

Raj Kumar Goel Institute of Technology, Ghaziabad



LABORATORY MANUAL

Faculty Name : Shailja Varshney

Department : CSE

Course Name : SE Lab

Course Code : RCS651

Year/Sem : 3rd/6th

NBA Code : C318

Email ID : shailfcs@rkgit.edu.in

Academic Year : 2020-21

Raj Kumar Goel Institute of Technology, Ghaziabad
Department of Computer Science & Engineering

VISION OF THE INSTITUTE

To continually develop excellent professionals capable of providing sustainable solutions to challenging problems in their fields and prove responsible global citizens.

MISSION OF THE INSTITUTE

We wish to serve the nation by becoming a reputed deemed university for providing value based professional education.

VISION OF THE DEPARTMENT

To be recognized globally for delivering high quality education in the ever changing field of computer science & engineering, both of value & relevance to the communities we serve.

MISSION OF THE DEPARTMENT

1. To provide quality education in both the theoretical and applied foundations of Computer Science and train students to effectively apply this education to solve real world problems.
2. To amplify their potential for lifelong high quality careers and give them a competitive advantage in the challenging global work environment.

PROGRAM EDUCATIONAL OUTCOMES (PEOs)

PEO 1: Learning: Our graduates to be competent with sound knowledge in field of Computer Science & Engineering.

PEO 2: Employable: To develop the ability among students to synthesize data and technical concepts for application to software product design for successful careers that meet the needs of Indian and multinational companies.

PEO 3: Innovative: To develop research oriented analytical ability among students to prepare them for making technical contribution to the society.

PEO 4: Entrepreneur / Contribution: To develop excellent leadership quality among students which they can use at different levels according to their experience and contribute for progress and development in the society.

Raj Kumar Goel Institute of Technology, Ghaziabad
Department of Computer Science & Engineering

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to

Raj Kumar Goel Institute of Technology, Ghaziabad
Department of Computer Science & Engineering

comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: The ability to use standard practices and suitable programming environment to develop software solutions.

PSO2: The ability to employ latest computer languages and platforms in creating innovative career opportunities.

Raj Kumar Goel Institute of Technology, Ghaziabad
Department of Computer Science & Engineering

COURSE OUTCOMES (COs)

CO1	Identify ambiguities, inconsistencies and incompleteness from a requirements specification and state functional and non-functional requirement
CO2	Identify different actors and use cases from a given problem statement and draw use case diagram to associate use cases with different types of relationship
CO3	Draw a class diagram after identifying classes and association among them
CO4	Graphically represent various UML diagrams , and associations among them and identify the logical sequence of activities undergoing in a system, and represent them pictorially
CO5	Able to use modern engineering tools for specification, design, implementation and testing

CO-PO &PSOs MAPPING

CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO1 0	PO1 1	PO1 2	PSO 1	PSO 2
C318. 1	2	2	3	2	1		1		2	3	2	3	2	1
C318. 2	2	2	2	2	1		1		2	2	1	2	2	1
C318. 3	2	2	2	2	1		1		2	2	1	2	2	1
C318. 4	2	2	2	3	1		1		2	3	1	2	2	1
C318. 5	3	3	2	2	3		1		2	3	2	2	2	2
CO31 8	2.2	2.2	2.2	2.2	1.4		1		2	2.6	1.4	2.2	2	1.2

Raj Kumar Goel Institute of Technology, Ghaziabad
Department of Computer Science & Engineering

LIST OF EXPERIMENTS

Expt. No.	Title of experiment	Corresponding CO
1.	Prepare a SRS document in line with the IEEE recommended standards	C 318.1
2.	Draw the use case diagram and specify the role of each of the actors Also state the precondition, post condition and function of each use case for a given problem.	C 318.2
3.	Draw the activity diagram for a given problem	C 318.3
4.	Identify the classes. Classify them as weak and strong classes and draw the class diagram for a given problem.	C 318.3
5.	Draw the sequence diagram for any two scenarios for a given problem	C 318.3
6.	Draw the collaboration diagram for a given problem.	C 318.3
7.	Draw the state chart diagram for a given problem.	C 318.3
8.	Draw the component diagram for a given problem.	C 318.3
9.	Draw the deployment diagram for a given problem.	C 318.4

Content Beyond Syllabus		
10.	Draw Data Flow Diagram	C 318.3
11.	Write a program to compute cyclomatic complexity.	C 318.5
12.	Write a program to compute Halstead's Complexity	C 318.4

INTRODUCTION

Scope of Software Engineering: Software engineering is a discipline whose aim is the production of fault-free software that is delivered on time, within budget, and satisfies the user's needs. Software engineering is about teams. The problems to solve are so complex or large, that a single developer cannot solve them anymore. Software engineering is also about communication. Teams do not consist only of developers, but also of testers, architects, system engineers, customer, project managers, etc. Software projects can be so large that we have to do careful planning. Implementation is no longer just writing code, but it is also following guidelines, writing documentation and also writing unit tests. But unit tests alone are not enough. The different pieces have to fit together. And we have to be able to spot problematic areas using metrics. They tell us if our code follows certain standards. Once we are finished coding, that does not mean that we are finished with the project: for large projects maintaining software can keep many people busy for a long time. Since there are so many factors influencing the success or failure of a project?

We also need to learn a little about project management and its pitfalls, but especially what makes projects successful. There are four fundamental phases in most, if not all, software engineering methodologies. These phases are analysis, design, implementation, and testing. These phases address what is to be built, how it will be built, building it, and making it high quality.

INTRODUCTION TO UML (UNIFIED MODELING LANGUAGE)

The UML is a language for specifying, constructing, visualizing, and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics. The rules and semantics of a model are expressed in English in a form known as OCL (Object Constraint Language). OCL uses simple logic for specifying the properties of a system. The UML is not intended to be a visual programming language. However, it has a much closer mapping to object-oriented programming languages, so that the best of both can be obtained. The UML is much simpler than other methods preceding it. UML is appropriate for modelling systems, ranging from enterprise information system to distributed web-based application and even to real time embedded system. It is a very expensive language addressing all views needed to develop and then to display system even though understand to use. Learning to apply UML effectively starts forming a conceptual mode of languages which requires learning.

Department of Computer Science & Engineering

The primary goals in the design of UML are:

1. Provides users ready to use, expressive visual modeling language as well so they can develop

2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. To be independent of particular programming languages and development processes.
4. Provide formal basis for understanding the modelling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts.
7. Integrate best practices and methodologies. Every complex system is best approached through a small set of nearly independent views of model. Every model can be expressed at different levels of fidelity. The best models are connected to reality. The UML defines nine graphical diagrams:

1. Class diagram
2. Use-case diagram
3. Behaviour diagram
 - 3.1. Interaction diagram
 - 3.1.1. Sequence diagram
 - 3.1.2. Collaboration diagram
 - 3.2. State chart diagram
 - 3.3. Activity diagram
4. Implementation diagram
 - 4.1 component diagram
 - 4.2 deployment diagram

1. UML class diagram:

The UML class diagram is also known as object modeling. It is a static analysis diagram. These diagrams show the static structure of the model. A class diagram is a connection of static model elements, such as classes and their relationships, connected as a graph to each other and to their contents.

2. Use-case diagram:

The functionality of a system can be described in a number of different use cases, each of which represents a specific flow of events in a system. It is a graph of actors, a set of use-cases enclosed in a boundary, communication, associations between the actors and the use-cases, and generalization among the use-cases.

Department of Computer Science & Engineering

3. Behavior diagram:

It is a dynamic model unlike all the others mentioned before. The objects of an object-oriented system are not static and are not easily understood by static diagrams. The behaviour of the class's instance (an object) is represented in this diagram. Every use-case of the system has an associated behavior diagram that indicates the behaviour of the object. In conjunction with the use-case diagram we may provide a script or interaction diagram to show a time line of events. It

4. Interaction diagram:

It is the combination of sequence and collaboration diagram. It is used to depict the flow of events in the system over a timeline. The interaction diagram is a dynamic model which shows how the system behaves during dynamic execution.

5. State chart diagram:

It consists of state, events and activities. State diagrams are a familiar technique to describe the behaviour of a system. They describe all of the possible states that particular object can get into and how the object's state changes as a result of events that reach the object. In most OO techniques, state diagrams are drawn for a single class to show the lifetime behavior of a single object.

6. Activity diagram:

It shows organization and their dependence among the set of components. These diagrams are particularly useful in connection with workflow and in describing behavior that has a lot of parallel processing. An activity is a state of doing something: either a real-world process, or the execution of a software routine.

7. Implementation diagram:

It shows the implementation phase of the systems development, such as the source code structure and the run-time implementation structure. These are relatively simple high level diagrams compared to the others seen so far. They are of two sub diagrams, the component diagram and the deployment diagram.

8. Component diagram:

These are organizational parts of a UML model. These are boxes to which a model can be decomposed. They show the structure of the code itself. They model the physical components such as source code, user interface in a design. It is similar to the concept of packages.

Department of Computer Science & Engineering

PREFACE

This manual is intended for the Third year students of Computer Science & Engineering in the subject of Software Engineering. This manual typically contains practical/Lab Sessions related Software Engineering covering various aspects related the subject to enhanced understanding. The manual contains procedures, and pre-experiment questions to help students prepare for experiments

Raj Kumar Goel Institute of Technology, Ghaziabad

Although, as per the syllabus, study of SDLC is prescribed, we have made the efforts to cover various aspects of Software Engineering covering different development models , Cost Estimations ,software engineering principles to develop the project which contains necessary documents such as SRS , Design details ,User interface, neatly documented code, testing methods etc elaborative understandable concepts and conceptual visualization.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

Though all the efforts have been made to make this manual error free, yet some errors might have crept in inadvertently. Suggestions from the readers for the improvement of the manual are most welcomed.

Good Luck for your Enjoyable Laboratory Sessions

Ms. Harshita Bhardwaj , Assistant Professor, Dept. of CSE

Ms. Shailja Varshney, Assistant Professor, Dept. of CSE

Mr. Piyoush Gupta, Assistant Professor, Dept. of CSE

Department of Computer Science & Engineering

DO'S AND DONT'S

DO's

1. Conform to the academic discipline of the department.
2. Enter your credentials in the laboratory attendance register.
3. Read and understand how to carry out an activity thoroughly before coming to the laboratory.
4. Ensure the uniqueness with respect to the methodology adopted for carrying out the experiments.
5. Shut down the machine once you are done using it.

Raj Kumar Goel Institute of Technology, Ghaziabad

DONT'S

1. Eatables are not allowed in the laboratory.
2. Usage of mobile phones is strictly prohibited.
3. Do not open the system unit casing.
4. Do not remove anything from the computer laboratory without permission.
5. Do not touch, connect or disconnect any plug or cable without your faculty/laboratory technician's permission.

Raj Kumar Goel Institute of Technology, Ghaziabad
Department of Computer Science & Engineering

GENERAL SAFETY INSTRUCTIONS

1. Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.
2. Report fire or accidents to your faculty /laboratory technician immediately.
3. Report any broken plugs or exposed electrical wires to your faculty/laboratory technician immediately.
4. Do not plug in external devices without scanning them for computer viruses.

GUIDELINES FOR LABORTORY RECORD PREPARATION

While preparing the lab records, the student is required to adhere to the following guidelines:

Contents to be included in Lab Records:

1. Cover page
2. Vision
3. Mission
4. PEOs
5. POs
6. PSOs
7. COs
8. CO-PO-PSO mapping
9. Index
10. Experiments
 - Aim
 - Source code
 - Input-Output

A separate copy needs to be maintained for pre-lab written work.

The student is required to make the Lab File as per the format given on the next two pages.

Raj Kumar Goel Institute of Technology, Ghaziabad



SOFTWARE ENGINEERING LAB FILE (KCS 651)

Name	
Roll No.	
Section- Batch	

INDEX

Experiment No.	Experiment Name	Date of Conduction	Date of Submission	Faculty Signature

Raj Kumar Goel Institute of Technology, Ghaziabad
GUIDELINES FOR ASSESSMENT

Students are provided with the details of the experiment (Aim, pre-experimental questions, procedure etc.) to be conducted in next lab and are expected to come prepared for each lab class. Faculty ensures that students have completed the required pre-experiment questions and they complete the in-lab programming assignment(s) before the end of class. Given that the lab programs are meant to be formative in nature, students can ask faculty for help before and during the lab class.

Students' performance will be assessed in each lab based on the following Lab Assessment Components:

Assessment Criteria-1: Performance (Max. marks = 5)

Assessment Criteria-2: VIVA (Max. marks = 5)

Assessment Criteria-3: Record (Max. marks = 5)

In each lab class, students will be awarded marks out of 5 under each component head, making it total out of 15 marks.

Aim: Prepare a SRS document in line with the IEEE recommended standards for a given problem.

Pre-Experiment Questions:

1. Define Software.
2. Define Software engineering.
3. What does requirement gathering and analysis mean?
4. How mini project topic was selected.
5. What analysis you have done before finalizing the topic.
6. What is scope of project?

Procedure:

Step 1:

Introduction:

Purpose

Identify the product whose software requirements are specified in this document. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem. Describe the different types of user that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

Project Scope

Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. An SRS that specifies the next release of an evolving product should contain its own scope statement as a subset of the long-term strategic product vision.

Step 2:

Overall Description

Product Perspective

Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

Product Features

Summarize the major features the product contains or the significant functions that it performs or lets the user perform. Only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or a class diagram, is often effective.

User Classes and Characteristics

Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the favored user classes from those who are less important to satisfy.

Operating Environment

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

Design and Implementation Constraints

Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).

Step 3:

System Features

This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.

System Feature 1

Don't really say "System Feature 1." State the feature name in just a few words.

1 Description and Priority

Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).

2 Stimulus/Response Sequences

List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.

3 Functional Requirements

Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary.

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

Step 4:

External Interface Requirements

User Interfaces

Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

Hardware Interfaces

Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

Software Interfaces

Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

Communications Interfaces

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

Nonfunctional Requirements

Performance Requirements

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific

as possible. You may need to state performance requirements for individual functional requirements or features.

Safety Requirements

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

Security Requirements

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

Software Quality Attributes

Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

Other Requirements

Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.

Post-Experiment Questions:

1. Document the SRS of your project.
2. Why we need SRS in any Project.
3. Which part of SRS is more important?
4. What is the difference between functional and nonfunctional requirement.

Aim: Draw the use case diagram and specify the role of each of the actors Also state the precondition, post condition and function of each use case for a given problem.

Description:

Introduction: The Use Case Approach

- Ivar Jacobson & others introduced Use Case approach for elicitation & modeling.
- Use Case – give functional view. Use Cases are structured outline or template for the description of user requirements modeled in a structured language like English.
- Use case Scenarios are unstructured descriptions of user requirements.
- Use case diagrams are graphical representations that may be decomposed into further levels of abstraction.
- Actor: An actor or external agent, lies outside the system model, but interacts with it in some way.

Actor→Person, machine, information System

- Jacobson & others proposed a template for writing Use cases as shown below:

1. Introduction

Describe a quick background of the use case.

2. Actors

List the actors that interact and participate in the use cases.

3. Pre-Conditions

Pre-conditions that need to be satisfied for the use case to perform.

4. Post Conditions

Define the different states in which we expect the system to be in, after the use case executes.

5. Flow of events

5.1 Basic Flow

List the primary events that will occur when this use case is executed.

5.2 Alternative Flows

Any Subsidiary events that can occur in the use case should be separately listed. List each such event as an alternative flow.

A use case can have many alternative flows as required.

6. Special Requirements

Business rules should be listed for basic & information flows as special requirements in the use case narration. These rules will also be used for writing test cases. Both success and failures scenarios should be described.

7. Use Case relationships

For Complex systems it is recommended to document the relationships between use cases. Listing the relationships between use cases also provides a mechanism for traceability.

Pre-Experiment Questions:

Q1. What are functional requirements?

Q2. What is USE CASE Diagram?

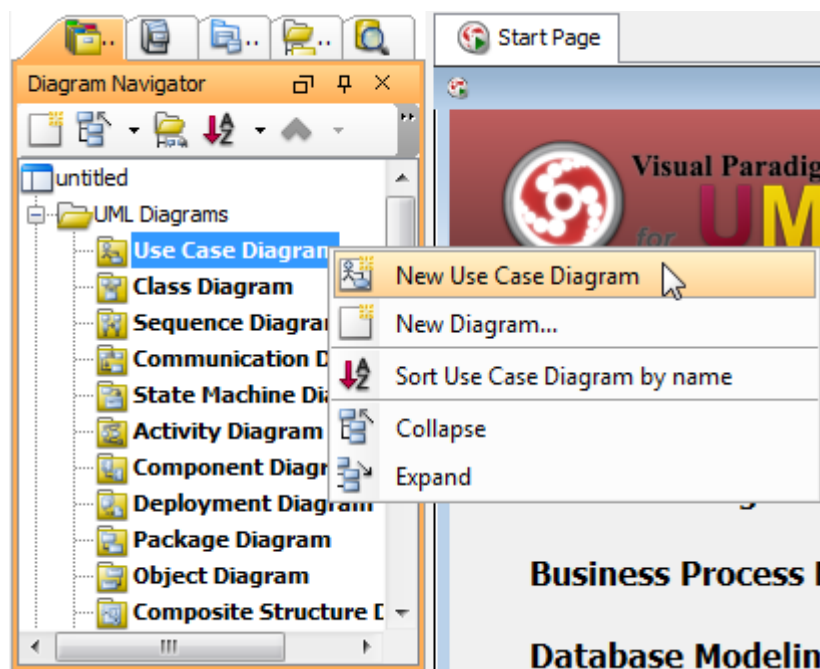
Software Required:-

Visual Paradigm for UML 8.2

Procedure:

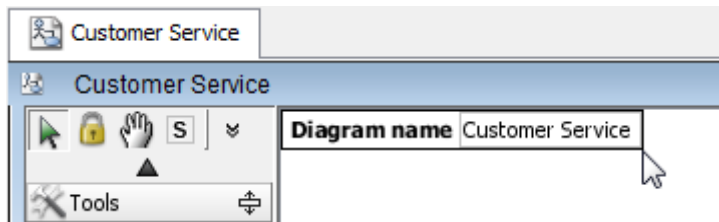
You can draw use case diagrams in VP-UML as well as to document the event flows of use cases using the flow-of-events editor of UML 8.2. The steps are as follows.

Step 1:- Right click **Use Case Diagram** on **Diagram Navigator** and select **New Use Case Diagram** from the pop-up menu.



Step 2:-

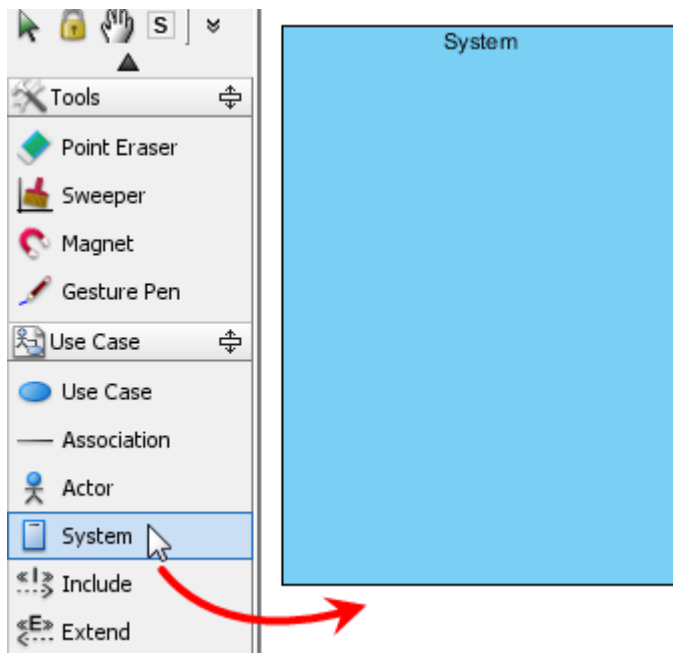
Enter name for the newly created use case diagram in the text field of pop-up box on the top left corner.



Step 3:

Drawing a system

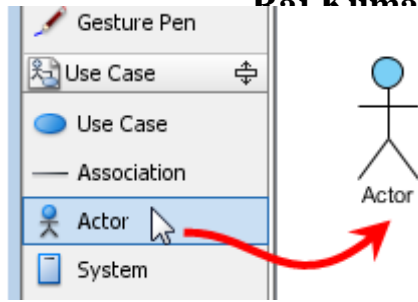
To create a system, select **System** on the diagram toolbar and then click it on the diagram pane. Finally, name the newly created system when it is created



Step 4:

Drawing an actor

To draw an actor, select Actor on the diagram toolbar and then click it on the diagram pane. Finally, name the newly created actor when it is created

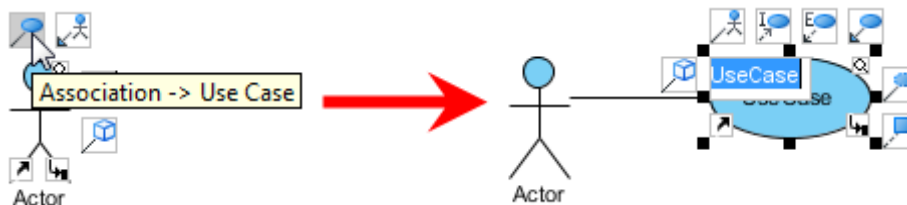


Step 5:-

Drawing a use case

Besides creating a use case through diagram toolbar, you can also create it through resource icon.

Move the mouse over a shape and press a resource icon that can create use case. Drag it and then release the mouse button until it reaches to your preferred place. The source shape and the newly created use case are connected. Finally, name the newly created use case.



Step 6:-

Create a use case through resource icon

Line wrapping use case name

If a use case is too wide, for a better outlook, you may resize it by dragging the filled selectors. As a result, the name of use case will be line-wrapped automatically.

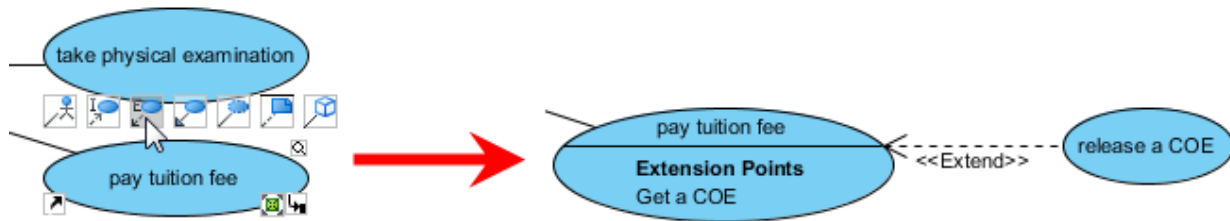


Step 7:-

Resize a use case

To create an extend relationship, move the mouse over a use case and press its resource icon **Extend -> Use Case**. Drag it to your preferred place and then release the mouse button. The use case with extension points and a newly created use case are connected. After you

name the newly created use case, a pop-up dialog box will ask whether you want the extension point to follow the name of use case. Click Yes if you want it to do so; click NO if you want to enter another name for extension point.

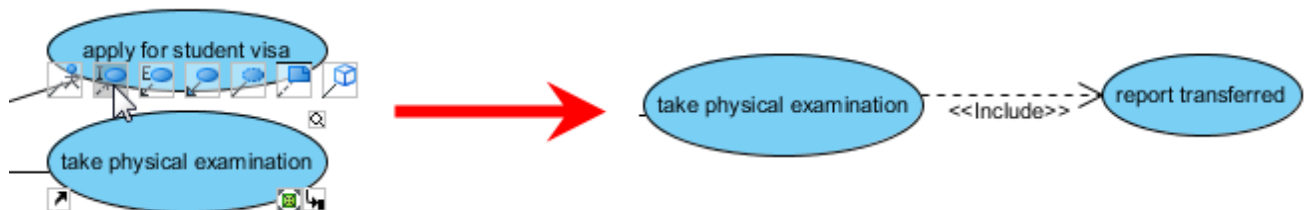


Step 8:-

Create an extend relationship

Drawing <<Include>> relationship

To create an include relationship, mouse over a use case and press its resource icon Include -> Use Case. Drag it to your preferred place and then release the mouse button. A new use case together with an include relationship is created. Finally, name the newly created use case.



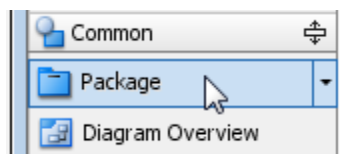
Step 9:-

Include relationship is created

Structuring use cases with package

You can organize use cases with package when there are many of them on the diagram.

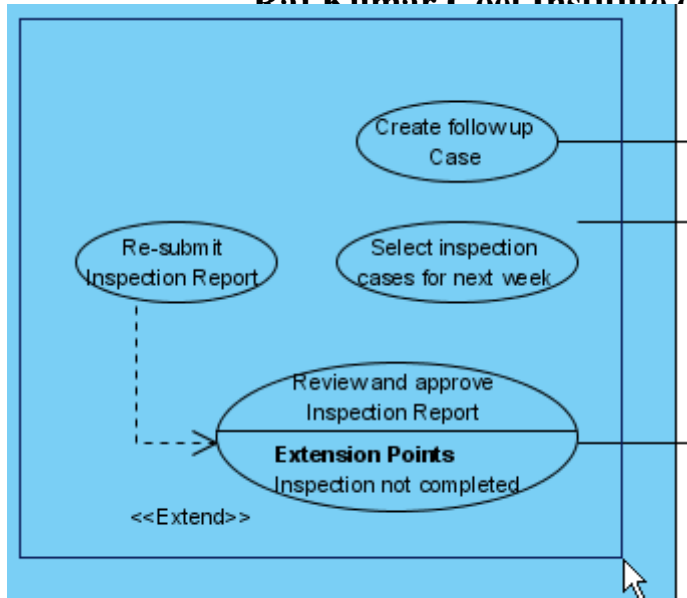
Select Package on the diagram toolbar (under Common category).



Step 10:-

Create a package

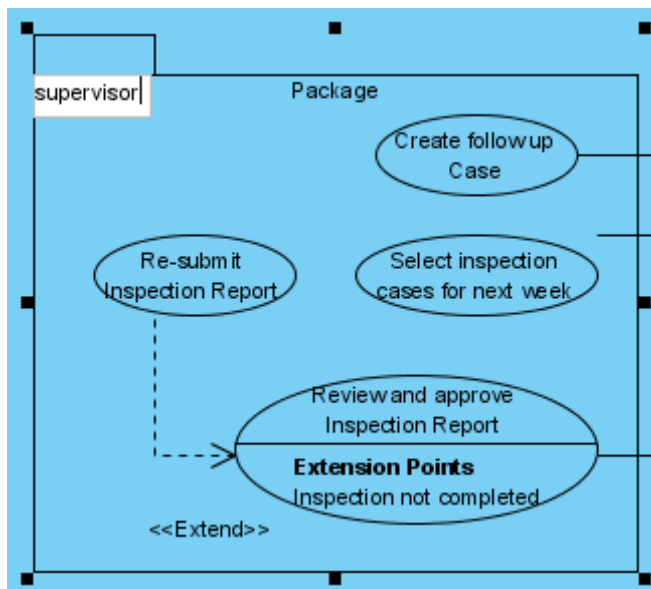
Drag the mouse to create a package surrounding those use cases.



Step 11:

Surround use cases with package

Finally, name the package.



Step 12

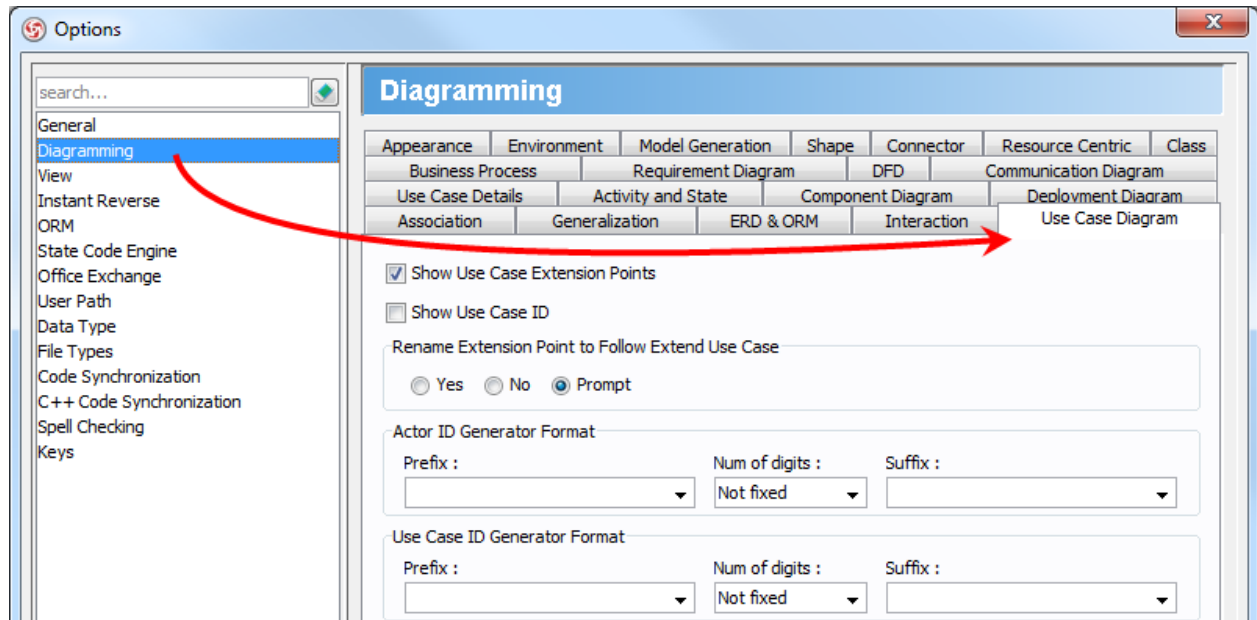
Name the package

Assigning IDs to actors/Use cases

You may assign IDs to actors and use cases. By default, IDs are assigned with the order of object creation, starting from one onwards. However, you can define the format or even enter an ID manually.

Defining the format of ID

To define the format of ID, select **Tools > Options** from the main menu to unfold the **Options** dialog box. Select **Diagramming** from the list on the left hand side and select the **Use Case Diagram** tab on the right hand side. You can adjust the format of IDs under **Use Case Diagram** tab. The format of ID consists of prefix, number of digits and suffix.



Step 13:-

Use Case Diagram tab

The description of options for ID generator format is shown below.

Option

Description

Prefix The prefix you enter in **Prefix** text field will be inserted before the number.

Num of digits The number of digits for the number. For example, when digit is 3, ID "1" will become "001".

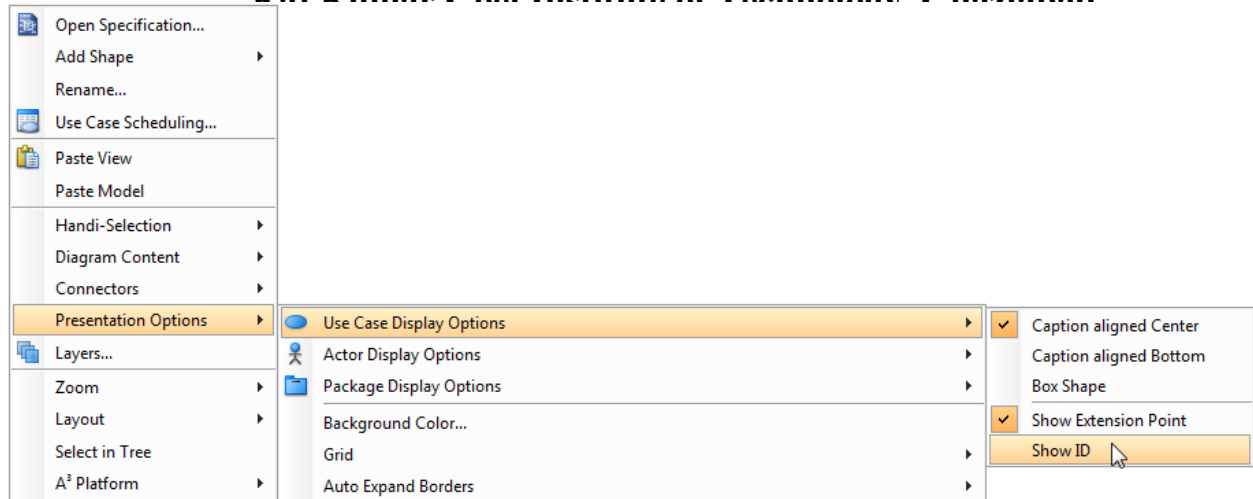
Suffix The suffix you enter in **Suffix** text field will be inserted behind the number.

Options for formatting ID

Showing ID on diagram

By default, ID is just a text property. It usually doesn't appear on diagram. However, you can make it shown within a use case.

Right click on the diagram background, select **Presentation Options** and the specific model element display option from the pop-up menu.



Step 14:-

Show ID on diagram

As a result, the use case is displayed with ID.



A use case with ID displayed

NOTE: The feature of showing ID does only support for use case, but not for actor.

ID assignment

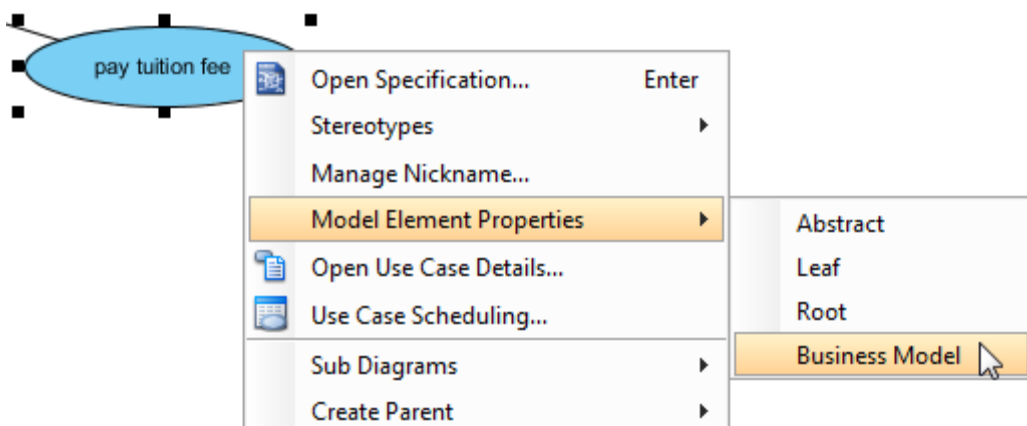
There are several ways that you can assign an ID to a model element, including:

Through the specification dialog box (Right click on the selected model element and select **Open Specification...** from the pop-up menu)

Through the **Property Pane**

Drawing business use case

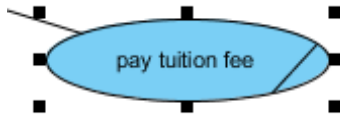
Right click on a use case and select Model Element Properties > Business Model from the pop-up menu.



Step 15:-

Raj Kumar Goel Institute of Technology, Ghaziabad
Click **Business Model**

After selected, an extra slash will be shown on the left edge of the use case.



Business model

And Finally The Use case Diagram is ready

Post-Experiment Questions:

1. Who are the actors in a UML.
2. What is boundary in a USE CASE.

Aim: Draw the activity diagram for a given problem.

Descriptions:

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deal with all types of flow control by using different elements like fork, join etc.

The basic purposes of activity diagrams are similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flow chart. Although the diagram looks like a flow chart but it is not. It shows different flows like parallel, branched, concurrent and single. So the purposes can be described as:

Draw the activity flow of a system.

Describe the sequence from one activity to another.

Describe the parallel, branched and concurrent flow of the system.

How to draw Activity Diagram?

Activity diagrams are mainly used as a flow chart consists of activities performed by the system. But activity diagrams are not exactly a flow chart as they have some additional capabilities. These additional capabilities include branching, parallel flow, swim lane etc.

Before drawing an activity diagram we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities we need to understand how they are associated with constraints and conditions. So before drawing an

activity diagram we should identify the following elements:

Activities

Association

Conditions

Constraints

Once the above mentioned parameters are identified we need to make a mental layout of the entire flow. This mental layout is then transformed into an activity diagram.

Problem: - Example

The following is an example of an activity diagram for order management system. In the diagram four activities are identified which are associated with conditions. One important point should be clearly understood that an activity diagram cannot be exactly matched with the code. The activity diagram is made to understand the flow of activities and mainly used by the business users. The following diagram is drawn with the four main activities:

Send order by the customer

Receipt of the order

Confirm order

Dispatch order

Activity diagram

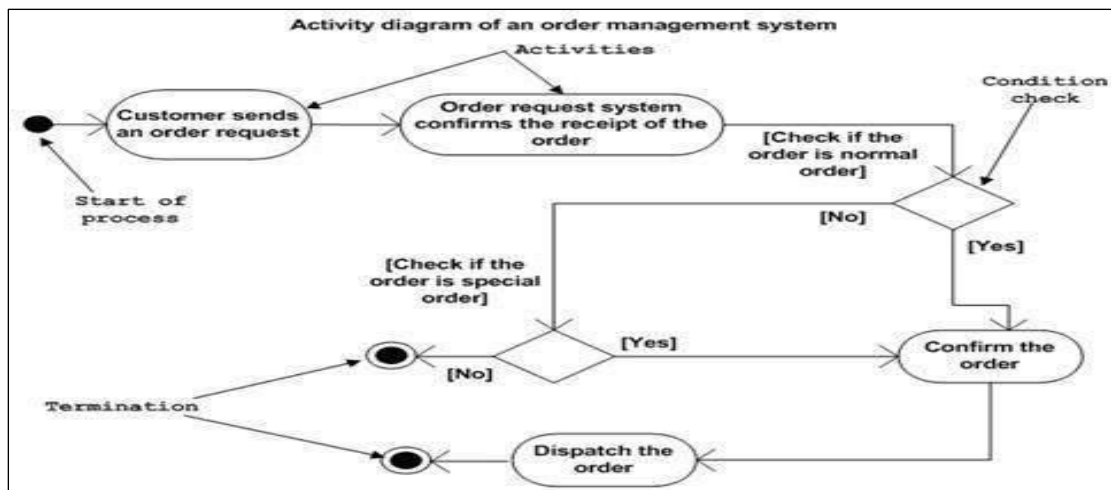


Fig.: activity diagram for order management system

Result & Conclusion

After receiving the order request condition checks are performed to check if it is normal or special order. After the type of order is identified dispatch activity is performed and that is marked as the termination of the process.

EXPERIMENT - 4

Aim: Identify the classes. Classify them as weak and strong classes and draw the class diagram for a given problem.

Descriptions:

Class diagram: describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes.

Pre-Experiment Questions:

Q1. What is static view of an application?

Q2. What is CLASS Diagram?

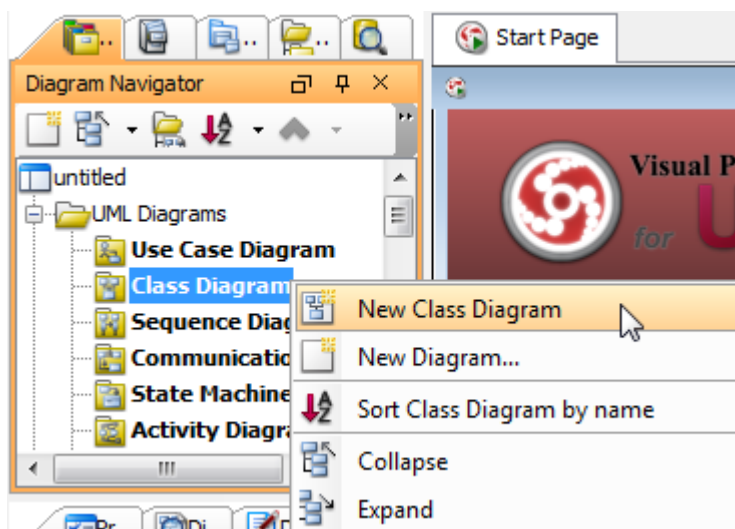
Software Required:-

Visual Paradigm for UML 8.2

Procedure:-

Step 1:-

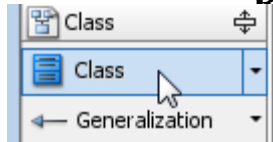
Right click **Class Diagram** on **Diagram Navigator** and select **New Class Diagram** from the pop-up menu to create a class diagram.



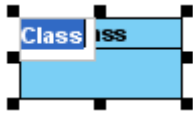
Step 2:-

Creating class

To create class, click **Class** on the diagram toolbar and then click on the diagram.

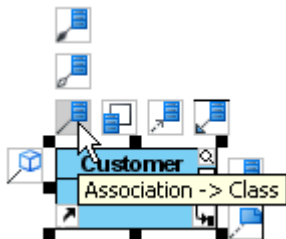


A class will be created.



Creating association

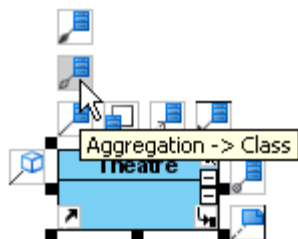
To create association from class, click the Association -> Class resource beside it and drag.



Drag to empty space of the diagram to create a new class, or drag to an existing class to connect to it. Release the mouse button to create the association.

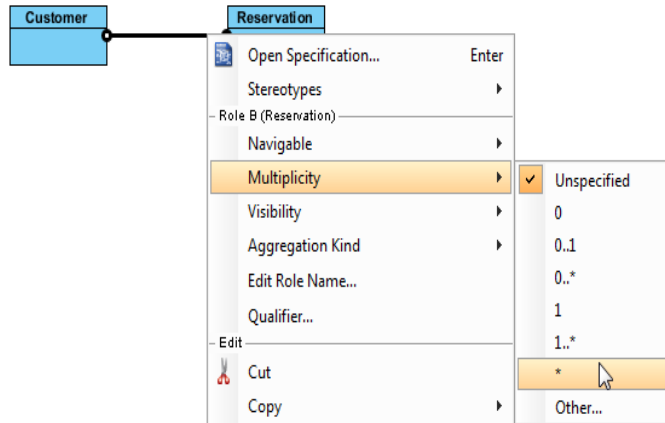


To create aggregation, use the **Aggregation -> Class** resource instead.

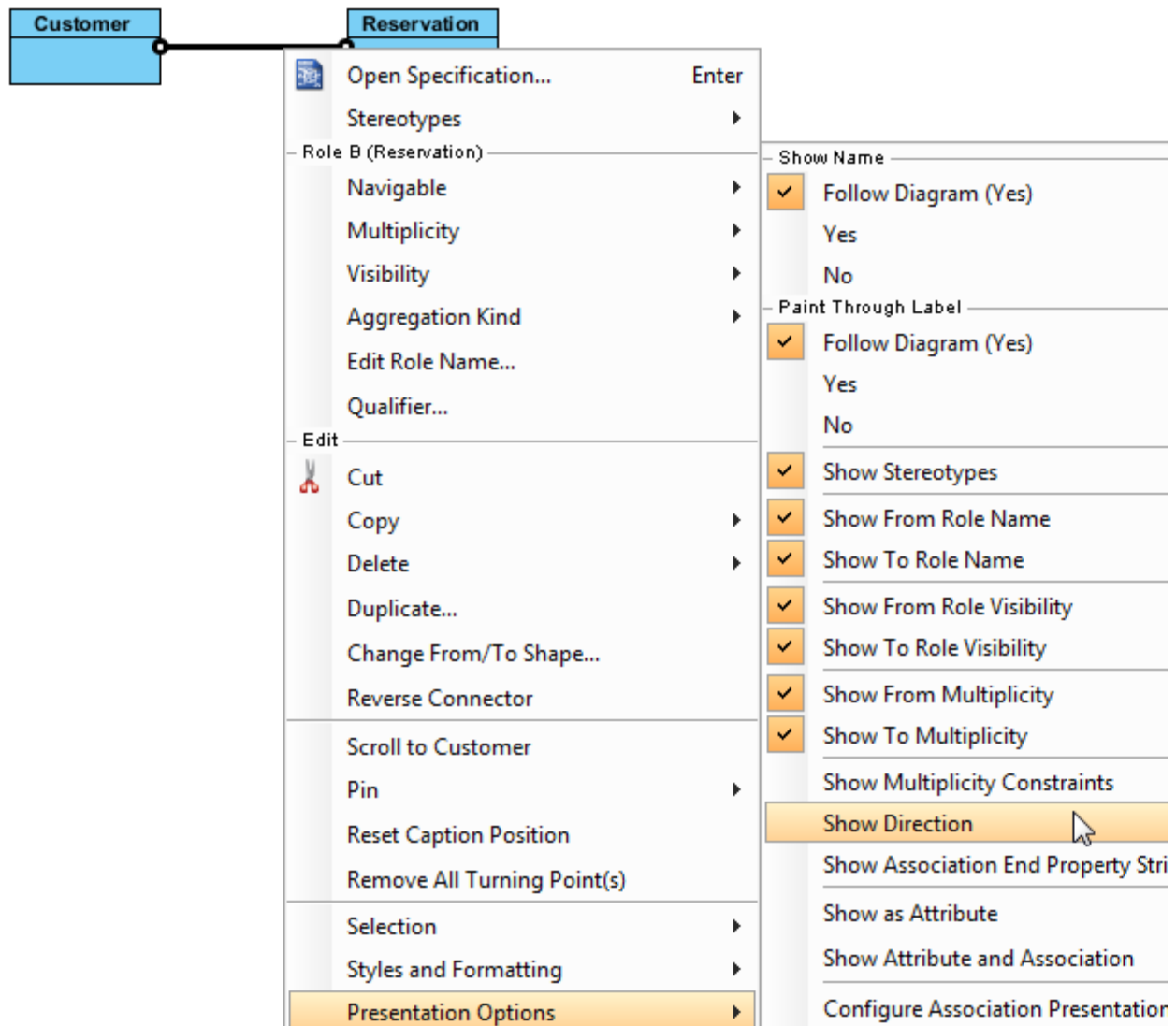


Step 3:-

To edit multiplicity of an association end, right-click near the association end, select Multiplicity from the popup menu and then select a multiplicity.



To show the direction of an association, right click on it and select Presentation Options > Show Direction from the pop-up menu.



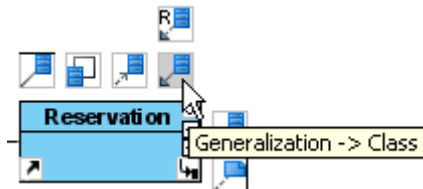
Step 4:-

The direction arrow is shown beside the association.

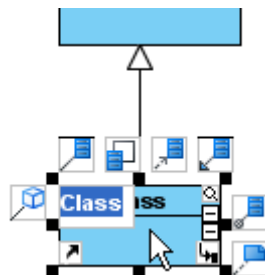


Creating generalization

To create generalization from class, click the Generalization -> Class resource beside it and drag.

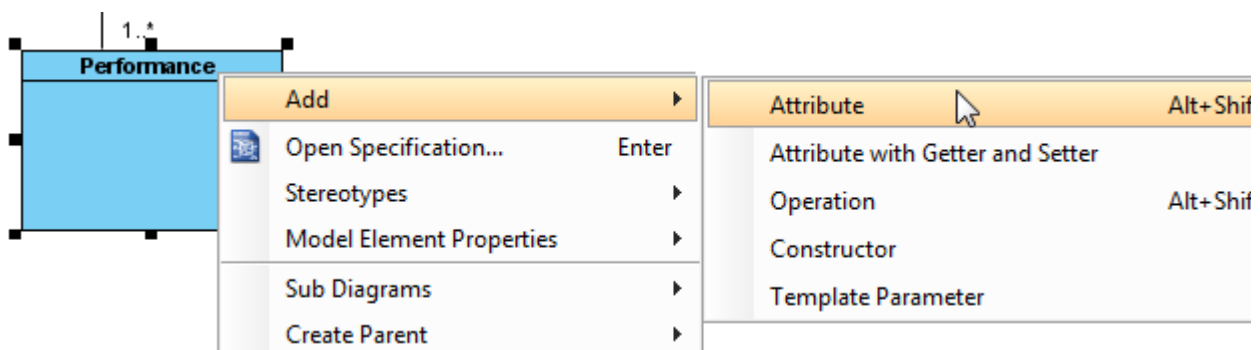


Drag to empty space of the diagram to create a new class, or drag to an existing class to connect to it. Release the mouse button to create the generalization.

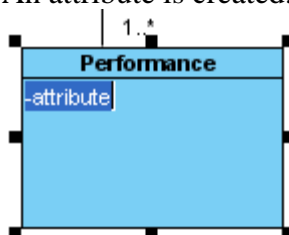


Creating attribute

To create attribute, right click the class and select Add > Attribute from the pop-up menu.



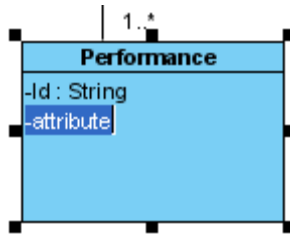
An attribute is created.



Creating attribute with enter key

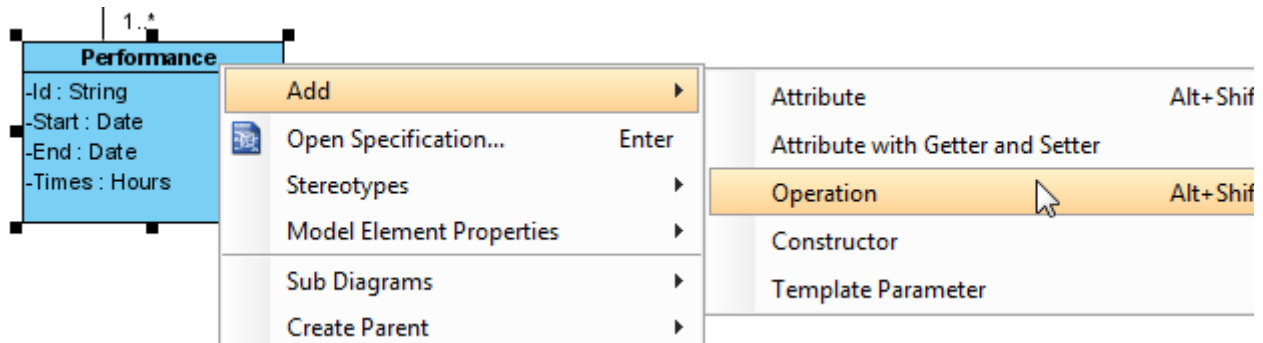
After creating an attribute, press the Enter key, another attribute will be created. This method

lets you create multiple attributes quickly and easily.

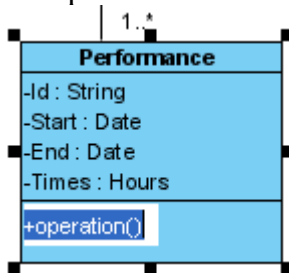


Creating operation

To create operation, right click the class and select Add > Operation from the pop-up menu.



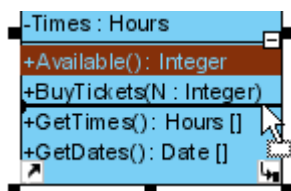
An operation is created.



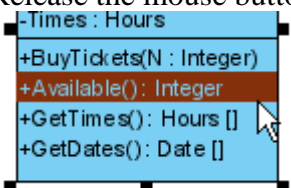
Similar to creating attribute, you can press the Enter key to create multiple operations continuously.

Drag-and-Drop reordering, copying and moving of class members

To reorder a class member, select it and drag within the compartment, you will see a thick black line appears indicating where the class member will be placed.

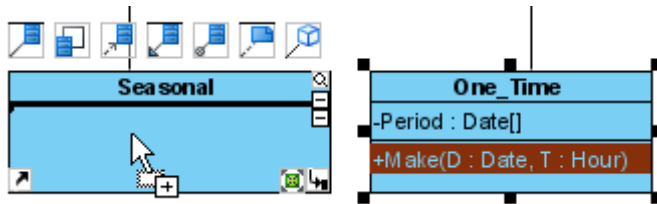


Release the mouse button, the class member will be reordered.

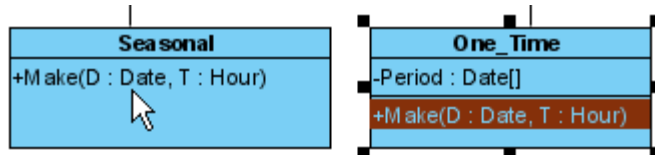


To copy a class member, select it and drag to the target class while keep pressing the Ctrl key,

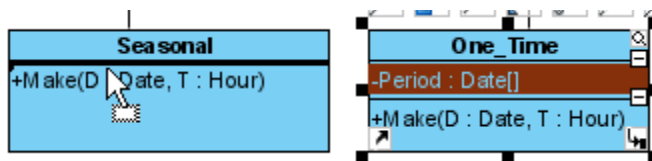
you will see a thick black line appears indicating where the class member will be placed. A plus sign is shown beside the mouse cursor indicating this is a copy action.



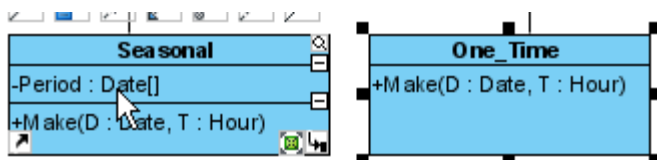
Release the mouse button, the class member will be copied.



To move a class member, select it and drag to the target class, you will see a thick black line appears indicating where the class member will be placed. Unlike copy, do not press the Ctrl key when drag, the mouse cursor without the plus sign indicates this is a move action.

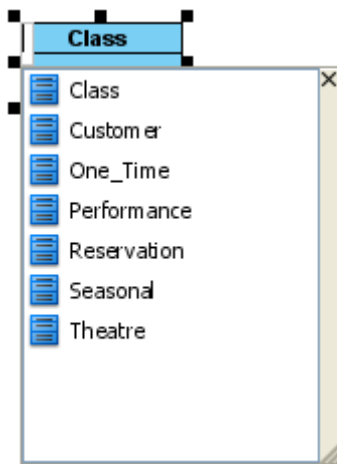


Release the mouse button, the class member will be moved.

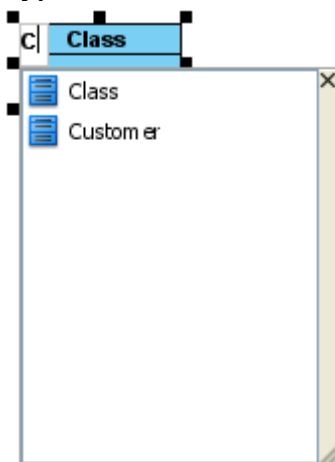


Model name completion for class

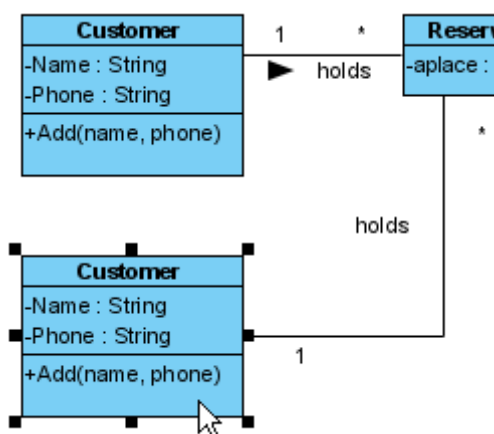
The model name completion feature enables quick creation of multiple views for the same class model. When create or rename class, the list of classes is shown.



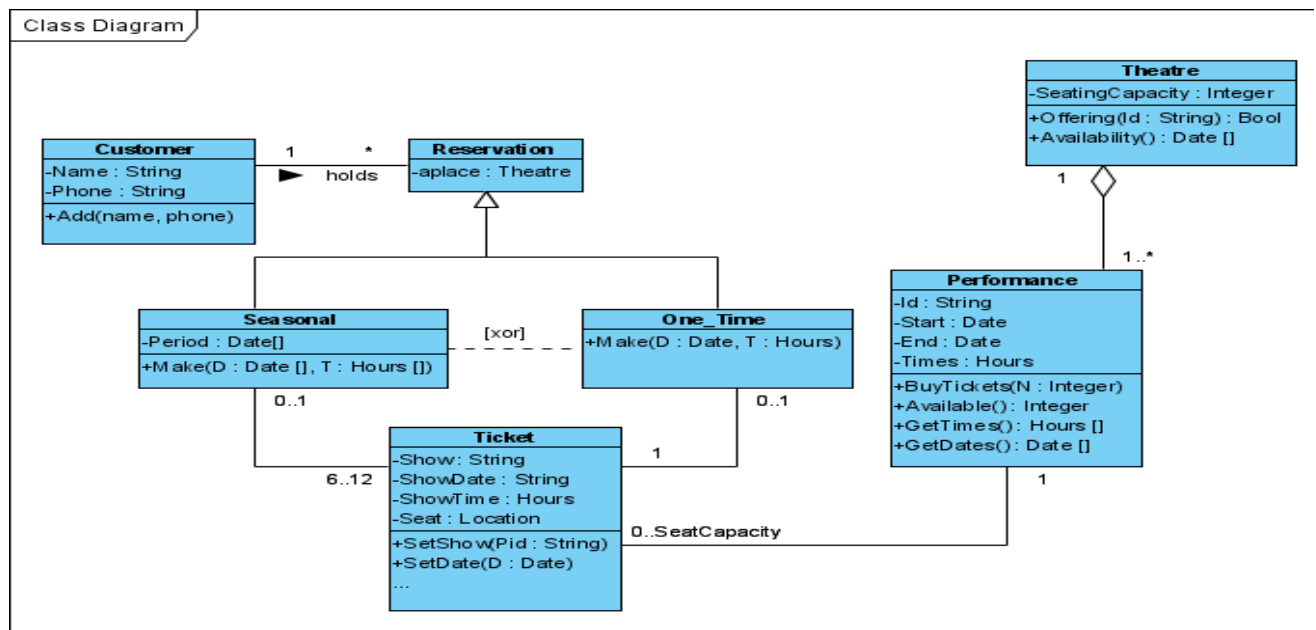
Type text to filter classes in the list.



Press up or down key to select class in the list, press Enter to confirm. Upon selecting an existing class, all class members and relationships are shown immediately.

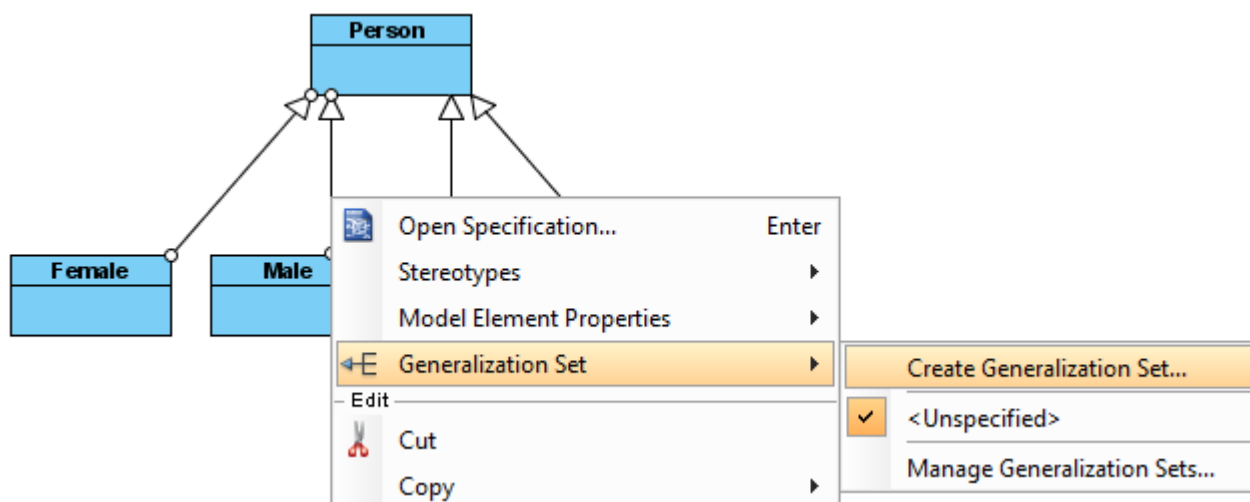


Step 5:-



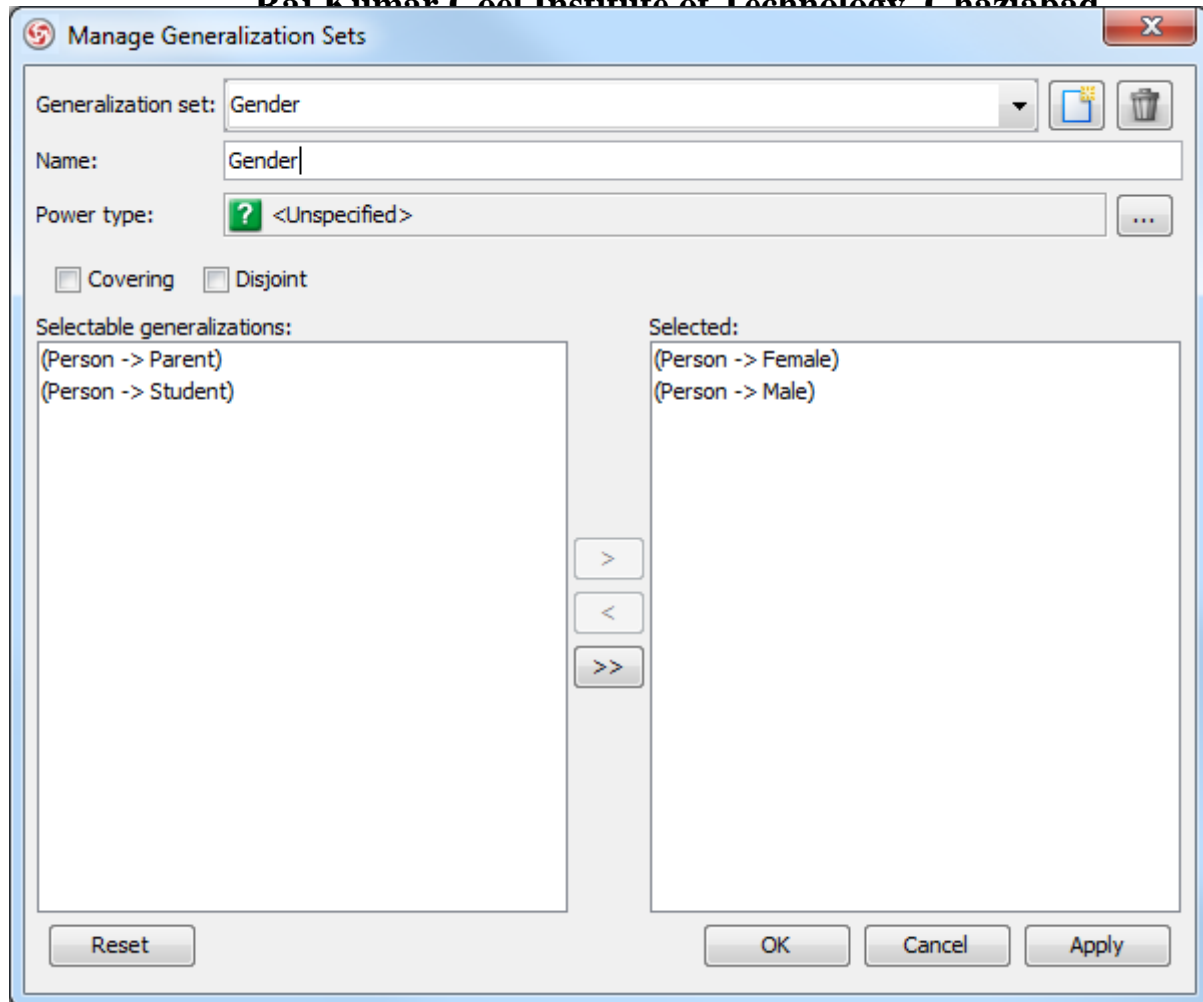
Generalization set

A generalization set defines a particular set of generalization relationships that describe the way in which a general classifier (or superclass) may be divided using specific subtypes. To define a generalization set, select the generalizations to include, right click and select Generalization set > Create Generalization Set... from the popup menu.

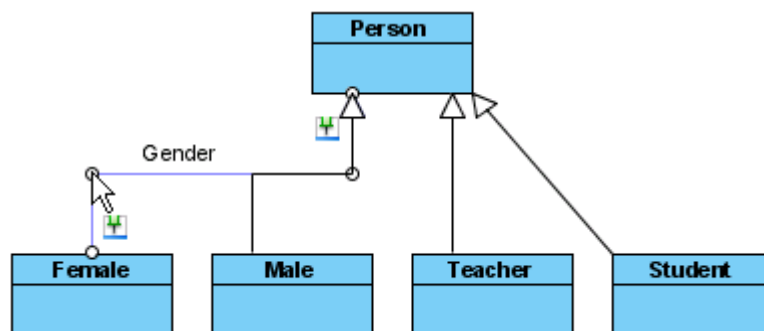


Step 6:-

Name the set in the Manage Generalization Sets dialog box, and confirm by pressing OK.

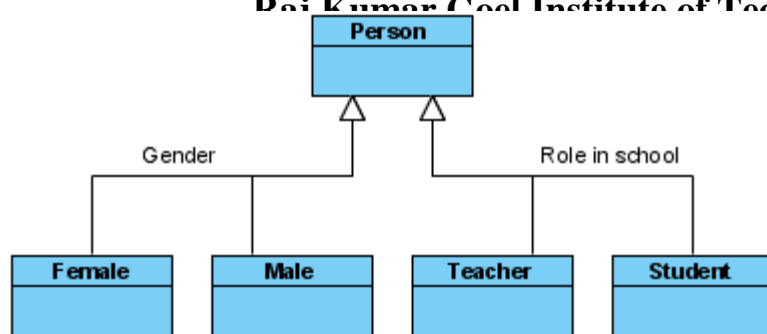


The selected generalizations are grouped. Adjust the connector to make the diagram tidy.



Repeat the steps for other generalizations.

Dai Kumar Cool Institute of Technology, Ghaziabad



Aim: Draw the sequence diagram for any two scenarios for a given problem.

Software Required:-

Visual Paradigm for UML 8.2

Pre-Experiment Question:

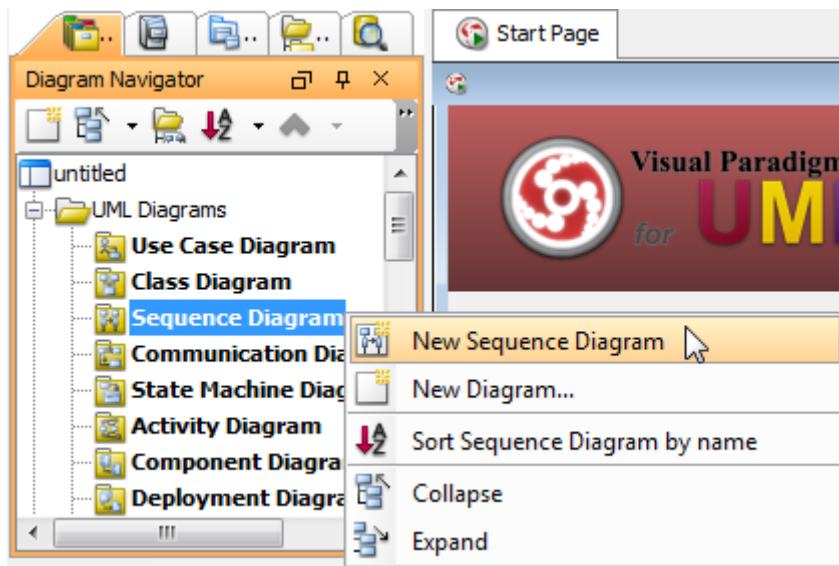
1. What is sequence diagram?
2. Why we need this?

Procedure :-

A sequence diagram is used primarily to show the interactions between objects that are represented as lifelines in a sequential order.

Step 1:-

Right click **Sequence diagram** on **Diagram Navigator** and select **New Sequence Diagram** from the pop-up menu to create a sequence diagram.

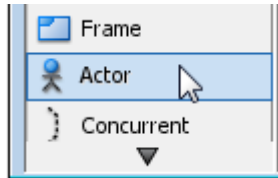


Step 2:-

Enter name for the newly created sequence diagram in the text field of pop-up box on the top left corner.

Creating actor

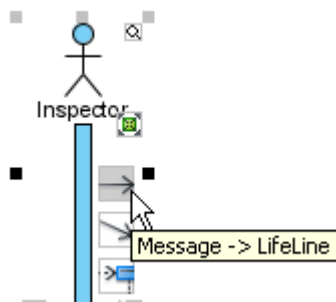
To create actor, click **Actor** on the diagram toolbar and then click on the diagram.



Creating lifeline

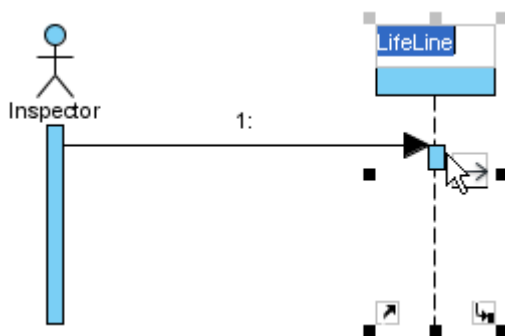
To create lifeline, you can click LifeLine on the diagram toolbar and then click on the diagram.

Alternatively, a much quicker and more efficient way is to use the resource-centric interface. Click on the Message -> LifeLine resource beside an actor/lifeline and drag.



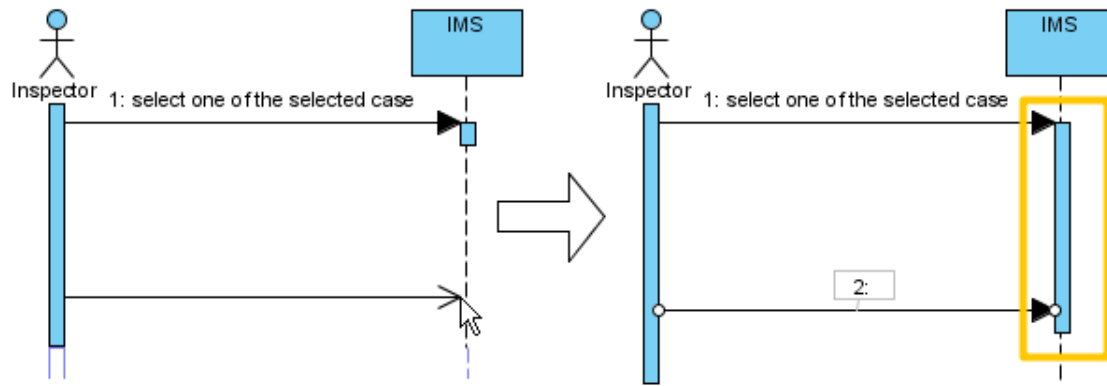
Step 3:-

Move the mouse to empty space of the diagram and then release the mouse button. A new lifeline will be created and connected to the actor/lifeline with a message.



Auto extending activation

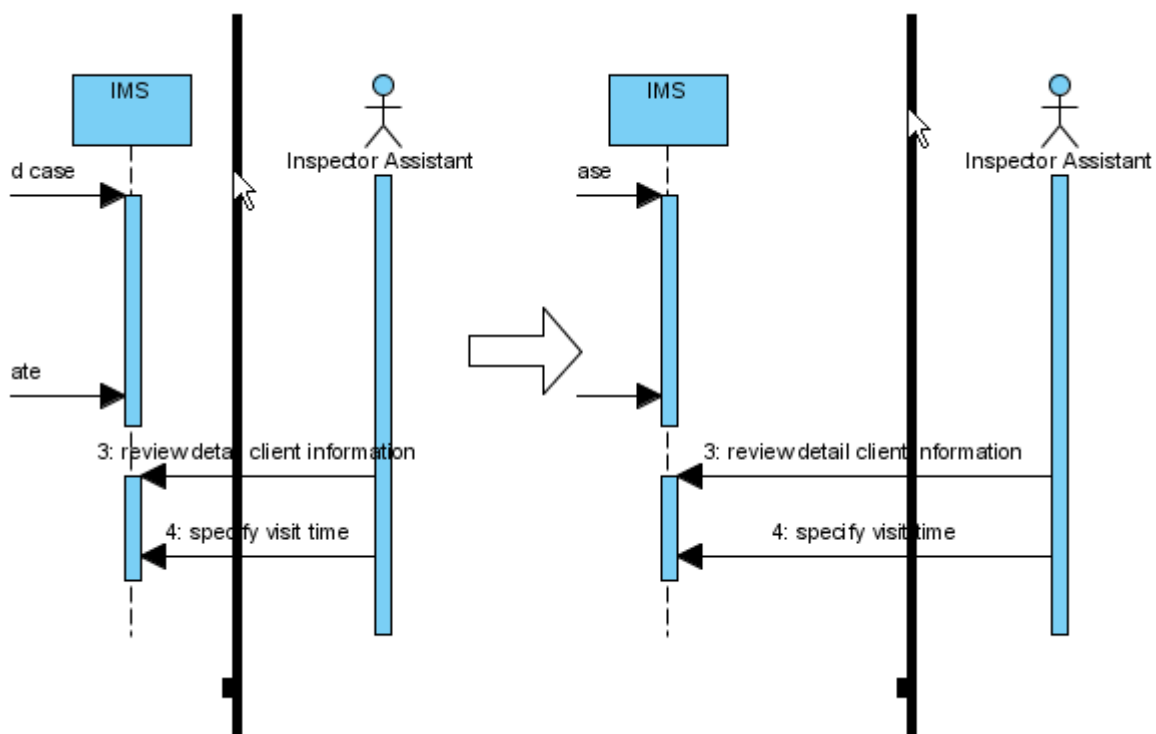
When create message between lifelines/actors, activation will be automatically extended.



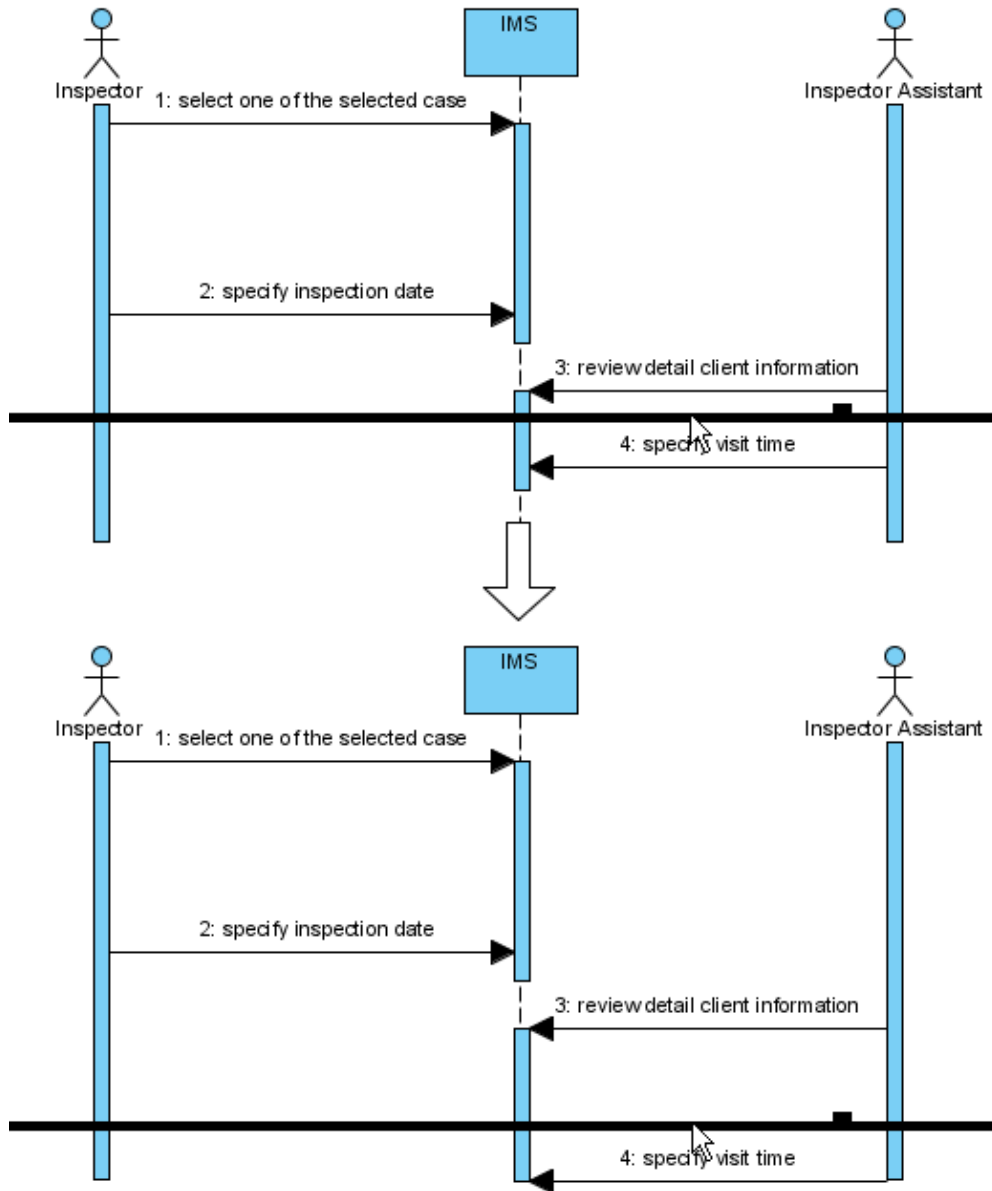
Step 4:-

Using sweeper and magnet to manage sequence diagram

Sweeper helps you to move shapes aside to make room for new shapes or connectors. To use sweeper, click **Sweeper** on the diagram toolbar (under the **Tools** category).

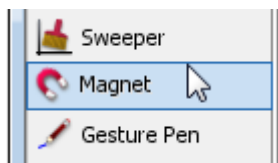


The picture below shows the message *specify visit time* is being swept downwards, thus new room is made for new messages



Step 5:-

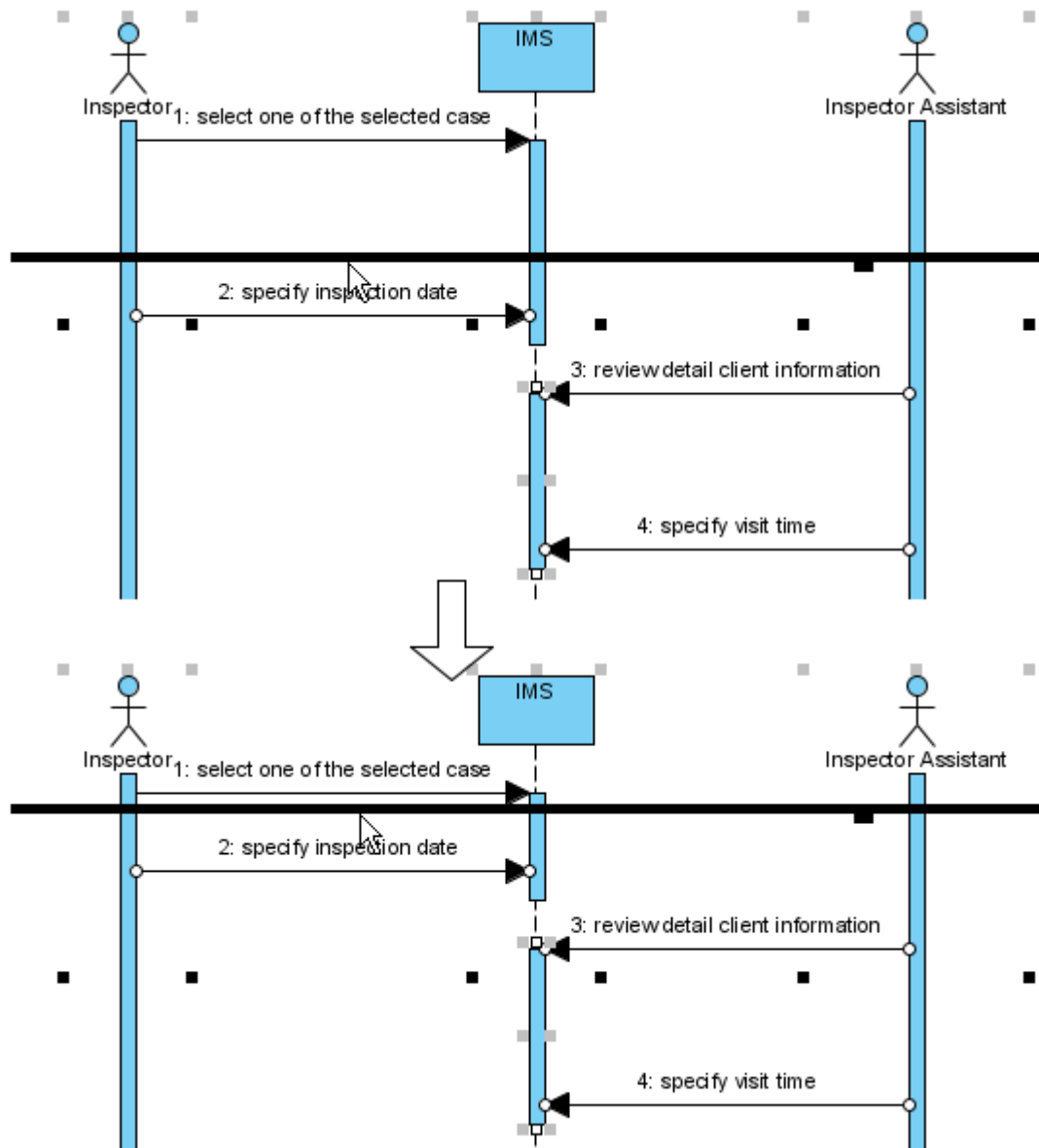
You can also use magnet to pull shapes together. To use magnet, click Magnet on the diagram toolbar (under the Tools category).



Magnet

Click on empty space of the diagram and drag towards top, right, bottom or left. Shapes affected will be pulled to the direction you dragged.

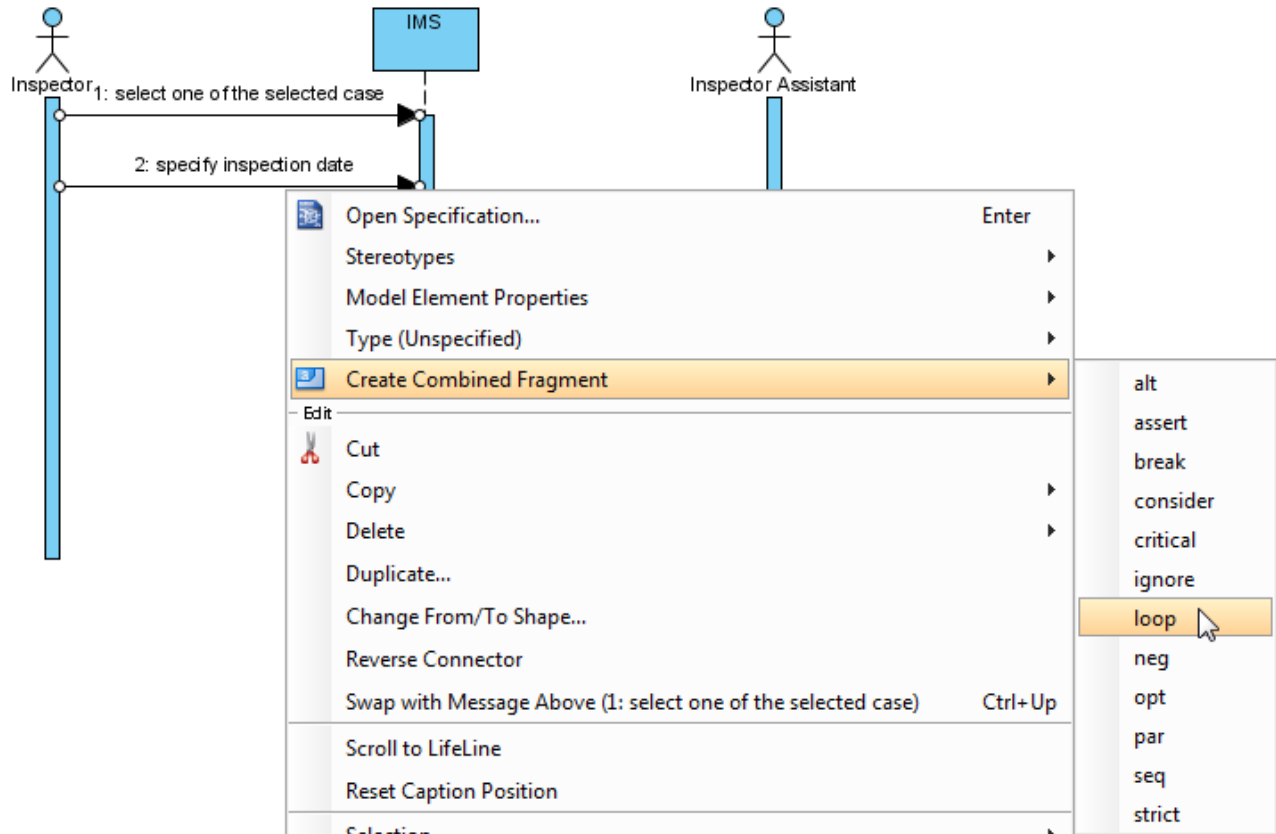
The picture below shows when drag the magnet upwards, shapes below dragged position are pulled upwards.



Step 6:-

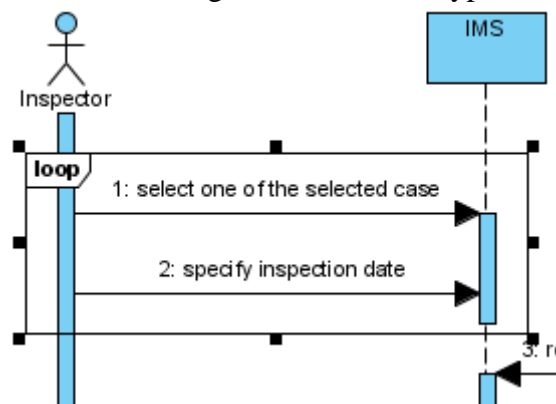
Creating combined fragment for messages

To create combined fragment to cover messages, select the messages, right-click on the selection and select **Create Combined Fragment**, and then select a combined fragment type (e.g. loop) from the popup menu.



Step 7:-

A combined fragment of selected type will be created to cover the messages.

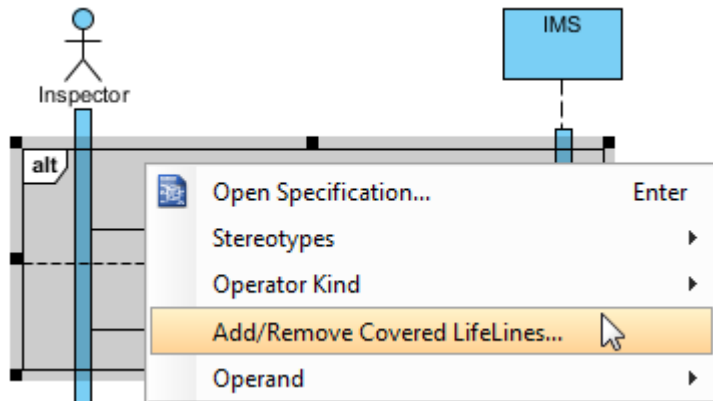


Step 8:-

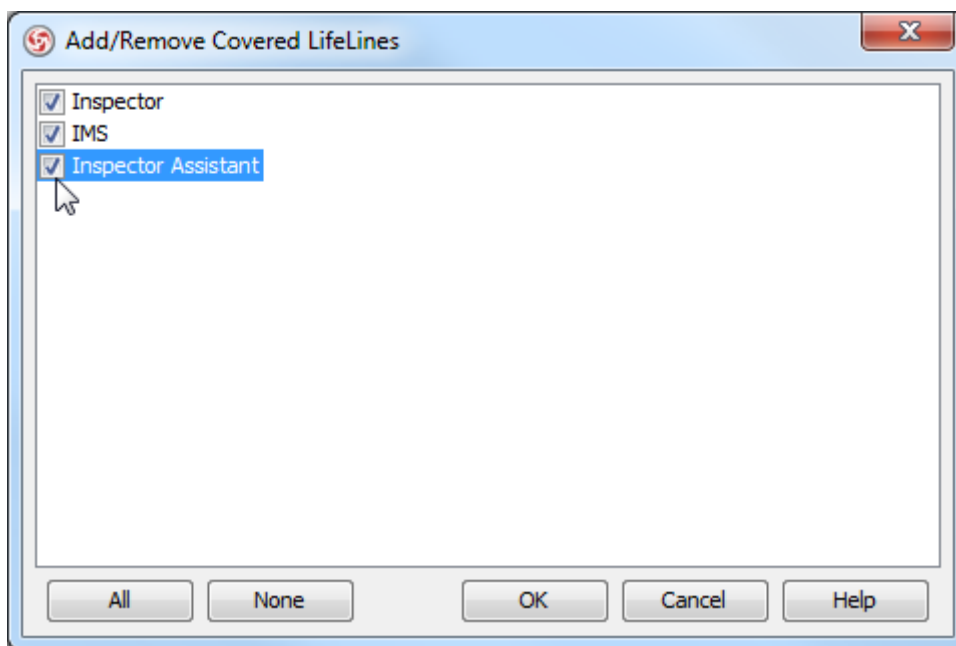
Adding/removing covered lifelines

1. After you've created a combined fragment on the messages, you can add or remove the covered lifelines.

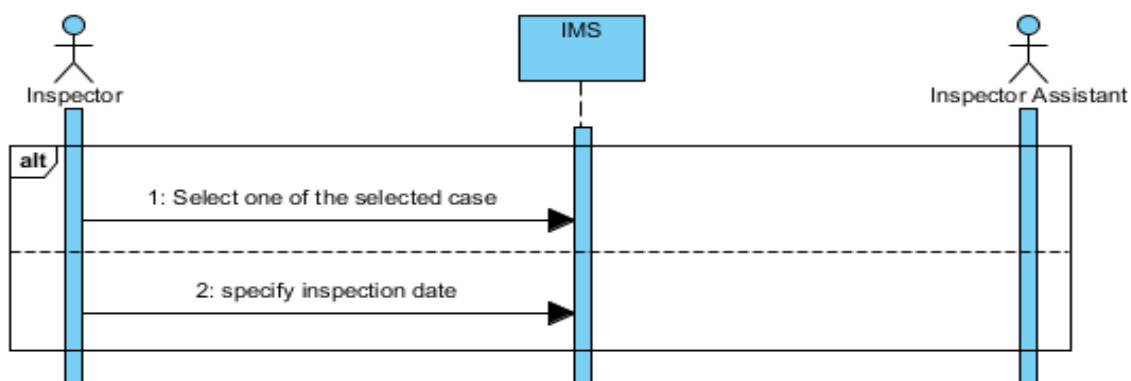
Move the mouse over the combined fragment and select Add/Remove Covered Lifeline... from the pop-up menu.



2. In the Add/Remove Covered Lifelines dialog box, check the lifeline(s) you want to cover or uncheck the lifeline(s) you don't want to cover. Click OK button.



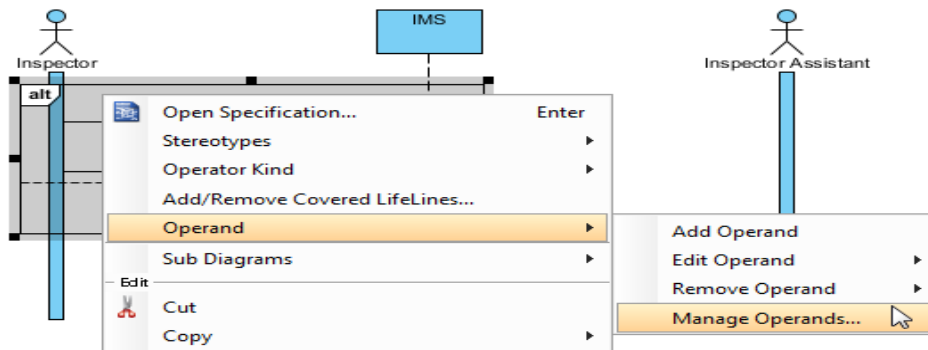
As a result, the area of covered lifelines is extended or narrowed down according to your selection.



Managing Operands

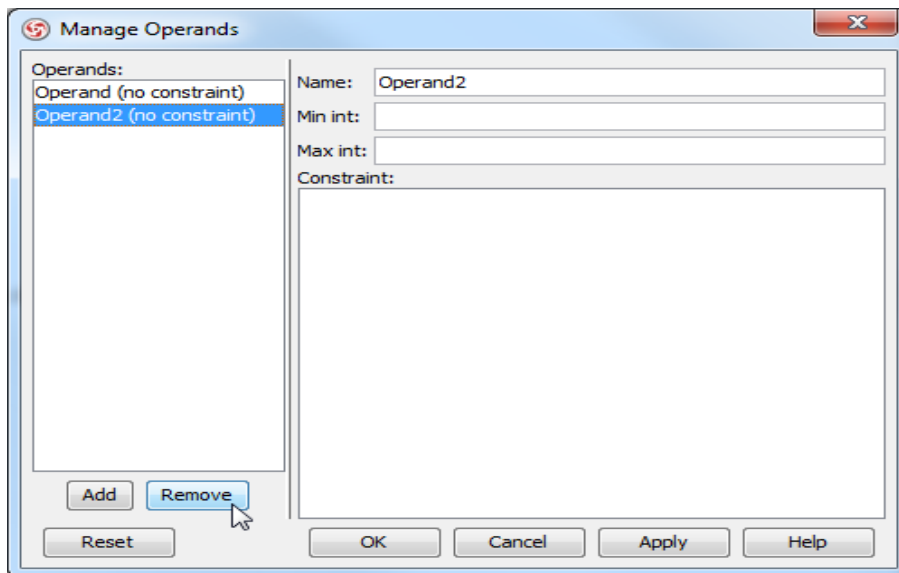
After you've created a combined fragment on the messages, you can also add or remove operand(s).

Move the mouse over the combined fragment and select **Operand > Manage Operands...** from the pop-up menu.



Step 9:-

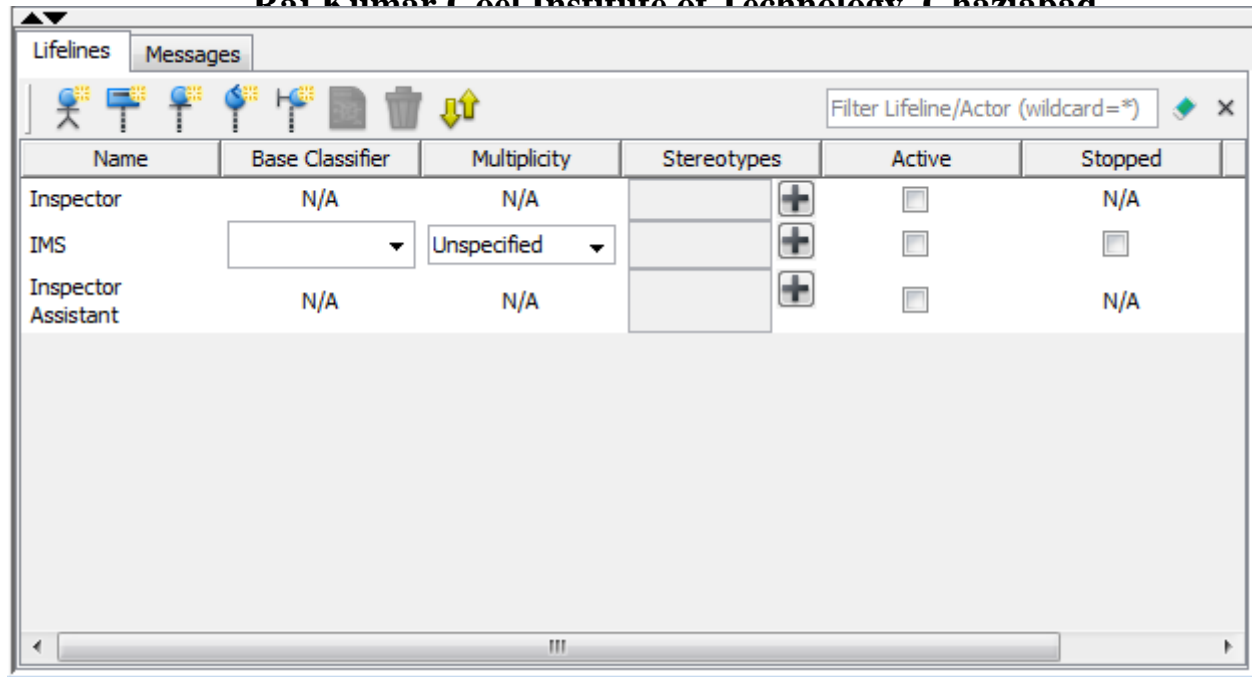
1. To remove an operand, select the target operand from **Operands** and click **Remove** button. Click **OK** button.



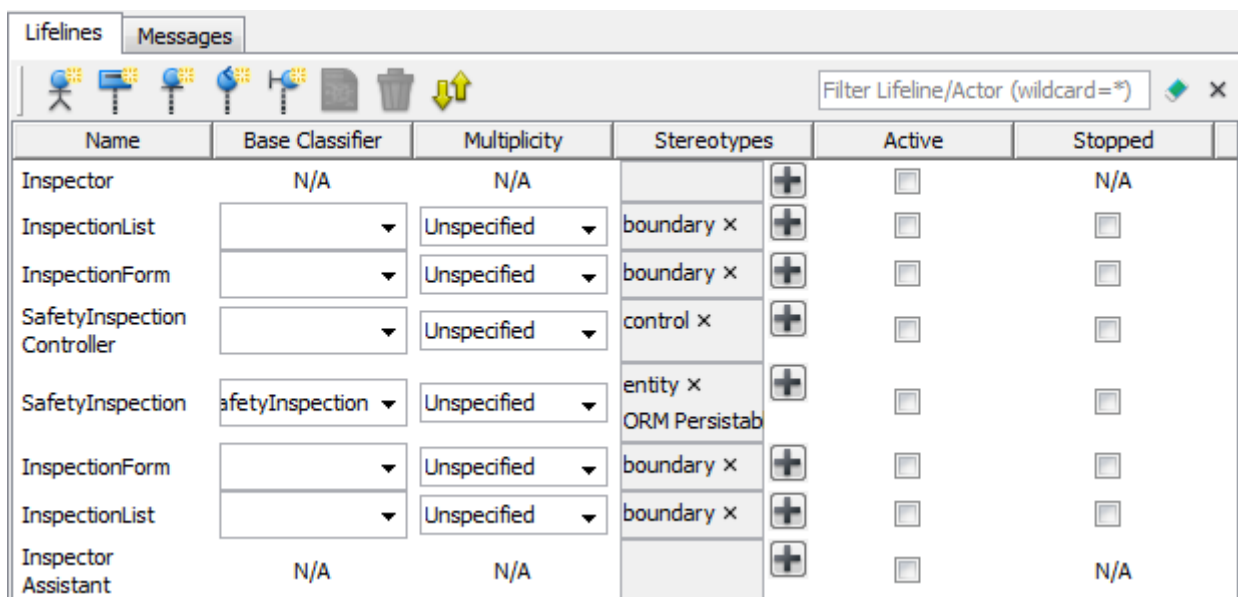
2. Otherwise, click **Add** button to add a new operand and then name it. Click **OK** button.

Developing sequence diagram with quick editor or keyboard shortcuts

In sequence diagram, an editor appears at the bottom of diagram by default, which enables you to construct sequence diagram with the buttons there. The shortcut keys assigned to the buttons provide a way to construct diagram through keyboard. Besides constructing diagram, you can also access diagram elements listing in the editor.



There are two panes, **Lifelines** and **Messages**. The **Lifelines** pane enables you to create different kinds of actors and lifelines.

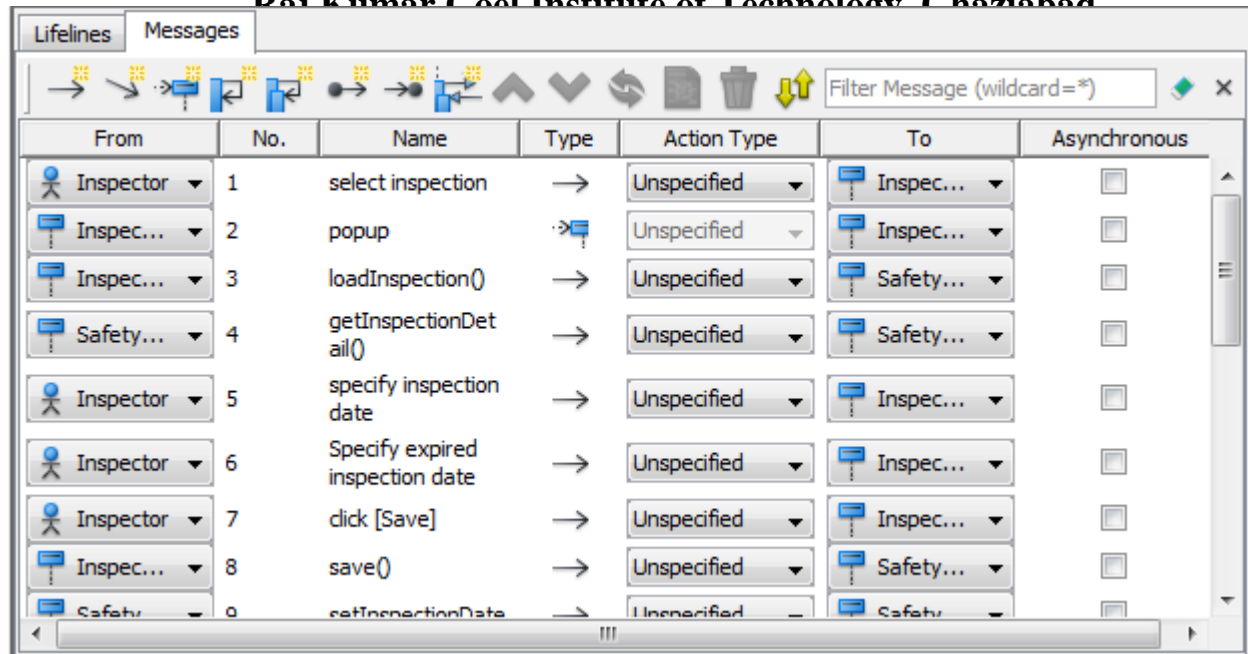


Step 10:-

Buttons in Lifelines pane

Editing messages















The **Messages** pane enables you to connect lifelines with various kinds of messages.



Messages pane in quick editor

Button

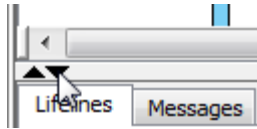
Short
cut Description

-  Alt-Shift-M To create a message that connects actors/lifelines in diagram
-  Alt-Shift-D To create a duration message that connects actors/lifelines in diagram
-  Alt-Shift-C To create a create message that connects actors/lifelines in diagram
-  Alt-Shift-S To create a self message on an actor/lifeline in diagram
-  Alt-Shift-R To create a recursive message on an actor/lifeline in diagram
-  Alt-Shift-F To create a found message that connects to an actor/lifeline
-  Alt-Shift-L To create a lost message from an actor/lifeline
-  Alt-Shift-E To create a reentrant message that connects actors/lifelines in diagram
-  Ctrl-Shift-Up To swap the chosen message with the one above
-  Ctrl-Shift-Down To swap the chosen message with the one below
-  Ctrl-R To revert the direction of chosen message
-  Alt-Shift-O To open the specification of the message chosen in quick editor
-  Ctrl-Del To delete the message chosen in quick editor
-  Ctrl-L To link with the diagram, which cause the message to be selected when selecting a message in editor, and vice versa

Buttons in Messages pane

To hide the editor, click on the down arrow button that appears at the bar on top of the quick editor.

To expand, click on the up arrow button.



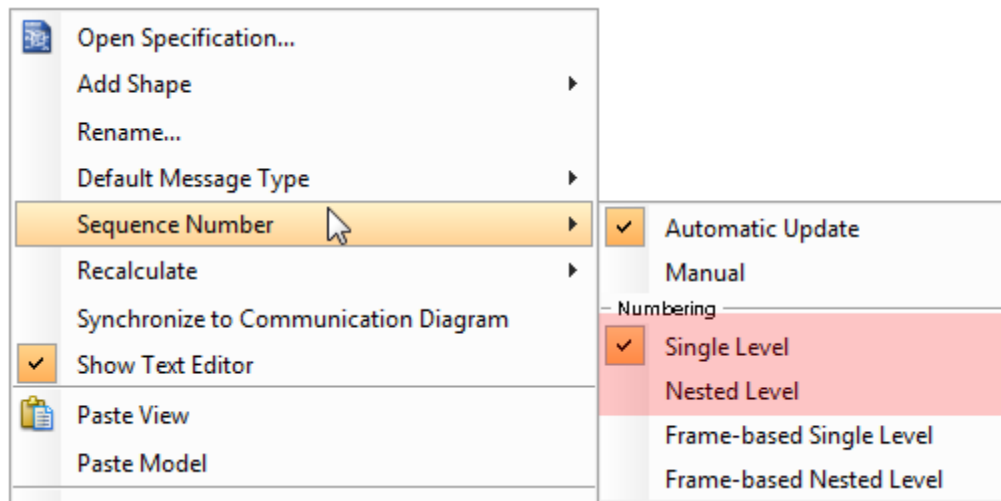
Collapse the quick editor

Setting different ways of numbering sequence messages

You are able to set the way of numbering sequence messages either on diagram base or frame base.

Diagram-based sequence message

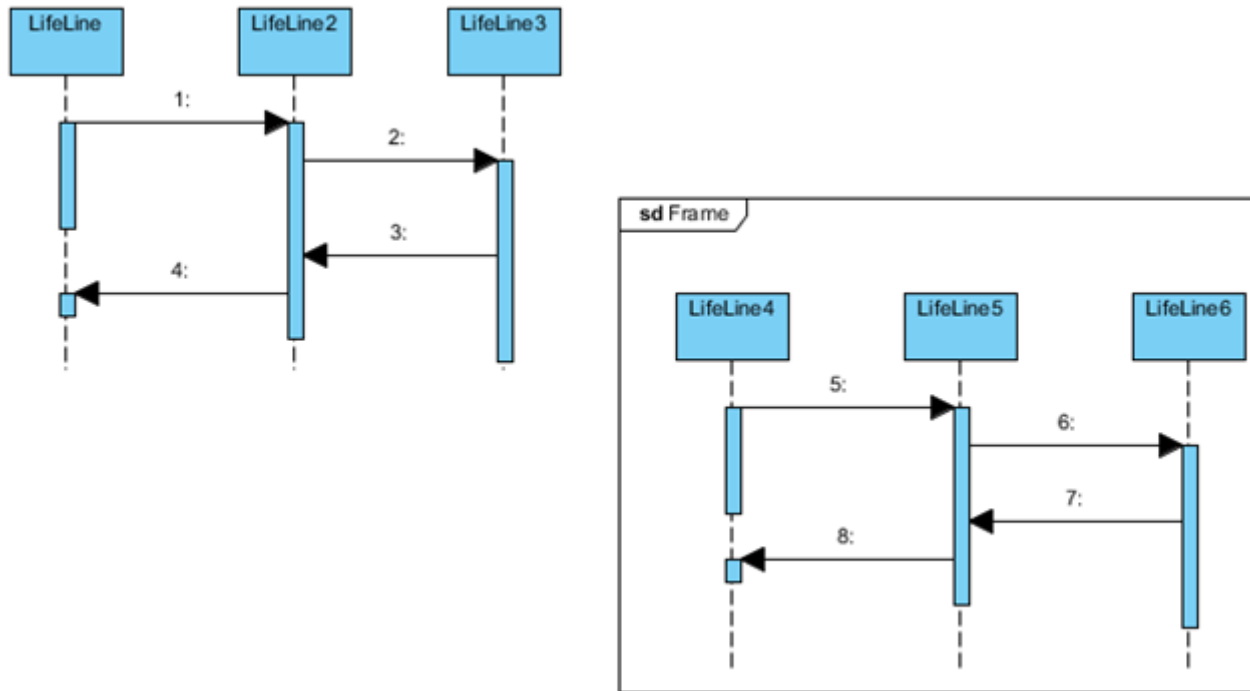
Right click on the diagram's background, select **Sequence Number** and then either **Single Level** or **Nested Level** from the pop-up menu.



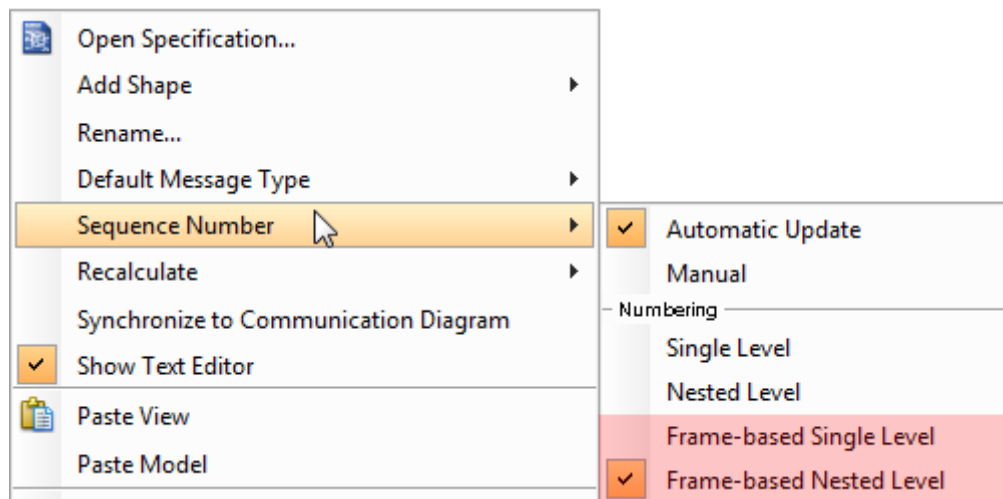
Step 11:-

If you choose **Single Level**, all sequence messages will be ordered with integers on diagram base.

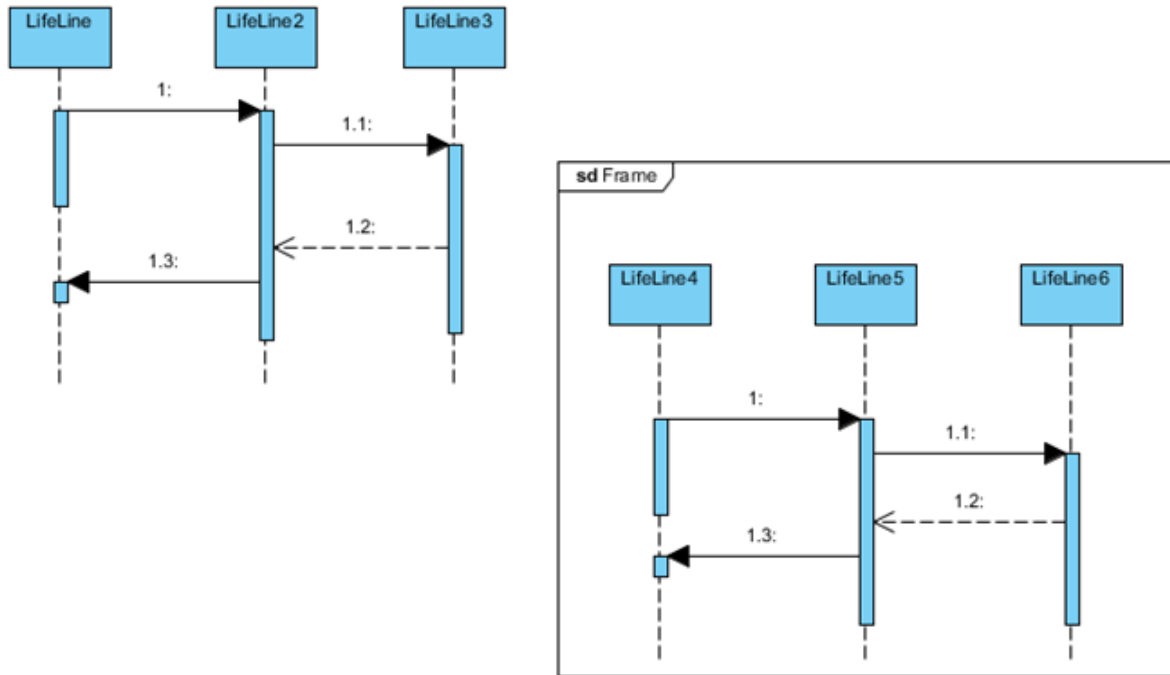
On the other hand, if you choose **Nested Level**, all sequence messages will be ordered with decimal place on diagram base.



Right click on the diagram's background; select **Sequence Number** and then either **Frame-based Single Level** or **Frame-based Nested Level** from the pop-up menu.



When you set the way of numbering sequence messages on frame base, the sequence messages in frame will restart numbering sequence message since they are independent and ignore the way of numbering sequence message outside the frame.



Post-Experiment Questions:

1. Draw the Sequence diagram of College Automation System.
2. What is the need of sequence diagram in a project?
3. What is the difference between nested level and single level sequence?
4. Draw the Sequence diagram of Banking Management system

Aim: Draw the collaboration diagram for a given problem.

Description:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult. A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Pre-Experiment Questions:

Q1. Explain collaboration diagram?

Q2. How method call is to be done in collaboration Diagram?

Procedure:

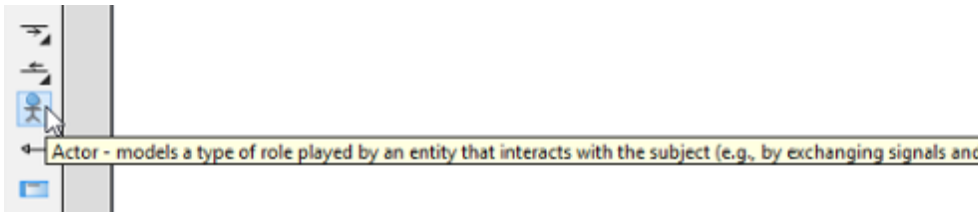
A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). Collaboration diagrams are best suited to the portrayal of simple interactions among relatively small numbers of objects. As the number of objects and messages grows, a collaboration diagram can become difficult. A collaboration diagram resembles a flowchart that portrays the roles, functionality and behaviour of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects is shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

Perform the steps below to create a UML communication diagram in Visual Paradigm.

1. Select **Diagram** > **New** from the application toolbar.
2. In the **New Diagram** window, select **Communication Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Step 1: Creating actor

To create an actor, click **Actor** on the diagram toolbar and then click on the diagram.



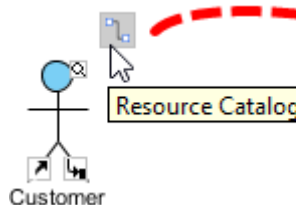
Create actor

Step 2: Creating lifeline

To create lifeline, you can click **LifeLine** on the diagram toolbar and then click on the diagram.

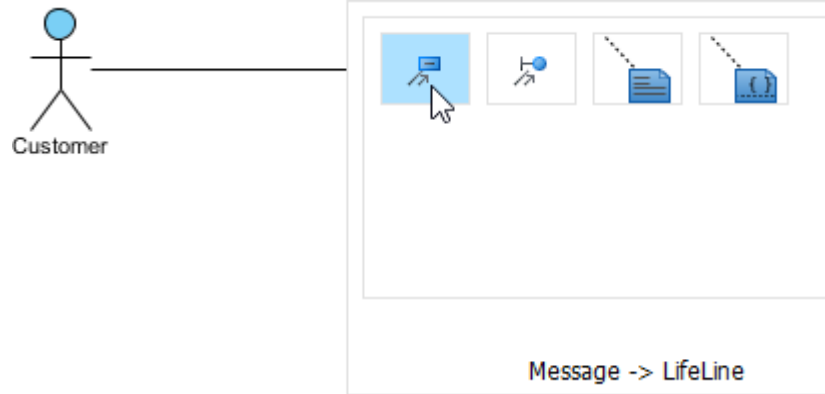
Alternatively, a much quicker and more efficient way is to use Resource Catalog:

1. Move your mouse pointer over the source lifeline.
2. Press on the **Resource Catalog** button and drag it out.



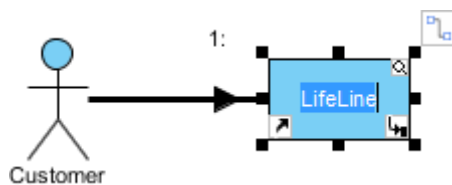
Using Resource Catalog

3. Release the mouse button at the place where you want the lifeline to be created.
4. Select **Message** -> **LifeLine** from Resource Catalog.



To create a lifeline

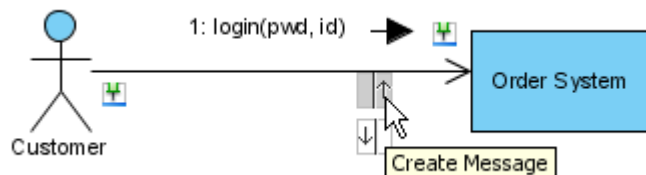
5. A new lifeline will be created and connected to the actor/lifeline with a message. Enter its name and press **Enter** to confirm editing.



Lifeline created

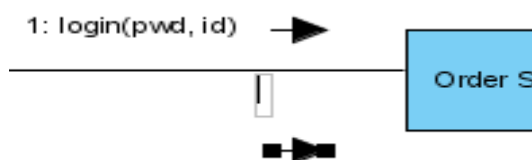
Step 3: Creating message on link

To create message on link, click its **Create Message** resource.



Create message on link

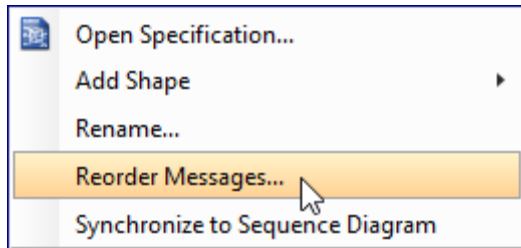
A message will be created on the link.



Message created on link

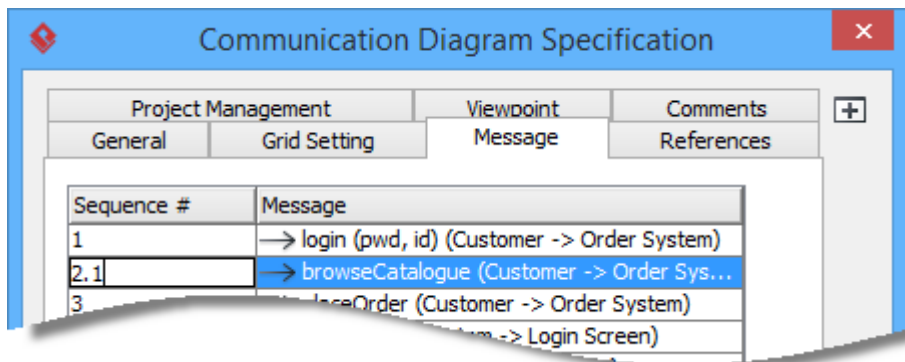
Step 4: Editing sequence number of messages

To edit sequence number of messages, for example, to show certain messages are in nested level of interaction, right-click the diagram and select **Reorder Messages...** from the pop-up menu.



Reorder messages

When the collaboration diagram **Specification** window appears, the **Message** tab is opened by default. Double click on the **Sequence #** cell of a message to edit it. Click **OK** button to apply the changes.



Edit sequence number of messages

Post-Experiment Questions:

- Q1. Can you generate a collaboration diagram using sequence diagram in UML?
- Q2. How to represent setting a variable's and attribute's value to a specified value?

EXPERIMENT - 7

Aim: Draw the state chart diagram for a given problem.

Description:

A state diagram, also called a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modelling Language (UML). In this context, a state defines a stage in the evolution or behavior of an object, which is a specific entity in a program or the unit of code representing that entity. State diagrams are useful in all forms of object-oriented programming (OOP).

State-chart diagram model elements

The common model elements that state chart diagrams contain are:

- States
- Start and end states
- Transitions
- Entry, do, and exit actions

A state represents a condition during the life of an object during which it satisfies some condition or waits for some event. Start and end states represent the beginning or ending of a process. A state transition is a relationship between two states that indicates when an object can move the focus of control on to another state once certain conditions are met. In a statechart diagram, a transition to self-element is similar to a state transition. However, it does not move the focus of control. A state transition contains the same source and target state.

Pre-Experiment Questions:

Q1. What is difference between actions and states?

Q2. What do you mean by transition in state chart diagram?

Procedure:

Step 1: Creating state machine diagram

Perform the steps below to create a UML state machine diagram in Visual Paradigm.

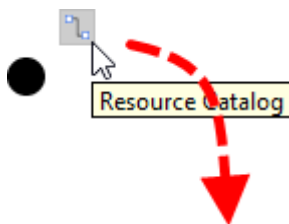
1. Select **Diagram > New** from the application toolbar.
2. In the **New Diagram** window, select **State Machine Diagram**.

3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Step 2: Creating states and transitions

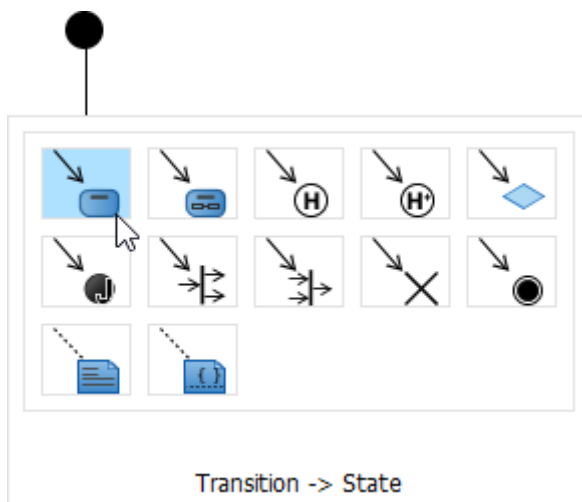
After creating a state machine diagram, an initial pseudo state appears by default. You can create other states by using Resource Catalog:

1. Move your mouse pointer over the source state.
2. Press on the **Resource Catalog** button and drag it out.



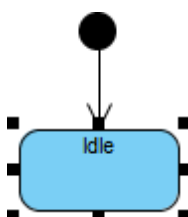
Using Resource Catalog

3. Release the mouse button at the place where you want the state to be created.
4. Select the state to be created from Resource Catalog.



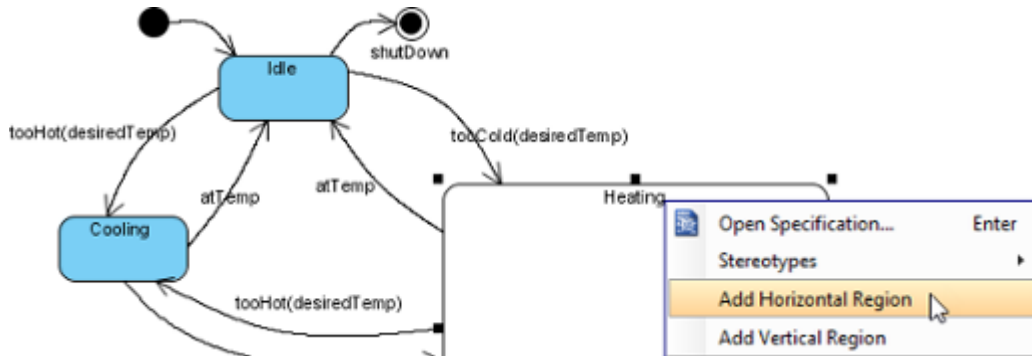
To create a state

5. A new state will be created and is transited from the source state. Enter its name and press **Enter** to confirm editing.



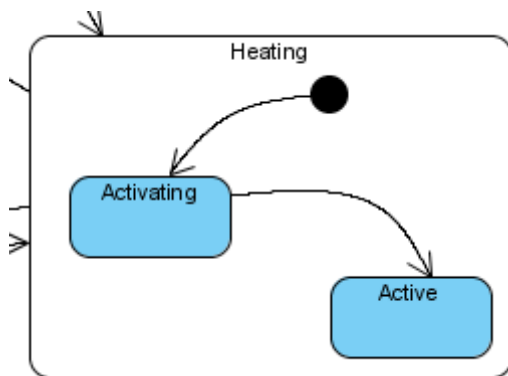
Step 3: Adding region to state

To model substates of a composite state, you need to add one or more regions to it. To add a region, right-click the state and select **Add Horizontal Region** from the popup menu.



Add region to state

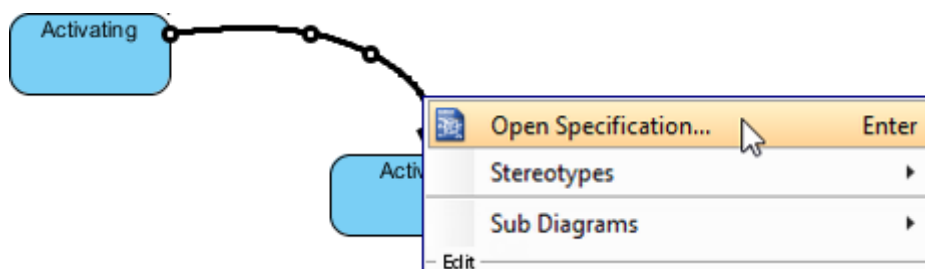
Next, you can draw the substates inside the region.



Substates in a composite state

Step 4: Modeling properties of transition

To model properties of transition such as effect and guard, right-click the transition and select **Open Specification...** from the pop-up menu.



Open specification of transition

When the **Transition Specification** pops out, you can edit its name, effect and guard. Next, select **Create Activity...** from the **Effect** property.

Transition Specification

General Triggers

Name: ready

Source: Activating ...

Target: Active ...

Kind: External

Effect: <Unspecified> ...

Redefined transition: <Unspecified> ...

Guard: Browse...

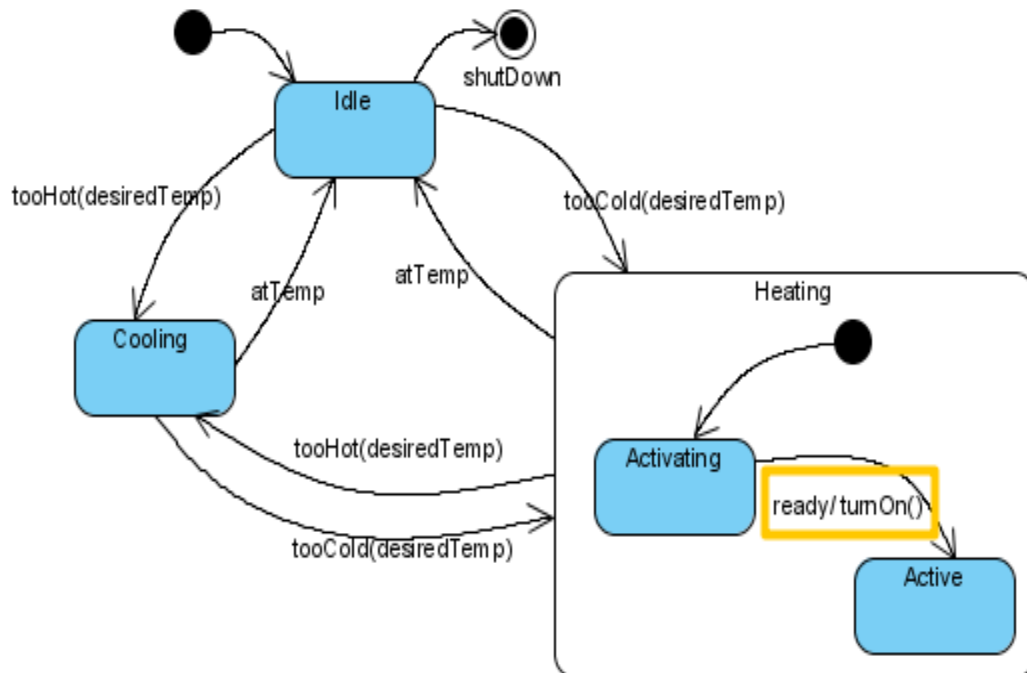
Select Constraint...

Description:

Create Activity from transition

In **Activity Specification (Effect)** window, change its name and then click **OK** button to apply the change.

Click **OK** in the **Transition Specification** to close it. The name and effect are shown on the transition caption.



Post-Experiment Questions:

Q1. What is the main usage of state chart diagram?

Q2. How state chart is different from collaboration Diagram?

EXPERIMENT - 8

Aim: Draw the component diagram for a given problem.

Description:

Component diagrams are different in terms of nature and behaviour. Component diagrams are used to model physical aspects of a system. Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node. So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems. Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make that functionalities. So from that point component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files etc. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

So the purpose of the component diagram can be summarized as:

- Visualize the components of a system.
- Construct executable by using forward and reverse engineering.
- Describe the organization and relationships of the components.

Pre-Experiment Questions:

Q1. Where to use component Diagram?

Q2. What is its importance in software engineering?

Procedure:

Step 1: Component diagram is a kind of UML diagram. shows the physical aspect of an object-oriented software system. It illustrates the architectures of the software components and dependencies between them.

Creating component diagram

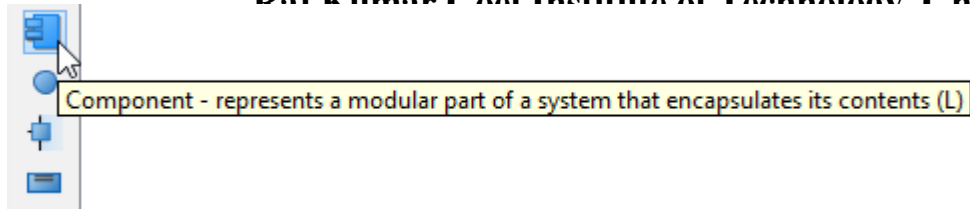
Perform the steps below to create a UML component diagram in Visual Paradigm.

1. Select **Diagram > New** from the application toolbar.
2. In the **New Diagram** window, select **Component Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**

Step 2:

Creating component

To create component in component diagram, click **Component** on the diagram toolbar and then click on the diagram.



Create component

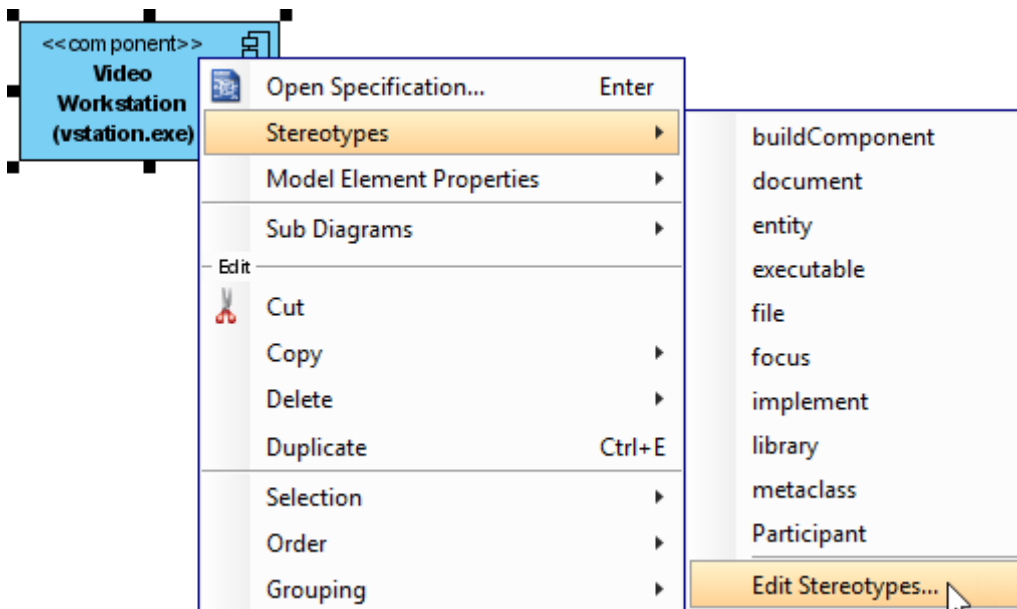
A component will be created.



Component created

Step 3: Assigning stereotypes

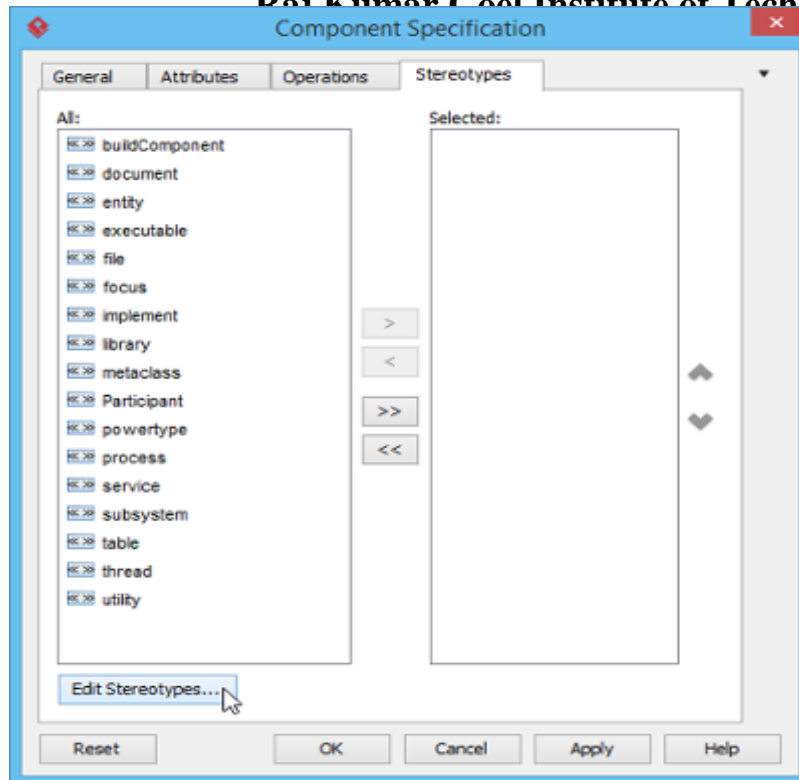
Right click on the package and select **Stereotypes > Edit Stereotypes...** from the pop-up menu.



Assign stereotypes

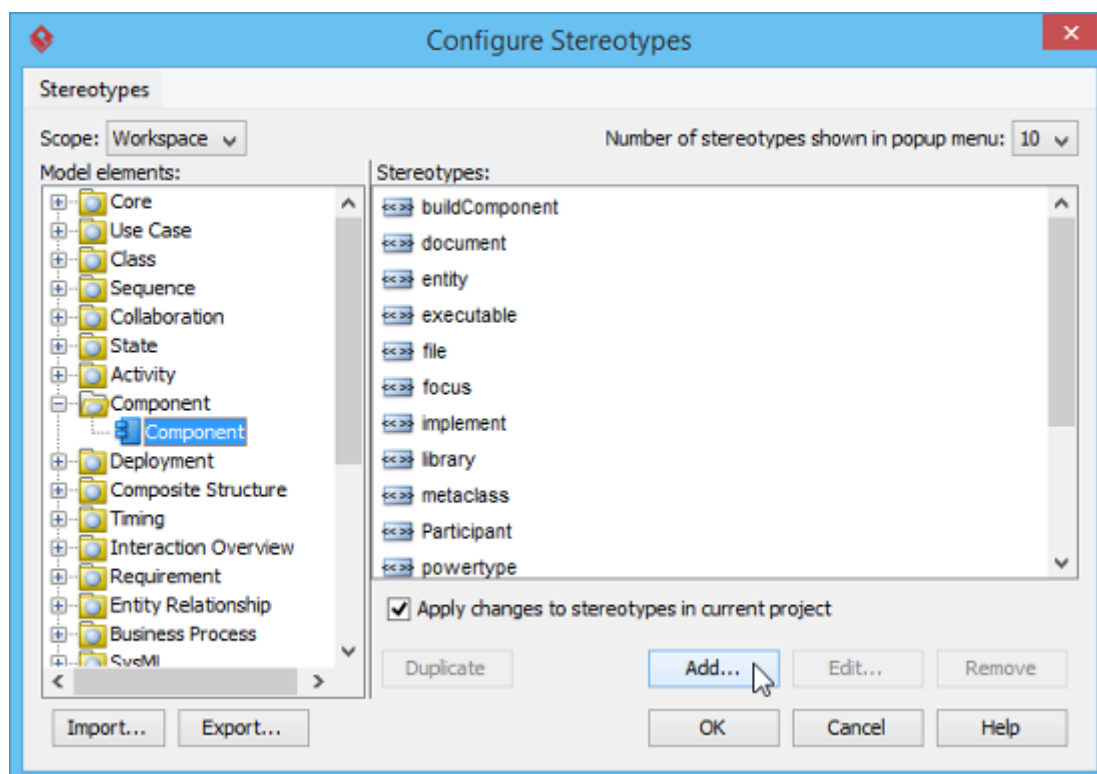
When the **Component Specification** window is opened, with the **Stereotypes** tab selected. The list on the left shows the selectable stereotypes.

If the stereotype you want to use is not on the list, click **Edit Stereotypes...** button.



Edit stereotypes

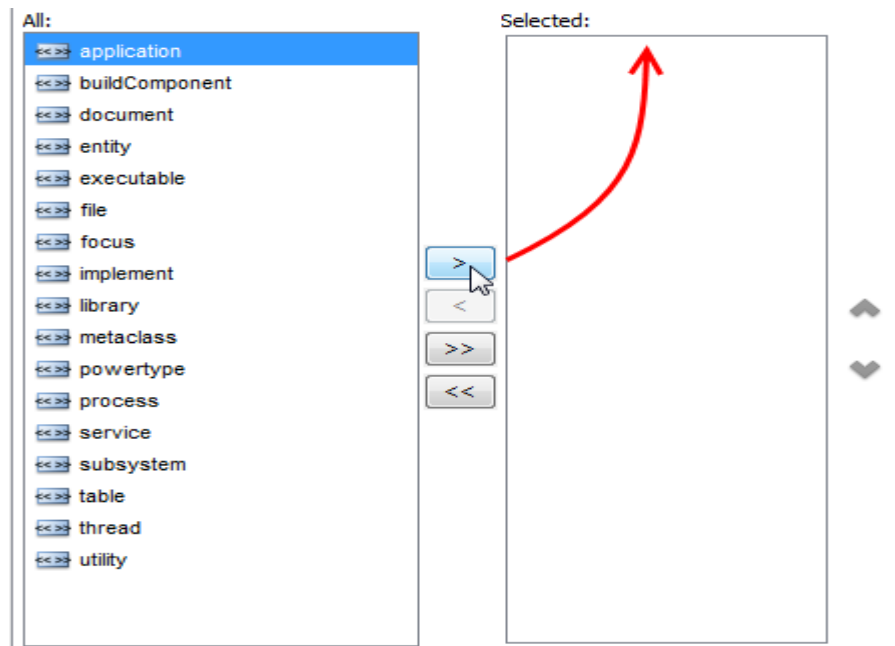
Click **Add...** button in the **Configure Stereotypes** window.



Add stereotype

Name the stereotype (e.g. *application*) in the **Stereotype Specification** window and then click **OK** button to close it. Click **OK** button in the **Configure Stereotypes** window. The added

stereotype will then be shown on the list in the **Component Specification** window. Select it and click **Add Selected** button. Finally, click **OK** button to confirm.



Step 4: Add selected stereotypes

Close the specification window. Stereotypes will be applied to the package.



Stereotypes assigned

Creating provided interface

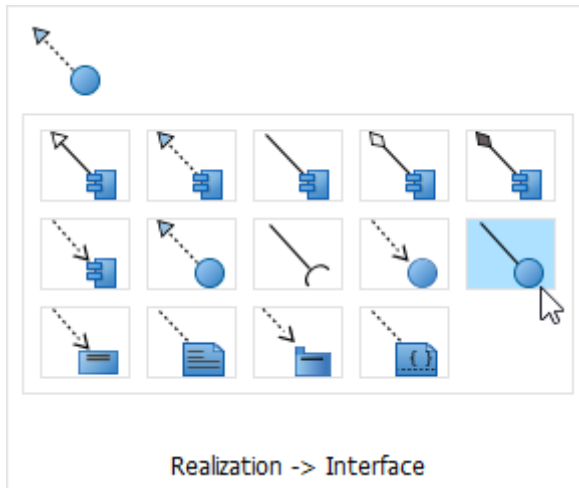
To create provided interface for a component:

1. Move your mouse pointer over the source component.
2. Press on the **Resource Catalog** button and drag it out.



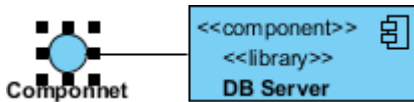
3. Release the mouse button at the place where you want the interface to be created.

4. Select **Realization** -> **Interface** from Resource Catalog.



To create a provided interface

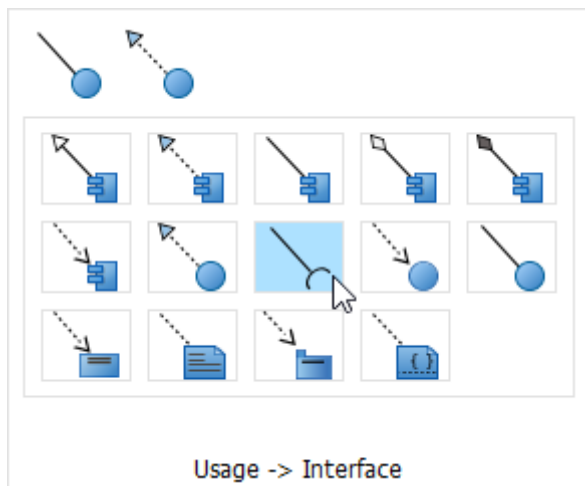
5. A new interface will be created and is connected to the source component. Enter its name and press **Enter** to confirm editing.



Interface created

Step 5: Creating required interface

To create required interface for a component, just follow the steps described above for creating provided interface, but change to select **Usage**-> **Interface** in Resource Catalog.



Step 6: Creating dependency

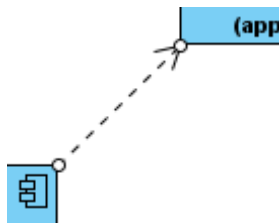
To create dependency, click **Dependency** on the diagram toolbar.



Dependency - a relationship between two model elements where changes in one of them will similarly affect the other

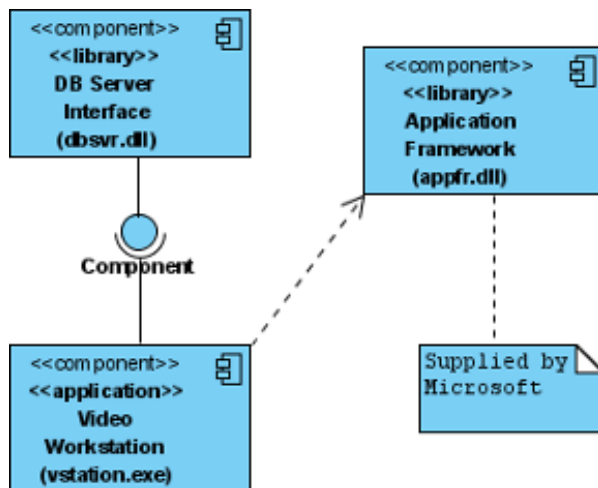
Create dependency

Drag from the source shape, move the mouse over the target shape and then release the mouse button to create the dependency.



Dependency created

Continue to complete the diagram



Completed diagram

Step 7: Showing/hiding attributes in component

Per diagram

You can add attributes to a component. To show/hide the attributes for all components in a diagram:

1. Right click on the background of the component diagram.
2. Select **Presentation Options > Component Display Options** from the popup menu.
3. Select/De-select **Show Attributes** to cause attributes to be shown or hidden.

Per component

You can add attributes to a component. To show/hide the attributes for a specific component:

1. Right click on the desired component.
2. Select **Presentation Options > Show Attributes Mode** from the popup menu.

3. Select **Follow Diagram/Show All/Hide All/Customized...** from the popup menu. If you have selected the **Customized** option, you can select the specific attribute(s) to be shown or hidden.

Step 8: Showing/hiding operations in component

Per diagram

You can add operations to a component. To show/hide the operations for all components in a diagram:

1. Right click on the background of the component diagram.
2. Select **Presentation Options > Component Display Options** from the popup menu.
3. Select/De-select **Show Operations** to cause attributes to be shown or hidden.

Per component

You can add operations to a component. To show/hide the operations for a specific component:

1. Right click on the desired component.
2. Select **Presentation Options > Show Operations Mode** from the popup menu.
3. Select **Follow Diagram/Show All/Hide All/Customized...** from the popup menu. If you have selected the **Customized** option, you can select the specific operation(s) to be shown or hidden.

Post-Experiment Questions:

- Q1. Differentiate between class and component diagram.
- Q2. Why to make a component diagram? What is its advantage in Software engineering?

Aim: To Draw the Deployment diagram for a given problem.

Description:

Deployment diagram is a structure diagram which shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process. Examples of artifacts are executable files, libraries, archives, database schemas, configuration files, etc.

Deployment target is usually represented by a node which is either hardware device or some software execution environment. Nodes could be connected through communication paths to create networked systems of arbitrary complexity.

Deployment diagrams could describe architecture at specification level (also called type level) or at instance level (similar to class diagrams and object diagrams).

Specification level deployment diagram shows some overview of deployment of artifacts to deployment targets, without referencing specific instances of artifacts or nodes.

Instance level deployment diagram shows deployment of instances of artifacts to specific instances of deployment targets. It could be used for example to show differences in deployments to development, staging or production environments with the names/ids of specific build or deployment servers or devices.

Some common types of deployment diagrams are:

- Implementation (manifestation) of components by artifacts,
- Specification level deployment diagram,
- Instance level deployment diagram,
- Network architecture of the system.

Pre-Experiment Questions:

Q1. What is deployment diagram in software engineering?

Q2. Explain the term deployment chart?

Procedure:

Deployment is a kind of UML diagram that shows the physical aspects of an object-oriented system. It also shows the configuration of run time processing nodes and artifacts.

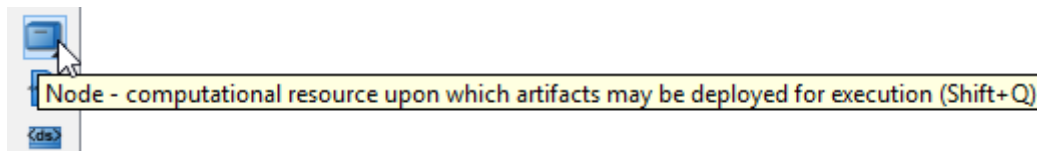
Step 1: Creating deployment diagram

Perform the steps below to create a UML deployment diagram in Visual Paradigm.

1. Select **Diagram** > **New** from the application toolbar.
2. In the **New Diagram** window, select **Deployment Diagram**.
3. Click **Next**.
4. Enter the diagram name and description. The **Location** field enables you to select a model to store the diagram.
5. Click **OK**.

Step 2: Creating node

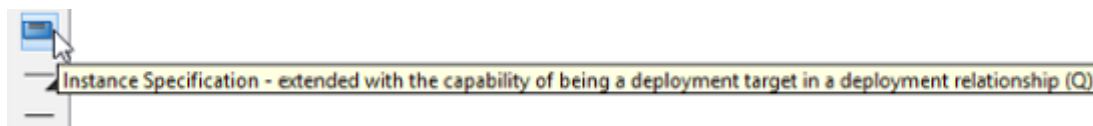
To create node in deployment diagram, click **Node** on the diagram toolbar and then click on the diagram.



Node selected

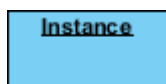
Step 3: Creating instance of node

To create instance of node, click **Instance Specification** on the diagram toolbar and then click on the diagram.



Create instance specification

An instance specification will be created.

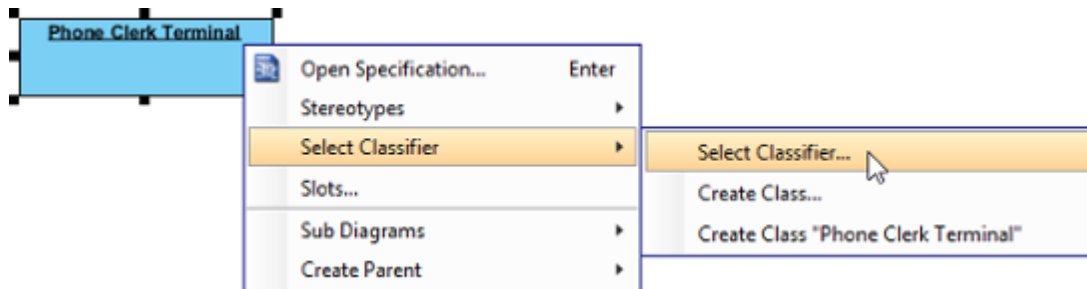


Instance specification created

Step 4: Selecting classifiers

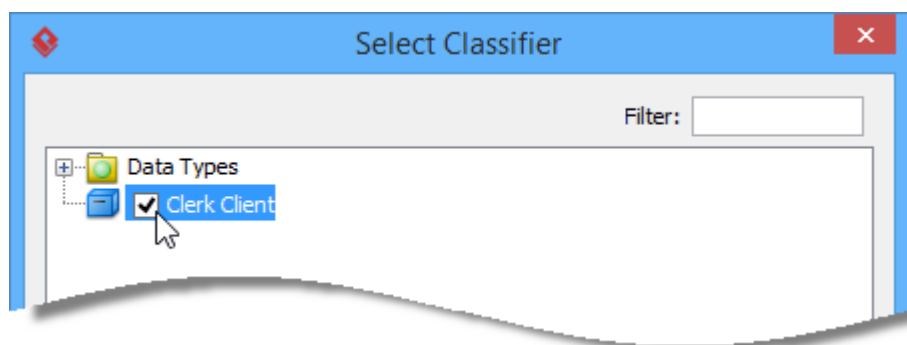
To specify classifiers for an instance specification, right-click it and select **Select Classifier** >

Select Classifier... from the pop-up menu.



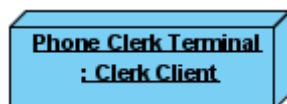
Select classifier

When the **Instance Specification Specification** window pops out, the **Classifiers** tab is opened by default. Click **Add...**. Then, select the classifier(s) in the popup window and click **OK**.



Select node

Click **OK** button to close the specification window. The selected classifiers are assigned to the instance specification.

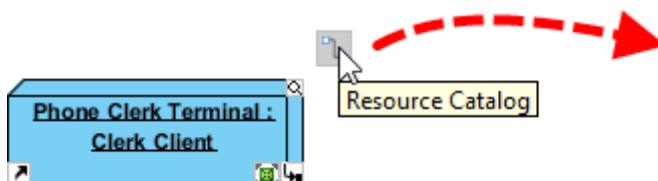


Classifiers assigned

Step 5: Creating link

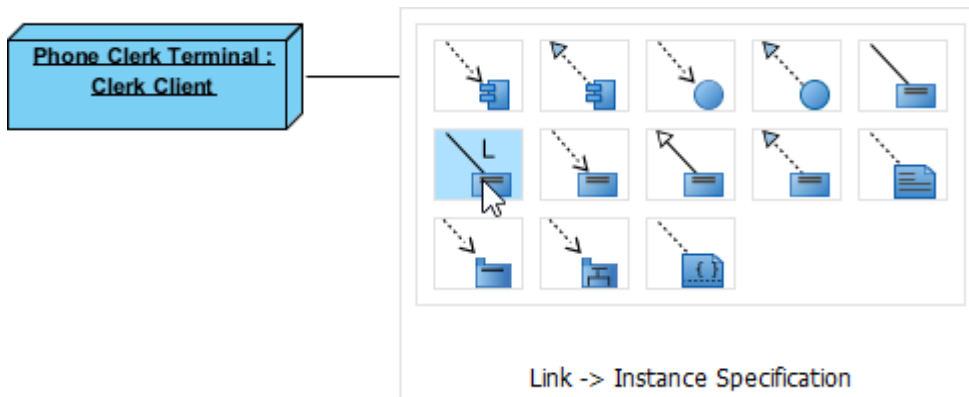
To create link from instance specification:

1. Move your mouse pointer over the source shape.
2. Press on the **Resource Catalog** button and drag it out.



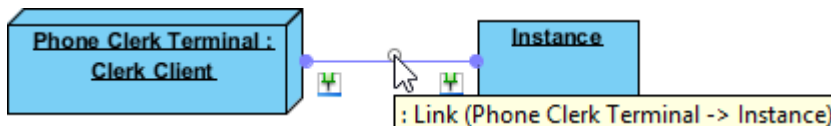
Using Resource Catalog

3. Release the mouse button at the place where you want the instance specification to be created.
4. Select **Link -> Instance Specification** from Resource Catalog.



To create an instance specification

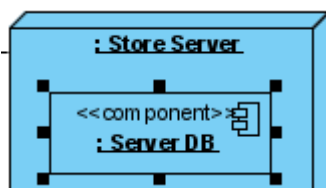
5. A new instance specification will be created and is connected to the source shape. Enter its name and press **Enter** to confirm editing.



Instance specification created

Step 6: Creating instance of component

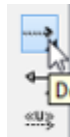
Similar to creating instance of node, you first create a component model element and then create an instance specification. However, this time assigns a component to the instance specification as classifier. After that the instance specification will be displayed as a component.



Instance of component

Step 7: Creating dependency

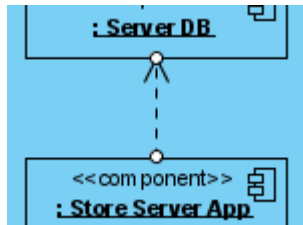
To create dependency, click **Dependency** on the diagram toolbar.



Dependency - a relationship between two model elements where changes in one of them will similarly affect the other

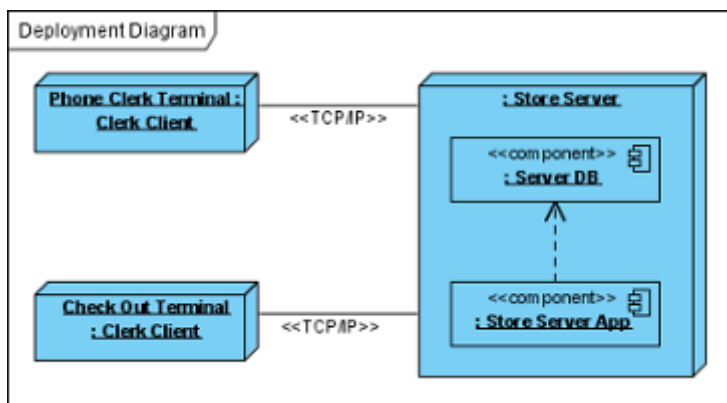
Create dependency

Drag from the source shape, move the mouse over the target shape and then release the mouse button to create the dependency.



Dependency created

Continue to complete the diagram.



Completed diagram

Post-Experiment Questions:

- Q1. Explain the use of deployment diagram in software & hardware components?
- Q2. How it is beneficial in software engineering?

EXPERIMENT - 10

Aim: Draw Data Flow Diagram for a given problem for a given problem.

Description:

Data analysis attempts to answer four specific questions:

- ☐ What processes make up a system?
- ☐ What data are used in each process?
- ☐ What data are stored?
- ☐ What data enter and leave the system?

Data drive business activities and can trigger events (e.g. new sales order data) or be processed to provide information about the activity. Data flow analysis, as the name suggests, follows the flow of data through business processes and determines how organisation objectives are accomplished. In the course of handling transactions and completing tasks, data are input, processed, stored, retrieved, used, changed and output. Data flow analysis studies the use of data in each activity and documents the findings in data flow diagrams, graphically showing the relation between processes and data.

Pre-Experiment Questions:

- 1 What are the symbols used in a DFD.
2. What is an external entity?
3. What is a context free diagram?

Procedure:

Physical and Logical DFDs

There are two types of data flow diagrams, namely physical data flow diagrams and logical data flow diagrams and it is important to distinguish clearly between the two:

Physical Data Flow Diagrams

An implementation-dependent view of the current system, showing what tasks are carried out and how they are performed. Physical characteristics can include:

Names of people

Logical Data Flow Diagrams

An implementation-independent view of the a system, focusing on the flow of data between processes without regard for the specific devices, storage locations or people in the system. The physical characteristics listed above for physical data flow diagrams will not be specified.

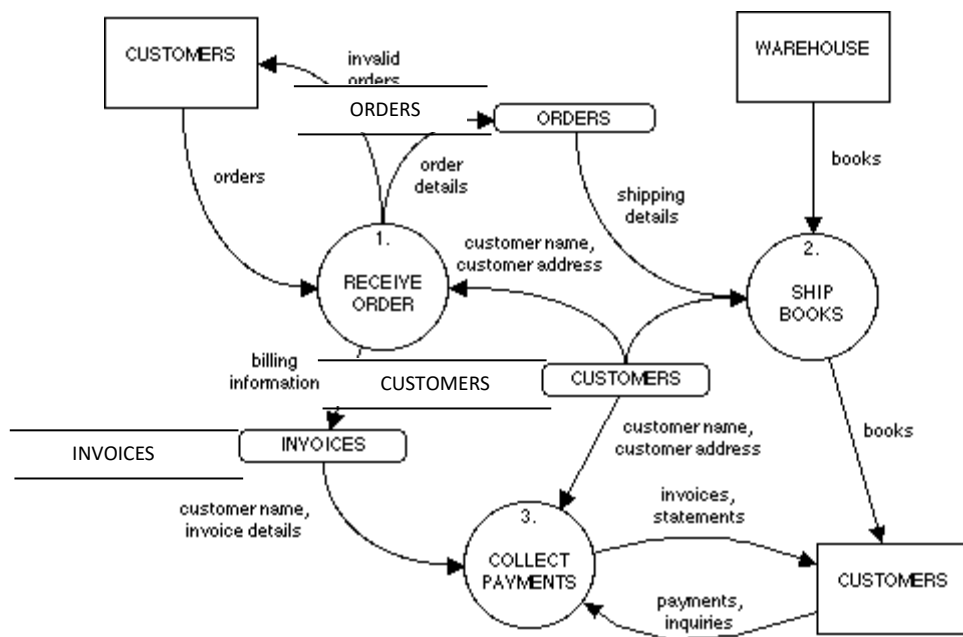


Fig. A typical DFD

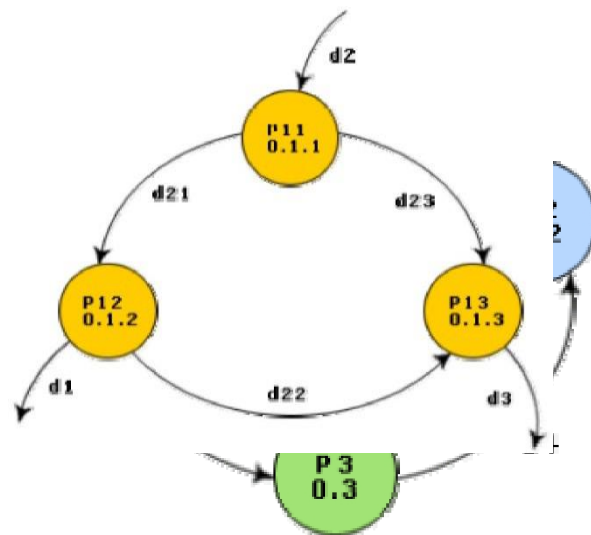
Data Flow Diagram (DFD)

The DFD (also known as a bubble chart) is a hierarchical graphical model of a system that shows the different processing activities or functions that the system performs and the data interchange among these functions. Each function is considered as a processing station (or process) that consumes some input data and produces some output data. The system is represented in terms of the input data to the system, various processing carried out on these data, and the output data generated by the system. A DFD model uses a very limited number of primitive symbols [as shown in fig. 5.1(a)] to represent the functions performed by a system and the data flow among these functions.

Symbols used for designing DFDs

Here, two examples of data flow that describe input and validation of data are considered. In Fig. 5.1(b), the two processes are directly connected by a data flow. This means that the 'validate-number' process can start only after the 'read-number' process had supplied data to it. However in

Fig 5.1(c), the two processes are connected through a data store. Hence, the operations of the two bubbles are independent. The first one is termed 'synchronous' and the second one 'asynchronous'.
Importance of DFDs in a good software design



The main reason why the DFD technique is so popular is probably because of the fact that DFD is a very simple formalism – it is simple to understand and use. Starting with a set of high-level functions that a system performs, a DFD model hierarchically represents various sub-functions. In fact, any hierarchical model is simple to understand. Human mind is such that it can easily understand any hierarchical model of a system – because in a hierarchical model, starting with a very simple and abstract model of a system, different details of the system are slowly introduced through different hierarchies. The data flow diagramming technique also follows a very simple set of intuitive concepts and rules. DFD is an elegant modeling technique that turns out to be useful not only to represent the results of structured analysis of a software problem, but also for several other applications such as showing the flow of documents or items in an organization

Data dictionary

Raj Kumar Goel Institute of Technology, Ghaziabad

A data dictionary lists all data items appearing in the DFD model of a system. The data items listed include all data flows and the contents of all data stores appearing on the DFDs in the DFD model of a system. A data dictionary lists the purpose of all data items and the definition of all composite data items in terms of their component data items. For example, a data dictionary entry may represent that the data grossPay consists of the components regularPay and overtimePay.

Balancing a DFD

The data that flow into or out of a bubble must match the data flow at the next level of DFD. This is known as balancing a DFD. The concept of balancing a DFD has been illustrated in fig. 5.3. In the level 1 of the DFD, data items d1 and d3 flow out of the bubble 0.1 and the data item d2 flows into the bubble 0.1. In the next level, bubble 0.1 is decomposed. The decomposition is balanced, as d1 and d3 flow out of the level 2 diagram and d2 flows in.

Post-Experiment Questions:

1. What is Data-dictionary?
2. Why balancing of DFD is required?

EXPERIMENT - 11

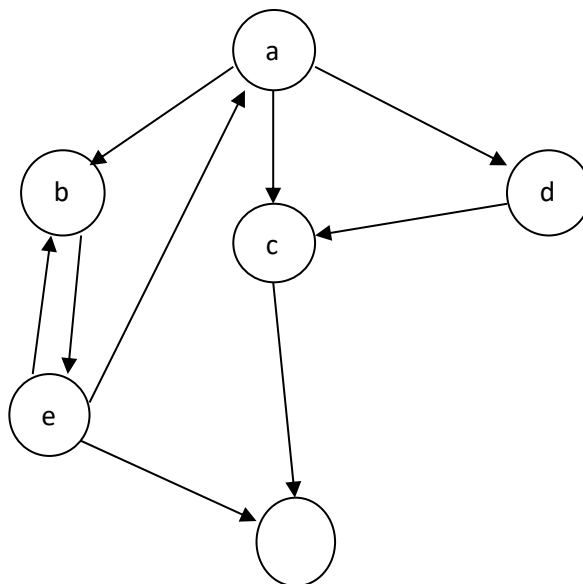
Aim: Write a program to compute cyclomatic complexity.

Description:

The cyclomatic complexity is also known as structural complexity because it gives internal view of the code. This approach is used to find the number of independent paths through a program. This provides us the upper bound for the number of tests that must be conducted to ensure that all statements have been executed at least once and every condition has been executed on its true or false side.

McCabe's cyclomatic matrices $V(G)$ of a graph G with n vertices, e edges and P connected components is $V(G)=e-n+2P$

Example:



The value of cyclomatic complexity is :

$$V(G) = 9 - 6 + 2 \times 1 = 5$$

Where: $e=9$, $n=6$ and $P=1$

Raj Kumar Goel Institute of Technology, Ghaziabad
Pre-Experiment Questions:

1. What is software metrics?
2. What are advantages of cyclomatic complexity?

PROGRAM :

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int e,n,P,result;
clrscr();
printf("\nEnter the value of no of nodes edges , vertices and connected component/n");
scanf("%d%d%d",&e,&n,&P);
result=(e-n+(2*P));
printf("\nThe cyclomatic complexity is V(G)=");
printf(result);

}
```

Post-Experiment Questions:

1. What is P in Cyclomatic Complexity?

EXPERIMENT - 12

Aim: Write a program to compute Halstead's Complexity.

Theory in Brief

Halstead complexity measures are software metrics introduced by Maurice Howard Halstead in 1977 as part of his treatise on establishing an empirical science of software development. Halstead makes the observation that metrics of the software should reflect the implementation or expression of algorithms in different languages, but be independent of their execution on a specific platform. These metrics are therefore computed statically from the code.

Halstead's goal was to identify measurable properties of software, and the relations between them. This is similar to the identification of measurable properties of matter (like the volume, mass, and pressure of a gas) and the relationships between them (such as the gas equation). Thus his metrics are actually not just complexity metrics.

Pre-Experiment Questions:

1. What is software metrics?
2. What are advantages of Halstead software science?

Calculation

For a given problem, Let: First we need to compute the following numbers, given the program:

- n_1 = the number of distinct operators
- n_2 = the number of distinct operands
- N_1 = the total number of operators
- N_2 = the total number of operands

From these numbers, several measures can be calculated:

Program vocabulary: $n = n_1 + n_2$

- Program length: $N = N_1 + N_2$
- Calculated program length: $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$
- Volume: $V = N * \log_2 n$
- Difficulty : $D = (n_1/2) * (N_2/n_2)$
- Effort: $E = D * V$

Raj Kumar Goel Institute of Technology, Ghaziabad

PROGRAM :

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int n1 , n2 , N1 , N2 , N ,V, N , D ,E ;
printf("\n Enter the value of no of distinct operators , no of distinct operands , total no of
operators , total no of operands /n ");
printf("\n Program Vocabulary : n =");
printf(n1 + n2);
printf("\n Program length : N=");
printf(N1 + N2);
printf("\n Calculated program length :N=");
printf((n1*log2n1)+(n2*log2n2));
printf("\n Volume : V=");
V = N * log2n;
printf(V);
printf("\n Difficulty : D=");
D=((n1/2)*(N2/n2));
printf(D);
printf("\n Effort :E=");
printf(D*V);
}
```

Post-Experiment Questions:

1. How we calculate Length and vocabulary?
2. How we calculate Effort?

References

1. RS Pressman, Software Engineering: A Practitioners Approach, McGraw Hill.
2. Pankaj Jalote, Software Engineering, Wiley
3. Rajib Mall, Fundamentals of Software Engineering, PHI Publication.
4. KK Aggarwal and Yogesh Singh, Software Engineering, New Age International Publishers.

APPENDIX

AKTU SYLLABUS

RCS 651: SOFTWARE ENGINEERING LAB

For any given case/ problem statement do the following;

1. Prepare a SRS document in line with the IEEE recommended standards.
2. Draw the use case diagram and specify the role of each of the actors. Also state the precondition,
post condition and function of each use case.
3. Draw the activity diagram.
4. Identify the classes. Classify them as weak and strong classes and draw the class diagram.
5. Draw the sequence diagram for any two scenarios.
6. Draw the collaboration diagram.
7. Draw the state chart diagram.
8. Draw the component diagram.
9. Perform forward engineering in java. (Model to code conversion)
10. Perform reverse engineering in java. (Code to Model conversion)
11. Draw the deployment diagram