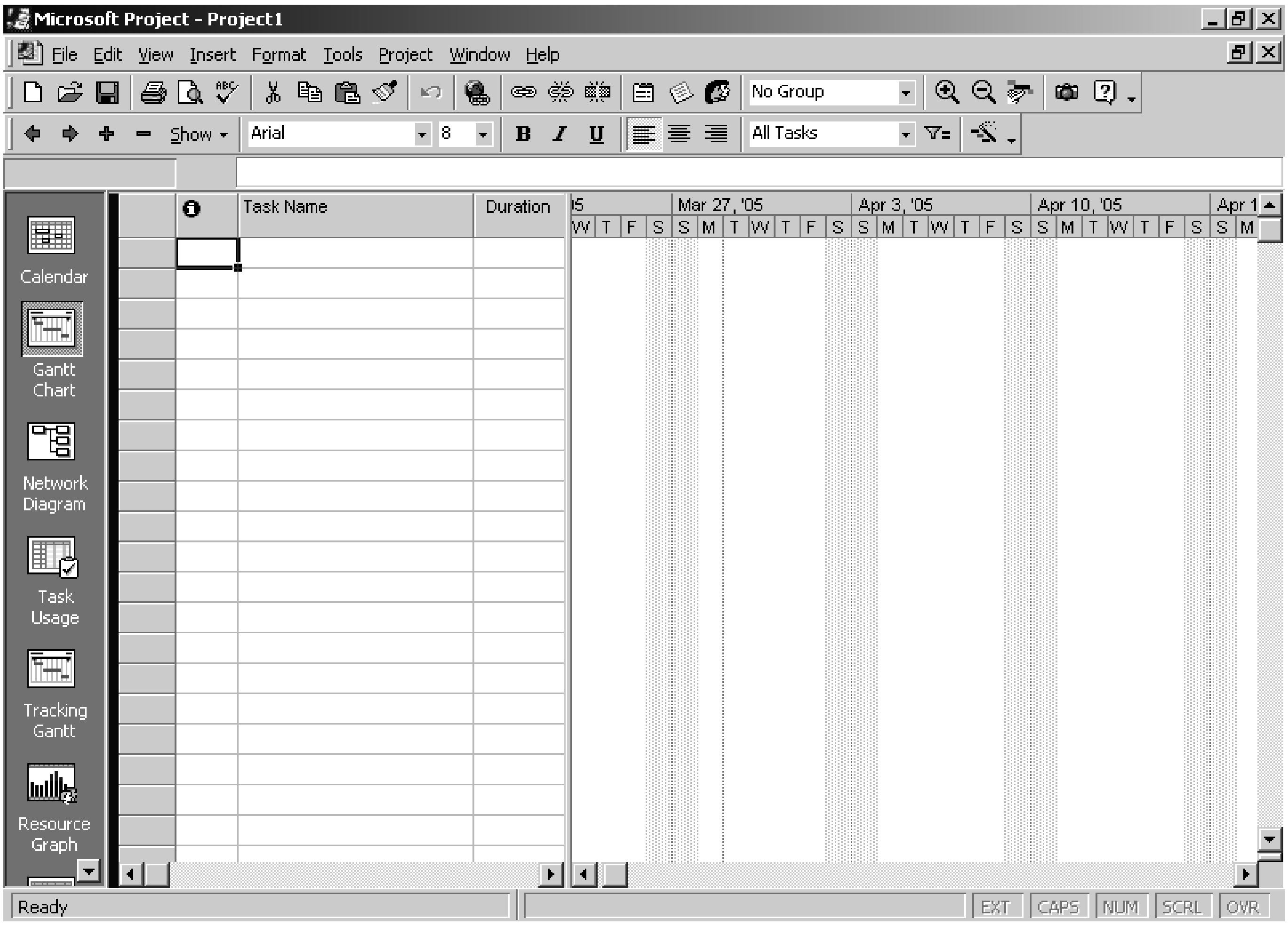


Figure: Main window

Views

Views allow you to examine your project from different angles based on what information you want displayed at any given time. You can use a combination of views in the same window at the same time.
Project Views are categorized into two types:

- Task Views (5 types)
- Resource Views (3 types)

The Project worksheet is located in the main part of the application window and displays different information depending on the view you choose. The default view is the Gantt Chart View.

Tasks
Goals of any project need to be defined in terms of tasks. There are four major types of tasks:
1. *Summary tasks* - contain subtasks and their related properties
2. *Subtasks* - are smaller tasks that are a part of a summary task
3. *Recurring tasks* - are tasks that occur at regular intervals
4. *Milestones* - are tasks that are set to zero duration and are like interim goals in the project

4. ***Entering Tasks and assigning task duration***

Click in the first cell and type the task name. Press enter to move to the next row. By default, estimated duration of each task is a day. But, there are very few tasks that can be completed in a day's time. You can enter the task duration as and when you decide upon a suitable estimate.
Double-clicking a task or clicking on the Task Information button on the standard toolbar opens the *Task Information box*. You can fill in more detailed descriptions of tasks and attach documents related to the task in the options available in this box.
To enter a *milestone*, enter the name and set its duration to zero. Project represents it as a diamond shape instead of a bar in the Gantt chart.
To copy tasks and their contents, click on the task ID number at the left of the task and copy and paste as usual.
You can enter *Recurring tasks* by clicking on Insert > Recurring task and filling in the duration and recurrence pattern for the task.
Any action you perform on a summary task - deleting it, moving or copying it apply to its subtasks too.

5. ***Outlining tasks***

Once the summary tasks have been entered in a task table, you will need to insert subtasks in the blank rows and indent them under the summary task. This is accomplished with the help of the outlining tool. Outlining is already active when you launch a project, and its tools are found at the left end of the Formatting bar. To enter a subtask, enter the task in a blank cell in the Task Name column and click the Indent button on the Outlining tool bar. The Show feature in this toolbar is drop down tool that gives you an option of different Outline levels.
A summary task is outdented to the left cell border, is bold and has a Collapse (-) (Hide subtasks) button in front of it and its respective subtasks are indented with respect to it.
Advantages of Outlining:
• It creates multiple levels of subtasks that roll up into a summary task
• Collapse and expand summary tasks when necessary
• Apply a Work Breakdown structure
• Move, copy or delete entire groups of tasks

LINKS
Tasks are usually scheduled to start as soon as possible i.e., the first working day after the project start date.

Dependencies
Dependencies specify the way two tasks are linked. Because Microsoft Project must maintain the way tasks are linked, dependencies can affect the way a task is scheduled. In a scenario where there are two tasks, the following dependencies exist:

Finish to Start – Task 2 cannot start until task 1 finishes.
Start to Finish – Task 2 cannot finish until task 1 starts.
Start to Start – Task 2 cannot start until task 1 starts.
Finish to Finish – Task 2 cannot finish until task 1 finishes.
Tasks can be linked by following these steps:
1. Double-click a task and open the Task Information dialog box.
2. Click the predecessor tab.
3. Select the required predecessor from the drop-down menu.
4. Select the type of dependency from drop down menu in the Type column.
5. Click OK.

The Split task button splits tasks that may be completed in parts at different times with breaks in their duration times.

UPDATE COMPLETED TASKS
If you have tasks in your project that have been completed as they were scheduled, you can quickly update them to 100% complete all at once, up to a date you specify.
1. On the View menu, click Gantt Chart.
2. On the Tools menu, point to Tracking, and then click Update Project
3. Click Update work as complete through and then type or select the date through which you want progress updated.
4. If you don't specify a date, Microsoft Project uses the current or status date.
5. Click Set 0% or 100% complete only.
6. Click Entire Project

RESOURCES
Once you determine that you need to include resources into your project you will need to answer the following questions:
• What kind of resources do you need?
• How many of each resource do you need?
• Where will you get these resources?
• How do you determine what your project is going to cost?

Resources are of two types - *work* resources and *material* resources.
Work resources complete tasks by expending time on them. They are usually people and equipment that have been assigned to work on the project.
Material resources are supplies and stocks that are needed to complete a project.
A new feature in Microsoft Project 2000 is that it allows you to track material resources and assign them to tasks.

Entering Resource Information in Project
The Assign Resources dialog box is used to create list of names in the resource pool.
To enter resource lists:
1. Click the Assign Resources button on the Standard Tool bar or Tools > Resources > Assign resources.
2. Select a cell in the name column and type a response name. Press Enter
3. Repeat step 2 until you enter all the resource names.
4. Click OK.

Resource names cannot contain slash (/), brackets [ ] and commas (,). Another way of defining your resource list is through the Resource Sheet View.
1. Display Resource Sheet View by choosing View > Resource Sheet or click the Resource Sheet icon on the View bar.
2. Enter your information. (To enter material resource, use the Material Label field)

To assign a resource to a task:
1. Open the Gantt Chart View and the Assign Resources Dialog box.
2. In the Entry Table select the tasks for which you want to assign resources.
3. In the Assign Resources Dialog box, select the resource (resources) you want to assign.
4. Click the Assign button.

4. Getting familiarized with the Unified Modeling Language (UML) Environment.

OBJECT

Getting familiarized with the Unified Modeling Language (UML) Environment.

THEORY

Once the Requirements have been gathered, it is necessary to convert them into an appropriate design. In order to bridge the requirements gathering and design phases, we use some modeling or specification tool, e.g. the Unified Modeling Language. A commonly available CASE tool for design through UML is the Rational Rose. Given below is an overview of the various Aspects in which development could be done using Rational Rose.

The Browser

The Browser contains a list of all the modeling elements in the current model. The Browser contains a tree view of all the elements in the current Rose model. Elements in the tree structure are either compressed or expanded.

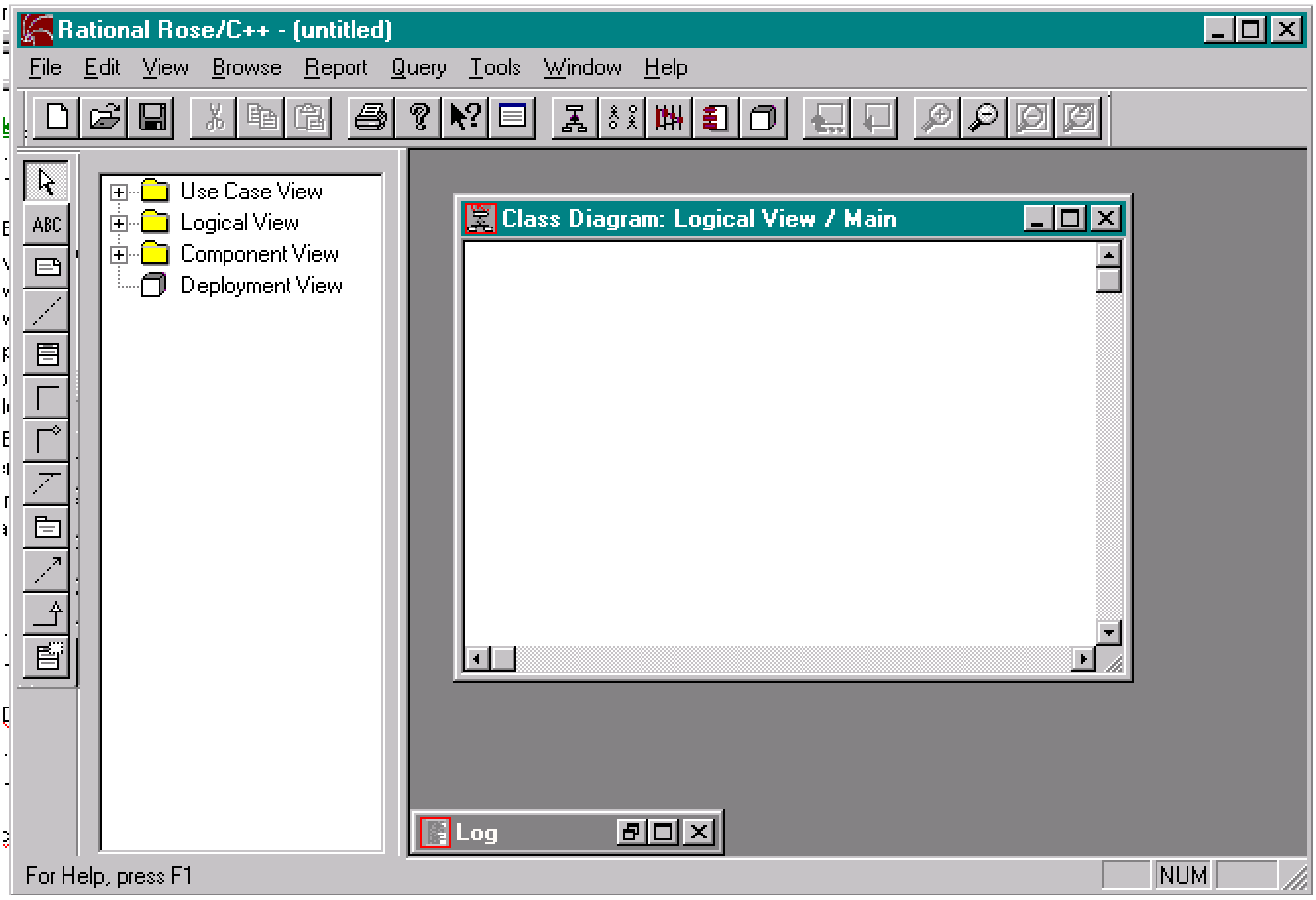

Rational Rose/C++ - (untitled)
File Edit View Browse Report Query Tools Window Help
Use Case View
Logical View
Component View
Deployment View
Class Diagram: Logical View / Main
Log
For Help, press F1
NUM

Figure: The Browser Windo

The Documentation Window

The Documentation Window may be used to create, review, and modify for any selected modeling element. If nothing is selected, the window displays the string “No selection”. The contents of the Documentation Window will change as different modeling elements are selected.
To create the documentation for a selected element, click to select the element and enter its documentation in the Documentation Window.
The Documentation Window may be docked or floating just like the Browser. Visibility of the window is controlled via the View: Documentation window command.

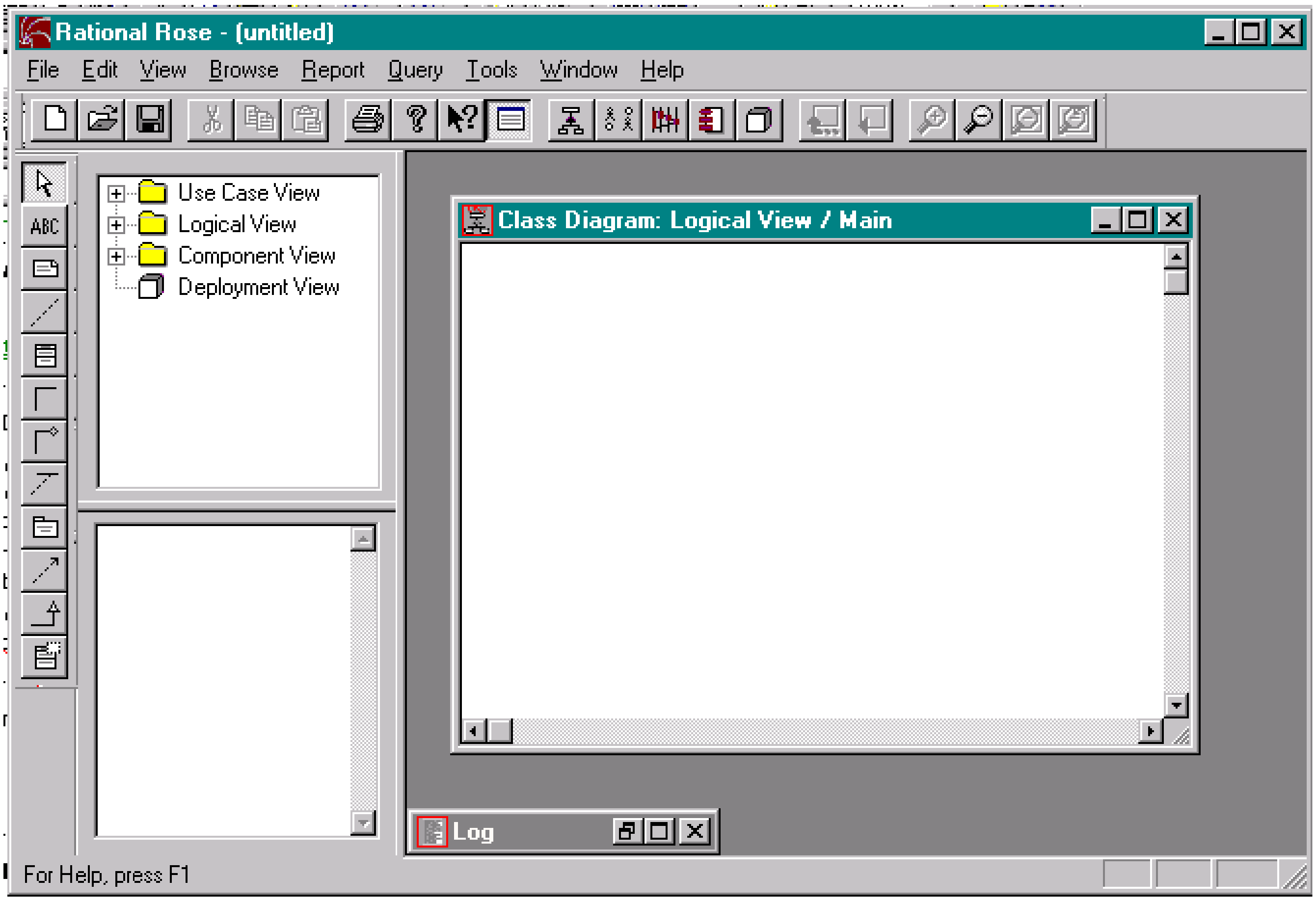

Rational Rose - (untitled)
File Edit View Browse Report Query Tools Window Help
Use Case View
Logical View
Component View
Deployment View
Class Diagram: Logical View / Main
Log
For Help, press F1

Figure: The Documentation Window

Diagram Windows

Diagram windows allow you to create and modify graphical views of the current model. Each icon on a diagram represents a modeling element. Since diagrams are used to illustrate multiple views of a model, each model element can appear in none, one, or several of a model's diagrams.

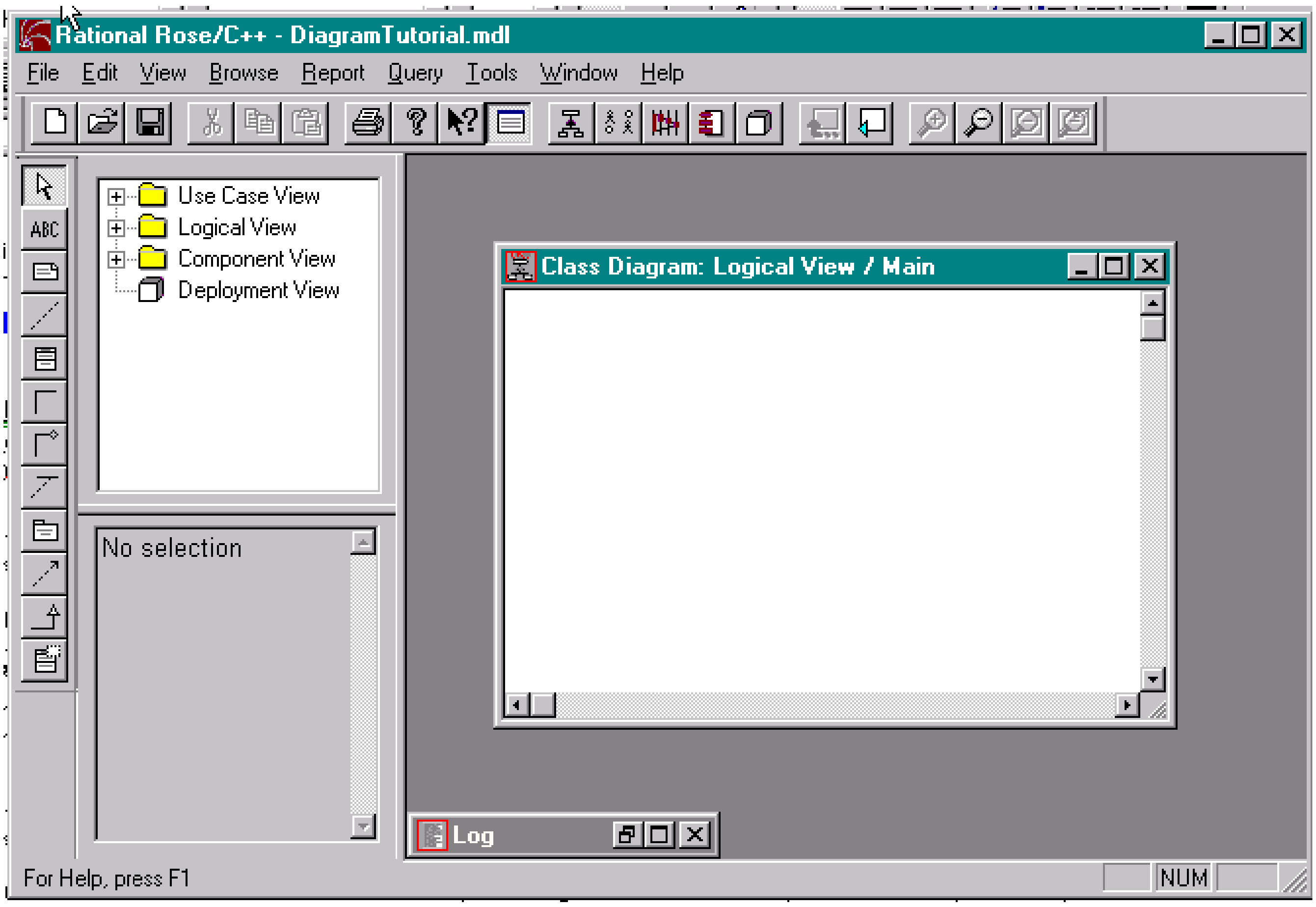

Rational Rose/C++ - DiagramTutorial.mdl
File Edit View Browse Report Query Tools Window Help
Use Case View
Logical View
Component View
Deployment View
No selection
Class Diagram: Logical View / Main
Log
For Help, press F1
NUM

Figure: The Diagram Windows

Views in Rational Rose

There are many views of a software project under development. Rational Rose is organized around the views of a software project:

1. The Use Case View
2. The Logical View
3. The Component View
4. The Deployment View.

Each view presents a different aspect of the visual model. Each view contains a Main diagram by default. Additional elements and diagrams are added to each view throughout the analysis and design process.

1. The Use Case View

The use case view of the system addresses the understandability and usability of the system. This view looks at actors and use cases along with their interactions. The diagrams in this view are use case diagrams, sequence diagrams and collaboration diagrams.

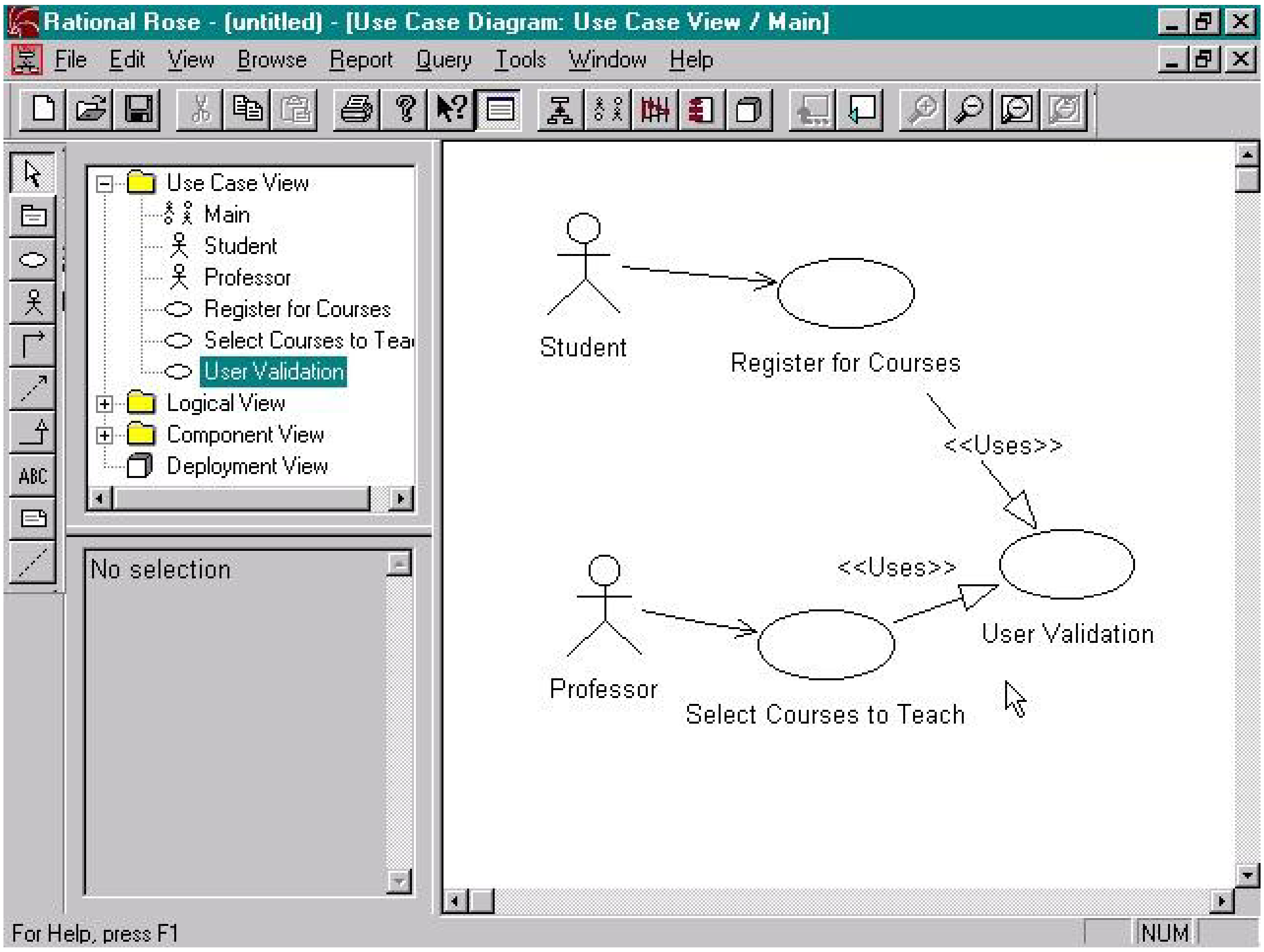


Figure: Use Case View □□Use Case Diagram

Packages in the Use Case View

Packages in the use case view can contain actors, use cases, sequence diagrams, and/or collaboration diagrams. To create a package:

1. Right-click on the parent modeling element (use case view or another package) to make the shortcut menu visible.
2. Select the New: Package menu command. This will add a new package called NewPackage to the browser.
3. While the new package is still selected, enter its name.

Once a package is created, modeling elements may be moved to the package. To move a modeling element to a package:

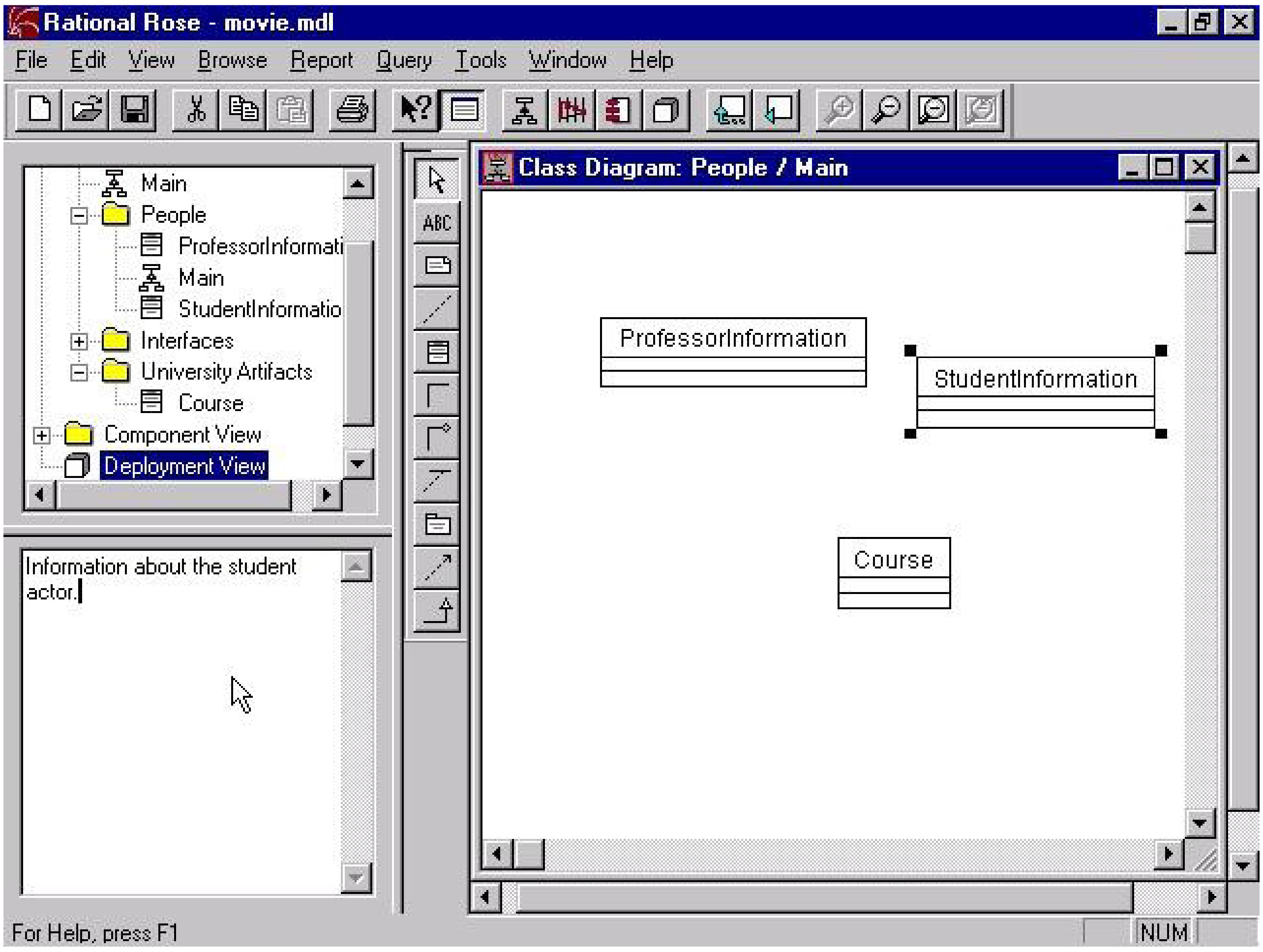
1. Click to select the modeling element to be relocated.
2. Drag the modeling element to the package.

2. The Logical View

The logical view of the system addresses the functional requirements of the system. This view looks at classes and their static relationships. It also addresses the dynamic nature of the classes and their interactions. The diagrams in this view are class diagrams and state transition diagrams.

2.1 Class Diagram

The browser provides a textual view of the classes in a system. Class diagrams are created to graphically view the classes and packages in the system. Rose automatically creates a class diagram called Main in the Logical View. This diagram may be opened by double-clicking on it in the browser. The Main class diagram typically contains packages thus, by the end of development it is a graphical view of the major architectural elements of the system. A package may be added to a class diagram by selecting it in the browser and dragging it onto the class diagram.



Information about the student actor.

Figure: Logical View Class Diagram

2.2 State Transition Diagram

State transition diagrams show the life history of a given class, the events that cause a transition from a state and the actions that result from a state change. They are created for classes whose objects exhibit significant dynamic behavior.

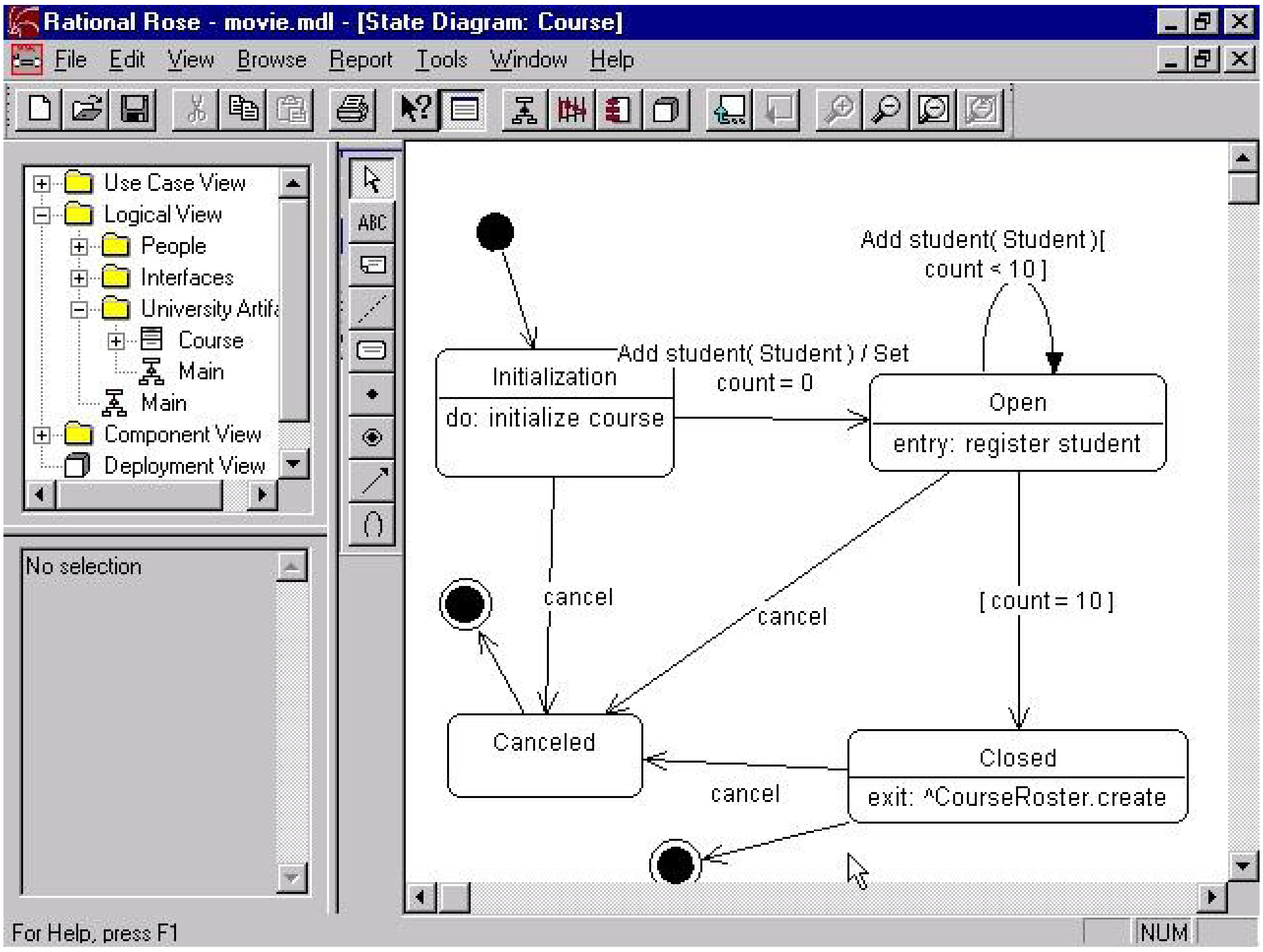


Figure: Logical View State Transition Diagram

3. The Component View

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

Component Diagram

A component diagram shows the organizations and dependencies among components. Most models contain many component diagrams. A component is represented as a rectangle with one small ellipse and two small rectangles protruding from its side.

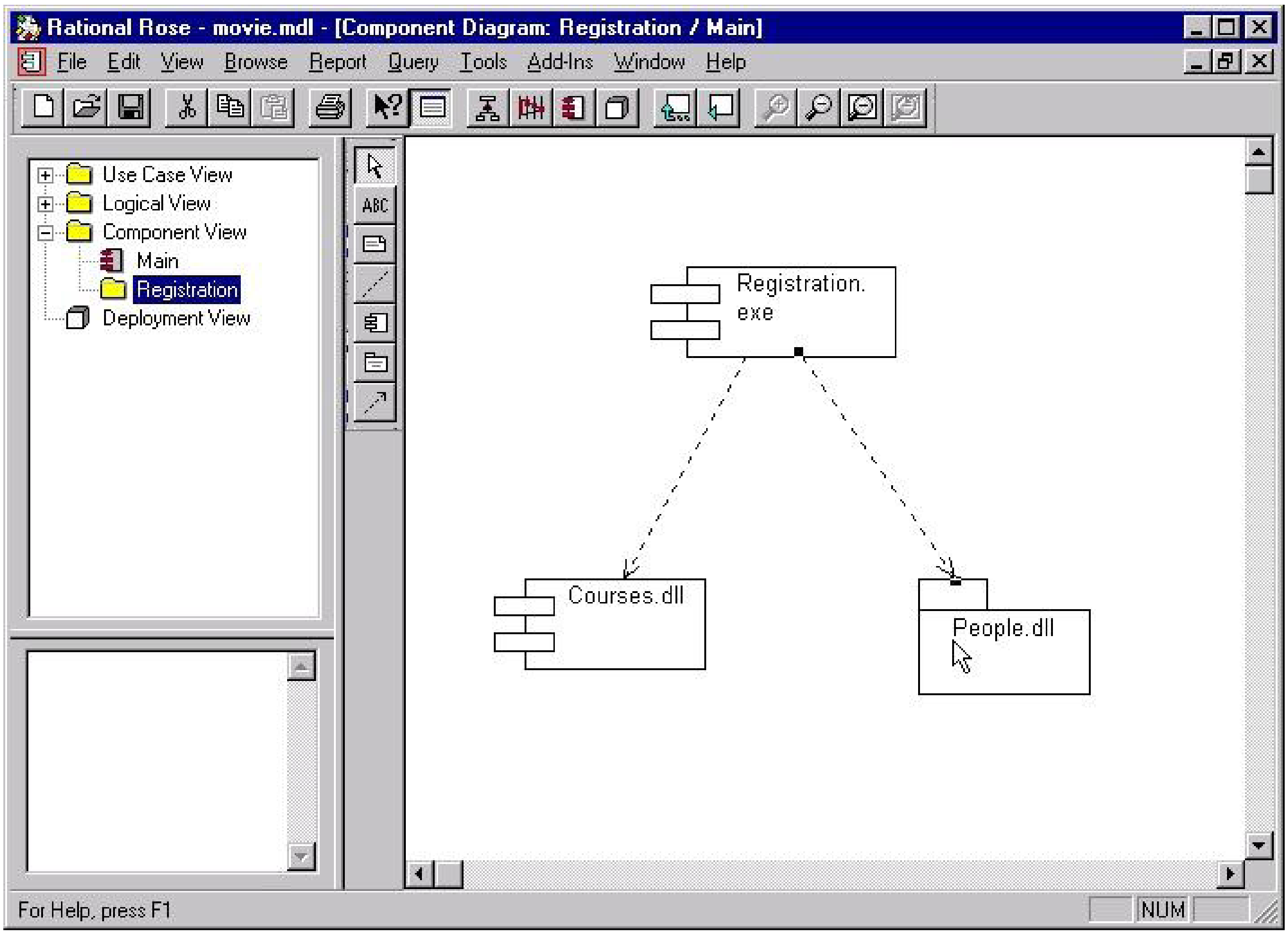


Figure: Component View

Rose automatically creates one component diagram called Main. To create an additional component diagram:
1. Right-click on the owning package (either the component view itself or a user created package) to make the shortcut menu visible.
2. Select the New: Component Diagram menu command. This will place a new component diagram called NewDiagram in the browser.
3. While the new diagram is still selected, enter its name.

Rose will automatically add the new diagram to the browser.

To open a component diagram:
1. Double-click on the diagram in the browser.

To create a component:
1. Click to select the package specification icon from the toolbar.
2. Click on the component diagram to place the component.
3. While the component is still selected, enter its name.

Rose will automatically add the new component to the browser.

To create a dependency relationship:
1. Click to select the dependency icon from the toolbar.
2. Click on the package or component representing the client.
3. Drag the dependency arrow to the package or component representing the supplier.

4. The Deployment View
The deployment view addresses the configuration of run-time processing nodes and the components, processes, and objects that live on them. This view contains only one diagram – the deployment diagram.
Deployment Diagram
The deployment diagram contains nodes and connections. Rose provides two icons that can be used to draw a node – the processor and the device. A processor is a node that has processing capacity.

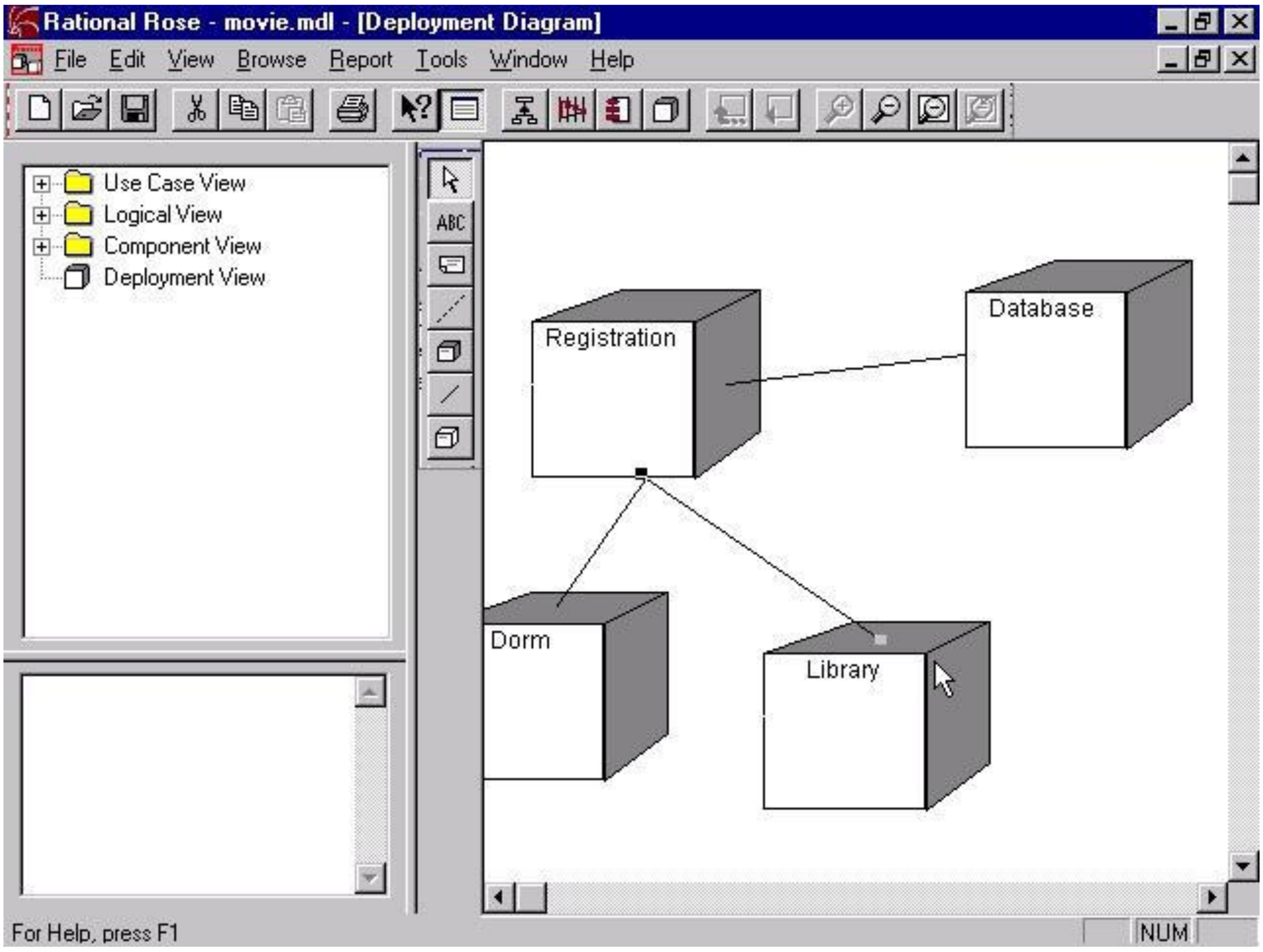


Figure: Deployment View

To open the deployment diagram:
1. Double-click on the Deployment View in the browser.

To create a node:
1. Click to select the processor icon from the toolbar.
2. Click on the deployment diagram to place the node.
3. While the node is still selected, enter its name.

To create a connection:
1. Click to select the connection icon from the toolbar.
2. Click on the node representing the client.
3. Drag the connection line to the node representing the supplier.

5. Working with the Use-case View of UML.

OBJECT

Working with the Use-case View of UML.

THEORY

The use case view of the system addresses the understandability and usability of the system. This view looks at actors and use cases along with their interactions. The diagrams in this view are use case diagrams, sequence diagrams and collaboration diagrams.

Actor

An actor is represented by a stickman. To create an actor:

1. Right-click to select the Use Case View in the browser and make the shortcut menu visible.
2. Select the New: Actor menu command. This will add an actor called NewClass to the browser.
3. While the new class is still selected, enter the name of the actor.

As actors are created, their documentation is added to the model. To add the documentation for an actor:

1. Click to select the actor in the browser.
2. Position the cursor in the Documentation Window.
3. Enter the documentation for the actor.

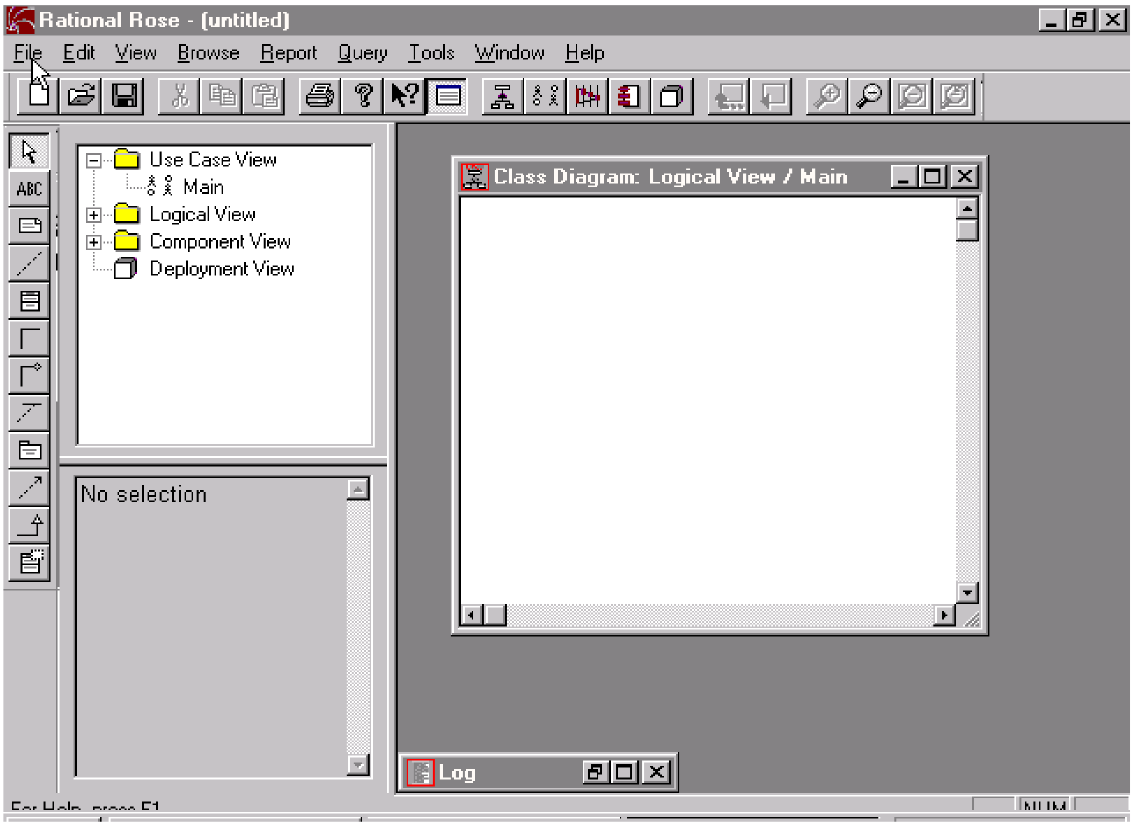

Rational Rose - [untitled]
File Edit View Browse Report Query Tools Window Help
Use Case View
Main
Logical View
Component View
Deployment View
No selection
Class Diagram: Logical View / Main
Log
For Help, press F1
NUM

Figure: Use Case View

Each modeling element is associated with a Specification window. The Specification contains additional information about the element. To view the Specification for an actor:
1. Right-click to select the actor in the browser and make the shortcut menu visible.
2. Select the Specification menu command.

When you view the Specification for an actor, you will notice that the title is "Class Specification for <actor>". This is since an actor is a class in Rose with a stereotype /*Extension of Metaclass*/ of Actor.

Use Case
A use case is represented by an oval. To create a use case:
1. Right-click to select the Use Case View in the browser and make the shortcut menu visible.
2. Select the New: Use Case menu command. This will add an unnamed use case to the browser.
3. While the new use case is still selected, enter the name of the use case.

Use cases are documented in two ways – they have a brief description and a flow of events. To add the brief description for a use case:
1. Click to select the use case in the browser.
2. Position the cursor in the Documentation Window.
3. Enter the brief description for the use case.

The flow of events is captured in documents and/or web pages external to Rose. The documents and web pointers are linked to the use case. To link an external document or web pointer to the use case:
1. Right-click to select the use case in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Files tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert File menu command and enter the name of the document to be linked to the use case.
6. Or, select the Insert URL menu command and enter the www pointer to be linked to the use case.
7. Click the Open button.

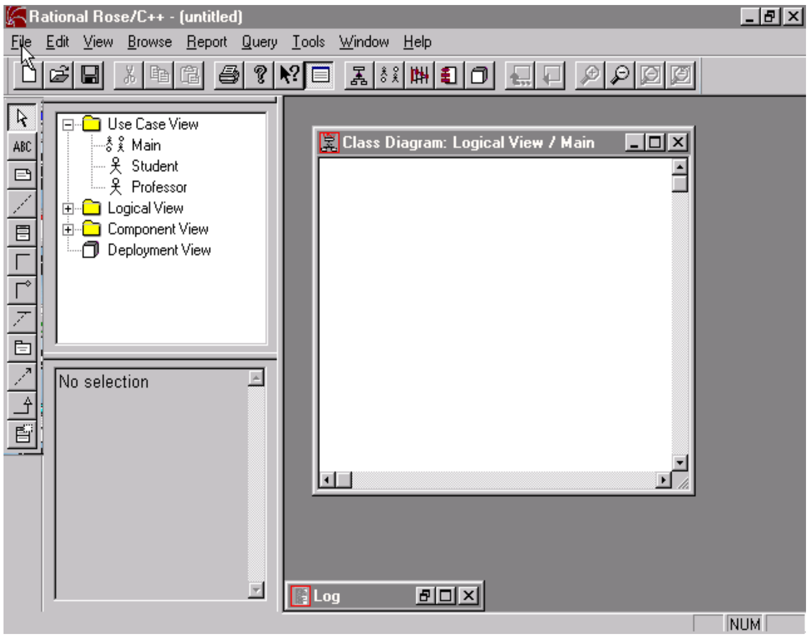

Rational Rose/C++ - (untitled)
File Edit View Browse Report Query Tools Window Help
Use Case View
Main
Student
Professor
Logical View
Component View
Deployment View
No selection
Class Diagram: Logical View / Main
Log
NUM

Figure: Use Cases

Use Case Diagrams

A use case diagram shows the relationships among actors and use cases within the system. The use case view is automatically created with one use case diagram called Main. A system may have other use case diagrams (e.g. a diagram to show all the use cases for one actor).

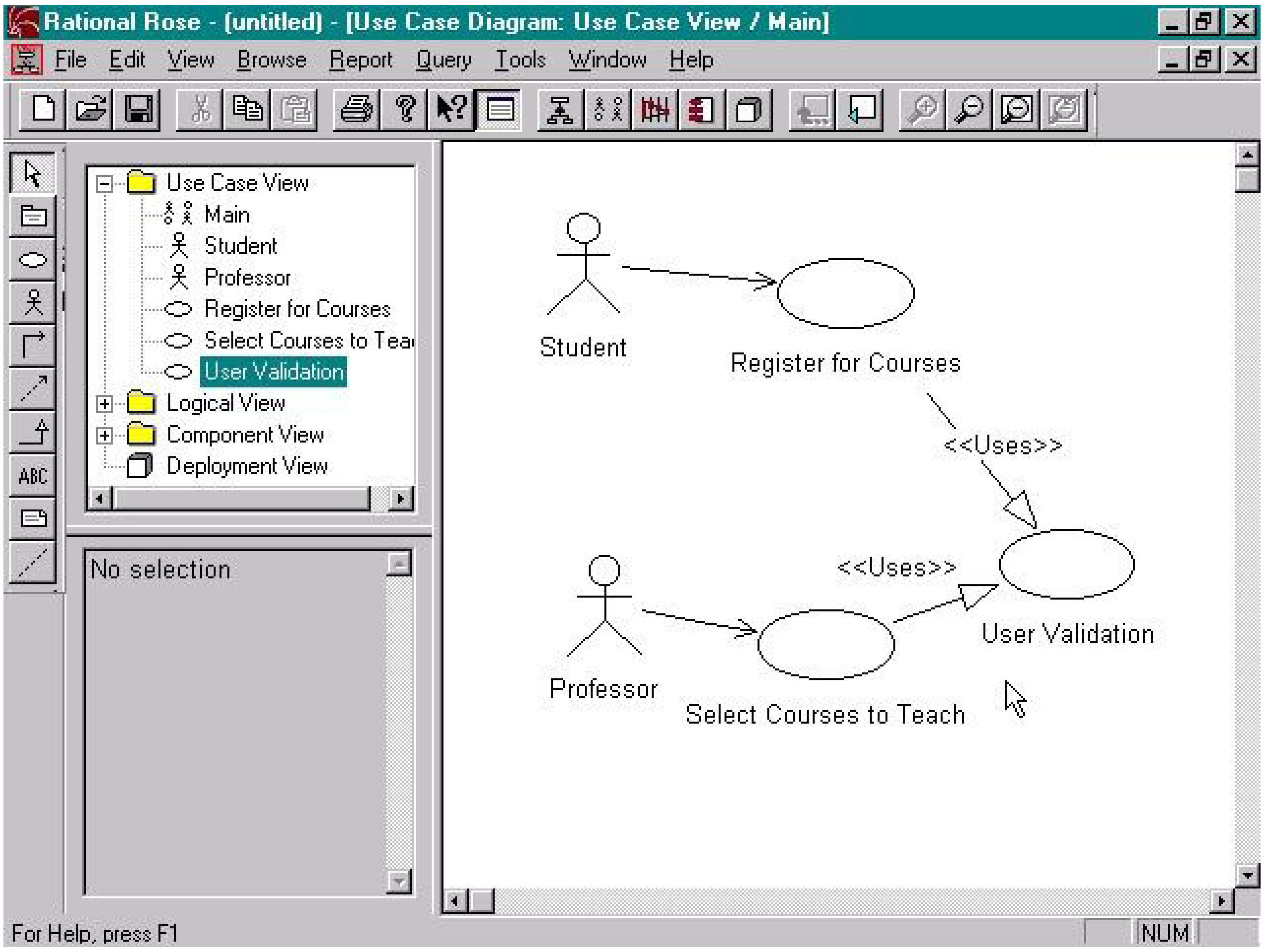


Figure: Use Case Diagram

To open a use case diagram:

1. Click the + next to the Use Case View in the browser to expand the view.
2. Double-click on the diagram you wish to open.
3. Re-size the diagram as needed.

To add an actor or use case to the diagram:

1. Click to select the actor or use case in the browser.
2. Drag the actor or use case to the diagram.

The main type of relationship is a communication relationship. This relationship is shown as a uni-directional association. To add communication relationships to the diagram:
1. Click to select the uni-directional icon from the toolbar.
2. Click on the actor or use case initiating the communication.
3. Drag the relationship line to the participating use case or actor.

A use case diagram may have two other types of relationships – extends and uses relationships. Extends and uses relationships are shown as generalization relationships with stereotypes. To create an extends or uses relationship:
1. Click to select the generalization icon from the toolbar.
2. Click on the using or extension use case.
3. Drag the relationship line to the used or extended use case.

4. Double-click on the relationship line to make the Specification visible.
5. Enter "uses" or "extends" in the Stereotype field.
6. Click the OK button to close the Specification.
7. Right-click on the generalization line to make the shortcut menu visible.
8. Select the Show Stereotype menu command.
Actors and use cases may also be created on a use case diagram. To create an actor or use case on the diagram:
1. Click to select the actor icon or the use case icon from the toolbar.
2. Click on the diagram to place the actor or use case.
3. While the actor or use case is still selected, enter its name.

The actor or use case is automatically added to the browser.

Sequence Diagrams
Use case diagrams present an outside view of the system. The functionality of a use case is captured in its flow of events. Scenarios are used to describe how use cases are realized as interactions among societies of objects. Scenarios are captured in sequence diagrams. Sequence diagrams are associated with use cases.

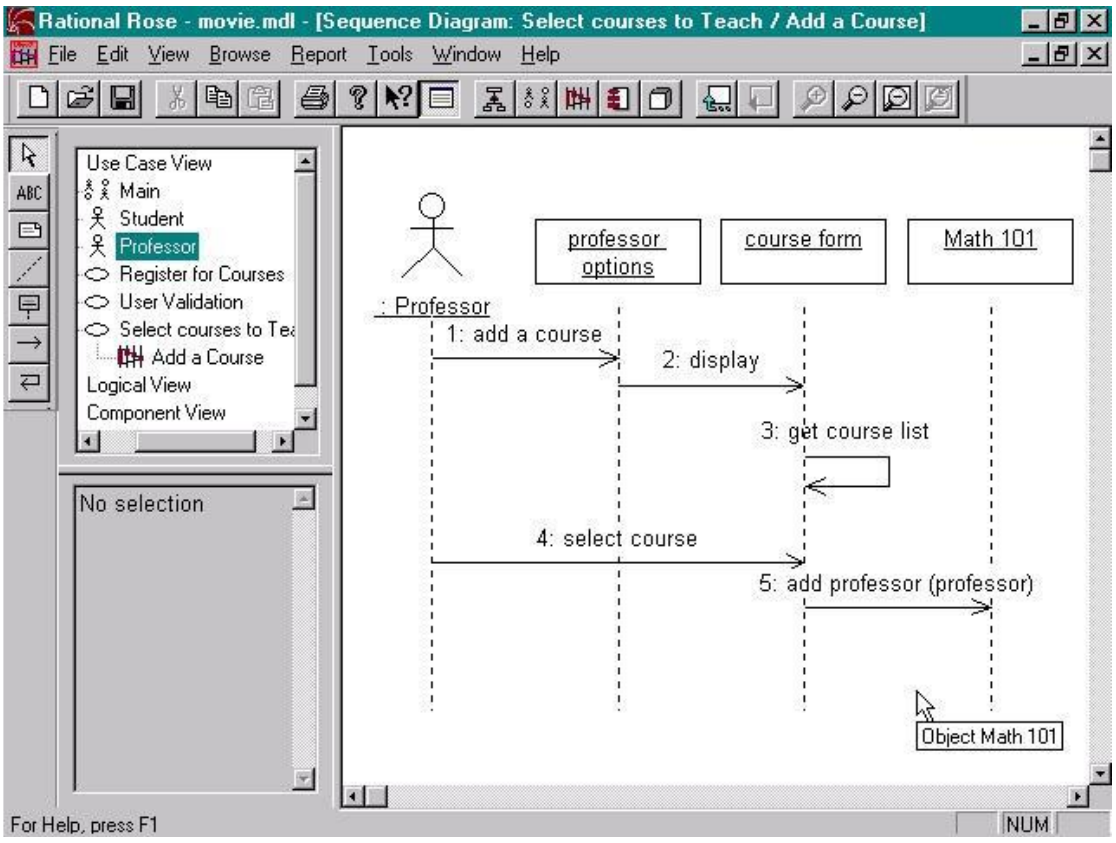


Figure: Sequence Diagram

To create a new sequence diagram:
1. Right-click on the use case in the browser to make the shortcut menu visible.
2. Select the New: Sequence Diagram menu command. This will add a new sequence diagram called NewDiagram to the browser.
3. While the new diagram is still selected, enter the name of the diagram.
4. Double-click on the diagram in the browser to open it.
5. Re-size the sequence diagram as needed.

Sequence diagrams contain actors, objects and messages.
To add an actor to a sequence diagram:
1. Click to select the actor in the browser.
2. Drag the actor onto the diagram.

To add an object to the sequence diagram:
1. Click to select the object icon from the toolbar.
2. Click on the diagram to place the object.
3. While the object is still selected, enter its name.
To create a message:
1. Click to select the message icon from the toolbar.
2. Click on the object representing the client (sender of the message).
3. Drag the message to the object representing the supplier (receiver of the message).
4. While the message line is still selected, enter the name of the message.

To create a reflexive message:
An object may send a message to itself. This is called a reflexive message.
1. Click to select the message to self icon from the toolbar.
2. Click on the object that needs a reflexive message.
3. While the message line is still selected, enter the name of the message.

Collaboration Diagrams
A scenario may also be represented in a collaboration diagram. Like sequence diagrams, collaboration diagrams are associated with use cases.

To create a collaboration diagram directly from a sequence diagram:
1. Select the Browse: Create Collaboration Diagram menu command or use the F5 accelerator.
2. Re-size the diagram as needed.
3. Re-arrange the objects as needed.

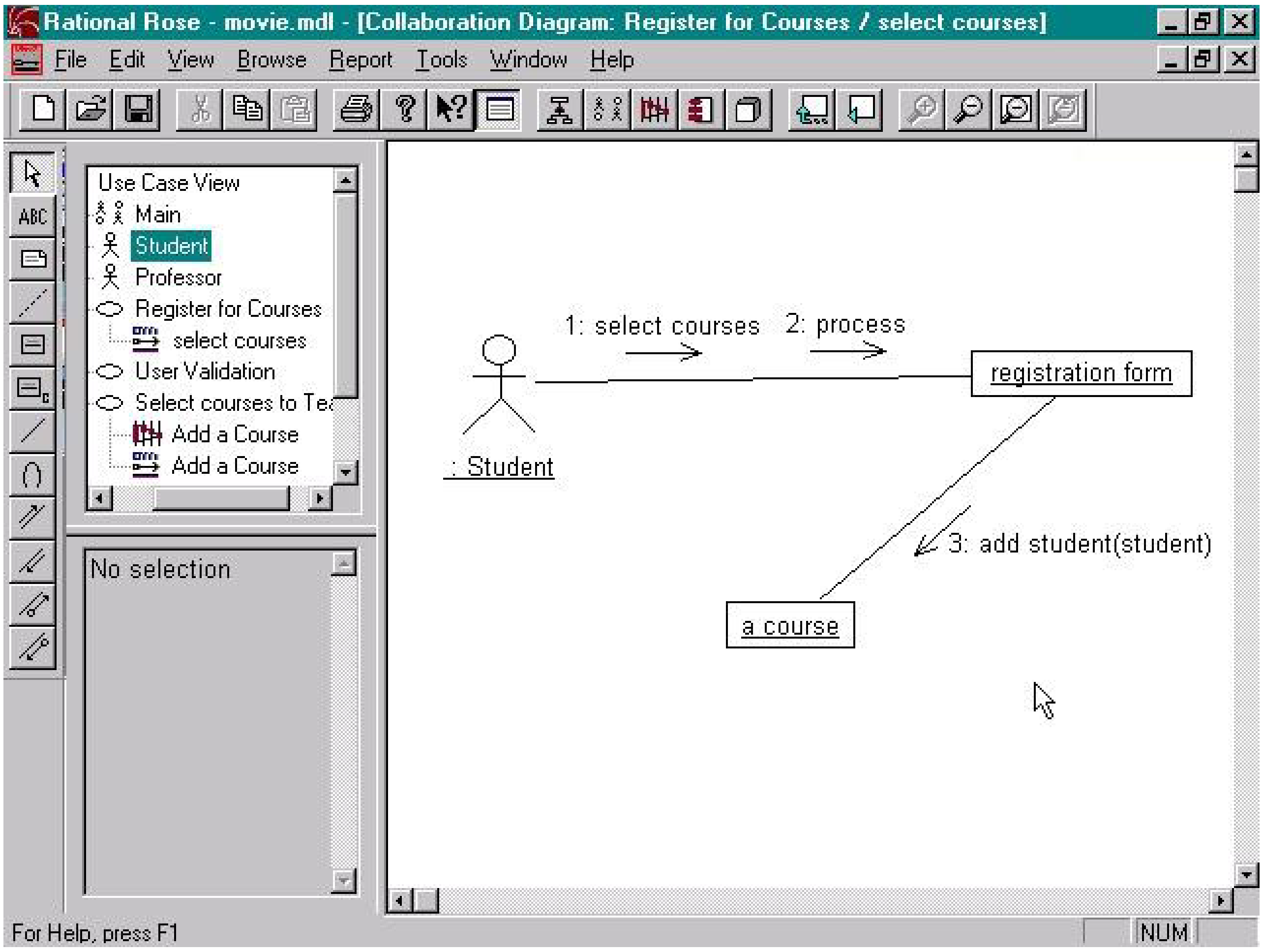


Figure: Collaboration Diagram

6. Working with the Class Diagrams of UML.

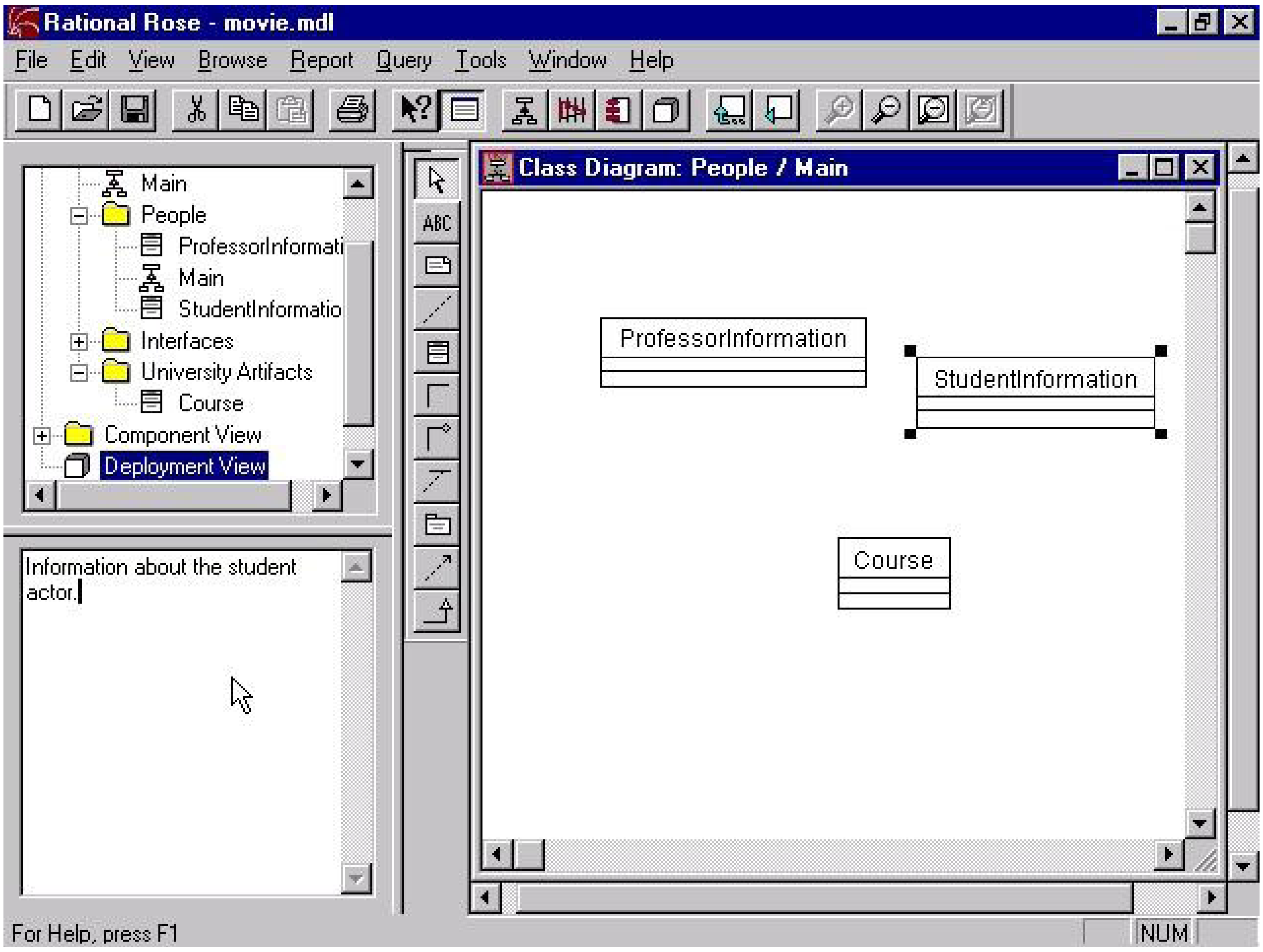
OBJECT

Working with the Class Diagrams of UML.

THEORY

Class Diagrams

The browser provides a textual view of the classes in a system. Class diagrams are created to graphically view the classes and packages in the system. Rose automatically creates a class diagram called Main in the Logical View. This diagram may be opened by double-clicking on it in the browser. The Main class diagram typically contains packages thus, by the end of development it is a graphical view of the major architectural elements of the system. A package may be added to a class diagram by selecting it in the browser and dragging it onto the class diagram.



Information about the student actor.

Figure: The Class Diagram

Each package typically has its own main diagram which is a picture of its key packages and classes. To create the Main class diagram for a package, double-click on the package on a class diagram. Once the Main diagram is created for a package, you can addpackages and classes to the diagram by selecting them in the browser and dragging them onto the diagram.

Classes from any package may be added to a class diagram by selecting the class on the browser and dragging it onto the open class diagram. If the Show Visibility option is set to true either as a default using the Tool: Options menu or individually by using the shortcut menu for the class, the name of the “owning” package is displayed. The package name will be visible for all classes that do not belong to the package owning the class diagram.

Packages and classes may also be created using the class diagram toolbar. To create a package or class using the toolbar:
1. Click to select the icon (package or class) on the class diagram toolbar.
2. Click on the class diagram to place the package or class.
3. While the new package or class is still selected, enter its name.
4. Multiple packages or classes may be created by depressing and holding the Shift key.

Packages and classes created on class diagrams are automatically added to the browser.
To create a class diagram:
1. Right-click to select the owning package and make the shortcut menu visible.
2. Select the New: Class Diagram menu command. This will add a class diagram called NewDiagram to the browser.
3. While the new class diagram is still selected, enter its name.
4. To open the class diagram, double-click on it in the browser.

Class Structure
The structure of a class is represented by its set of attributes. Attributes may be created in the browser, via the Class Specification or on a class diagram.

To create an attribute in the browser:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. To create an attribute, select the New: Attribute menu command. This will add an attribute called name to the browser.
3. While the new attribute is still selected, enter its name.
4. Attribute data types and default values may not be entered via the browser

To create an attribute using the Class Specification:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Attributes tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert menu command. This will insert an attribute called name.
6. While the new attribute is still selected, enter its name. Type and initial value may be filled in at this time or you may choose to fill in this information later in the development cycle.

To create an attribute on a class diagram:
1. Right-click to select the class on the class diagram and make the shortcut menu visible.
2. Select the Insert New Attribute menu command. This will insert an attribute in the form.
name : type = initval
3. While the attribute is still selected, fill in its name. Type and initial value may be filled in at this time or you may choose to fill in this information later in the development cycle.

Attributes of a class may be viewed in the browser. The class will be initially collapsed. To expand the class to view its attributes, click the + next to the class.
Attributes should be documented.

To add the documentation for an attribute:
1. Click to select the attribute in the browser.
2. Position the cursor in the Documentation Window. If the Documentation Window is not visible, select the View: Documentation menu command.
3. Enter the documentation for the attribute.

To delete an attribute:
1. Right-click to select the attribute in the browser or on the Attributes tab of the Class Specification and make the shortcut menu visible.
2. Select the Delete menu command.

Class Behavior
The behavior of a class is represented by its set of operations. Operations may be created in the browser, via the Class Specification or on a class diagram. To create an operation in the browser:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. To create an operation, select the New: Operation menu command. This will add an operation called opname to the browser.
3. While the new operation is still selected, enter its name.
4. The operation signature may not be entered via the browser

To create an operation using the Class Specification:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Operations tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert menu command. This will insert an attribute called opname.
6. While the new operation is still selected, enter its name. The signature and return value may be filled in at this time or you may choose to fill in this information later in the development cycle.

To create an operation on a class diagram:
1. Right-click to select the class on the class diagram and make the shortcut menu visible.
2. Select the Insert New Operation menu command. This will insert an attribute in the form
opname (argname : argtype = default) : return
3. While the operation is still selected, fill in its name. The signature and return value may be filled in at this time or you may choose to fill in this information later in the development cycle.

Operations of a class may be viewed in the browser. The class will be initially collapsed. To expand the class to view its operations, click the + next to the class.
To enter the signature of an operation:
1. Right-click to select the operation in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Detail tab.
4. Right-click in the Arguments field to make the shortcut menu visible.

5. Select the Insert menu command. This will insert an argument called argname of type argtype with a default value of default.
6. Click to select the name, type, or default and enter the desired information.
7. Click the OK button to close the Specification.

To delete an operation:
1. Right-click to select the operation in the browser or on the Operations tab of the Class Specification and make the shortcut menu visible.
2. Select the Delete menu command.

Operations should be documented.
To add the documentation for an operation:
1. Click to select the operation in the browser.
2. Position the cursor in the Documentation Window. If the Documentation Window is not visible, select the View: Documentation menu command.
3. Enter the documentation for the operation.

Relationships
Relationships provide a conduit for communication. There are four different types of relationships: association, aggregation, dependency, and inheritance. Relationships may only be created on a class diagram using the toolbar.

7. Working with the State Transition Diagrams of UML.

OBJECT

Working with the State Transition Diagrams of UML.

THEORY

State Transition Diagrams

State transition diagrams show the life history of a given class, the events that cause a transition from a state, and the actions that result from a state change. They are created for classes whose objects exhibit significant dynamic behavior.

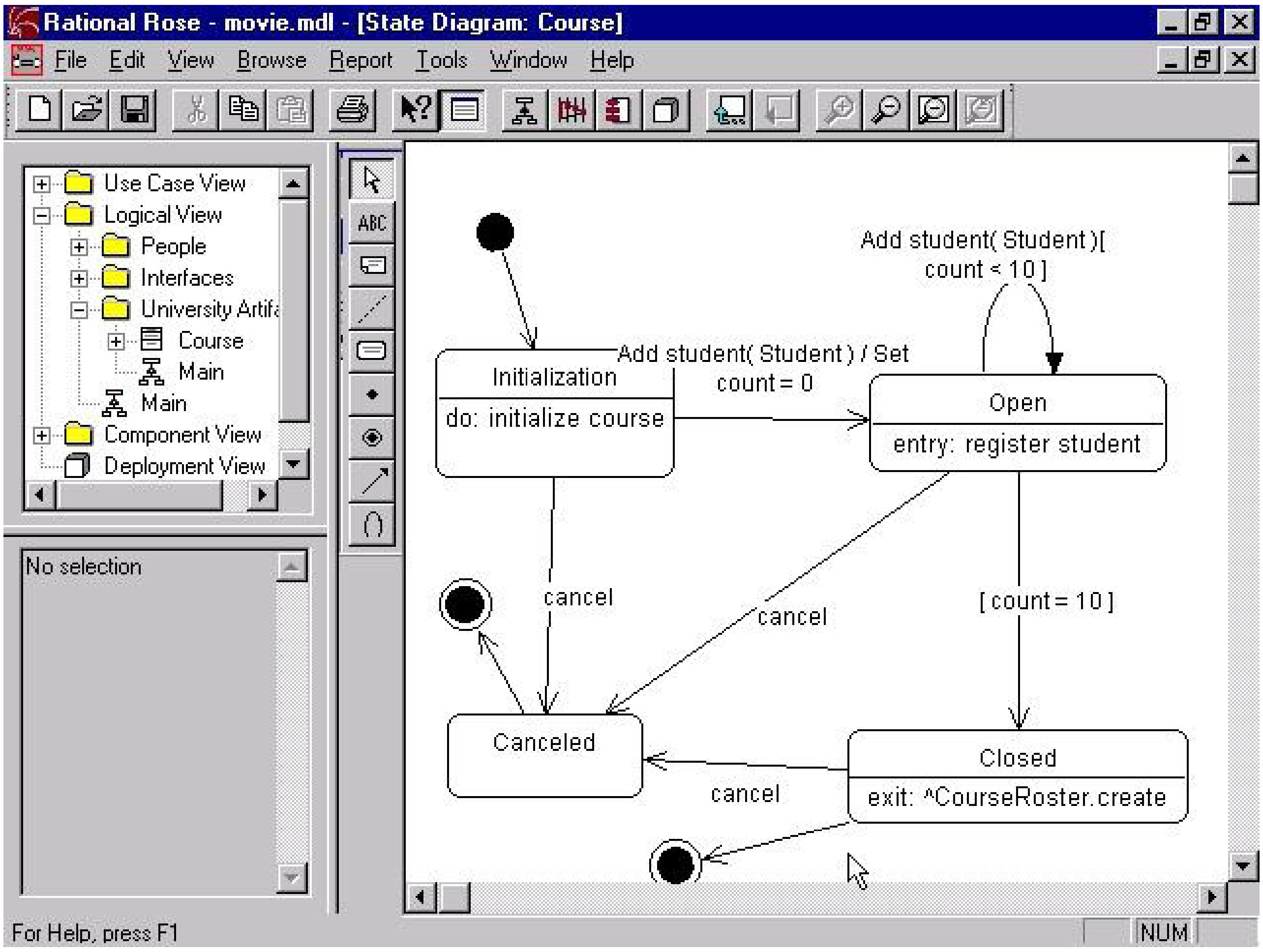


Figure: State Transition Diagram

To create a state transition diagram:

1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the State Diagram menu command.

To open a state transition diagram

1. Click the + next to the class to expand the tree
2. Double-click on the State Diagram for the class

States
A state is represented by an oval.

To create a state
1. Click to select the State icon from the toolbar.
2. Click on the state transition diagram to place the state.
3. While the state is still selected, enter its name.

State Transitions
A state transition is represented as an arrow which points from the originating state to the successor state.

To create a state transition
1. Click to select the state transition arrow from the toolbar.
2. Click on the originating state and drag the arrow to the successor state.

State Actions
Behavior that occurs while an object is in a state can be expressed in three ways: entry actions, activities , and exit actions . The behavior may be a simple event, or it may be an event sent to another object.

To create an entry action, exit action or activity.
1. Point to the state and double click to make the State Specification dialog box visible.
2. Select the Detail tab.
3. Click-right in the Actions field to make the pop-up menu visible.
4. Select the Insert menu choice to insert a new action called entry.
5. Double-click on the action to make the State Action Specification visible.
6. If the action is a simple action, enter the name of the action in the Action field.
7. If the action is a send event action, enter the name of the event to be sent in the Send Event field. If the event has arguments, enter the arguments in the Send Arguments field. Enter the name of the target object (object receiving the event) in the Send Target field.
8. Select the appropriate radio button in the When field (On Enty to create an entry action, On Exit to create an exit action, or Entry Until Exit to create an activity.
9. Click the OK button to close the Action Specification.
10. Click the OK button to close the State Specification.

Start and Stop States
There are two special states associated with state transition diagrams – the start state and the stop state. The start state is represented by a filled in circle and the stop state is represented by a bulls eye.

To create a start state
1. Click to select the start state icon from the toolbar.
2. Click on the diagram to place the start state on the diagram.
3. Click to select the state transition icon from the toolbar.
4. Click on the start state and drag the state transition arrow to the successor state.

To create a stop state
1. Click to select the stop state icon from the toolbar.
2. Click on the diagram to place the stop state on the diagram.
3. Click to select the state transition icon from the toolbar.
4. Click on the originating state and drag the state transition arrow to the stop state.

8. Identifying the Requirements from Problem Statements

Identifying the Requirements from Problem Statements

Objectives
After completing this experiment, you will be able to:
Identify ambiguities, inconsistencies and incompleteness from a requirement specification
Identify and state functional requirements.
Identify and state non-functional requirements.

This documentation will serve as reference for the subsequent design, implementation, and verification of the system.

It is necessary and important that before we start planning, design and implementation of the software system for our client, we are clear about it's requirements. If we don't have a clear vision of what is to be developed and what all features are expected, there would be serious problems, and customer dissatisfaction as well.
Characteristics of Requirements

Requirements gathered for any new system to be developed should exhibit the following three properties:

Unambiguity: There should not be any ambiguity what a system to be developed should do. For example, consider you are developing a web application for your client. The client requires that enough number of people should be able to access the application simultaneously. What's the "enough number of people"? That could mean 10 to you, but, perhaps, 100 to the client. There's an ambiguity.

Consistency: To illustrate this, consider the automation of a nuclear plant. Suppose one of the clients say that it the radiation level inside the plant exceeds R1, all reactors should be shut down. However, another person from the client side suggests that the threshold radiation level should be R2. Thus, there is an inconsistency between the two end users regarding what they consider as threshold level of radiation.

Completeness: A particular requirement for a system should specify what the system should do and also what it should not. For example, consider a software to be developed for ATM. If a customer enters an amount greater than the maximum permissible withdrawal amount, the ATM should display an error message, and it should not dispense any cash.
Categorization of Requirements

Based on the target audience or subject matter, requirements can be classified into different types, as stated below:
User requirements: They are written in natural language so that both customers can verify their requirements have been correctly identified.

System requirements: They are written involving technical terms and/or specifications and are meant for the development or testing teams.
Requirements can be classified into two groups based on what they describe:

Functional requirements (FRs): These describe the functionality of a system -- how a system should react to a particular set of inputs and what should be the corresponding output.

Non-functional requirements (NFRs): They are not directly related what functionalities are expected from the system. However, NFRs could typically define how the system should behave under certain situations. For example, a NFR could say that the system should work with 128MB RAM. Under such condition, a NFR could be more critical than a FR.

Non-functional requirements could be further classified into different types like:

Product requirements: For example, a specification that the web application should use only plain HTML, and no frames.

Performance requirements: For example, the system should remain available 24x7.

Organizational requirements: The development process should comply to SEI CMM level 4.
Functional Requirements

Identifying Functional Requirements
Given a problem statement, the functional requirements could be identified by focusing on the following points:

Identify the high-level functional requirements simply from the conceptual understanding of the problem. For example, a Library Management System, apart from anything else, should be able to issue and return books.
Identify the cases where an end user gets some meaningful work done by using the system. For example, in a digital library a user might use the "Search Book" functionality to obtain information about the books of his interest.
If we consider the system as a black box, there would be some inputs to it, and some output in return. This black box defines the functionalities of the system. For example, to search for a book, user gives title of the book as input and get the book details and location as the output.
Any high-level requirement identified could have different sub-requirements. For example, "Issue Book" module could behave differently for different class of users, or for a particular user who has issued the book thrice consecutively.

Preparing Software Requirements Specifications
Once all possible FRs and non-FRs have been identified, which are complete, consistent, and non-ambiguous, the Software Requirements Specification (SRS) is to be prepared. IEEE provides a template [iv], also available here, which could be used for this purpose. The SRS is prepared by the service provider and verified by its client. This document serves as a legal agreement between the client and the service provider. Once the concerned system has been developed and deployed, and a proposed feature was not found to be present in the system, the client can point this out from the SRS. Also, if after delivery, the client says a new feature is required, which was not mentioned in the SRS, the service provider can again point to the SRS. The scope of the current experiment, however, doesn't cover writing an SRS.