Tutor on Software Design

Fariza binti Halim

WEK010074

WXES 3182: Tutor on Software Design (Sequence Diagram)

Supervisor: Puan Nazean Jomhari

Moderator: Puan Siti Hafizah Ab Hamid

Faculty of Computer Science
And Information Technology

Abstract

Tutor on Software Design is a system that helps users design their own software using the Unified Modeling Language (UML). The UML diagrams involved are the use case diagram, Class diagram, and sequence diagram. It is a stand-alone system that requires operating system Windows 98 or above. This report especially focuses on designing software using sequence diagram. The system receives requirements using step-by-step wizard-like method. The requirements (called as elements) include actor, use case, association, class, object and sequence of events in the exact order. After all elements are entered, the system produces a professional looking sequence diagram. The produced diagram could be edited by users and saved. This system provides a lot of information, tips, tutorial and help regarding the system usage and sequence diagram that can be accessed any time.

Contents

List of Figures

List of Tables

Chapter 1

Project Introduction

1.1 Project Introduction

The purpose of this project is to develop a system called Tutor on Software Design. Tutor on Software Design is a stand-alone system developed to help design software using the UML (Unified Modeling Language) diagrams. The software provides step by step tutorials in drawing the diagrams. The diagrams included in this software are the use-case diagram, the class diagrams and the sequence diagrams.

This system receives user's requirements and translates them into the desired diagrams. The requirements entry process will take a few steps. These steps are taken to show users how the designing is done and also to ensure that the system gets the requirements correctly.

Each step requires users to enter their requirements into textboxes, or choose previously entered data from combo boxes and list boxes. The data entered are in the form of names of use cases, classes and messages to be passed between objects. This method of data entry is to help those who are not very familiar with drawing sequence diagram. This way, they only have to determine the objects and the flow of control of the system they are developing, and the system will show them what their diagram will look like. This method is also used to limit the data entered by users so that errors are minimized and the output would be reliable.

Based on requirements entered by users, this system produces a reliable and professional looking sequence diagram. The produced diagrams can be edited, saved and printed.

1.2 Project Objectives

Setting objectives to a project is very important as it helps to identify the system's requirements as well as it provides a guideline for the rest of the development process. For this project, the development objectives will cover the following aspects:

1. To build a system that can help software designers, including students, to design software using UML diagrams.
2. The system will be a tool and reference to software designers especially students.
3. This system will reduce the problem of learning to design software because users can learn as they design.
4. It will be an easy to use, easy to learn and user-friendly system.
5. This software will be useful for students who are developing software such as their thesis.
6. Software developers that have little knowledge of UML can learn quickly about using UML diagrams to design their software.

1.3 Project Motivations

UML is a very useful modeling language in designing software. It acts as an interface between the natural language of user requirements and the programming language of implementation. It is also the communication medium between software designers with programmers.

Designing software solely based on the user requirements is hard and dangerous. Points might be missed and concepts might be misunderstood. Representing user requirements in graphical notations makes the reviewing and understanding of the desired software easier. It will also minimize the risk of missed points and misunderstood concepts. In design phase of every methodology, user requirements should be translated into UML diagrams to increase efficiency during implementation phase.

A lot of people are involved in designing software. Some organizations appoint a certain specialized personnel to design the system and draw the UML diagrams. But in most, systems analysts, programmers and etcetera will have to design and draw themselves. By using this system, it would not be a problem.

In the case of students doing their thesis, that student is the system analyst, the system designer, and the programmer, all in one. Therefore, they will have to design the software and draw the diagrams by themselves. Designing a software and drawing UML diagrams for it manually is a chore. Doing it using CASE tools requires them to

1.4 Project Scope

There are a lot of diagrams in UML. For instance, activity diagram, collaboration diagram, state diagram, entity relationship diagram (ERD) and data flow diagram (DFD). This system will only use three of the UML diagrams. The diagrams are the use-case diagram, the class diagram and the sequence diagram.

The system produces a sequence diagram based on user requirements. These requirements are entered step by step. Each step requires users to enter one element such as use case, class and object.

Tutor on Software Design is a stand-alone system. One system runs on one computer. It is not available on-line.

1.5 Target User

The target users for this system are system designers and system developers. Those who are developing software can benefit a lot from this system. They may have experience in designing software with UML manually or using the CASE tools, but this system might be a better and faster new way of doing it.

Students can use this system to help them design their software. The system is especially helpful if the students are developing a system such as their thesis. In most cases, students doing their thesis have not learned much about UML. With that little knowledge and experience, this software can save their time compared to having to learn how to design with UML from books and having to learn how to use the available CASE tools.

Teachers and lecturers may use this system to help them teach their students about using UML to design software. This software is easier to understand and exciting compared to books and boring lectures.

1.6 System Capability and Constraints

The Tutor on Software Design, as indicated by its name, tutors users to design software using three of the UML diagrams- use-case diagram, class diagram and sequence diagram. It is a private tutor for every user. Users will enter the requirements for their software. Then, the system will use the elements in the requirements to draw the diagrams. For example, actors, use-cases, objects, etcetera. Users will be asked a few questions element by element. This is done to ensure that the system gets the user's requirements correctly. This is also done to let users see the steps to be taken in designing software using the UML. The system will then draw the required diagrams.

Every time users change any data that has been entered, the diagram will change automatically according to the changes. The finished diagram can be saved in various image formats such as .bmp, .tif and etcetera.

1.7 Project Schedule

Project scheduling is done to manage time and tasks systematically. It is also done to avoid late delivery. The following Gantt chart represents schedule for this project:

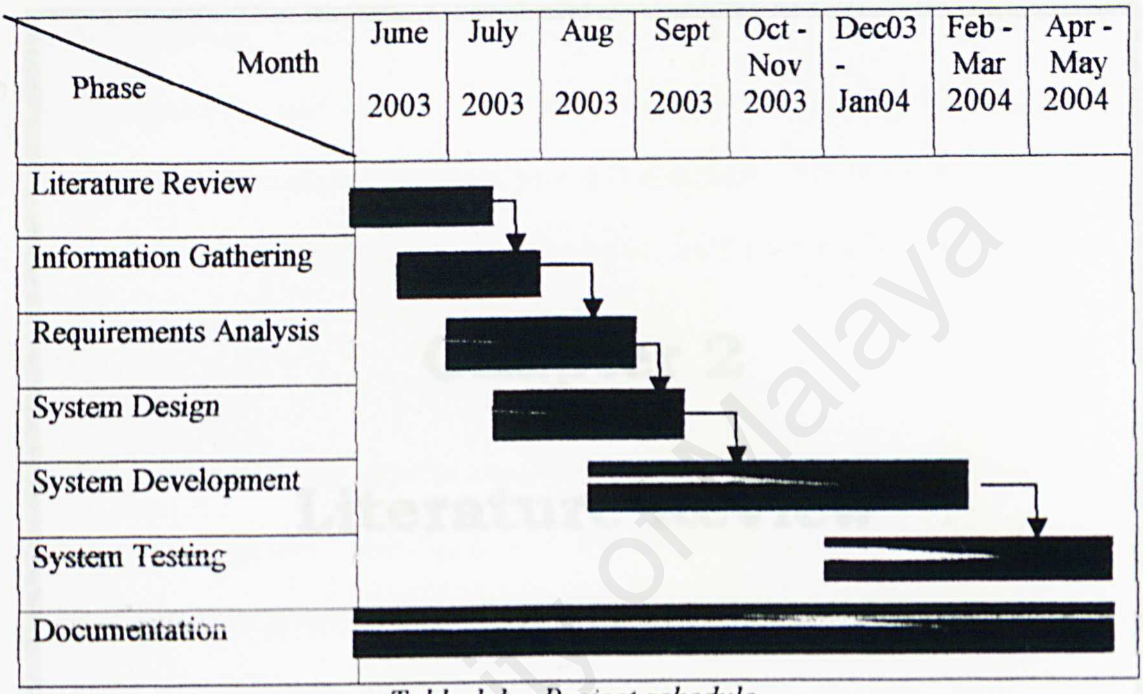


Table 1.1 – Project schedule.

Chapter 2

Literature Review

2.1 Introduction to Literature Review

A system is a collection of objects and activities, plus a description of a relationship that tie the objects and activities together. Typically, a system definition includes, for each activity, a lot of inputs required, actions taken and output produced. A system can be developed in different ways. Before developing a system, information about the characteristics and purposes of the system to be developed, the procedures involved, and the methodology used need to be gathered. There are many sources, which this valuable information can be gathered. Each source will provide different information and facts.

Literature review is the information gathering and research phase of this project. Information is gathered thoroughly as it will be used as guidance in planning the application development process. Information gathered in several ways, some of the methods are discussion with supervisor, document room, and discussions among colleagues, forums and Internet.

2.1.1 Discussion with Supervisor

Early discussion is more towards the requirements of the system, project scope and the objective of this project. By having a goal in mind, the group members start gathering information, planning and designing. All the while, reporting to supervisor face to face or by e-mails. Then, discussions are more to correcting whatever mistakes, errors and misunderstanding done.

2.1.2 Document Room

Another source for gathering information is the document room in Faculty of Computer Science and Information Technology, University of Malaya. This room contains thesis reports of past year students. These documents are very suitable as guidance for report writing. Basically, the format of most of the reports is almost the same. Some of the thesises are on tutor software in different topics such as C language. Others concerns UML such as Meta Modeling via UML. These relevant reports are very useful for this project.

2.1.3 Internet

The internet has become the indispensable source for searching any information. It has become one of the major sources for obtaining the latest information. Information can be gathered in the most cost-effective and time-efficient manner using the internet.

There are many virtual projects done by commercial companies or research institutes. These projects are published on the internet, providing very useful information. Besides, several websites of software companies were visited to gather further information about certain software for comparison.

Furthermore, journals, books and articles are now published electronically. A lot of these e-journals, e-books and e-articles can be accessed free via the UM library from inside the varsity.

Three software relevant to this project are discussed and compared in this report. The trial versions of these software were used for evaluation and first-hand experience. These software are available on the internet.

2.1.4 Internet forum

The forums give more specific answers to questions relating to this software that blindly searching in the internet. There are many people around the world who can give their opinions on my problems. I tried using their opinions and picked the best. Joining various forums on VB.Net also gives me more knowledge, and I also give my opinions whenever I can. Some of the forums that I joined are forums.devx.com, www.vbcity.com, www.vbforums.com, www.vbwm.com, www.wimdows.net.

2.1.5 Discussions Among Colleagues

Because this is a group project of three members, we have the advantage of discussing our problems and misunderstandings regarding the software development. We also discuss to get full understanding of UML. Furthermore, discussions must be held occasionally to avoid contradictions as we are developing the same system.

Although a lot of information gathered, many things remain unclear. Therefore, some researches need to be done to make them clear. Researches includes the conditions of each of the sequence diagram elements and notations, how to draw sequence diagram from user requirements, from use case diagrams and from class diagrams, problems getting the user requirements due to natural language, and handling the ever changing user requirements.

2.2 Introduction to Sequence Diagram

There are a few modeling in UML, such as use-case modeling, activity modeling, class modeling, interaction modeling and state chart modeling. Sequence diagram falls under the category of interaction diagram.

2.2.1 Interaction Modeling

Interaction modeling captures interactions between objects needed to execute a use case. Interaction models are used in more advanced stages of requirements analysis, when a basic class model is known, so that the references to objects are backed by the class model. The interaction modeling shows the sequencing of events (messages) between collaborating objects.

There are two kinds of interaction diagrams – the sequence diagram and the collaboration diagram. They can be used interchangeably and, indeed, many CASE tools support an automatic conversion from one model to the other. The difference is in emphasis. The sequence models concentrate on time sequences and the collaboration models on object relationships.

2.2.2 History of Sequence Diagram

The existence and evolution of sequence diagram is parallel with other diagrams in UML. UML actually starts with Unified Method 0.8 from the combination of Booch Method and James Rumbaugh's Object Modeling Technique (OMT).

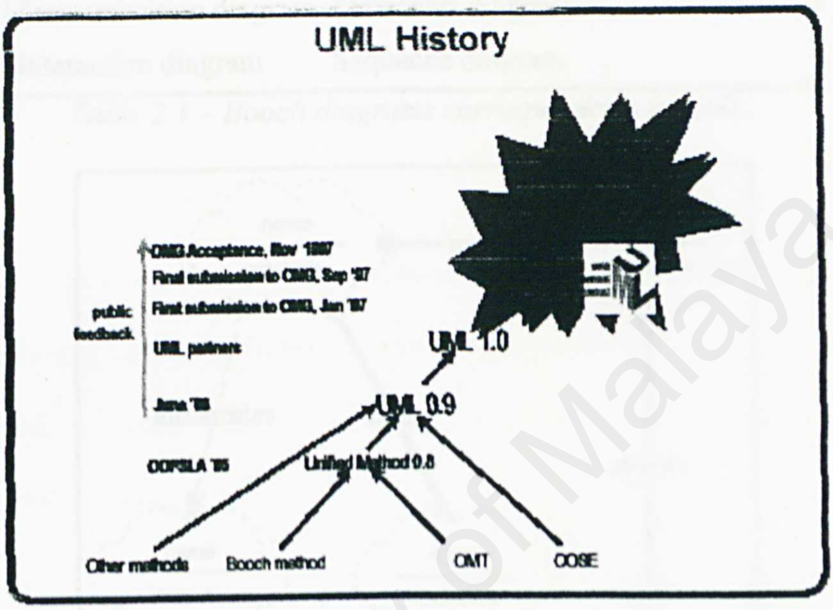


Figure 2.1 – History of UML.

Grady Booch's Object-Oriented Design (OOD), also known as Object-Oriented Analysis and Design (OOAD), is a precursor to the Unified Modeling (UML). The Booch method includes six types of diagrams: class, object, state transition, interaction, module, and process. Booch's class and object diagrams (classified as Booch's Static Diagrams) differentiate this methodology (at least in notation) from similar object oriented systems.

Booch's Dynamic Diagrams use state transition and interaction diagrams to illustrate the dynamic nature of an application. Below is a table that lists what each of the dynamic Booch diagrams corresponds to in UML.

Booch (OOD)	**Unified Modeling Language (UML)**
State transition diagram	Statechart diagram
Interaction diagram	Sequence diagram

Table 2.1 – Booch diagrams correspondents in UML.

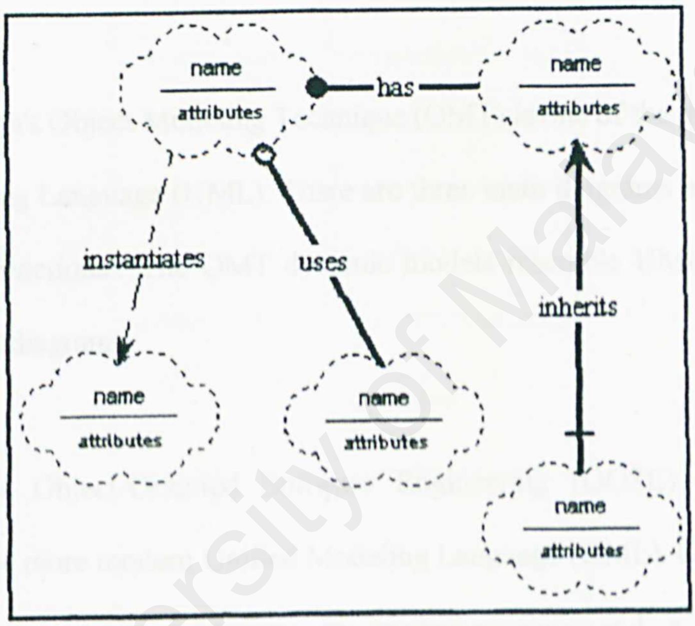
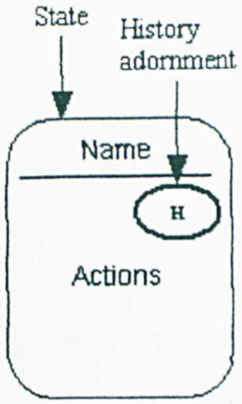


Figure 2.2 – Booch's class diagram.

Booch's Dynamic Diagram Notations



State

States represent situations during the life of an object. Draw a Booch state symbol using a rectangle with rounded corners and two compartments. Use the oval-shaped H symbol to indicate the most recently visited state.

James Rumbaugh's Object Modeling Technique (OMT) is one of the precursors to the Unified Modeling Language (UML). There are three main diagrams in OMT: object, dynamic, and functional. The OMT dynamic models resemble UML sequence and UML statechart diagrams.

Ivar Jacobson's Object-Oriented Software Engineering (OOSE) is one of the precursors to the more modern Unified Modeling Language (UML). OOSE includes a requirement, an analysis, a design, an implementation, and a testing model. Jacobson's design model shows how the system behaves. There are two types of diagrams under this model: interaction diagrams and state transition diagrams. Interaction diagrams are similar to UML's sequence diagrams. State transition diagrams are like UML statechart diagrams, but Jacobson also employs a number of unique symbols listed below.

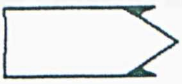
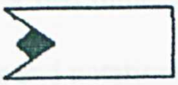
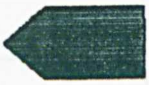
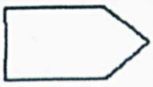
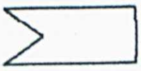
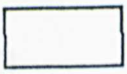
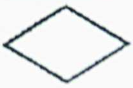
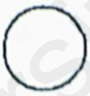
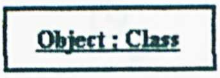
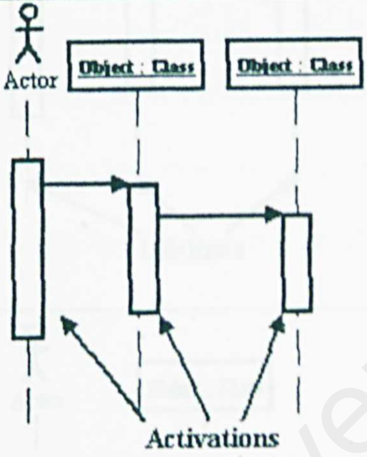
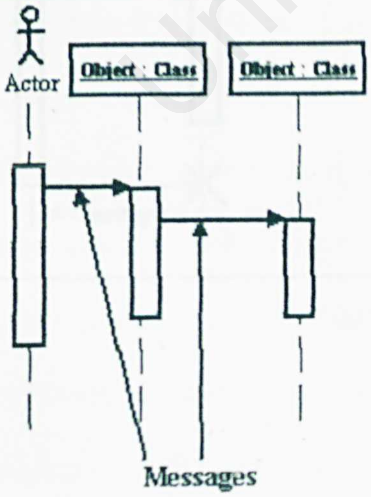
	**Send Message**
	**Receive Message**
	**Return Message**
	**Send Signal**
	**Receive Signal**
	**Perform Task**
	**Decision**
	**Label**

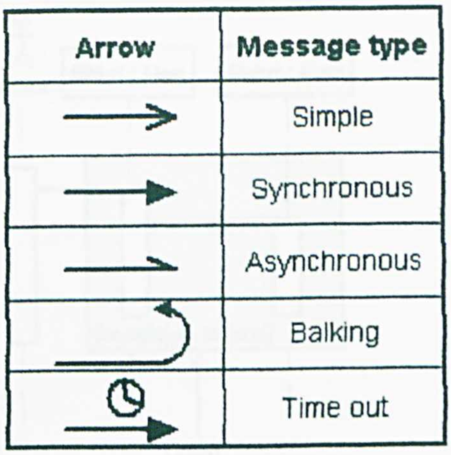
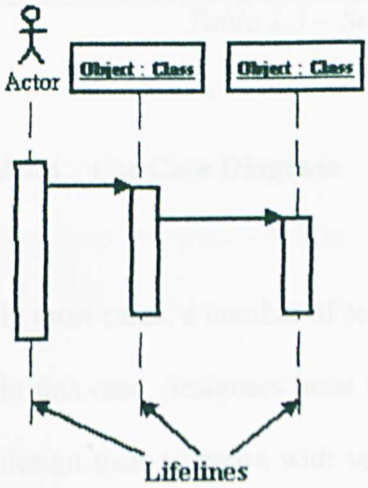
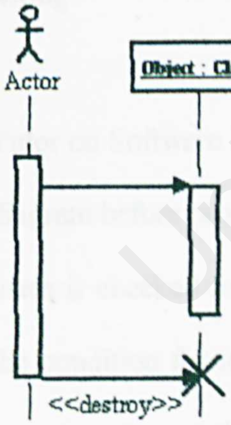
Table 2.2 – Unique symbols employed by Jacobson.

The combination of Ivar Jacobson's Object-Oriented Software Engineering (OOSE) with Unified Method 0.8 and other models produce Unified Modeling Language (UML) 0.9. This is followed by the other versions, UML 1.0, UML 1.1, UML 1.3 and finally UML 2.0.

2.2.3 Sequence Diagram Elements and Notations

The sequence diagram is a two dimensional graph with elements on horizontal and vertical axis. The elements and notations used in sequence diagrams are:

	**Class roles** Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.
	**Activation** Activation boxes represent the time an object needs to complete a task.
	**Messages** Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.

 Arrow | Message type Simple Synchronous Asynchronous Balking Time out	Various message types for Sequence and Collaboration diagrams
	**Lifelines** Lifelines are vertical dashed lines that indicate the object's presence over time.
	**Destroying Objects** Objects can be terminated early using an arrow labeled "< < destroy > >" that points to an X.

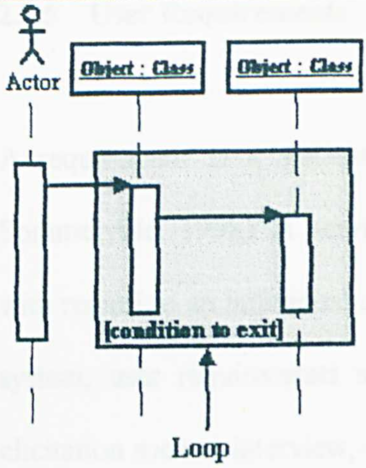
 Actor Object : Class Object : Class [condition to exit] Loop	**Loops** A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].

Table 2.3 – Sequence diagram elements and notations.

2.2.4 Use Case Diagram

In most cases, a number of sequence diagrams are derived from one use case diagram. In this case, designers must make sure that the use case is correct before moving to design their software with sequence diagram. Otherwise, the whole design would be wrong.

Tutor on Software Design has a way of reducing the risk of getting the wrong use case diagram before proceeding to the next diagram. Every instance of elements entered by users is checked for errors. Errors in this instance mean whether the elements fulfills the condition for it to be that elements. For example, actor must be a noun and use case must be a verb. If errors are detected, users will be prompted to correct the errors or confirm the elements if they persist to proceed. This is to make them aware of error or mistake occurrences and choose to correct them or not. The same principles implemented for sequence diagram to avoid or reduce the risk of wrong designs.

2.2.5 User Requirements

A requirement is a statement of a system service or constraint (Kotonya and Sommerville, 1998). A service statement describes how the system should behave with regard to an individual user or to the whole user community. Before designing a system, user requirements should be clear. This can be done by many ways of elicitation such as interview, questionnaire, observation, or others.

Elicited requirements are then documented into the requirements specification document. This document is usually presented in a detailed and precise natural language. Natural language is the language used in our day to day conversation. It can be English, Malay, or others. Some documents are presented in the structured language. This language is the structured version of natural language.

Tutor on Software Design is a system that helps users design their software. To do that, users will have to tell the system their requirements, so the system can draw three of the UML diagrams for them based on their requirements. However, this is not as easy to implement as it sounds.

There are two problems about getting user requirements for this system. First, what is the best way to get user requirements so it is possible for the system to translate them into diagrams? Second, what happens if the users tend to change their requirements ever so often? These problems are discussed in the following sections as how to get user requirements and how to handle the ever changing user requirements.

2.2.5.1 How to Get User Requirements

Due to the trouble of translating user requirements in the form of natural language into sequence diagram, another way has been figured out and developed. This problem can be solved by limiting and restricting the way users enter their requirements. Two methods that can do exactly that have been identified. The methods are structured language, and step-by-step requirements entry.

Structured Language

This method requires users to enter their requirements using a certain specified language and the system will design their software based on it. Tutor on Software Design would have to design a structured language that is easy for users to learn and use, and also easy to be translated into sequence diagrams.

An existing tool, EventStudio has already used this method. It uses a language called Feature Description Language (FDL). The language and tool will be discussed further later in this chapter.

Step By Step Requirements Entry

Another method is by providing specified boxes for users to enter their requirements. For example, the system specifies a certain text box for users to fill it in with an actor's name and click add. Then they can choose to enter another actor or proceed to enter use cases the same way they enter the actors. To enter object, users will have to choose which class the object belongs to using the combo box, give the object's name and click add. This is the method adopted by Tutor on Software Design. It will be discussed later in chapter 4, Analysis and Design, in the Graphical User Interface section.

2.2.5.2 How to Handle the Ever Changing User Requirements

User requirements are always changing. Therefore, Tutor on Software Design must support these changes. The system handles this problem by allowing users to go back to previous tasks to change what they have entered. For example, while the user is entering objects, he decided to enter another class. He can click on the back button to change, add or delete any classes.

Another feature of this system that support requirements changes is, users can still edit the diagrams produced by the system. Therefore, users would not have to worry if they realize they missed something, or suddenly would like to add something, after the system produces the sequence diagrams. In any case, users can always go to previous tasks for corrections or improvement. The system will produce a new diagram after changes.

2.3 Introduction to Existing UML Tools

Researches have to be conducted on existing UML tools to gather useful information for the system development project. The information gathered will give a preview of what the system will be. The tools studied do not have to be exactly the same with the system to be developed in this project. Research can be done based on the interface, system structure and etcetera. Basically, it is a way to identify possible requirements for the system.

2.3.1 SmartDraw

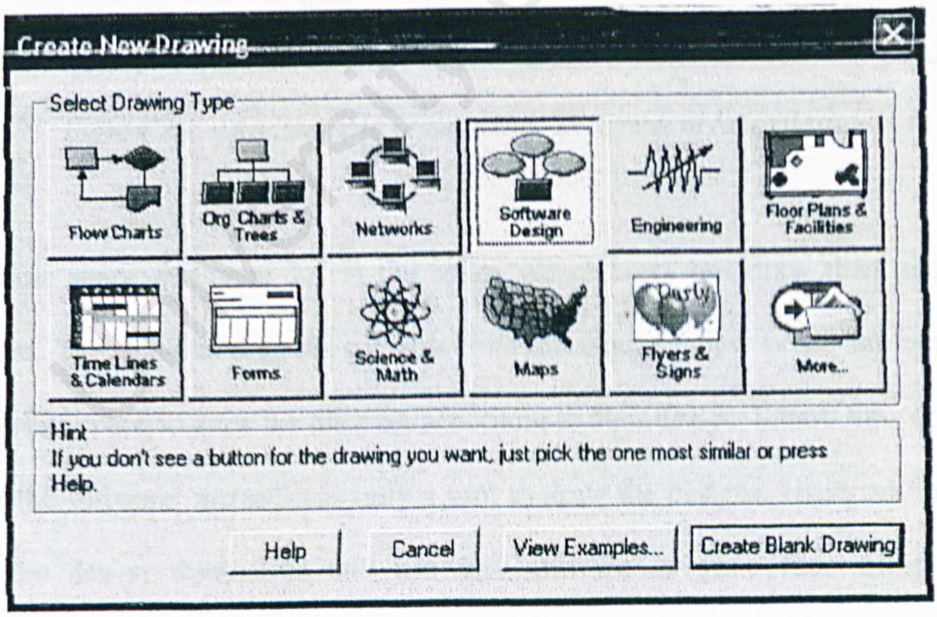
SmartDraw is the easy-to-use software for creating business charts and diagrams such as flow charts, organizational charts, networks, floor plans, timelines, software designs, forms and more. The UML diagrams are included in the software designs option **SmartDraw** can help illustrate a report, analyze a process, make a presentation, persuade others, document procedures, communicate clearly and help others "see what you mean".

SmartDraw helps users look like a graphic professional. No special skills are needed to draw charts and diagrams as the software uses the drag and drop technique. There are over 50 000 built in symbols and clip art images in this software for users to use in their diagrams and charts. Aside from the built in components, they can also import their own symbols and clip arts. This software also has stand-alone step-by-step tutorials and in depth tutorials accessed on-line.

Automatic alignments are provided for neat, crisp drawings. Users can use the templates and examples as reference. Anything done with this software can be printed or saved in GIF, JPG or HTML format. This software can easily convert drawings made in other software. Furthermore, it works hand-in-hand with Microsoft Office which means drawings done in SmartDraw can be copy-pasted into Microsoft Words, Excel, PowerPoint, and etcetera.

2.3.1.1 SmartDraw Interface

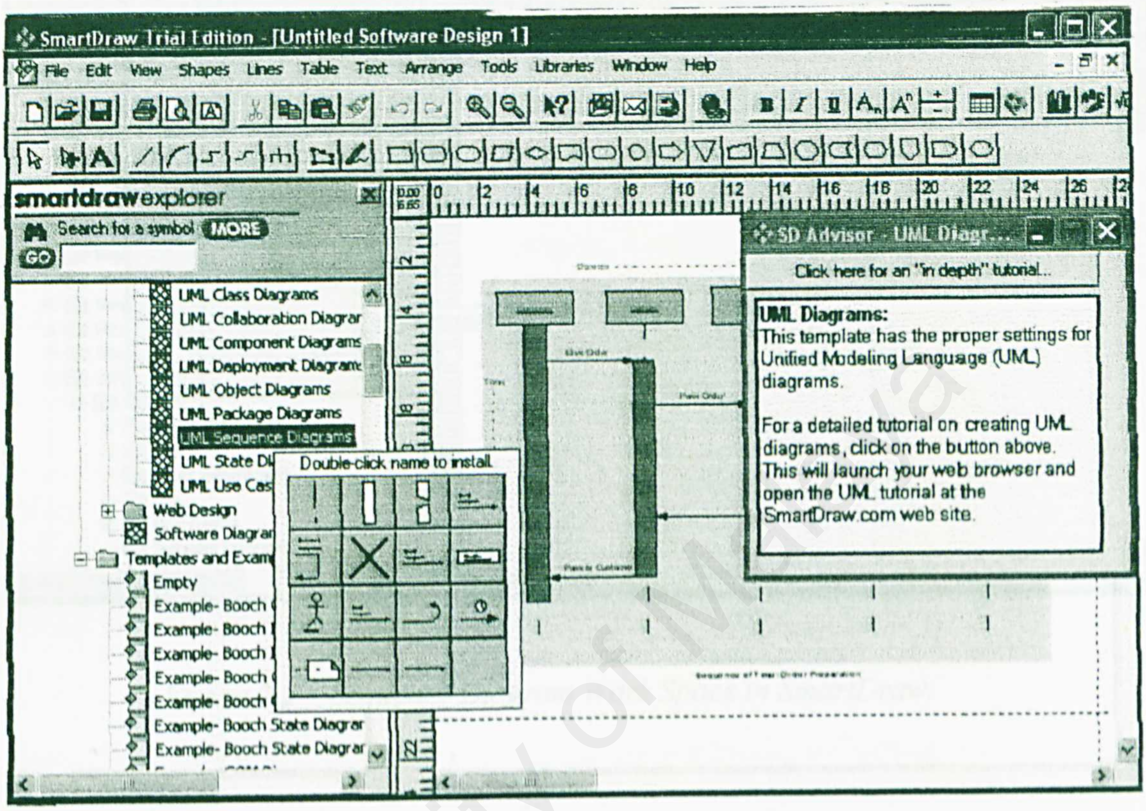
First, users will have to choose whether to open existing diagrams or charts. If they choose to draw a new diagram, they will then have to choose what kind of charts or diagrams they want to draw. The options available are as shown below. To draw a sequence diagram, users will have to choose the software design option.



If you don't see a button for the drawing you want, just pick the one most similar or press Help.

Figure 2.3 - Diagrams and Charts Options in SmartDraw.

The screenshot shown below is what users will get when they choose to see an example of a sequence diagram. The small window at the right hand side will bring users to an in-depth tutorial on-line with one mouse click. The highlighted 'UML Sequence Diagram' at the left hand side is an option for users to draw their own sequence diagram.



UML Diagrams:
This template has the proper settings for Unified Modeling Language (UML) diagrams.

For a detailed tutorial on creating UML diagrams, click on the button above. This will launch your web browser and open the UML tutorial at the SmartDraw.com web site.

Figure 2.4 – An Example of Sequence Diagram in SmartDraw.

The white space in Figure 2.5 is the space where users can draw their sequence diagrams. They have to drag the symbols from the small window at the left hand side to the white space to draw the diagram according to their design. SmartDraw does not design the software; instead it is only a tool to draw the designs. Users will have to make the design themselves and use this software to make their design look professional.

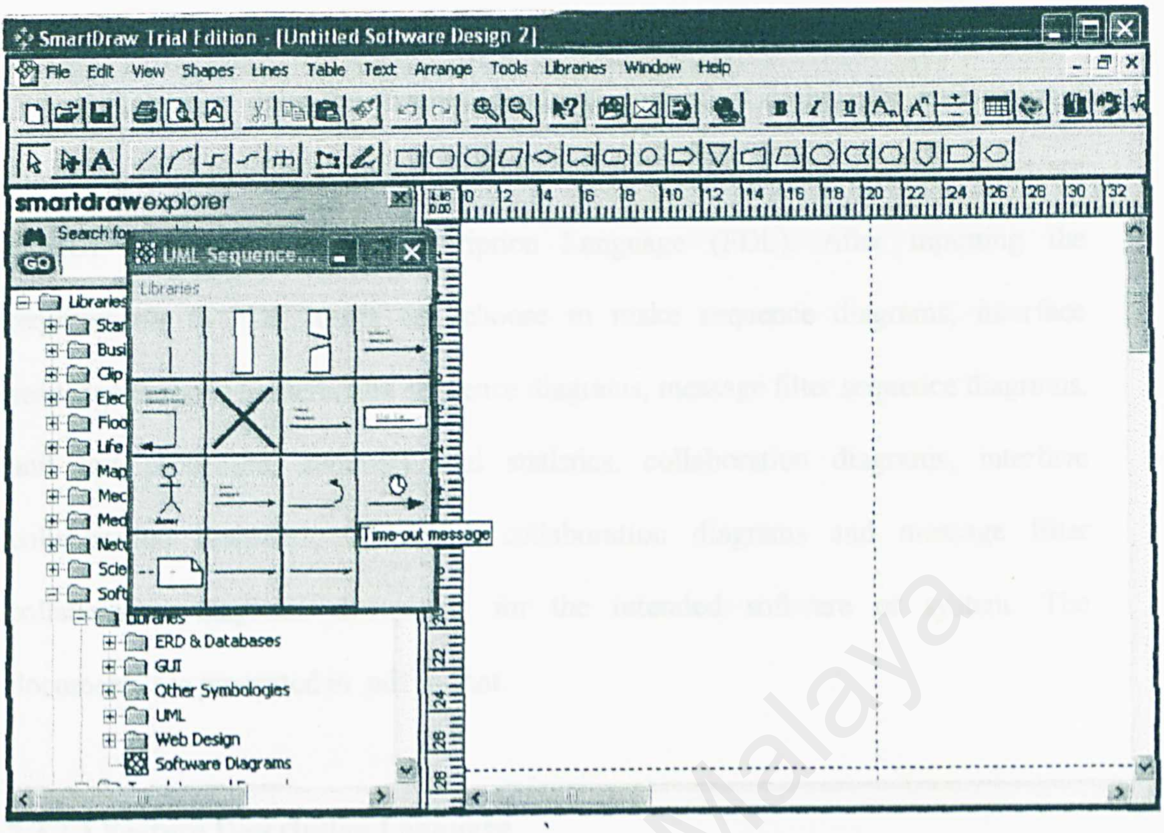


Figure 2.5 – Sequence Diagram Work Space in SmartDraw.

2.3.2 EventStudio

EventStudio is a CASE tool for distributed system design in object oriented as well as structured development environments. EventStudio supports multiple scenario use case and sequence diagram modeling. EventStudio is particularly suited for Message Sequence Charts (MSCs), Real-time and embedded system design, use case development, object sequence diagram development, protocol design and documentation, process flow diagrams, distributed system design, and business process workflows.

Unlike SmartDraw or any other CASE tools, this software does not use the drag and drop technique to draw the required diagrams. Instead, it designs the software and draws the diagrams based on requirements entered by users. These requirements are entered using the Feature Description Language (FDL). After inputting the requirements in FDL, users can choose to make sequence diagrams, interface sequence diagrams, interaction sequence diagrams, message filter sequence diagrams, unit test procedure, summary and statistics, collaboration diagrams, interface collaboration diagrams, interaction collaboration diagrams and message filter collaboration diagrams documents for the intended software or system. The documents are presented in .pdf format.

2.3.2.1 Feature Description Language

The software development involves the design state followed by coding. The output of the design stage cannot be verified for correctness by tools. However, the output of the coding stage can be verified by using compilers and linkers.

This represents a sort of impedance mismatch between the two development processes. Design is largely informal while coding is completely formal. The Feature Description Language (FDL) tries to bridge this gap by introducing a semi-formal development system for design. It tries to incorporate features from both the stages:

- FDL documents allow the user to express the system even when all the details of the system have not been defined.

- FDL documents allow the user to review the correctness of the system design by running an automated review process.

A very simple FDL program is shown below. It shows modules and processors defined in the system. Message interactions between processors are shown enclosed in the feature-endfeature block.

```
1.      module : customer, exchange
2.      processor : phone in customer
2.      processor : frontend in exchange, core in exchange
3.      feature "Call Setup"
4.              offhook : phone -> frontend
                dialtone : frontend -> phone
                digits : phone -> frontend
                setup_call : frontend -> core
                setup_complete : core -> frontend
                ringback : core -> phone
        endfeature
```

1. This program defines the message exchanges between a customer and a telephone exchange. The customer and the exchange have been declared with the module declaration.

2. The processor statements in the next two lines define different entities within the customer and the exchange. Here the customer contains a phone and the

exchange contains a frontend and a core processor. This relationship is specified using the **in** keyword.

3. The feature-endfeature block follows the declarations in FDL. A title for the feature is included in the feature declaration. The feature…endfeature block encloses all the feature interactions between the customer and the exchange.

4. Message interactions have been enclosed within the feature-endfeature block. The first message interaction in the sequence sends an offhook message from the phone to the frontend processor. This is followed by other message interactions involved in call setup. Messages are represented as arrows from the source to the destination.

FDL allows system partitioning into a three level hierarchy. At the highest level are modules. The system consists of modules. Modules contain processors and processors contain eternal and dynamic objects. The selection of modules, processors and object is best explained with examples:

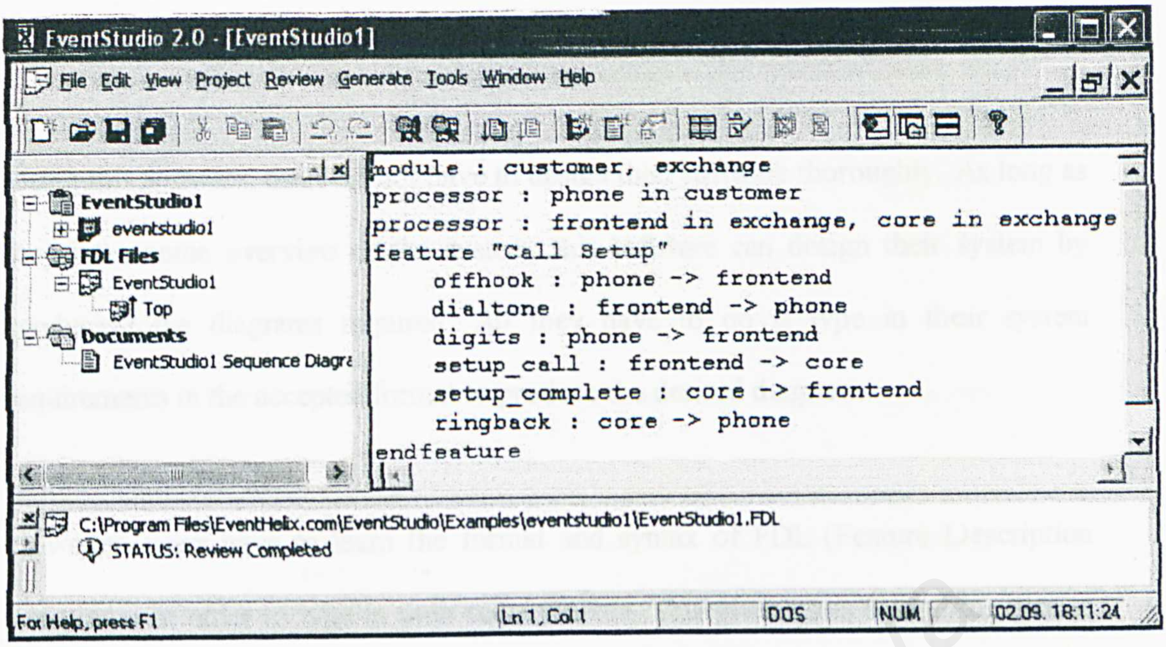
- Acme Inc. Recruiting:
  - Modules are Recruiters, Acme_Inc, Media, Other_Company etc.
  - Processors contained in Acme_Inc are the various departments in the company, e.g. Finance, HR, IT.
  - Objects contained in the HR department are HR_Secretary, Recruitment_Specialist.
- Highway System:
  - Modules are Highways, EntryRamp, TollBooth etc.

- Processors contained in Highway are Cars, Trucks, Motorbikes and etc.
- Object contained in a Car are steering, brakes, engine etc.

2.3.2.2 EventStudio Interface

The interface of EventStudio has some similarities with that of Visual C++. Users enter their requirements in the white space at the right hand side. The left bar are the list of files involved in the scenarios. Scenarios are all files and diagrams associated with the set of requirements. After the requirements are entered, the document must be reviewed. Reviewing is the same as compiling. This is done to check for errors.

If and when the requirements are free of errors, users can choose to produce the diagrams stated earlier. Clicking the new document icon at the toolbar will bring them to a wizard to create the diagrams. The diagrams will be produced in .pdf format and viewed in Adobe Acrobat. The EventStudio interface and the sequence diagram produced in .pdf format are as shown in Figure 2.6 and Figure 2.7.


EventStudio 2.0 - [EventStudio1]
File Edit View Project Review Generate Tools Window Help
EventStudio1
eventstudio1
FDL Files
EventStudio1
Top
Documents
EventStudio1 Sequence Diagra
For Help, press F1
Ln 1, Col 1
DOS
NUM
02.09. 18:11:24

```
module : customer, exchange
processor : phone in customer
processor : frontend in exchange, core in exchange
feature "Call Setup"
    offhook : phone -> frontend
    dialtone : frontend -> phone
    digits : phone -> frontend
    setup_call : frontend -> core
    setup_complete : core -> frontend
    ringback : core -> phone
endfeature
```

C:\Program Files\EventHelix.com\EventStudio\Examples\eventstudio1\EventStudio1.FDL
STATUS: Review Completed

Figure 2.6 – Feature Description Language (FDL)

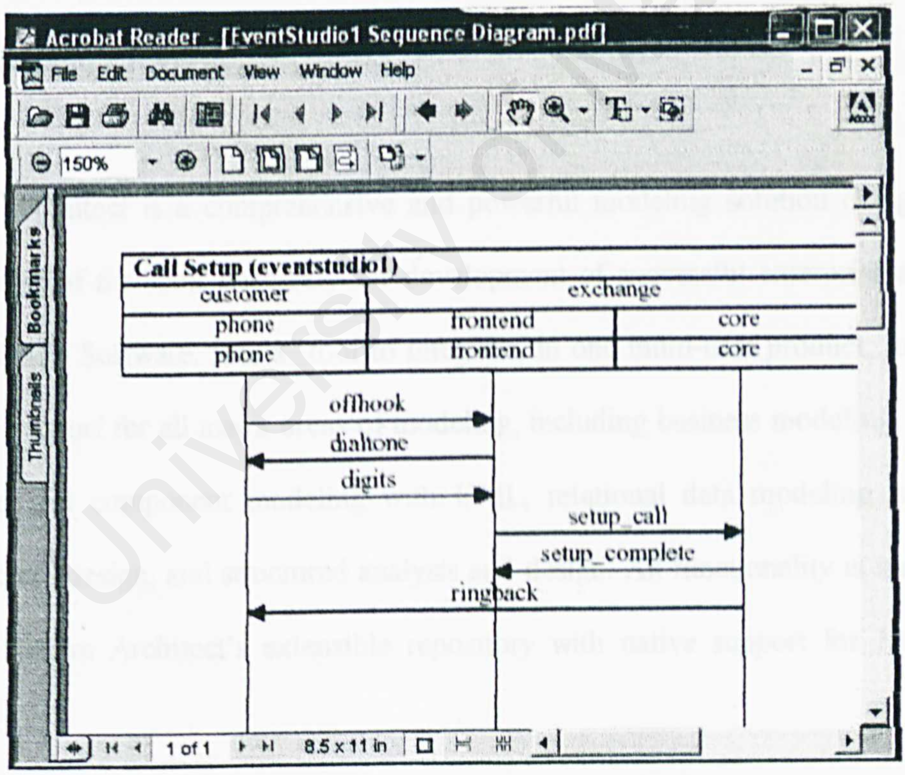


Figure 2.7 – Example of sequence diagram produced by EventStudio.

2.3.2.3 Advantages and Disadvantages of EventStudio

Using this software, users do not have to design their software thoroughly. As long as they have some overview of the system, this software can design their system by producing the diagrams required. All they have to do is type in their system requirements in the accepted format to produce the desired diagrams.

However, users have to learn the format and syntax of FDL (Feature Description Language) in order to type in their requirements. This gives users something else to learn before using this software.

2.3.3 System Architect

System Architect is a comprehensive and powerful modeling solution designed to provide all of the tools necessary for development of successful enterprise systems from Popkin Software. It is a tool to integrate, in one multi-user product, industry-leading support for all major areas of modeling, including business modeling, object-oriented and component modeling with UML, relational data modeling, network architecture design, and structured analysis and design. All functionality is harnessed within System Architect's extensible repository with native support for Microsoft VBA.

System Architect provides extensive support for **UML**, the industry standard for analysis and design of software systems and applications. UML may be used to perform high-level analysis with Use Cases, model of the dynamics of the system with UML Activity, Sequence, Collaboration, and State diagrams, and analyze and design the static structure of the system with UML Class, Component, and Deployment diagrams. System Architect supports code implementation and redesign through automatic generation and reversal of several languages, including Java, Visual Basic, and C++. System Architect also provides unmatched capability to extend its UML support through stereotypes, tagged values, and custom profiles.

System Architect also enables users to extend the UML using relational data modeling techniques and more comprehensive business modeling techniques. System Architect allows users to perform component-based and object-oriented analysis, design, and implementation of the system using the Unified Modeling Language (UML).

System Architect is a repository-based visual modeling tool that supports the following methodologies in a single product:-

- Enterprise Modeling (Strategy and Planning & Business Requirement Capture),
- Data Modeling,
- SSADM,
- Business Enterprise Modeling (Catalyst),
- Business Process Modeling (IDEF), and
- Object and Component Based Design with UML.

Popkin software provides a powerful e-business definition toolset. These tools and services integrate business, systems and data architecture into complete enterprise architecture. It provides comprehensive support for Enterprise Architecture within the ZachmanFramework

Users can use this tool if they need to do the following things for their organization:-

- Develop business goals
- Identify and model current business processes
- Align business processes with business goals
- Evaluate business goals and identify process weaknesses
- Generate new business processes using the criteria above
- Simulate new business processes
- Develop business and IT requirement specifications
- Generate relational database schemas from models
- Develop applications from models
- Re-engineer legacy systems

2.3.3.1 System Architect Interface

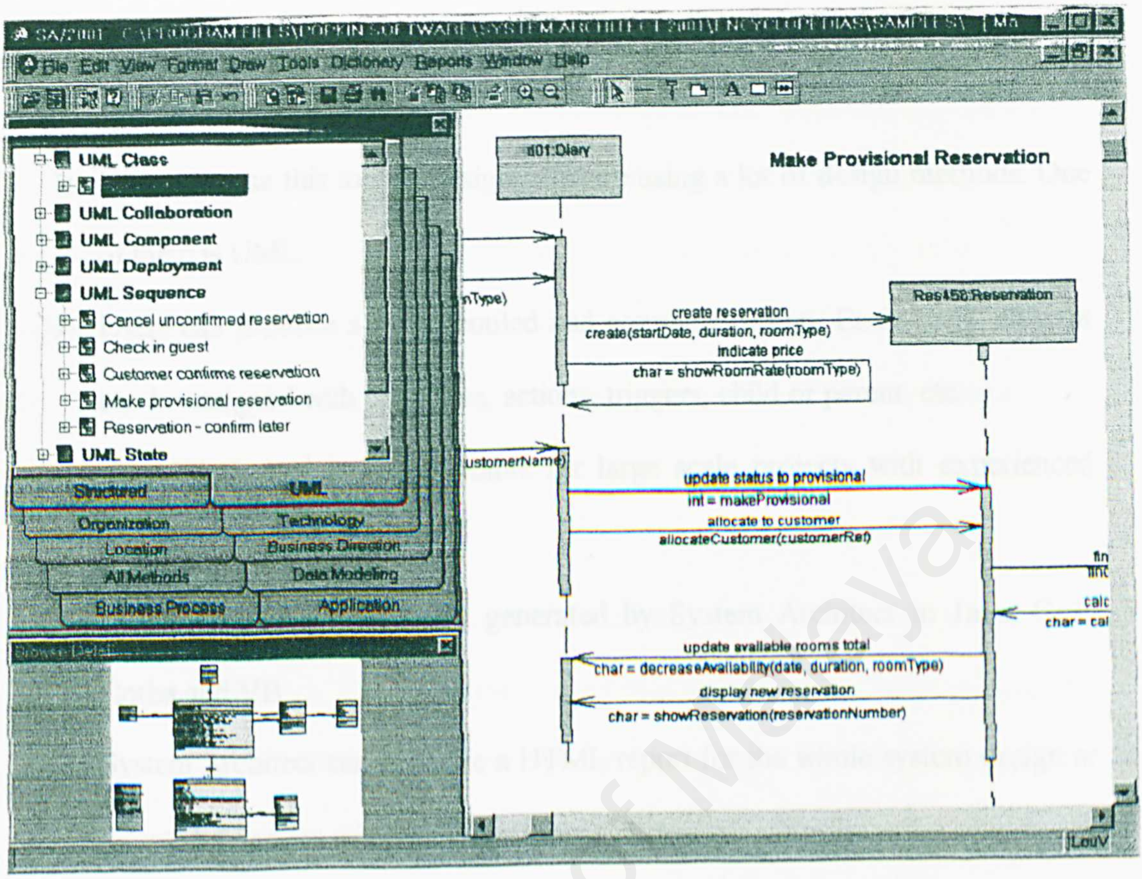

File Edit View Format Draw Tools Dictionary Reports Window Help
UML Class
UML Collaboration
UML Component
UML Deployment
UML Sequence
Cancel unconfirmed reservation
Check in guest
Customer confirms reservation
Make provisional reservation
Reservation - confirm later
UML State
Structured
UML
Organization
Technology
Location
Business Direction
All Methods
Data Modeling
Business Process
Application
d01:Diary
Make Provisional Reservation
Res458:Reservation
create reservation
create(startDate, duration, roomType)
indicate price
char = showRoomRate(roomType)
update status to provisional
int = makeProvisional
allocate to customer
allocateCustomer(customerRef)
update available rooms total
char = decreaseAvailability(date, duration, roomType)
display new reservation
char = showReservation(reservationNumber)

Figure 2.8 – System Architect interface.

Like any other application, System Architect has the menu bar and toolbar. Additionally, the toolbar contains the symbols to used fro diagram drawing. At the left-hand side is the browser to view the various designs such UML, Structured, Data Modeling, Business Direction, and etc.

2.3.3.2 Advantages and Disadvantages of System Architect

Advantages of System Architect:

- Users can use this tool to design software using a lot of design methods. One of them is UML.
- Users can produce a very detailed and complete design. Each UML element can be assigned with behaviors, actions, triggers, child or parent, etc.
- This CASE tool is very suitable for large scale projects with experienced designers.
- Codes can be automatically generated by System Architect in Java, C++, Corba and VB.
- System Architect can generate a HTML report for the whole system design or selected diagrams with just a few mouse click.
- It has built in encyclopedia that can be used as example, reference or starting point.
- It provides an easy to use, easy to follow, and complete lessons and tutorials on designing systems using System Architect.

Disadvantages of System Architect:

- It is a complex application to use. One would have to go through the tutorial and do a lot of practices to gain experience or take a formal class to be able to use this software.
- It is not suitable for beginners in designing as one would have to understand software design fully before taking full advantage of this tool.
- The diagrams, codes, reports and documentations really look professional.

2.3.4 Conclusion

Previously, three CASE tools have been introduced, namely, SmartDraw, EventStudio and System Architect. All three tools have different approaches in designing software using UML especially the sequence diagram. While SmartDraw is only a tool to draw diagrams, System Architect is what peple called as the real designing tool. However, EventStudio takes a totally different approach where it uses Feature Description Language (FDL) to draw diagrams.

In a sense, Tutor on Software Design a little similarities with EventStudio. However, this system simplifies the FDL by using step by step wizard to get the requirements. Furthermore, it uses combo boxes, text boxes, and buttons to ease the way users enter their requirements. Users would not have to learn any new language.

While diagrams produced by EventStudio are not editable, users of Tutor on Software Design can still edit their diagrams using the drag and drop technique as used in SmartDraw and System Architect.

2.4 Application Development Software

There are many kinds of application development software that can be used to develop Tutor on Software Design. Some of them are more suitable for it than others. Therefore, it has to be picked out carefully to avoid late delivery or unfulfilled requirements. Developers should take into consideration some aspects while choosing and deciding on which software to be used, including:

- It should be a software which the developers can learn to use it easily and quickly. For this project, time constraint is certainly the biggest problem and it will be a bigger problem if too much time is spent to study the software to be used. It will also be a problem if the developers does not fully understand the software and only realize it in the middle of the project.
- The selected software should have the feature needed by the system to be developed. For example, if the system requires database access, the selected software should support this feature.
- The selected software should be able to produce output in the format required by the system to be developed. For example, Tutor on Software Design requires the sequence diagrams be saved in .jpg format; does the application development software support this feature?
- In terms of prototyping, the criteria for choosing the right software would be efficiency, rapidity and ease of use.
- The selected software should be interactive.

2.4.1 Visual Basic 6.0

Visual Basic (VB) has been around for a long time--since its early versions some 10 plus years ago. Many improvements over the years have raised VB to the level of a real knock-out contender. Visual Basic 6.0(VB6) is bundled with Visual Studio 6.0, which includes many fine tools for enhancing productivity, both with stand-alone and Web apps. VB6 comes in three flavors: Learning Edition, Enterprise Edition and Professional Edition. Visual Basic 6.0 will work with Win95/98/2000/NT. In the early years of VB, one had to rely heavily on programming experience and it was very code-intensive.

Feature Rich Means Ease of Use

VB6 is packed with new and enhanced features, allowing for quick deployment and access to data using the Microsoft Data Engine (MSDE), freely redistributable as part of the application, allowing full compatibility with large SQL Server databases. There is a new Report Writer, which allows development of very sophisticated, hierarchical reports with drag-and-drop ease. Data access is a snap with VB6 because of the new Data Environment, which automatically allows for data binding. Easily build applications for mobile users and client/server applications on a LAN or Web. There is also support for Microsoft universal data access using ActiveX Data Objects.

Another nice new feature is the enhanced FlexGrid control; this enhancement will allow one to expands, collapse, hide or show various information sets. Many new integrated visual database tools are included, enabling common database activities without leaving the environment. View tables, modify data and create SQL queries for any ODBC or OLE DB-compliant database. Visually design and modify live database schemas and other objects for Microsoft SQL Server 6.5 and Oracle 7.3.3 databases. Language performance has been sped up by as much as 20 times over Visual Basic 4.0. Another great new feature included with VB6 is the "retain-in-memory" option, which keeps component structures cached in memory for server-distributed applications.

2.4.2 Visual Basic.Net

Visual Basic is a hugely popular programming language that is suitable for students and beginners as well as professional development. The .NET version is significantly different from older variants of Basic. Visual Basic .NET is the next version of Visual Basic. Rather than simply adding some new features to Visual Basic 6.0, Microsoft has reengineered the product to make it easier than ever before to write distributed applications such as Web and enterprise n-tier systems. Visual Basic .NET has two new forms packages (Windows Forms and Web Forms); a new version of ADO for accessing disconnected data sources; and streamlined language, removing legacy keywords, improving type safety, and exposing low-level constructs that advanced developers require.

Visual Basic .NET is fully integrated with the other Microsoft Visual Studio .NET languages. Not only can application components be developed in different programming languages, classes can now inherit from classes written in other languages using cross-language inheritance. With the unified debugger, multiple language applications can be debugged, irrespective of whether they are running locally or on remote computers. Whatever language used, the Microsoft .NET Framework provides a rich set of APIs for Microsoft Windows and the Internet.

Two things make Visual Basic .NET Standard 2003 easy to learn and use. One is the language itself, which is designed to be closer to natural English than others. The other is the array of tools and wizards that Microsoft provides, including a visual form designer for both Windows and Web projects. The web technology is called ASP.NET, and enables easy creation of web page that query and update databases, although note that a Windows web server running .NET is required. The programming environment is slick, with convenient features like docking and tabbed windows, project wizards, auto-completion and pop-up help in the code editor. The .NET version of Visual Basic benefits from full object-orientation and a rich class library. It also supports advanced features like multi-threading, which is a way of writing code to do background tasks.

Whereas Visual Basic 6.0 and earlier version needed a small runtime library, this .NET edition requires the .NET Framework, a runtime engine and class library that manages memory and enforces security. Framework applications perform well, since they are compiled to native code at runtime, but there is an overhead in terms of memory usage and the Framework runtime must be installed. These factors, together with less than perfect code compatibility, have made some Visual Basic developers reluctant to switch. While that's understandable, the .NET technology is now maturing. It is still important to note the heavy system requirements, and that .NET applications do not run on Windows 95.

Visual Basic .NET provides the easiest, most productive language and tool for rapidly building applications for Microsoft Windows and the Web. Ideal for existing Visual

Basic developers as well as new developers in the Microsoft .NET development environment, Visual Basic .NET 2003 delivers enhanced visual designers, increased application performance, and a powerful integrated development environment (IDE).

2.4.3 Conclusion

Discussed above are two choices of programming language that can be used as application development software. Each one of them has advantages and disadvantages. The Visual Basic.Net is chosen as the application development software for developing Tutor on Software Design. Besides all the extra features stated in the above section, Visual Basic.Net has a new and easier way of implementing the drag and drop technique. This technique will be used for users to edit the produced diagrams.

2.5 Database

Tutor on Software Design receives data from users. Therefore, it should save the data in an organized way. The best way to organize and keep track of information is using the database.

Database is an organized collection of data. A database management system (DBMS) such as Access, FileMaker Pro, Oracle or SQL Server provides the software tools to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents.

The DBMS to be used for this system is Microsoft Access. Other aspects of concern are data provider and how to access the database file programmatically from VB.Net (the selected development tool). Data provider could be SQL server or OLEDB, meanwhile ADO can be used to access database.

2.5.1 Microsoft Access

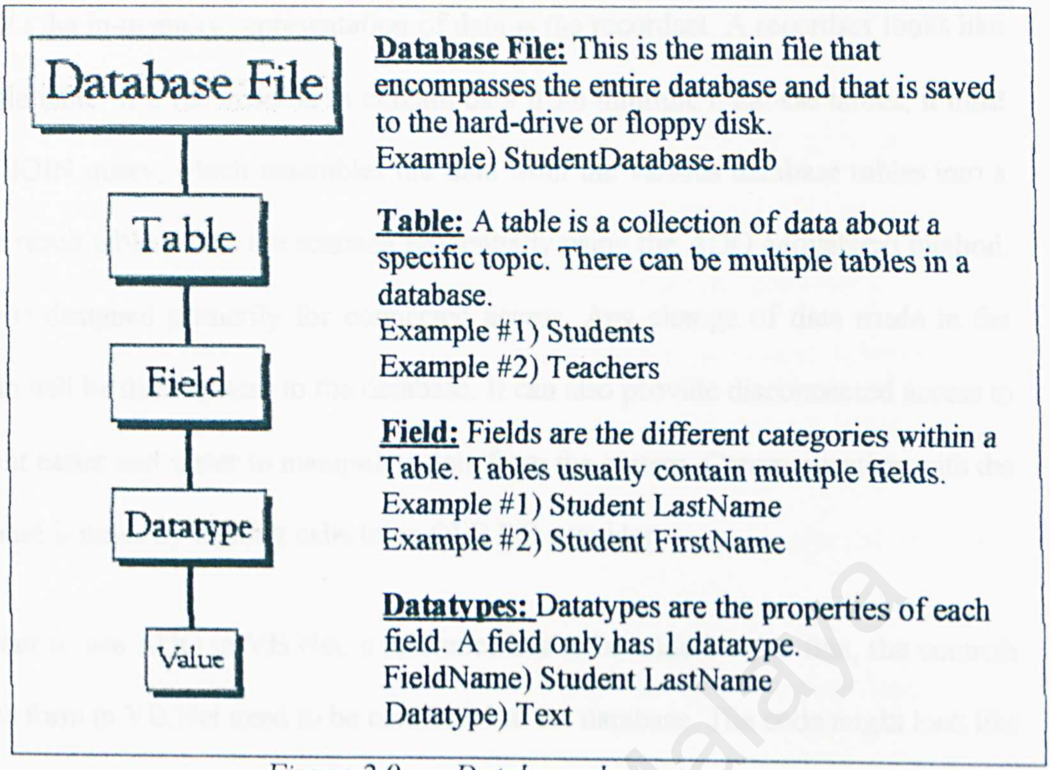
Microsoft Access is a computer application used to create and manage computer-based databases on desktop computers and/or on connected computers (a network). Microsoft Access can be used for personal information management (PIM), in a small business to organize and manage all data, or in an enterprise to communicate with servers.

Microsoft Access provides users with one of the simplest and most flexible DBMS solutions on the market today. Regular users of Microsoft products will enjoy the familiar Windows "look and feel" as well as the tight integration with other Microsoft Office family products. An abundance of wizards lessen the complexity of administrative tasks and the ever-present Microsoft Office Helper is available for those who care to use it.

Microsoft generally likes to incorporate as many features as possible into its products. For example, the Access package contains the following elements:

- a **relational database system** that supports two industry standard query languages: Structured Query Language (SQL) and Query By Example (QBE);
- a full-featured **procedural programming language**— essentially a subset of Visual Basic,
- a simplified procedural **macro language** unique to Access;
- a **rapid application development environment** complete with visual form and report development tools;
- a sprinkling of **objected-oriented extensions**; and,
- Various **wizards and builders** to make development easier.

There are several keywords that are commonly used when creating and maintaining a database such as *table*, *fields*, *value* and *datatype*. The relationship between these keywords can be illustrated as the following figure 2.9.



Database File: This is the main file that encompasses the entire database and that is saved to the hard-drive or floppy disk.
Example) StudentDatabase.mdb

Table: A table is a collection of data about a specific topic. There can be multiple tables in a database.
Example #1) Students
Example #2) Teachers

Field: Fields are the different categories within a Table. Tables usually contain multiple fields.
Example #1) Student LastName
Example #2) Student FirstName

Datatypes: Datatypes are the properties of each field. A field only has 1 datatype.
FieldName) Student LastName
Datatype) Text

Figure 2.9 Database elements

Other datatypes are number, autonumber, date, time, etc.

2.5.2 ADO

ADO is a set of ActiveX controls that provide programmatic access to Microsoft's latest underlying data access technologies. ADO is based on OLEDB. This is a defined set of interfaces that all data sources can implement through special drivers (or providers).

In other words, ADO gives a standard way of managing data from all kinds of data stores, not just relational databases. The ever-increasing role and importance of the Internet in application development has also driven the design concepts of ADO. It provides a range of ways that remote data access can be achieved over the Internet, using a Web browser.

In ADO the in-memory representation of data is the recordset. A recordset looks like a single table. If a recordset is to contain data from multiple database tables, it must use a JOIN query, which assembles the data from the various database tables into a single result table. Rows are scanned sequentially using the ADO MoveNext method. ADO is designed primarily for connected access. Any change of data made in the system will be directly sent to the database. It can also provide disconnected access to make it easier and faster to manipulate data from the system. Communication with the database is made by making calls to an OLE DB provider.

In order to use ADO in VB.Net, a reference has to be added. After that, the controls on the form in VB.Net need to be connected to the database. The code might look like this:

```
Dim conn As String="PROVIDER=SQLOLEDB;INITIAL CATALOG=Northwind;"& _
                     "SERVER=localhost;UID=sa;PWD=;"
Dim cmd as String = "select CustomerId from Customers"
Dim adoRS as New ADODB.Recordset()

' Open the RecordSet and retrieve the Customer IDs
adoRS.Open(cmd, conn, ADODB.CursorTypeEnum.adOpenForwardOnly, _
            ADODB.LockTypeEnum.adLockReadOnly, 0)

' Loop through the RecordSet and add the IDs to the ComboBox
Do While Not adoRS.EOF
   cboCustomerID.Items.Add(adoRS.Fields("CustomerID").Value)
   adoRS.MoveNext()
Loop

' Must close the Recordset
adoRS.Close()
```

2.5.3 OleDB

OLEDB is a COM-based data access object which provides access to all types of data, and even provides access to disconnected data stores (for example, if you're on your laptop, you can easily view a snapshot of the database from the last time you synced up). It is a data provider which is the mechanism that connects us to the physical data store while ADO is a data consumer because it uses data provided by OLEDB.

OLE DB interfaces provide applications with uniform access to data stored in diverse information sources, or data stores. These interfaces support the amount of DBMS functionality appropriate to the data store, enabling the data store to share its data.

2.5.4 Conclusion

The aspects of concern in managing data for Tutor on Software designed are DBMS, data provider and method of manipulating data between the system and the DBMS. The solutions have been discussed above.

The DBMS to be used is Microsoft Access while the data provider is OleDB. ADO will be used to manipulate data in the database from the system.

Chapter 3

Methodologies

3.1 Introduction to Methodologies

Methodologies may be defined as a collection of procedures, techniques, tools and documents aids. They help the developers to speed up and simplify the development process (P. Sellapan, 2000). Thus, it is important to choose a suitable methodology while developing the system.

3.2 Methodology Analysis and Consideration

Software development involves a lot of tasks. These tasks must be managed systematically to ensure that the project is successful. A software project is said to be successful if the system is delivered on time, within budget, and fulfills the requirements.

Software development methodologies are used to get the right system requirements and provide a systematic way of developing systems. Methodologies allow developers to monitor the progress of development. Documentation can be produced and maintained easily. Methodologies are important so that developers can detect changes as early as possible along the system development lifecycle.

Three methodologies will be discussed and considered in this section. They are the Waterfall Model, the Prototype Model, and the Waterfall with Prototyping Model. In the end, the most suitable methodology will be chosen for this project.

3.2.1 Waterfall Model

Waterfall model is one of the oldest model, most of the new ones are the modified versions of this model. The waterfall model is shown in Figure 3.1.

According to this model, one process should be completed before the next one begins, which means the developers should develop the system step by step by following the sequence. Thus, when all of the requirements are elicited from customers, analyzed for completeness and consistency, and documented in a requirements document, the development team can go on to the system design activities. This model enables the developers to view what is going on during the development process.

The waterfall model has been used to prescribe software development activities in various contexts. For instance, it was the basis for software development deliverables in U.S Department of Defense contracts for many years, defined in Department of Defense Standard 2167-A (Pfleeger, 2000).

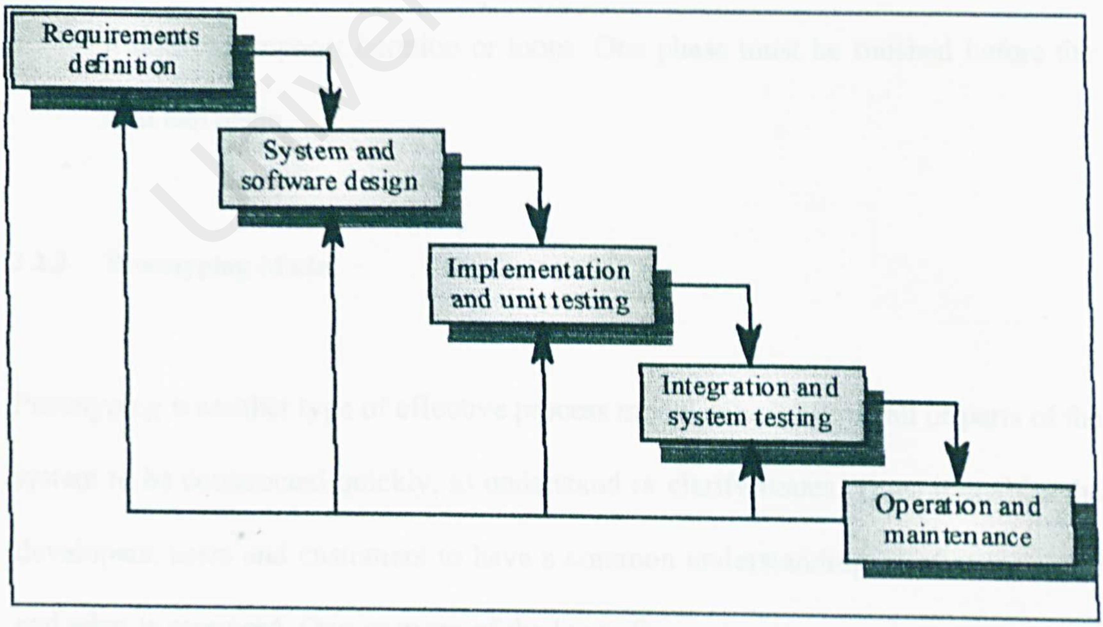

Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 3.1 – Waterfall model.

3.2.1.1 Advantages and Disadvantages of Waterfall Model

Advantages of using the waterfall model:

- It makes the explanation to the customer or a person who is not familiar with software development easy and convenient.
- This model can increase the confidence of the software developers during the development process.
- Most of the later models are built based on this model.

Disadvantages of using the waterfall model:

- It does not show how the basic coding is designed or created unless the requirements are fully understood.
- It does not have any reference when any sudden changes happen to the products or activities.
- Fails to perceive the software as a problem solving process because the waterfall model is actually modified from hardware development process.
- It does not support iteration or loops. One phase must be finished before the next can begin.

3.2.2 Prototyping Model

Prototyping is another type of effective process model, which allows all or parts of the system to be constructed quickly, to understand or clarify issues. Thus, it enables the developers, users and customers to have a common understanding of what is needed and what is proposed. One or more of the loops for prototyping requirements, design, or system may be eliminated, depending on the goals of the prototyping. However, the overall goal is to reduce risks and uncertainty in development. The prototyping model is shown in Figure 3.2.

3.2.2.1 Advantages and Disadvantages of the Prototyping Model

The advantage of using the prototyping model:

- Reduce risks and uncertainty in the system development.

The disadvantages of using the prototyping model:

- The quality of the software may be ignored when the product is produced within a limited time. Thus, more time will be needed for maintenance.
- The developers may develop the system within unsuitable platform or programs.

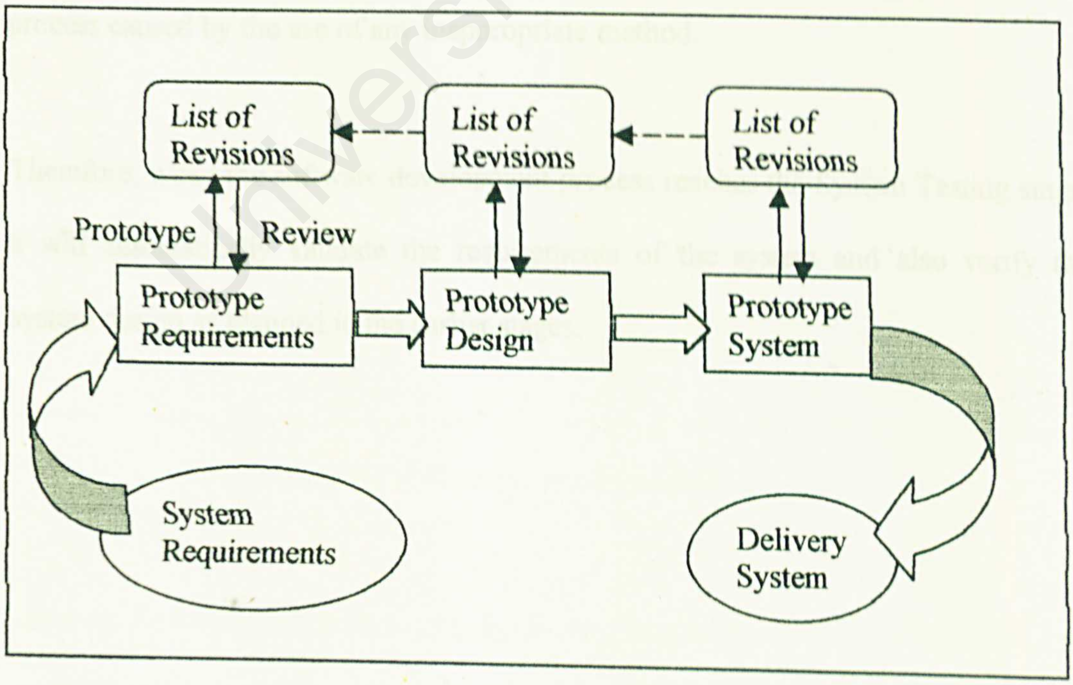


Figure 3.2 – Prototyping model.

3.2.2.2 How Does the Prototype Resolve the Problem of Traditional Waterfall Model?

The uniqueness of this model is that it allows the developers to quickly create a prototype to verify the needs and target of the particular process, thus enables revision to be made at the requirements stage rather than the more costly testing stage. This is because a software development process may involve a lot of iteration processes. The developers sometimes have to use the 'try and error' method to get the best result and if the result is not feasible or failed to hit the target, they will have to start all over from the beginning of the process again and again.

Thus, the prototyping stage is used to examine some aspects of the proposed system in the first few stages. This model offers a systematic way to manage the development process and avoids the use of inappropriate method for the system development. By verifying these stages, it will effectively reduce the possibilities of repeating any process caused by the use of any inappropriate method.

Therefore, when the software development process reaches the System Testing stage, it will automatically validate the requirements of the system and also verify the system design as planned in the earlier stages.

3.2.3 Waterfall Model with Prototyping

Both waterfall model and prototyping model have an extremely high potential to be merged to obtain the best process model needed to meet this project's needs. This is because they can improve the quality of the development process management by reducing the disadvantages of each other.

The model consists of several phases such as system requirements analysis, system design, implementation and unit testing, system integration and testing, and operation and maintenance. Each stage is well defined before the next stage begins. As shown in Figure 3.3, each process has to be finished before the following process starts.

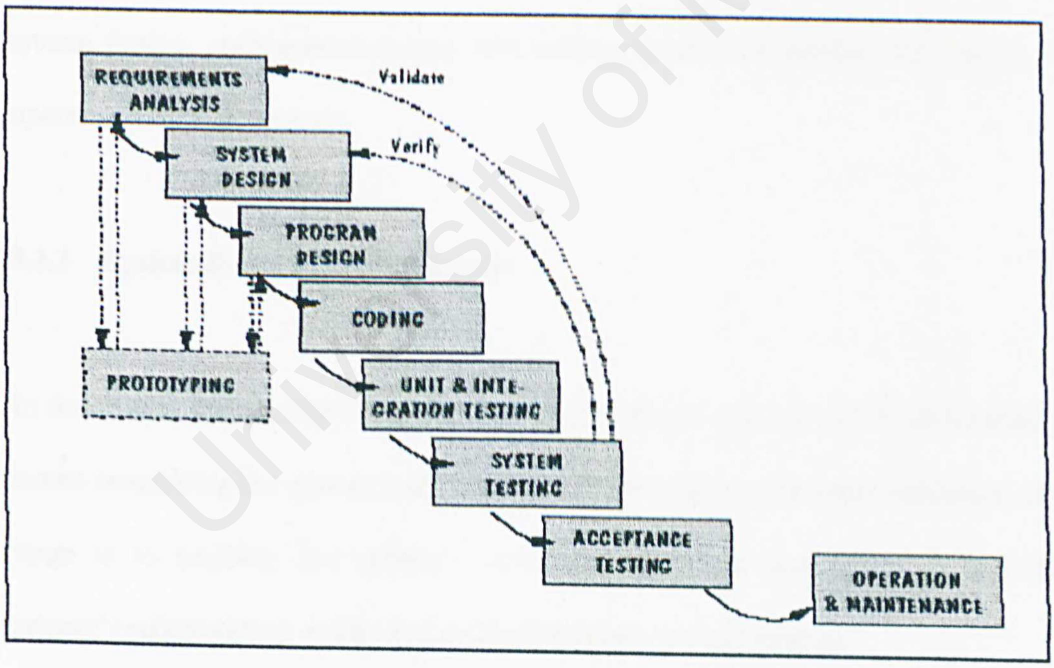


Figure 3.3 – Waterfall model with prototyping.

Tutor on Software Design is a complex system as it contains different modules built by different person. The prototyping is incorporated into the waterfall model because testing of the functionality of its modules before the development process gets to the implementation stage is vital. The usage of prototype will also allow potential users to test the system and necessary modifications can be made before it is implemented. Another reason why the waterfall model with prototyping is used is it offers a mean of making the development process more visible compared to other models. Through this model, every detail about requirements and functions can be known in advance before the system is developed and these details remain stable throughout the development process.

3.3 Focus on Waterfall Model with Prototyping Stages

The five stages in waterfall model with prototyping are system requirements analysis, system design, implementation and unit testing, system integration and testing, and operation and maintenance.

3.3.1 System Requirements Analysis

In this stage, studies and researches of the system are carried out to understand the issues concerning the system and the nature of the system. The main objective of this stage is to establish the system's services, constraints and goals. In this stage, project's requirements, needs and constraints have to be identified.

3.3.2 System Design

At this stage, the overall system architecture is established. System modules are determined from the architecture. Requirements determined in the previous stage are partitioned into hardware and software requirements. System functions are also depicted. In this stage, preparation of various diagrams of system modules that logically represent the system to be developed is done.

3.3.3 Implementation and Unit Testing

In this stage, every module of the system has to be constructed using selected programming language. Each function will then be tested to verify that it is working according to its specifications. During this stage, various bugs shall be eliminated.

3.3.4 System Integration and Testing

Each module of the system developed separately will be integrated and tested as a whole system. The main objective of testing is to make sure that the system meets the user's requirements and therefore, ensures the usefulness of the system being developed. During this stage, the bugs will be encountered and the problems concerning the interface between modules may arise. Enhancement will also be made to improve the quality of the system.

3.3.5 Operation and Maintenance

The system will be developed and installed in this stage. Maintenance of the system includes fixing bugs that are discovered, will have to be carried out. Maintenance is crucial to ensure that the system remains useful.

Chapter 4

Analysis and Design

4.1 Introduction to Analysis and Design

System requirements analysis phase, as discussed in the previous chapter, is the earliest phase in the system development lifecycle. The purpose of this phase is to get knowledge and clarification regarding critical aspects to system development. Analysis activities require thorough investigation of the system including from the aspects of users, task analysis, and requirements specification. This analysis is vital to ensure that the system do and support requirements and existing policies. To achieve this, system requirements specification covers the functional, non-functional, hardware and software requirements.

System design on the other hand, covers all tasks and functions that give priority to detailed and in-depth specification based on computer-oriented problem solving. It leads to data, interface component, and process oriented technical and implementation aspects of the system. Amongst the popular approaches are structured and object-oriented approaches.

4.2 System Requirements Specification

A requirement is a statement of a system service or constraint (Kotonya and Sommerville, 1998). A service statement describes how the system should behave with regard to an individual user or with regard to the whole user community. In the latter case, a service statement really defines a business rule that must be obeyed at all times. For example, fortnightly salaries are paid on Wednesdays. A service statement may also be some computation that system must carry out. For example, calculate salesperson's commission based on the sales in the last fortnight using a particular formula.

Getting the accurate requirements requires an iterative process and a lot of user involvement. Accurate requirements are very important to determine system capabilities. Meanwhile, design determines how those requirements should be implemented. Requirements specification to be discussed here are the functional, non-functional, hardware and software requirements.

4.2.1 Functional Requirements

A functional requirement describes interactions between the system and its environment (Pfleeger, 2001). The functional requirements that I will develop involve three modules, namely, Sequence Diagram, User Requirements, and Help.

Module	Functional Requirements
User Requirements	1. Requirements entered by users using drop-down buttons (combo box) and text boxes. 2. Requirements captured element by element (actor → use case → association → class → object → sequence). 3. Users can always return to previous tasks using the back buttons. 4. Users can decide names for actors, use cases, classes and objects. 5. Users prompted deleting any object or sequence.
Sequence Diagram	1. Get elements needed from the User Requirements modules. 2. Produce a sequence diagram based on those elements. 3. Diagrams produced should be editable by users. Users can change the diagram output by reentering their requirements. 4. The final diagrams can be saved in various image format. 5. The system produces a professional looking diagram.
Help	1. Contains information and instructions on using the system. 2. Provide tutorials on sequence diagram. 3. Provide tips to get a good sequence diagram.

Table 4.1 – Functional requirements

4.2.2 Non-Functional Requirements

A non-functional requirement is a description of the features, characteristics, and attributes of the system as well as any constraints that may limit the boundaries of the proposed solution. The followings are the non-functional requirements identified in developing Tutor on Software Design.

Reliability

The system to be developed must be able to perform required functions and tasks correctly. The diagrams produced by the system must be accurate and drawn exactly according to the requirements entered by users.

Interactivity

Tutor on Software Design requires a lot of inputs from users. Users can choose which diagram to design their software with or all of them. They can go to previous tasks easily. Users are prompted for error corrections or confirmations whenever needed. Help and tutorial are available and can be accessed anytime.

4.2.3 Hardware and Software Requirements

Choosing the right hardware and software to be used for system development is very important to ensure the system succeeds. It is also vital as it can ensure that the system fulfills the system objectives. The task of choosing hardware and software is carefully done to make sure the system can fulfill the system requirements specification. The following tables shows the hardware and software requirements for developing this system.

Hardware	**Requirements**
Monitor	Sunbosch 15"
Processor	Intel Pentium III 434 MHz
Memory	128 MB
Hard disk	100 MB free space
Input device	Keyboard and mouse
Printer	Canon BJC-1000sp

Table 4.2 – Hardware requirements

Software	**Requirements**
Operating system	Windows 98/2000/Me/NT/XP
Database management system	Microsoft Access 2000
Programming language	Visual Basic.Net
Project documentation	Microsoft Word 2000

Table 4.3 – Software requirements

4.3 Application Architecture Design

An application architecture defines the technologies to be used by (and used to build) one, more or all information system in terms of its data, processes, interfaces and network components. Thus designing the application architecture involves considering network technologies and making decisions on how the systems' DATA, PROCESSES, and INTERFACES are to be distributed among the business locations.

4.3.1 System Development Structure Chart

Structure chart illustrates a top-down hierarchy of software modules that would conform to accepted principles of good software design. A module is a group of instructions – a paragraph, block, subprogram, and subroutine.

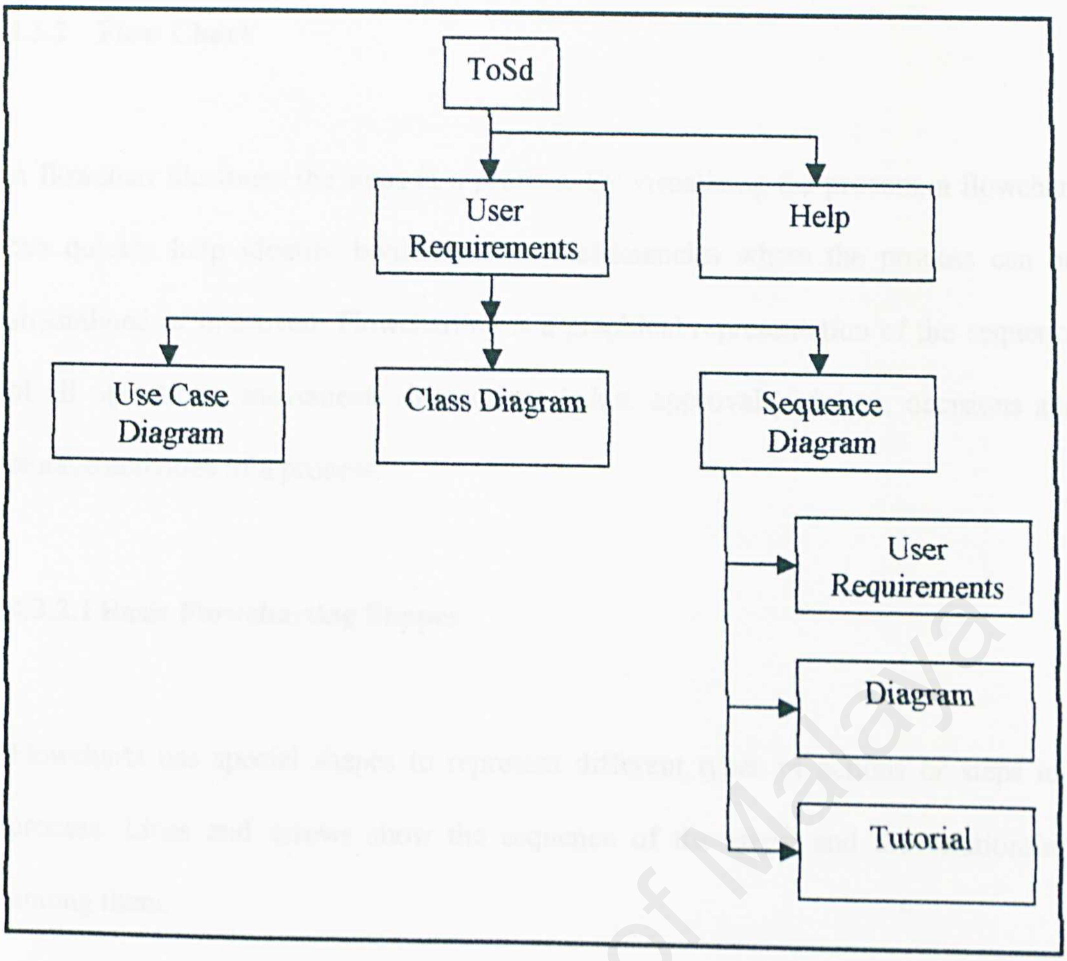


Figure 4.1 – Structure Chart

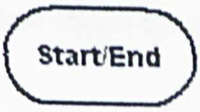
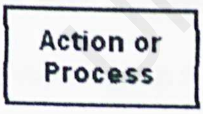
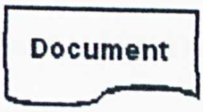
Tutor on Software Design are partitioned into five modules with User Requirements and Help at the top hierarchy and the three diagrams under User Requirements. The Sequence Diagram module has three submodules, namely, User Requirements, Diagram and Tutorial.

4.3.2 Flow Chart

A flowchart illustrates the steps in a process. By visualizing the process, a flowchart can quickly help identify bottlenecks or inefficiencies where the process can be streamlined or improved. Flowcharting is a graphical representation of the sequence of all operations, movements, inspections (a.k.a. approvals), delays, decisions and storage activities of a process.

4.3.2.1 Basic Flowcharting Shapes

Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them.

	The terminator symbol marks the starting or ending point of the system. It usually contains the word "Start" or "End."
	A box can represent a single step ("add two cups of flour"), or and entire sub-process ("make bread") within a larger process.
	A printed document or report.

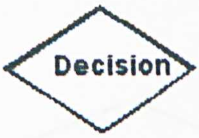
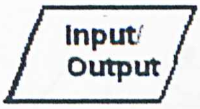
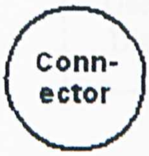
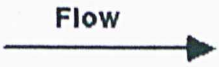
	A decision or branching point. Lines representing different decisions emerge from different points of the diamond.
	Represents material or information entering or leaving the system, such as customer order (input) or a product (output).
	Indicates that the flow continues on another page, where a matching symbol (containing the same letter) has been placed.
	Lines indicate the sequence of steps and the direction of flow.

Table 4.4 – Flow Chart's symbols and shapes

4.3.2.2 Flow Chart for Tutor on Software Design

As shown in Figure 4.2, the system starts by getting the general requirements which is used by all three diagrams including actor, use case, and association. Next, user can choose which diagram to draw. If the user chooses to design with sequence diagram, user will have to input the sequence requirements including class, object and sequence of events. Then, the user can save and print the diagrams. Users can choose to draw another diagram based on the requirements already entered, or new requirements or end the system.

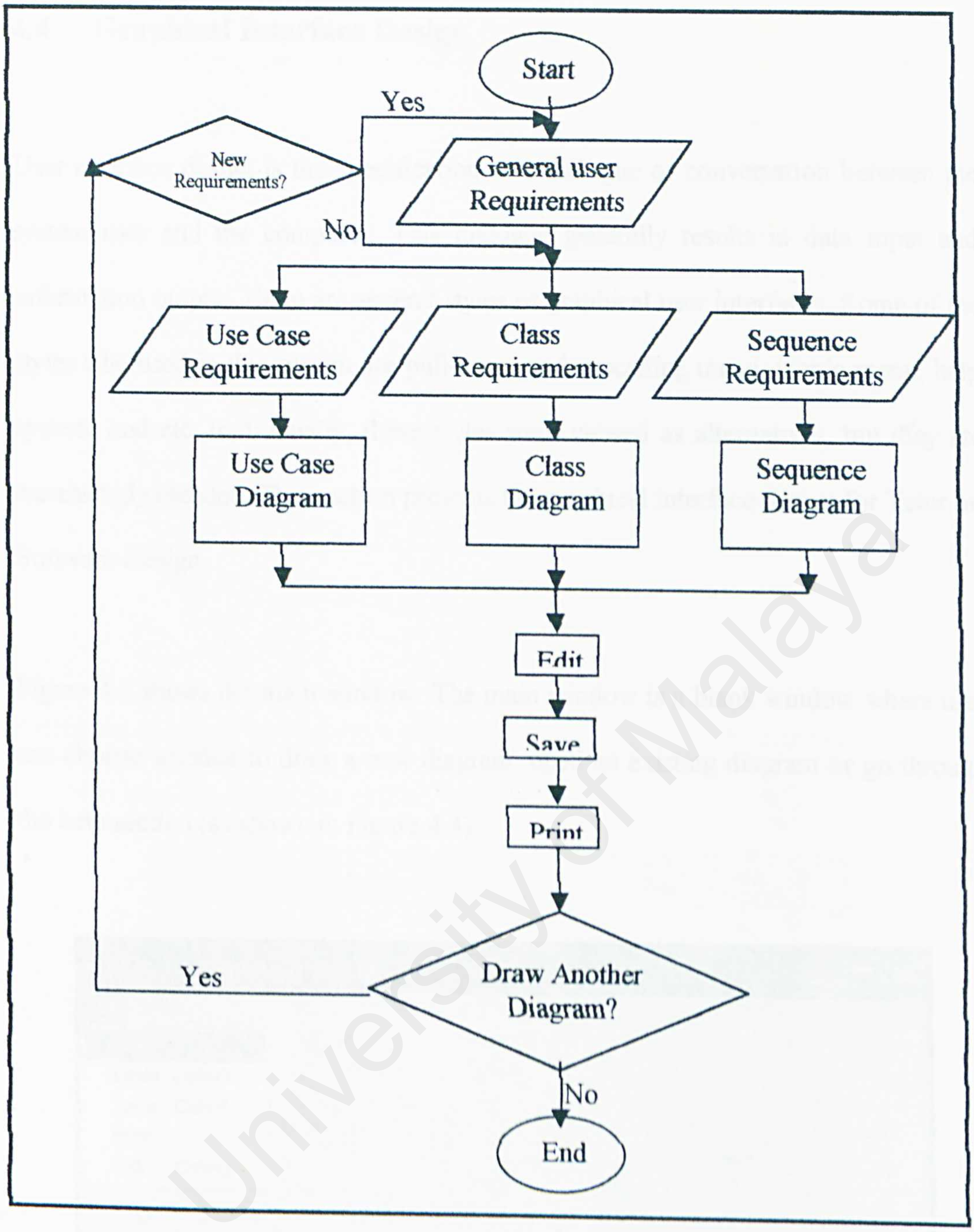


Figure 4.2 – Flow Chart

4.4 Graphical Interface Design

User interface design is the specification of a dialogue or conversation between the system user and the computer. This dialogue generally results in data input and information output. There are several styles of graphical user interfaces. Some of the styles t be used in this system are pull-down and cascading menu, iconic menu, help system, and etc. traditionally, these styles were viewed as alternatives, but they are increasingly blended. This section presents the graphical interface design for Tutor on Software Design.

Figure 4.3 shows the main window. The main window is a blank window where user can choose whether to draw a new diagram, open an existing diagram or go through the help section (as shown in Figure 4.4).

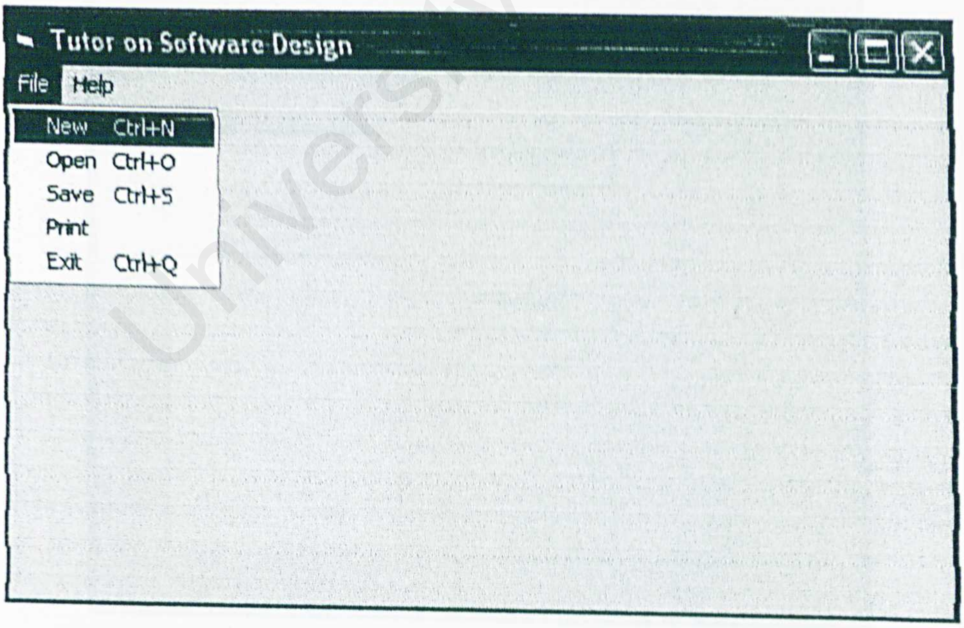


Figure 4.3 – Main window with File menu.

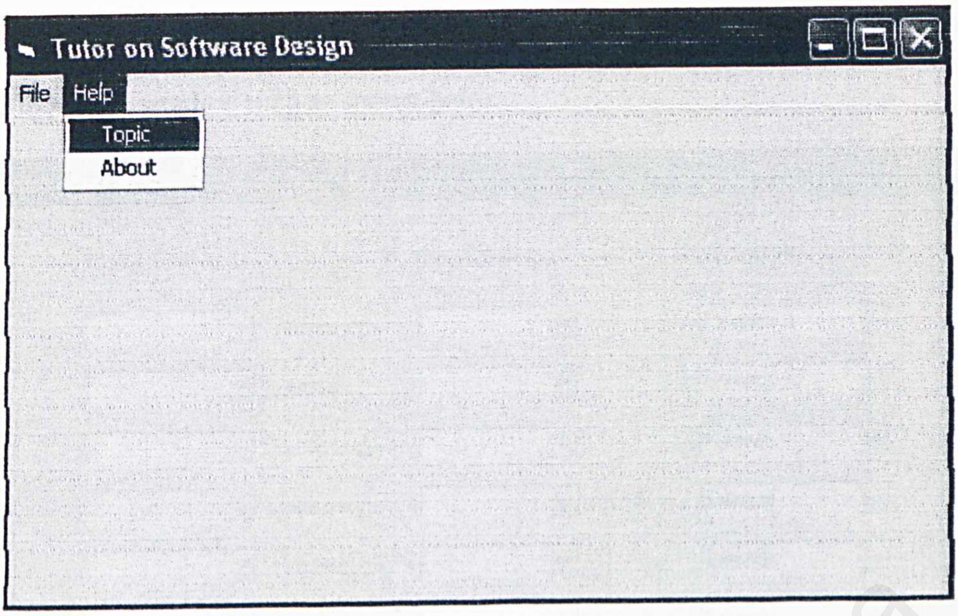


Figure 4.4 – Main window with Help menu.

Actors are specified by users in a window such as in Figure 4.5. Users can decide on the name of actors to be used in their software.

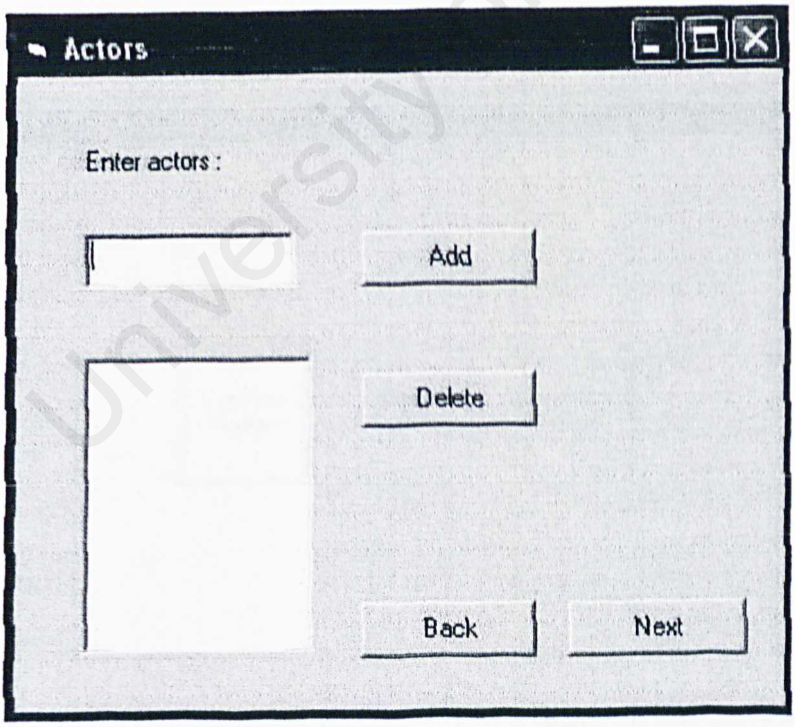
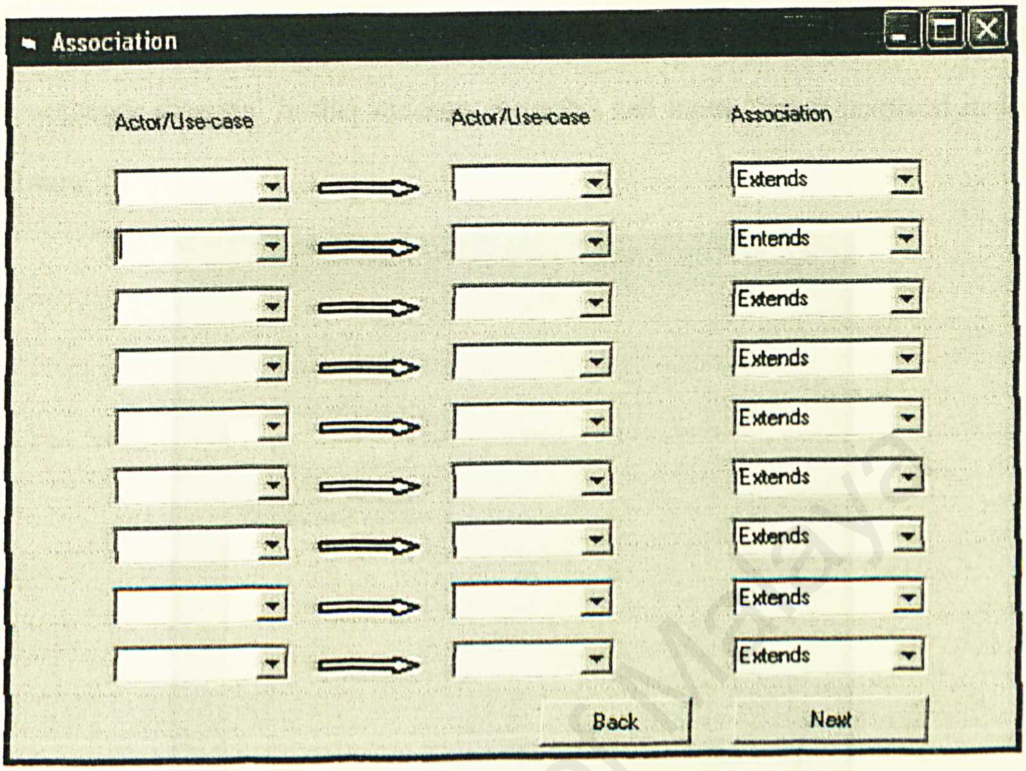


Figure 4.5 – Actors window.

Associations between actors and use cases along with the type of associations can be specified in the window such as shown below.



Extends
Entends
Extends
Extends
Extends
Extends
Extends
Extends
Extends

Figure 4.6 – Association window.

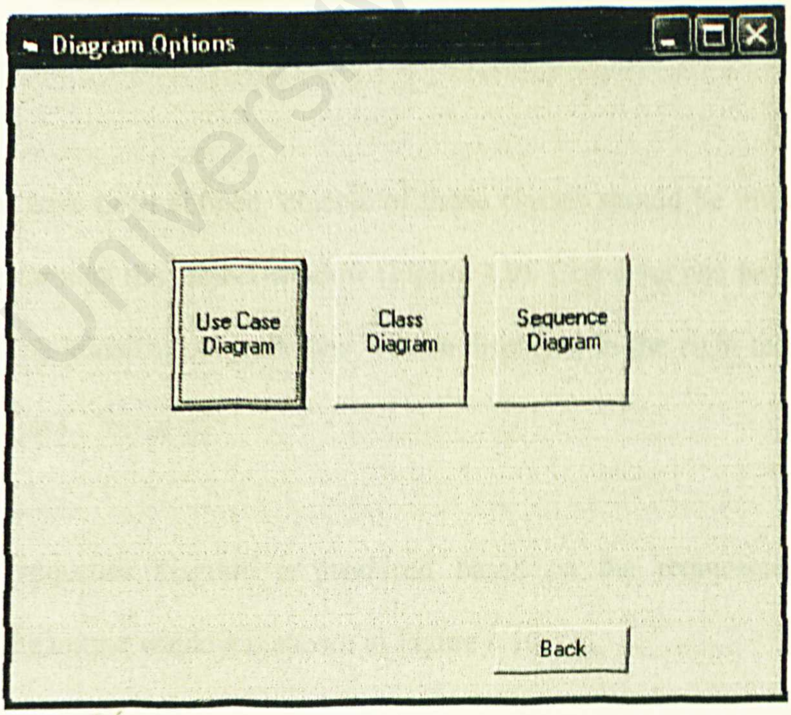


Figure 4.7 – Diagram Options window.

Users then have to choose which diagram to choose, whether the use case, class or sequence diagram by clicking one of the buttons in the Diagram Options window (Figure 4.7). The Class window (Figure 4.8) will appear if the user chooses to draw the sequence diagram. In this window, the users can input classes involved in their software.

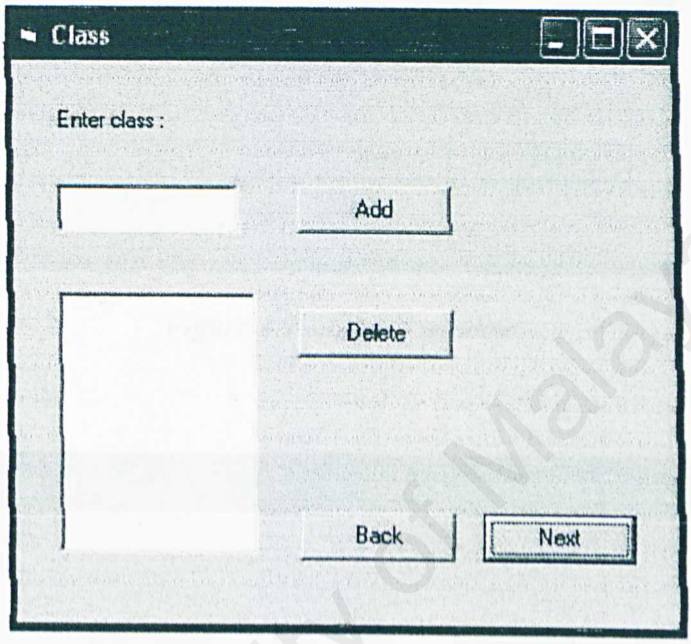


Figure 4.8 – Class window.

After classes have been defined, objects of those classes should be initialized. This can be done through the Object window (Figure 4.9). One class can have more than one object. The initialized objects then will be displayed in the right-most box with the syntax `class: object`.

Finally, the sequence diagram is produced based on the requirements entered previously. The output window is shown in Figure 4.10.

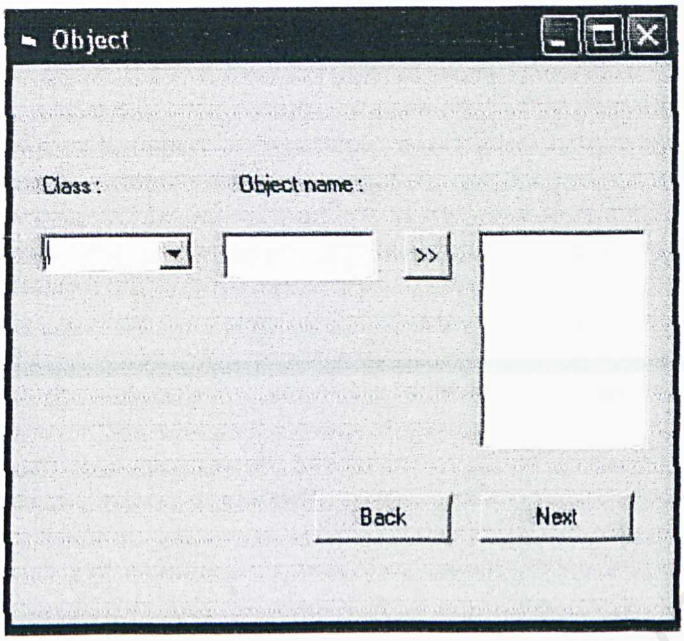


Figure 4.9 – Object window.

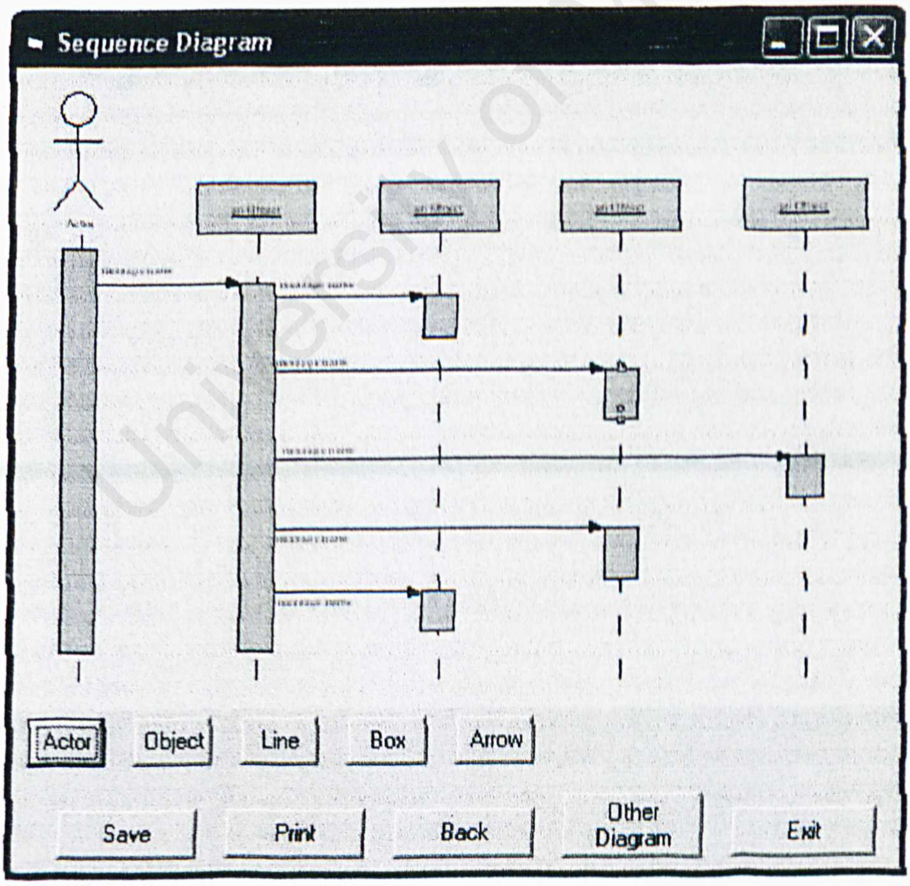


Figure 4.10 – Sequence Diagram window.

Chapter 5

System Implementation

5.1 Introduction to System Implementation

System implementation phase is a phase where the system designs are converted into executable codes. In this case, designs for Tutor on Software Design as shown in Chapter 4 are converted into a real program. This chapter discusses the hardware and software used to code the system, coding approach and style, and a brief description of system modules and functionality.

5.2 Development Environment

Development environment has certain impact on the development of a system. Using the suitable hardware and software will not only help to speed up the system development but also determine the success of the project. After implementing the system, the requirement of hardware and software that was stated in the previous chapter can be finalized.

5.2.1 Hardware

Hardware	**Requirements**
Monitor	Dell 17"
Processor	Intel® Pentium® IV 2.60 GHz
Memory	256 MB
Hard disk	80 GB
Input device	Keyboard and mouse
Printer	Canon BJ200sp

Table 5.1 Hardware used in system development

5.2.2 Software

Software	**Requirements**
Operating system	Windows XP
Database management system	Microsoft Access 2002
Programming language	Visual Basic.Net
Project documentation	Microsoft Word 2002
Image Editing	Adobe Photoshop

Table 5.2 Software used in system development

5.3 Coding Approach

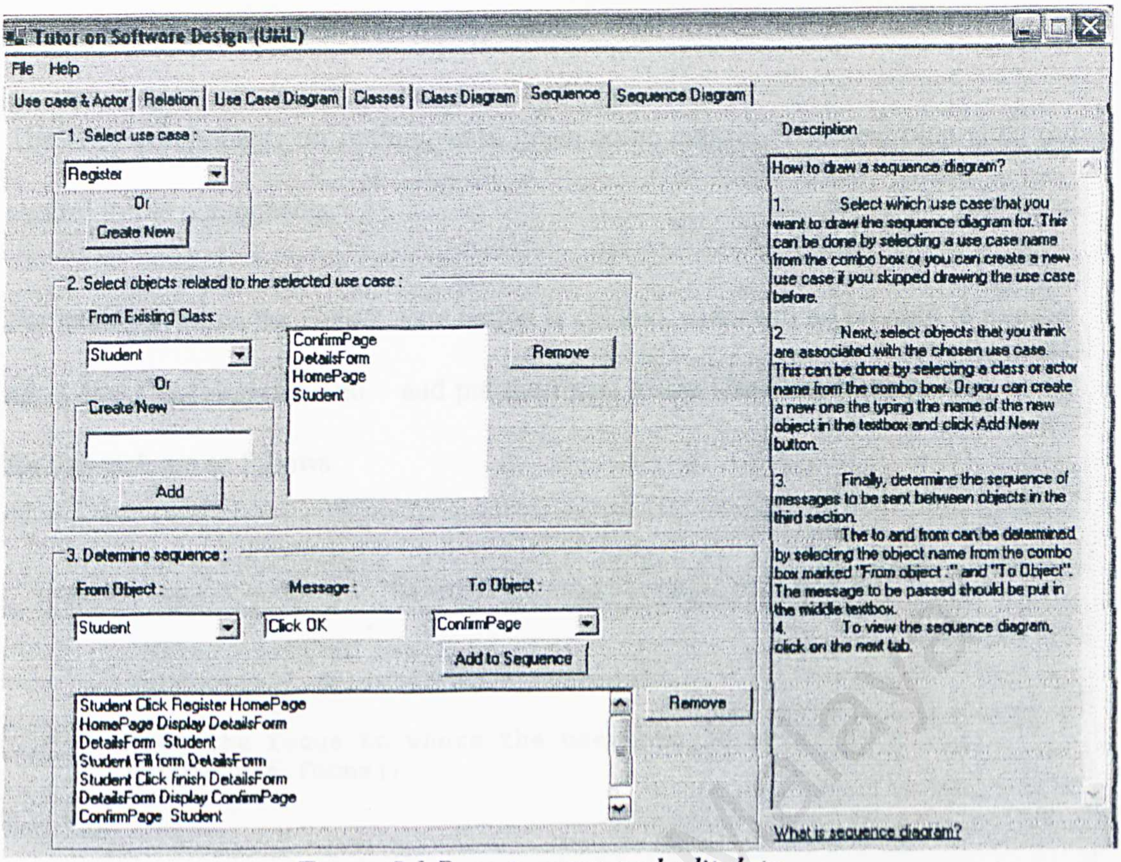
The coding process for Tutor on Software Design was done in different steps. Along the way, a lot of try-and-error method was used as I; along with the other team members were inexperienced in programming. The steps involved are developing the Graphical User Interface, putting controls to the GUI components, writing the processes involved, making additions, and making improvements.

5.3.1 Developing Graphical User Interface

In chapter 4, the prototype for this system is shown. It uses the concept of parent-child forms, in which buttons *Next* and *Back* will bring users to different forms. In the actual system implementation, the concept of tabs was used. It allows a lot of different pages in one form. Users can switch from one page to another easily by clicking the desired tabs.

The tab control was easily put onto the form and managed as VB.Net provided a built in tool for it. The tab just has to be dragged from the toolbox onto the form and managed via the property windows. There are 7 tabs on the form and sequence diagram uses 2 of them – *Sequence* and *Sequence Diagram*.

The *Sequence* tab as shown in Figure 5.1 has comboboxes, listboxes, textboxes and buttons for data entry by users. The *Sequence Diagram* tab as shown in Figure 5.2 only has a picture box that fills the entire tab page. It is used to display the output diagram.



1. Select use case :

Register

Or

2. Select objects related to the selected use case :

From Existing Class:

Student

Or

Create New

ConfirmPage
DetailsForm
HomePage
Student

3. Determine sequence :

From Object : Student

Message : Click OK

To Object : ConfirmPage

Student Click Register HomePage
HomePage Display DetailsForm
DetailsForm Student
Student Fill form DetailsForm
Student Click finish DetailsForm
DetailsForm Display ConfirmPage
ConfirmPage Student

Description

How to draw a sequence diagram?

1. Select which use case that you want to draw the sequence diagram for. This can be done by selecting a use case name from the combo box or you can create a new use case if you skipped drawing the use case before.

2. Next, select objects that you think are associated with the chosen use case. This can be done by selecting a class or actor name from the combo box. Or you can create a new one the typing the name of the new object in the textbox and click Add New button.

3. Finally, determine the sequence of messages to be sent between objects in the third section.

The to and from can be determined by selecting the object name from the combo box marked "From object :" and "To Object". The message to be passed should be put in the middle textbox.

4. To view the sequence diagram, click on the next tab.

What is sequence diagram?

Figure 5.1 Page to enter and edit data

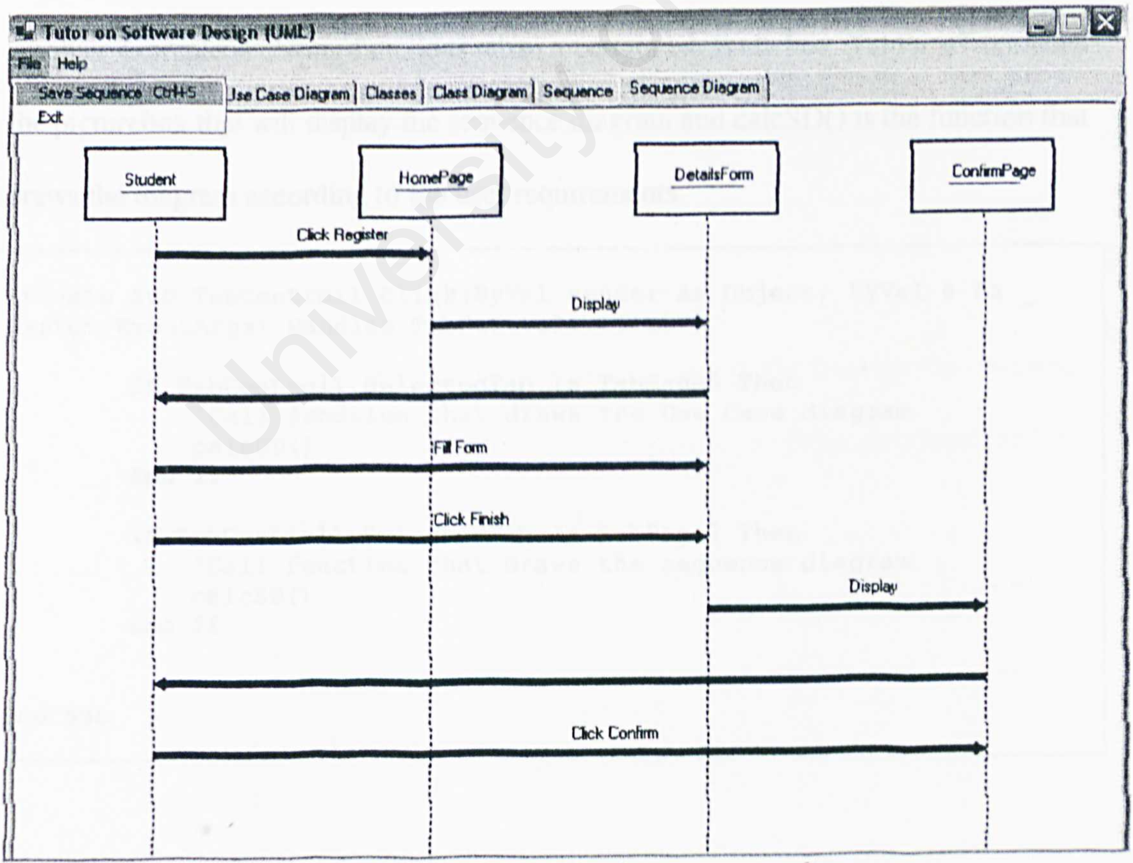


Figure 5.2 Picture box to display sequence diagram

5.3.2 Putting Control to GUI Components

The GUI components do nothing until given some control. The next step is to put control to the components.

For example, when the *Create New* button is clicked, users will be brought to another tab – *Use Case & Actor* Tab – and put the focus to the Use Case textbox. The codes for this task are as follows:

```
Private Sub btnNewUseCase_Click(ByVal sender As System.Object, ByVal
_ e As System.EventArgs) Handles btnNewUseCase.Click

        'Bring users to use case page
        TabControl1.SelectedTab = TabPage1

        'Set the focus to where the user should enter data
        txtUseCase.Focus()

End Sub
```

Another example is calling functions when a certain tab is clicked. TabPage7 contains the picturebox that will display the sequence diagram and calcSD() is the function that draws the diagram according to the user requirements.

```
Private Sub TabControl1_Click(ByVal sender As Object, ByVal e As _
System.EventArgs) Handles TabControl1.Click

        If TabControl1.SelectedTab Is TabPage5 Then
            'Call function that draws the Use Case diagram
            calcUD()
        End If

        If TabControl1.SelectedTab Is TabPage7 Then
            'Call function that draws the sequence diagram
            calcSD()
        End If

End Sub
```

5.3.3 Writing Processes

Processes refer to codes that are written to receive data from users and turn it into diagrams. They are divided into three – data manipulation codes, drawing codes and codes for saving.

5.3.3.1 Data Manipulation

Data entered by user as their system requirements has to be saved into the database. Users are allowed to add, choose or delete data anytime they want. Therefore, codes are assigned to do various tasks concerning data.

a. Database connection.

```
' stores the connection to an Access database globally
Public myDB As New ADODB.Connection
```

```
' connect to the Access database
myDB.Open("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
Application.StartupPath & "/db2.mdb")
```

b. Create table.

When users choose which use case they would like to draw the sequence diagram for, two tables are created – one to store classes involved, and the other to store the sequence of events.

```
'Remove space, - and _ from use case name and add
"sequence 'at the end
'This will be the table name to store relevant classes
strNew = comSeqUse.Text.Replace(" ", "")
strNew = strNew.Replace("-", "")
strNew = strNew.Replace("_", "")
strNew = strNew + "Sequence"

'Initialize the SQL statement to create table
```

```
   mySQL = "Create table " & strNew & "(ObjSeqID number, " &
   _ strNew & " char)"

   Try
        ' execute the SQL command
        myDB.Execute(mySQL)
   Catch ex As Exception

   End Try


   'Remove spaces, - and _ from use case name
   'Add "SeqUse" in front of it
   'This will be the name of the sequence table
   strNew = comSeqUse.Text.Replace(" ", "")
   strNew = strNew.Replace("-", "")
   strNew = strNew.Replace("_", "")
   strNew = "SeqUse" + strNew

   'Initialize SQL statement to create table
   mySQL = "Create table " & strNew & "(ID number, Obj1
   char,_ Obj2 char, Msg char, Type char)"
```

c. Fill combobox with data from database.

There are 4 comboboxes on the Sequence page – one for selecting use case, one for selecting class to associate with the use case, and two for users to determine sequence of events for their software. These comboboxes are filled with data from relevant tables in the database. The codes to fill the first combobox are as follows.

```
Dim myRS As New ADODB.Recordset

'Open the recordset
myRS.Open("Select UseCase from UseCase", myDB)

' clear old results
comSeqUse.Items.Clear()

' place results in the list
While Not myRS.EOF


    ' put data in actor table into combobox
    comSeqUse.Items.Add(myRS.Fields("UseCase").Value)

    ' go to next record in the results
    myRS.MoveNext()
```

```
     End While

     'Close the recordset
     myRS.Close()
```

d. Fill listbox with data from database.

Following are the codes to fill a listbox with data from database. The data are classes associated with the selected use case. The codes are called as soon as users select a use case.

```
        ' clear old results
         lstObj.Items.Clear()

        ' place results in the list
         While Not myRS.EOF
                'Put data from database to listbox
                lstObj.Items.Add(myRS.Fields(strNew).Value)

                ' go to next record in the results
                myRS.MoveNext()

         End While
```

e. Fill listbox with data from three fields.

Another listbox at the bottom of the page is used to displays the sequence of events of the users' software entered. It is filled as soon as a use case is selected and it changes everytime user add or remove a sequence. It is also affected when the user removes a class that is involved in the sequence. Each line combines data from 3 fields in the database that represent message sender, message and message receiver of a particular sequence.

```
      ' clear old results
      lstLeft.Items.Clear()
      lstTop.Items.Clear()

      ' place results in the list
      While Not myRS.EOF
          lstLeft.Items.Add(myRS.Fields("Obj1").Value)
          strLstTop = Trim(myRS.Fields("Obj1").Value) & " " &
```

```
            _ Trim(myRS.Fields("Msg").Value) & " " & _
            Trim(myRS.Fields("Obj2").Value)
           lstTop.Items.Add(strLstTop)
           ' go to next record in the results
           myRS.MoveNext()
       End While

       'Close recordset
       myRS.Close()
```

f. Add new data into database.

When the user adds a sequence, the following codes are executed.

```
    'Call function to create new record
     currentID = CreateNewRecordSeq()

     'Call function to fill the new record with data
     UpdateRecordSeq(currentID, comObj1.Text, comObj2.Text, _
    txtMsg.Text, comType.Text)
```

```
    Private Function CreateNewRecord() As Integer
    ' create a new record in the database

     'Initialization
     Dim strNew As String
     Dim myRS As New ADODB.Recordset
     Dim icount As Integer

     If comSeqUse.Text <> "" Then
         'Get table name
         strNew = comSeqUse.Text.Replace(" ", "")
         strNew = strNew.Replace("-", "")
         strNew = strNew.Replace("_", "")
         strNew = strNew + "Sequence"
    End If

     ' open the record set and
    'have it contain all records in the database table
    myRS.Open(strNew, myDB, ADODB.CursorTypeEnum.adOpenKeyset, _
   ADODB.LockTypeEnum.adLockOptimistic)

    If Not myRS.EOF Then

         'Move to the last record
         myRS.MoveLast()
         icount = myRS.Fields("ObjSeqID").Value + 1
    Else
         icount = 0
    End If

    ' add a new empty record
```

```
        myRS.AddNew("ObjSeqID", icount)

        ' update the recordset to include the new entry
        myRS.Update()
        myRS.Requery()

        ' go to the record we just created
        myRS.MoveLast()

        ' grab the record's ID value so we can edit it
       'return this from the function
        CreateNewRecord = myRS.Fields("ObjSeqID").Value

        ' close the record set
        myRS.Close()

       End Function
```

```
Public Function UpdateRecordSeq(ByVal currentid As Integer, _
ByVal Obj1 As String, ByVal Obj2 As String, ByVal Msg As String, _
ByVal Type As String)
        ' Edit a record in the database

        Dim strNew As String
        Dim myRS As New ADODB.Recordset

        If comSeqUse.Text <> "" Then
            'Get table name
            strNew = comSeqUse.Text.Replace(" ", "")
            strNew = strNew.Replace("-", "")
            strNew = strNew.Replace("_", "")
            strNew = "SeqUse" & strNew
        End If

        ' open a recordset from the table named address
        myRS.Open(strNew, myDB, ADODB.CursorTypeEnum.adOpenKeyset, _
       ADODB.LockTypeEnum.adLockOptimistic)

        myRS.Find("ID=" & currentid)

        ' change fields to update them
        myRS.Fields("Obj1").Value = Obj1
        myRS.Fields("Obj2").Value = Obj2
        myRS.Fields("Msg").Value = Msg
        myRS.Fields("Type").Value = Type


        ' update and close the recordset
        myRS.Update()
        myRS.Close()


    End Function
```

g. Remove selected data from database.

There are two Remove buttons on the page. Each for removing classes and sequences. Removing classes will call two functions – one to delete the selected class and one to delete the affected sequences. Removing sequence will also call two functions – one to delete the selected sequence and one to rearrange the ID of the remaining sequences.

The Renew function is used to reassign ID of sequences after one of the records has been deleted. This is done to ensure that the sequence is always correct. The idea is to set the first record to have 1 as ID, 2 for the second record and so on.

```
' Button to remove class

'Set selected class name to a variable
Dim str As String = Trim(lstObj.SelectedItem)

'Call function to delete the record of classes with current
ID
DeleteRecord(currentID)

'Call function to delete the affected sequence
DeleteSeqClass(str)
```

```
' Button to remove Sequence

While Not myRS.EOF

    If lstLeft.SelectedIndex + 1 = myRS.Fields("ID").Value
Then

        ' Set the ID of record to be deleted
        ID = lstLeft.SelectedIndex + 1

        ' Display confirmation dialog
        result = MessageBox.Show(message, caption, _
        MessageBoxButtons.OKCancel)
              If result = DialogResult.OK Then
                  ' Call function to delete selected sequence
                  ' Send ID as parameter
                  DeleteRecordSeq(ID)
              Else
                  Exit Sub
```

```
                End If
        Else
            ID = 0
        End If

        ' go to next record in the results
        myRS.MoveNext()
    End While

    ' Call function to rearrange ID
    RenewSeqID(myRS)
```

```
Public Function DeleteRecord(ByVal currentID As Integer)
        ' delete a record from the database

        'Initialization
        Dim strNew As String
        Dim item As String = CStr(lstObj.SelectedItem)

        If comSeqUse.Text <> "" Then
            'Get table name
            strNew = comSeqUse.Text.Replace(" ", "")
            strNew = strNew.Replace("-", "")
            strNew = strNew.Replace("_", "")
            strNew = strNew + "Sequence"
        End If

        ' create a string variable that will hold an SQL statement
        mySQL = "Delete * from " & strNew & " where " & strNew & "='"
        _ & item & "'"

        ' execute the SQL command
        myDB.Execute(Trim(mySQL))


End Function
```

```
Public Function DeleteSeqClass(ByVal item As String)
        ' delete a record from the database

        ' Initialization
        Dim strNew As String
        Dim mySQL1 As String
        Dim mySQL2 As String

        If comSeqUse.Text <> "" Then
            ' Get table name
            strNew = comSeqUse.Text.Replace(" ", "")
            strNew = strNew.Replace("-", "")
            strNew = strNew.Replace("_", "")
            strNew = "SeqUse" & strNew
        End If

        ' Set SQL statement to two variables
        mySQL1 = "Delete * from " & strNew & " where Obj1='" & item &
        _"'"
        mySQL2 = "Delete * from " & strNew & " where Obj2='" & item &
        _"'"

        ' execute the SQL command
        Try
            myDB.Execute(mySQL1)
            myDB.Execute(mySQL2)
        Catch ex As Exception

        End Try

End Function
```

```
Public Function RenewSeqID(ByVal myRS As ADODB.Recordset)

        'Initialization
        Dim strNew As String
        Dim count As Integer = 1

        If comSeqUse.Text <> "" Then
            'Get table name
            strNew = comSeqUse.Text.Replace(" ", "")
            strNew = strNew.Replace("-", "")
            strNew = strNew.Replace("_", "")
            strNew = "SeqUse" & strNew
        End If

        Try
            If Not myRS.BOF Then
                'Go to first record in the recordset
                myRS.MoveFirst()
            End If
        Catch ex As Exception

        End Try

        While Not myRS.EOF
            'Call function to replace existing ID with new one
            'Parameters : existing ID and new ID (Sequential)
            UpdateSeqID(myRS.Fields("ID").Value(), count)

            'Increase count
            count += 1

            'Move to next record
            myRS.MoveNext()
        End While

End Function
```

```
Public Function UpdateSeqID(ByVal currentID As Integer, _
ByVal newID As Integer)
        ' Edit a record from the database

        ' Variable initialization
        Dim strNew As String

        If comSeqUse.Text <> "" Then
            ' Get table name
            strNew = comSeqUse.Text.Replace(" ", "")
            strNew = strNew.Replace("-", "")
            strNew = strNew.Replace("_", "")
            strNew = "SeqUse" & strNew
        End If

        ' Set SQL statement to a variable
        mySQL = "Update " & strNew & " set ID=" & newID & " where
ID="_
        & currentID

        ' execute the SQL command
        myDB.Execute(mySQL)

End Function
```

5.3.3.2 Drawing

One of the functional requirements of Tutor on Software Design is, the system is able to draw sequence diagram based on user requirements. The above section has shown has user requirements can be manipulated. This section will show how those data can be converted into relevant sequence diagram. It involves a lot of steps as shown below.

a. Import namespace.

The namespace is necessary in order to use drawing functions to draw various shapes such as rectangle, lines, circles, etc.

```
        Imports System.Drawing.Imaging
```

b. Initialize bitmap and graphic objects.

```
        ' Initialize a bitmap object
        ' Set the size with the picturebox size
        Dim newBitmap As Bitmap = New Bitmap(784, 544, _
        Imaging.PixelFormat.Format24bppRgb)
```

```
' Set the bitmap to a graphic object
Dim lbl As Graphics = Graphics.FromImage(newBitmap)
```

c. Call function to draw.

```
If TabControl1.SelectedTab Is TabPage7 Then
    'Call function that draws the sequence diagram
    calcSD()
End If
```

d. Run function to draw.

The following tasks are done within `Public Function calcSD()`.

i. Draw a white rectangle on the bitmap object as the background.

```
Dim rect As Rectangle = New Rectangle(0, 0, 784, 544)
lbl.FillRectangle(New SolidBrush(Color.White), rect)
```

ii. Initialize and set coordinate values to points.

```
Dim p1, p2, p3, p4 As Point

p1.X = 98
p1.Y = 50

p2.X = 294
p2.Y = 50

p3.X = 490
p3.Y = 50

p4.X = 686
p4.Y = 50
```

iii. Initialize and set values to objects representing data. Data taken from a table in the database are set to 4 objects.

```
Dim drwObj1 As String = ""
Dim drwObj2 As String = ""
Dim drwObj3 As String = ""
Dim drwObj4 As String = ""
```

```
        While Not myRS.EOF
                'Fill drwObj1
                If drwObj1 = "" Then
                     If myRS.Fields("Obj1").Value <> drwObj2 And _
                      myRS.Fields("Obj1").Value <> drwObj3 And _
                      myRS.Fields("Obj1").Value <> drwObj4 Then
                         drwObj1 = myRS.Fields("Obj1").Value
                     End If
                End If

                If drwObj1 = "" Then
                     If myRS.Fields("Obj2").Value <> drwObj2 And _
                      myRS.Fields("Obj2").Value <> drwObj3 And _
                      myRS.Fields("Obj2").Value <> drwObj4 Then
                         drwObj1 = myRS.Fields("Obj2").Value
                     End If
                End If

                'Fill drwObj2, drwObj3 and drwObj4 with the same way

        End While
```

iv. Call function to draw objects.

```
        If drwObj1 <> "" Then
             recseqpic(p1, drwObj1)
        End If

        If drwObj2 <> "" Then
             recseqpic(p2, drwObj2)
        End If

        If drwObj3 <> "" Then
             recseqpic(p3, drwObj3)
        End If

        If drwObj4 <> "" Then
             recseqpic(p4, drwObj4)
        End If
```

```
Public Function recseqpic(ByVal c As Point, ByVal str As String)

        Dim rec As Rectangle
        Dim dotstart, dotend, strlen As Point
        Dim p As New Pen(Color.Black, 2)
        Dim dot As New Pen(Color.Black)
        Dim txt As SizeF, txtfont As Font = Me.Font
        Dim dashValues As Single() = {3, 2, 3, 2}
        Dim MyLen As Integer

        MyLen = 10 'Len(str)
        dot.DashPattern = dashValues

        rec = New Rectangle(c.X - 50, c.Y - 25, 100, 50)

        dotstart.X = c.X
        dotstart.Y = c.Y + 27

        dotend.X = c.X
        dotend.Y = c.Y + 800

        strlen.X = c.X + (MyLen * 2) + 5
        strlen.Y = c.Y + 5

        lbl.DrawString(str, txtfont, New SolidBrush(Color.Black), _
       c.X - 25, c.Y - 10)
        lbl.DrawLine(dot, dotstart, dotend)
        lbl.DrawLine(New Pen(Color.Black), _
       New Point(c.X - 25, c.Y + 5), strlen)
        lbl.DrawRectangle(p, rec)

End Function
```

v. Call function to draw arrows.

```
        'Initialization
        Dim temp1, temp2, temp3, temp4 As Point
        Dim obj1 As String
        Dim obj2 As String
        Dim str As String

        'Set coordinates to points
        temp1 = p1
        temp2 = p2
        temp3 = p3
        temp4 = p4

        While Not myRS.EOF
            'Get data from table
            obj1 = myRS.Fields("Obj1").Value
            obj2 = myRS.Fields("Obj2").Value
            str = myRS.Fields("Msg").Value

            'Set points 50 units down
            temp1.Y += 50
            temp2.Y += 50
            temp3.Y += 50
            temp4.Y += 50

            Select Case obj1 'From object
                Case drwObj1
                    Select Case obj2 'To object
                        Case drwObj1
                            drawBackArrow(temp1, str)
                        Case drwObj2
                            drawArrow(temp1, temp2, str)
                        Case drwObj3
                            drawArrow(temp1, temp3, str)
                        Case drwObj4
                            drawArrow(temp1, temp4, str)
                    End Select

                'Continue through drwObj4
                End Select
                myRS.MoveNext()
            End While
```

```
Public Function drawArrow(ByVal point1 As Point, _
ByVal point2 As Point, ByVal str As String)
        ' Create pen.
        Dim blackPen As New Pen(Color.Black, 5)

        ' Initialize the arrow cap
        blackPen.EndCap = Drawing2D.LineCap.ArrowAnchor

        'Calculate position of string
        Dim pos As Point

        If point1.X > point2.X Then
            pos.X = ((point1.X - point2.X) / 2) + point2.X
        Else
            pos.X = ((point2.X - point1.X) / 2) + point1.X
        End If

        'Draw line to screen with a string on it.
        lbl.DrawString(str, Me.Font, New SolidBrush(Color.Black), _
         pos.X, point1.Y - 20)
        lbl.DrawLine(blackPen, point1, point2)

End Function
```

```
Public Function drawBackArrow(ByVal p1 As Point, ByVal str As String)
        'Create pen
        Dim blackpen As New Pen(Color.Black, 5)
        Dim p As New Pen(Color.Black, 5)
        Dim p2, p3, p4 As Point

        p1.Y = p1.Y - 8
        p2.X = p1.X + 26 : p2.Y = p1.Y
        p3.X = p2.X : p3.Y = p2.Y + 26
        p4.X = p1.X : p4.Y = p3.Y

        'Initialize the arrow cap
        blackpen.EndCap = Drawing2D.LineCap.ArrowAnchor

        lbl.DrawLine(p, p1, p2)
        lbl.DrawLine(p, p2, p3)
        lbl.DrawLine(blackpen, p3, p4)
        lbl.DrawString(str, Me.Font, New SolidBrush(Color.Black), _
         p2.X + 5, p2.Y + 13)

End Function
```

vi. Set bitmap to picturebox.

```
        PictureBox2.Image = newBitmap
```

5.3.3.3 Save Diagram

The output of this system is the required sequence diagram. This diagram can be saved in various image formats anywhere on the computer the user wants. Executing the following codes does it.

```
Private Sub MnuSaveSeq_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MenuSaveSeq.Click

        Dim extension As String = dlgSave.FileName

        PictureBox2.Image = newBitmap

        If dlgSave.ShowDialog() = DialogResult.OK Then
        extension = extension.Substring(extension.LastIndexOf(".") + _
        1).ToLower

        Select Case extension
                Case "bmp"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Bmp)
                Case "jpg", "jpeg"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Jpeg)
                Case "gif"
                        PictureBox2.Image.Save(dlgSave.FileName,_
                        ImageFormat.Gif)
                Case "ico"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Icon)
                Case "emf"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Emf)
                Case "wmf"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Wmf)
                Case "png"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Png)
                Case "tif", "tiff"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Tiff)
                Case "exif"
                        PictureBox2.Image.Save(dlgSave.FileName, _
                        ImageFormat.Exif)
                End Select
            End If
    End Sub
```

5.3.4 Improvements.

In the beginning, all the codes were written carelessly. Then, same improvements were made. For example, variables that were initialized in many functions were made global.

Other than that, some error handling were also put into place. An example of error handling is:

```
Try
      ' execute the SQL command
      myDB.Execute(mySQL)
Catch ex As Exception

End Try
```

Redundant codes were made into separate function. For example, the functions *RenewSeqID()* and *UpdateSeqID()*.

5.4 Coding Style

There are some styles in coding that were purposely followed to make the codes maintainable and readable. One of the styles is using meaningful variable names. For instance, a button to add a class is named *btnAddClass*, a string variable to hold a new name is called *strNew*, and the menu to save the sequence diagram is named *mnuSaveSeq*, and so on.

To make the codes even more readable, certain lines were indented where necessary and keyword (or reserved word) are coloured blue and non-keyword are black. Comments were made in green. Fortunately, VB.Net automatically applies this style. An example is the following function:

```
ic Function DeleteRecord(ByVal currentID As Integer)
     ' delete a record from the database

     'Initialization
     Dim strNew As String
     Dim item As String = CStr(lstObj.SelectedItem)

     If comSeqUse.Text <> "" Then
          'Get table name
          strNew = comSeqUse.Text.Replace(" ", "")
          strNew = strNew.Replace("-", "")
          strNew = strNew.Replace("_", "")
          strNew = strNew + "Sequence"
     End If

     ' create a string variable that will hold an SQL statement
     mySQL = "Delete * from " & strNew & " where " & strNew & "='"
     _ & item & "'"

     ' execute the SQL command
     myDB.Execute(Trim(mySQL))


unction
```

y, comments were added where necessary as internal documentation. This will make any enhancement easier.

5.5 Description of System Modules and Functionality.

In order to draw the sequence diagram, 4 modules are used, namely –use case, class, sequence, and drawing module.

5.5.1 Use Case

The first thing that users have to go through in order to get the desired sequence diagram is to select a use case, which the diagram is drawn for. They can choose from the existing use case from the drop down list or they can add a new use case.

5.5.2 Class

Next, they will have to choose which class involved in the diagram. Like the use case, users can choose existing classes or create new ones.

5.5.3 Sequence

The next module allows users to put in the sequence of events for their system. They can add and remove any sequence using existing classes.

5.5.4 Drawing

This module simply takes data from database and executes a few functions to draw the required sequence diagram.

5.6 Conclusion

This chapter describes the implementation of Tutor on Software Design. It shows how designs from the previous chapter are turned into executable program and how it fulfils the functional requirements of this system as stated earlier in planning phase.

The next chapter documents the testing done on the system.

Chapter 6

System Testing

6.1 Introduction to System Testing

Many programmers view testing as a way to demonstrate how their program performs properly. However, the idea of demonstrating correctness is really the reverse of that testing is all about. We test a program to demonstrate the existence of a fault. Because our objective is to find faults, we consider a test successful only when a fault is discovered. This is achieved by using carefully planned test strategies and realistic data so that the entire testing process can be methodically and rigorously carried out.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and code generation. In this chapter, software testing fundamentals, testing strategies and software debugging methods will be presented.

Following are some of the objectives of software testing:

- Testing is a process if executing a program with the intent of finding an error.
- A good test case is noted that has a high probability of finding an as-yet-undiscovered error.
- A successful test is one that uncovers as-yet undiscovered error.

6.2 Fault

The objective of testing is to find error and fault. Fault identification is the process of determining what fault or faults caused the failure, and fault correction or removal is the process of making changes to the system so that the fault are removed.

6.2.1 Types of Fault

When no obvious fault exists, program is tested to isolate more faults by creating conditions where the code does not react as planned. Therefore, it is important to know kind of faults to seek.

Faults can be categorized as algorithmic faults, syntax faults and documentation faults described as below:

1. **Algorithmic faults**

   Algorithmic faults occur when a component's algorithm or logic does not produce the proper output for given input because something is wrong with the processing steps. These faults are easy to spot by reading through the program (call desk checking) or by submitting input data from each of the different classes of data that we expect the program to receive during its regular working.

   Typical algorithmic faults include:

   i. Testing for the wrong condition.

   ii. Forgetting to initialize variables or set loop invariants.

   iii. Forgetting to test for a particular condition (such as when division by zero might occur).

2. **Syntax faults**

Syntax faults can be checked while parsing for algorithmic faults. This will ensure that the construct of programming language is used properly. Microsoft Interdev does come with a compiler to catch syntax faults before a program is executed. Therefore, syntax faults within the program can be traced before the program is executed.

3. **Documentation faults**

   When the documentation does not match what the application does, the application has documentation faults. Usually, documentation is derived from system design and provides a clear description of what the programmer would like the program to do, but the implementation of these functions is faulty. Such faults can lead to other faults later.

6.3 Test Planning

The purpose of having test planning is to help in designing and organizing tests, so that testing is carried out appropriately and thoroughly.

A test plan has the following steps:

1. **Establishing test objectives**

   At the beginning, we have to know what we are going to test on. So we have to establish our test objectives that tell us what kinds of test cases to generate.

2. **Designing test cases**

   After establishing test objectives, we begin to design the test cases that are used to test the system. If test cases are not representative and do not thoroughly exercise the functions that demonstrate the correctness and validity of the system, then the reminder of the testing process is useless.

3. **Writing test cases**

   After designing, we have to start writing the test cases.

4. **Testing test cases**

   At the same time, we also test the test cases.

5. **Executing tests**

   After all testing have been done, we execute our tests on the system.

6. **Evaluating test results**

   After executing tests, we evaluate the test results.

6.4 Testing Technique

To test a component, a range of inputs and conditions are chosen. The component of the software will be allowed to manipulate the data, and the output will be observed. A particular input is chosen will demonstrate the behavior of the code behind the entire GUI. A test point or a test case is a particular choice of input data to be used in testing program. However, the data are entered with the express intent of determining whether the system will process them correctly.

Different test cases are needed on different type of testing strategies. There are four categories of test cases that are created for testing purposes namely erroneous test data, normal test data, extreme test data and condition test data. These categories are further explained in the following section.

6.4.1 Erroneous Test Data

Using test data that are erroneous is a good way to determine how the system handles such errors and how it behaves under such situation. For example, the drawing module of this system can only set coordinates for 8 sequences. What happens if users enter 9 sequences? Therefore, it is used as erroneous test data.

6.4.2 Normal Test Data

The normal test case is use to check whether the system will work well under normal condition. That is means to test whether a given correct data will produce the expected results. For example, assume that there are four classes associated with a use case called Register. So, if a user wants to generate a sequence diagram using these four classes, the program will allow it. This type of test data serves as a preliminary test of the system.

6.5 Testing Strategy

Testing is a process of exercising or evaluating a system by manual or automatic means to verify that it has satisfied requirements or to identify differences expected and actual results. Testing is probably the least understood part of a software development project. A bug is any unexpected, questionable, or undesired aspect or behavior displayed, facilitated or caused by the software being tested. Testing can uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. The strategies used for testing are unit testing, module testing, integration testing and system testing.

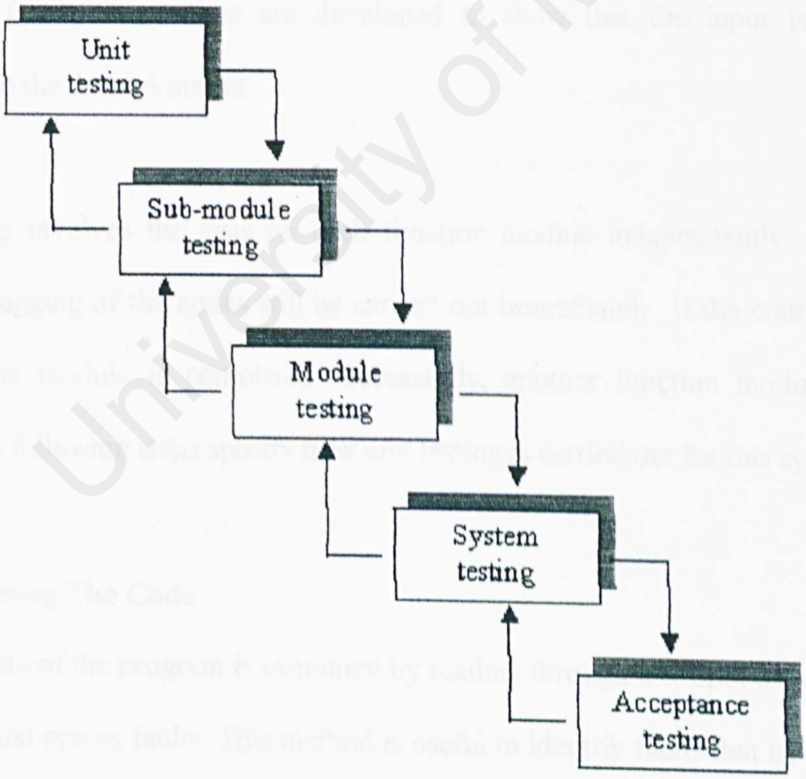


Figure 6.1: Testing process

6.5.1 Unit Testing

Historically, quality software is relied on testing each function or module. Unit testing is sometimes referred to as function testing or component testing, which is extremely time-consuming. For Tutor on Software Design, unit testing was done during the coding phrase.

The first step is to examine the program code by reading through it, trying to spot algorithm, data and syntax faults. Followed by comparing the code with specifications and with the design to make sure that all relevant cases have been considered. Next, the program is executed to view result and then eliminate remaining syntax faults if necessary. Finally, test cases are developed to show that the input is properly converted to the desired output.

Unit testing involves the tests on each function module independently. If error is found, debugging of the codes will be carried out immediately. If the compilation of the function module is completed successfully, another function module will be coded. The following steps specify how unit testing is carried out for this system:

1. **Examining The Code**

   The code of the program is examined by reading through it to spot for algorithmic faults and syntax faults. This method is useful to identify faults that have been left out by the programmer.

2. **Control Objects Testing**

   Command buttons are clicked to test their functionality and text boxes are tested with different data types and also null value to make sure invalid data will not cause any fault.

3. **Different Data Type Testing**

   Different data types like numbers, characters or date is used to test certain function because some control objects will only accept certain data type, invalid data type can be traced by the system without causing any error.

4. **Choosing Test Cases**

   Test cases are developed to ensure that the input is properly converted to the desired output. So, to test a component, input data and condition are chosen. Then the component is allowed to manipulate the data, and output is observed.

6.5.1.1 Example Of Unit Testing

There were too many unit test cases involved. Therefore, only a few will be shown as example. The rest are attached as Appendix A.

Unit Test Case Example 1

The Add Class function in this system is used to add new record of class into database. Unit Testing was carried out to ensure that the record was added successfully. The table below shows the test case for unit testing on the function of adding the record.

Unit:	**Add New Class**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Type a class name in the *Create New* textbox. Class name: ConfirmPage	The newly added class is listed in the listbox and drop down list.	ConfirmPage is listed in the listbox but not in drop down list.
2.	Click *Add*		

Table 6.1 Test Case For Adding Class

6.5.2 Module Testing

A module is a collection of dependent components. A module encapsulates all of the related components. Module testing enables each module to be tested independently. This testing will ensure that the module calling sequence in this project is systematic.

In module testing, two or more units in which either unit that use output data from or provide input data for another unit were tested in collection. These units have related characteristics to perform a common goal or function such as the drawing functions that uses data entered previously.

6.5.2.1 Example Of Module Testing

Module Test Case Example 1

The drawing function in this system is used to draw images, which are used to create sequence diagram. Module Testing was carried out to ensure that the image was being drawn successfully. Table below shows the test case for module testing on the function of drawing diagram.

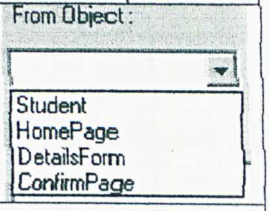
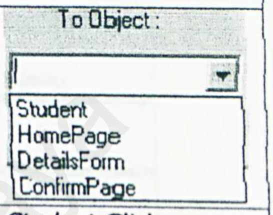
No.	Testing Procedure	Expected Output	Actual Output
1.	Click left drop down list in the third section	All added classes should be listed.	 From Object : Student HomePage DetailsForm ConfirmPage
2. 3.	Choose a class Class name: Student Type a message in the centre textbox Message: Click Register		
4.	Click right drop down list	All added classes should be listed.	 To Object : Student HomePage DetailsForm ConfirmPage
5. 6.	Choose a class Class name: HomePage Click Add to *Sequence button*	The sequence should be listed.	Student Click Register HomePage is listed.
7. 8.	Repeat steps 1 to 6 with different combination on input Click Sequence Diagram tab	A page displaying a sequence diagram.	Figure 6.2

Table 6.2 Test Case For Drawing Diagram

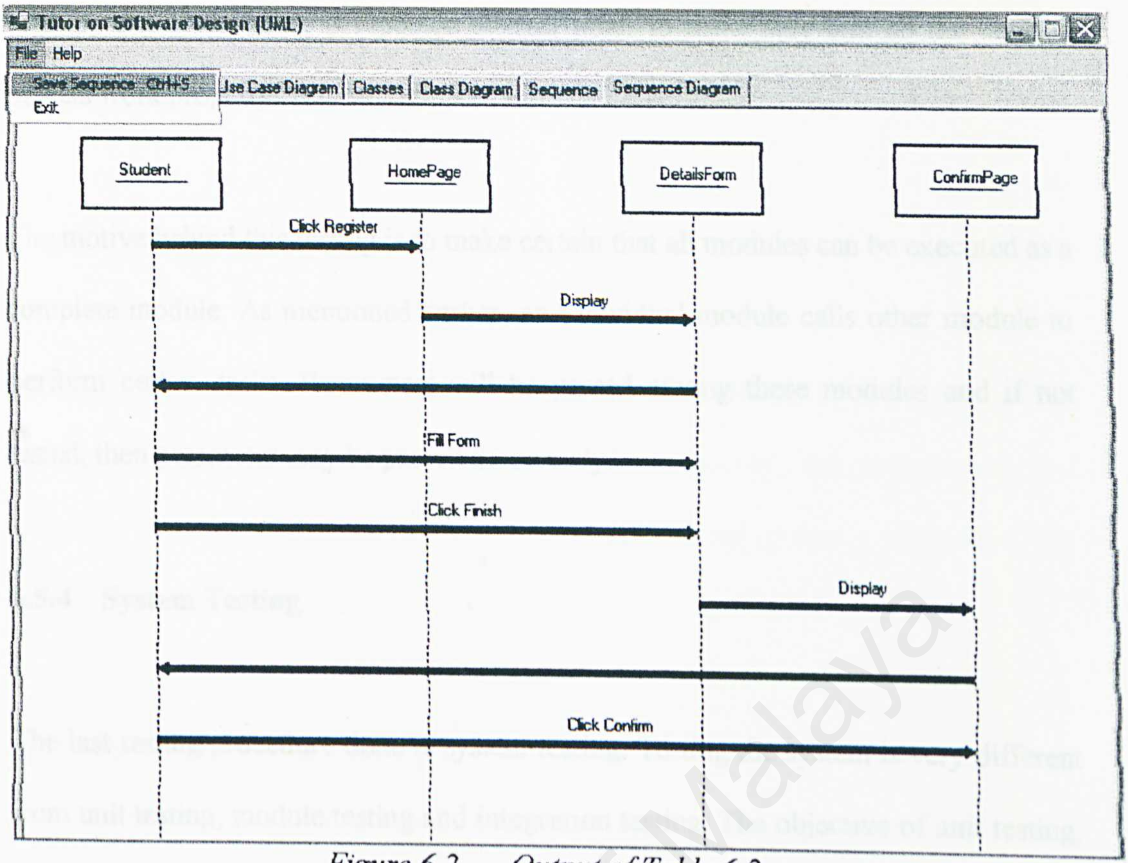


Figure 6.2 Output of Table 6.2

6.5.3 Integration Testing

When the individual components are working correctly and meet the objectives, these components are combined into a working system. In other words, integration testing is the process of verifying that the system components work together as described in the system and program design specifications. This integration is planned and coordinated so that when a failure occurs, some idea of what caused it can be got.

Sandwich integration testing approach is used for this system. This approach combines top-down integration with bottom-up integration. The testing starts from the first section of the system and down to the lowest level form of functions and from the function up. This testing is repeated several times to make sure that all the control objects work properly.

The motive behind this testing is to make certain that all modules can be executed as a complete module. As mentioned earlier, an individual module calls other module to perform certain tasks. Parameters will be passed among these modules and if not tested, then parameter may be passed incorrectly.

6.5.4 System Testing

The last testing procedure done is system testing. Testing the system is very different from unit testing, module testing and integration testing. The objective of unit testing, module testing and integration testing is to ensure that the code has implemented the design properly. In other words, the code is written to do what the design specifications intended. In system testing, a very different objective is to be achieved, that is to ensure that the system does what the users want it to do.

Tutor on Software Design involves two kinds of system testing. They are function testing and performance testing.

6.5.4.1 Function Testing

Function testing is based on the system functional requirements. In other words, a function test is used to check that whether the integrated system performs its functions as specified in the requirements. The tests are carried out for data module and drawing module in this system. The module is tested individually to determine whether the system performs as required.

6.5.4.2 Performance Testing

Performance testing addresses the nonfunctional requirements of the system. That means once the functions are convinced work as specified, the performance test compares the integrated components with the nonfunctional system requirements. The types of performance tests are:

a) Compatibility Tests

This test was performed to find out that the interface functions perform according to the requirements. The accuracy of data retrieval was high in this system. Besides, the speed of data retrieval was acceptable too.

b) Human Factors Tests

This test was performed to investigate requirements dealing with the user interface to the system. In this system, simple forms and related messages are displayed to determine user friendliness. These tests are sometimes called **usability tests**.

c) Timing Tests

This test was performed to evaluate the requirements dealing with the time to respond to a user and time to perform a function. The response time of this system is acceptable.

6.6 Summary

Testing is one of the important steps in developing a system. Precision and accuracy of output data is considered during this process. Unit, module, integration and system testing has been carried out for the Tutor on Software Design. These testing approaches lead to delivering a quality system to users. The objective of a system will only be achieved after all the thorough testing done by different user with different aspects.

Chapter 7

System Evaluation

7.1 Introduction to System Evaluation

In this chapter, the system evaluation will be discussed. There were many techniques that used to evaluate the final system. In this chapter, system's features and strengths, system's limitation and constraints, the future enhancement, problems and solutions and lastly knowledge and experience gained will be described.

7.2 System Strengths

Followings are the features and strengths that can be found in the Tutor on Software Design:

7.2.1 Different Approach From Existing Systems

As described in Chapter 2 – Literature Review, Tutor on Software Design is different from other existing systems that draw sequence diagrams. Instead dragging and dropping shapes onto a canvas or using certain language, this system uses drop down lists, textboxes and lists. This makes it a different and easy to use system to draw a sequence diagram.

7.2.2 Documented

This system is documented internally and externally. Internal documentation refers to the comments put in suitable places. This ensures the codes are readable and it is maintainable as anyone can easily understand the codes and make future enhancement. External documentation refers to this document.

7.2.3 Support Various Image Formats

Users of this system can save their diagram in various formants, namely, *.bmp, *.jpg, *.gif, *.ico, *.emf, *.wmf, *.png, *.tif.

7.3 System Limitations and Constraints

Even though there are many features provided by the system, it is still not perfect. Due to the problem of time constraint and technologies, some of the feature cannot be implemented. The limitation is as listed below:

7.3.1 Static Image Size

The size of the picture box that contains the diagram is predetermined and fixed. The same goes with every object or line drawn. Coordinates of everything on the picture box are also fixed.

7.3.2 Limited Domain

The picture box only displays four objects in the diagram produced. This is caused by the fixed size of the picture box.

7.3.3 Limited Interaction Between System and User

Users can only enter data by typing in certain texboxes, or choosing from drop down lists instead of typing away their requirements. This limitation is done on purpose to avoid misunderstanding of users requirements by the system. With this limitation, erroneous output is minimized.

7.3.4 Unattractive Output

The shapes, lines and strings on the output diagram are drawn solely with VB codes, instead of drawing it using any drawing applications and import it to the system. However, this method ensures that the size of the image saved is smaller.

7.3.5 Insecure Database

The data entered by users are stored in an unsecured database, which can be edited from outside.

7.4 Future Enhancement

Due to the limitations found in the system, in the future, enhancement will be applied to the system to improve the ability of the system.

7.4.1 Improve Output Appearance

The output should look more interesting. Instead of the rigid black and white shapes, users should be given some choices regarding colors and sizes. This can be done by making the drawing algorithm more dynamic. Right now, the objects represented on the diagram are just rectangles. In the future, there should be more. For example, the symbols of actor, entity, boundary and various more objects.

7.4.2 Improve Data Management

More methods of storing data to be used by the system should be explored, instead of insecure Microsoft Access. The database should be design thoroughly to ensure integrity of data, with relations between tables and etc. Users must not be allowed to change data in the database from outside the system.

7.4.3 Complete Help File

This system should have a complete help file as it should teach users about sequence diagram.

7.4.4 Allow User to Print Diagram

So far, the system can only allow users to save the diagram produced. Then they will have to print it from other available software like Photoshop or Words. In the future, this system should have a menu that users can choose to print their diagram.

7.5 Problems and Solutions

During the process of development of the system, there are a few problem encountered. Some of them could be overcome through certain solution while some of them were not. The following are some of the problems that arose during the development process.

7.5.1 Unfamiliar with Development Tools

This is the first time that I use VB.Net and first time of developing a system bigger than any C++ lab exercises. Therefore, I need to learn from scratch without a teacher. However, I have friends whom I could ask questions and there are always forums on the Internet. Furthermore, there are a lot of sources of books whether on paper or electronic.

7.5.2 Difficulties in Choosing Approaches

As already said before, the implementation of this system involved a lot of try-and-error. Another cause for it is there are different approaches to implement one task. For example, there are a lot of ways to store data. Among them are ADO, ADO.Net, *Struct*, notepad, Microsoft Access, MySQL, and so many more. I had to try some of them that I think I could use. I ended up using Microsoft Access and ADO for various reasons.

Another example is the approaches to take in producing the drawings. VB.Net allows drawing directly onto a picturebox, or a graphic surface, or a bitmap object. Another way is storing the pictures in a database and import them when needed. The method used for this system is drawing everything on a bitmap object and transferring it to the picturebox to display it. The output is better than drawing it directly onto the picturebox and more dynamic than importing pictures from database.

7.5.3 Repetition of Codes

There are some redundant codes that I failed to separate into functions or class. I tried but errors were encountered. Therefore, I left them as they are.

7.6 Knowledge and Experience Gained

During the development of the Tutor on Software Design, I gained a lot of knowledge and experience. The following are some of the knowledge I gained after developing this system:

7.6.1 Programming Experience

In order to implement this system, I had to start from scratch, programming-wise. Therefore, I got the experience of exploring the language that I never knew. I had the experience of putting the language that I just learned to use. Now I know about data manipulation and drawing using VB.Net.

7.6.2 First Hand Experience of SDLC

I am all too familiar with Software Development Lifecycle from various subjects that I took, namely, Software Engineering, Project Management, Software Requirements Engineering, and Software Quality. However, I have never been involved in actually applying that knowledge, even when I was doing my industrial training. Thanks to this project, I thoroughly understand what I learned.

7.6.3 Sharing Opinions

Along the SDLC, opinions and ideas were given and taken. Opinions go back and forth between team members, with lecturers, and among unknown friends in the Internet forum.

7.6.4 Self Expression

Developing Tutor on Software Design has really given me a great chance to express myself in designing and coding. Finally, before I graduate, I have a chance to build real application software myself. Now, I know more than just theory. Doing this project has greatly improved my self-esteem and self-confidence.

7.6.5 Thesis Making

Through this project, I understand the process of completing a thesis. I realize that there is more to software development than programming. I understand that research is essential to ensure the quality of the produced software.

7.6.6 Experience of Working Under Pressure

The pressure comes from the deadline. It is stressful to finish the system and report by a certain date. Though I feel I can do more for this system, I could not because of the time constraint. I had to prioritize everything that I have to do, and do it from the top down. I had to be satisfied with what I managed to finish. However, this makes me realize the importance of time management.

7.6.7 Independence

I did not have a certain teacher to teach me VB.Net. Instead, I had to find my own resources. I started with books, Internet sources, and then I asked around. It really makes me feel independent.

7.7 Conclusion

This chapter describes about the system in terms of its strengths, its constraints and limitations, and enhancement that could be done in the future. It also lists down the problems encountered in developing this system and they are solved, and knowledge and experience gained along the whole SDLC.

This system is able to help students learn about drawing a sequence diagram. It is easier to use than existing tools to draw the diagram. Hopefully, enhancement will be done to make this system better.

Bibliography

Whitten, J. L., Bentley, L. D. and Dittman, K. C (2002). System Analysis and Design Methods, 5th Ed., McGraw Hill.

Sommerville, I (2001). Software Engineering, 6th Ed., Addison Wesley.

Maciaszek, L. A (2001). Requirements Analysis and System Design : Developing Information System with UML, Addison Wesley.

Olson, D. L (2001). Information System Project Management, McGraw Hill.

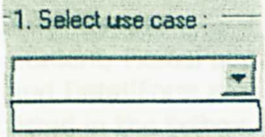
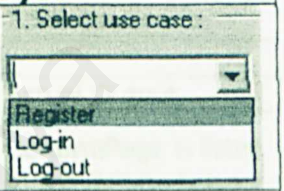
Appendix A

Test Cases

Test Cases for Tutor on Software Design

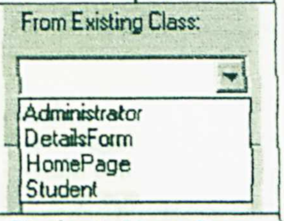
Ditulis oleh: **Fariza binti Halim** Tarikh: **May 16, 2004**

Module: **Use Case**

Unit:	**Select Use Case**		
Assumption:	**Database is blank (First time use)**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Click the uppermost drop down list.	Nothing is listed	 1. Select use case :
Assumption:	**Use cases have been added with no class or sequence yet**		
2.	Click the uppermost drop down list.	All use cases are listed	 1. Select use case : Register Log-in Log-out
3.	Click 1 use case name. Use case name: Register	All controls on the page are enabled displaying any class or sequence associated with the use case previously.	All controls on the page are enabled and blank.

Unit:	**Add New Use Case**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Click *Add New* button	Use Case page is displayed.	Use Case page is displayed.
2. 3. 4.	Type a use case name. Use case name: Register Click *Add* button Repeat steps 2 and 3 twice. Use case names: Log-in and Log-out	Newly entered use case names are listed.	Register, Log-in and Log-out are listed.
5. 6.	Click *Sequence* tab Click the uppermost drop down list	Newly entered use case names are listed.	Register, Log-in and Log-out are listed.

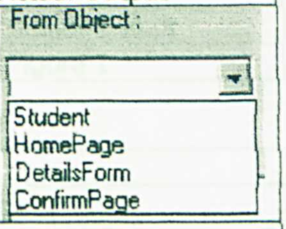
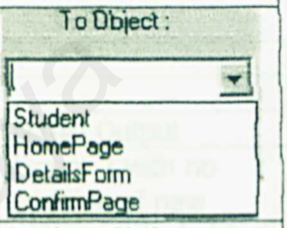
Module: **Class**

Unit:	**Select Class**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Click the second drop down list	Existing classes are listed	 From Existing Class: Administrator DetailsForm HomePage Student
2.	Choose 3 classes Classes names: Student, HomePage and DetailForm	The selected classes are listed in the listbox.	Student, HomePage and DetailForm are listed in the listbox.

Unit:	**Add New Class**		
No.	Testing Procedure	Expected Output	Actual Output
1. 2.	Type a class name in the *Create New* textbox. Class name: ConfirmPage Click *Add*	The newly added class is listed in the listbox and drop down list.	ConfirmPage is listed in the listbox but not in drop down list.

Unit:	**Remove Class**		
No.	Testing Procedure	Expected Output	Actual Output
1. 2.	Type a dummy class name in the *Create New* textbox Class name: Anything Click *Add*	The newly added class is listed.	Anything is listed.
3. 4.	Click Anything in the listbox Click *Remove*	A confirmation dialog box is displayed.	A confirmation dialog box is displayed.
5.	Click *Cancel*	Nothing happens.	Nothing happens.
6.	Click *OK*	The selected class is deleted.	Anything is no longer listed.

Module: **Sequence**

Unit:	**Add Sequence**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Click left drop down list in the third section	All added classes should be listed.	 From Object : Student HomePage DetailsForm ConfirmPage
2. 3.	Choose a class Class name: Student Type a message in the centre textbox Message: Click Register		
4.	Click right drop down list	All added classes should be listed.	 To Object : Student HomePage DetailsForm ConfirmPage
5. 6.	Choose a class Class name: HomePage Click Add to *Sequence button*	The sequence should be listed.	Student Click Register HomePage is listed.

Unit:	**Delete Sequence**		
Assumption:	**8 sequences have been added**		
No.	Testing Procedure	Expected Output	Actual Output
1. 2.	Choose 1 sequence Sequence: ConfirmPage Student Click *Remove*	A confirmation dialog box is diplayed.	A confirmation dialog box is diplayed.
3.	Click *Cancel*	Nothing happens.	Nothing happens.
4.	Repeat steps 1 and 2	A confirmation dialog box is diplayed.	A confirmation dialog box is diplayed.
5.	Click *OK*	The selected sequence is deleted.	ConfirmPage Student is no longer listed.

Module: **Drawing**

Unit:	**View Sequence Diagram**		
Assumption:	**No sequence has been added**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Click Sequence Diagram tab	A blank page	Figure 1

Unit:	**View Sequence Diagram**		
Assumption:	**8 sequences have been added**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Click Sequence Diagram tab	A page displaying a sequence diagram.	Figure 2

Unit:	**View Sequence Diagram**		
Assumption:	**1 sequence and 1 object class more added**		
No.	Testing Procedure	Expected Output	Actual Output
1.	Click Sequence Diagram tab	A page displaying a sequence diagram.	Figure 2 with no addition of new sequence and object class.

Unit:	**Save Sequence Diagram**		
No.	Testing Procedure	Expected Output	Actual Output
1.	With Sequence Diagram tab selected, click File -> Save Sequence	A standard save dialog is displayed.	A standard save dialog is displayed.
2. 3. 4. 5.	Choose a folder Folder: Pic Try on Desktop Type a name for the diagram Name: sequence1 Choose an image format Format: BMP Click Save	Diagram is saved in the selected format with the chosen name at the selected location.	A diagram named sequence1 is saved as sequence1.bmp in Pic Try folder on desktop.
6.	Repeat steps 1 to 5, 8 times. Each time with different image formats. Image formats: JPEG, GIF, Icon, EMF, WMF, PNG, TIFF and Exif.	Diagram is saved in the selected formats with the chosen name at the selected location.	A diagram named sequence1 is saved as sequence1.jpg, sequence1.gif, sequence1.ico, sequence1.emf, sequence1.wmf, sequence1..png and sequence1.tif in Pic Try folder on desktop. sequence1.exif cannot be viewed.

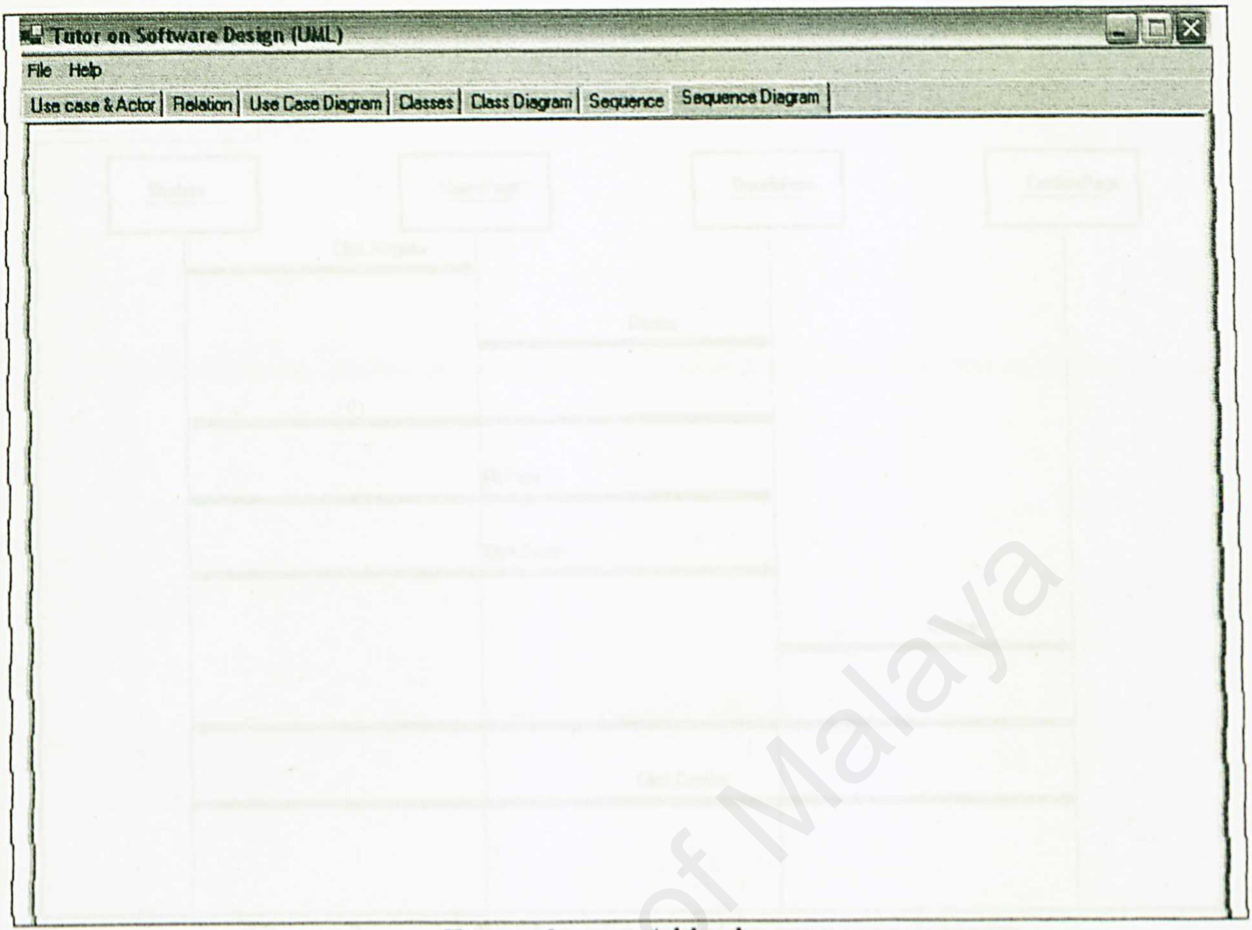


Figure 1 A blank page

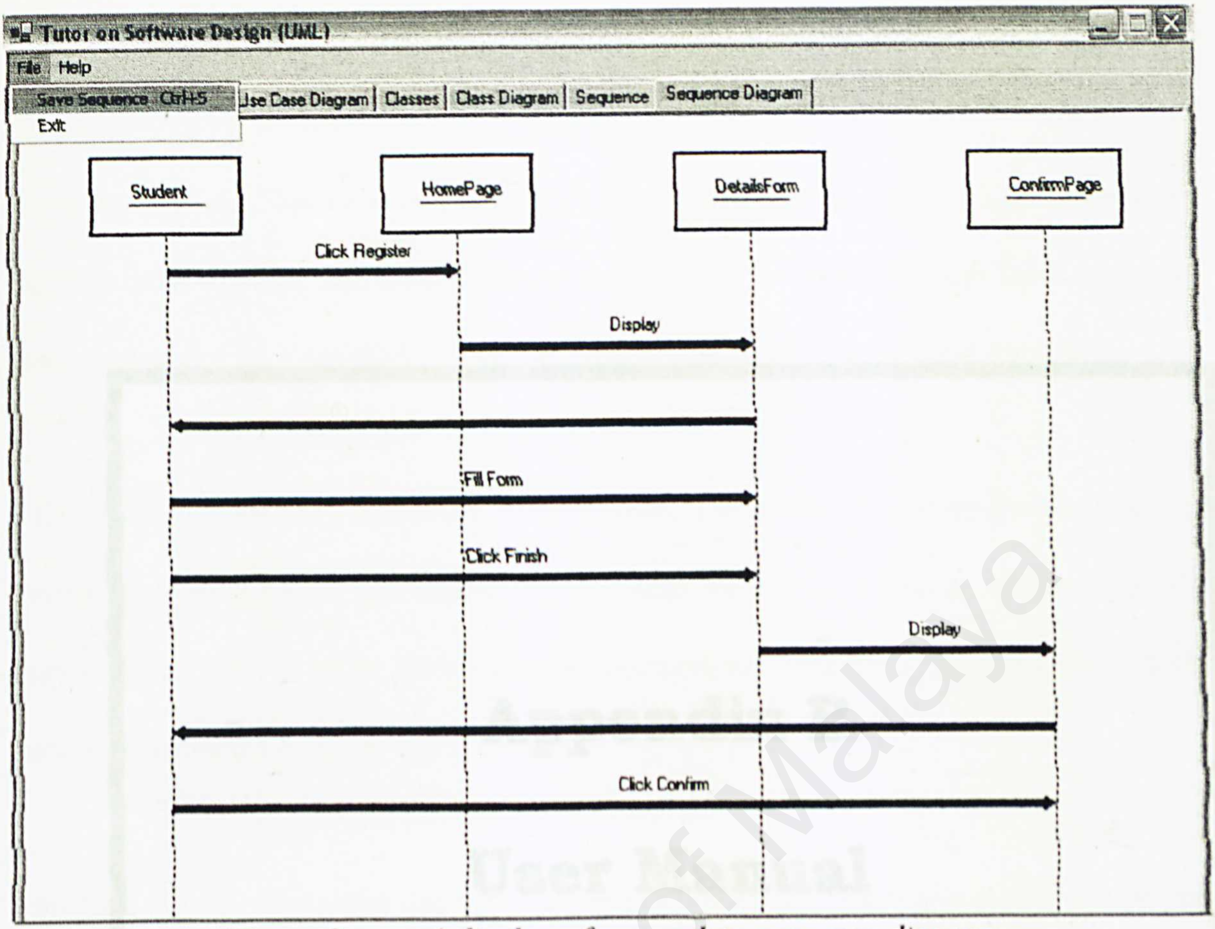


Figure 2 A display of a complete sequence diagram

Appendix B

User Manual

1.0 Introduction

Tutor on Software Design is a system that shows users how to draw a sequence diagram for their own software and actually draw the diagram for them. Though there are many systems that can do more than this system, this system applies a different approach.

Other software use drag and drop method where users have to decide the sequence of events for their software and where each symbols should go. Tutor on Software Design only requires users to enter their software requirements, and the system will show them how their sequence diagram will look like.

This user manual document describes the requirements necessary to run the system and how to use the system.

2.0 System Specification

This section lists down hardware and software requirements necessary in order to use Tutor on Software Design.

2.1 Hardware Specification

- ❑ Intel Pentium 100 Mhz
- ❑ Memory 64 MB RAM
- ❑ Hard disk 100 MB space
- ❑ Mouse
- ❑ Keyboard

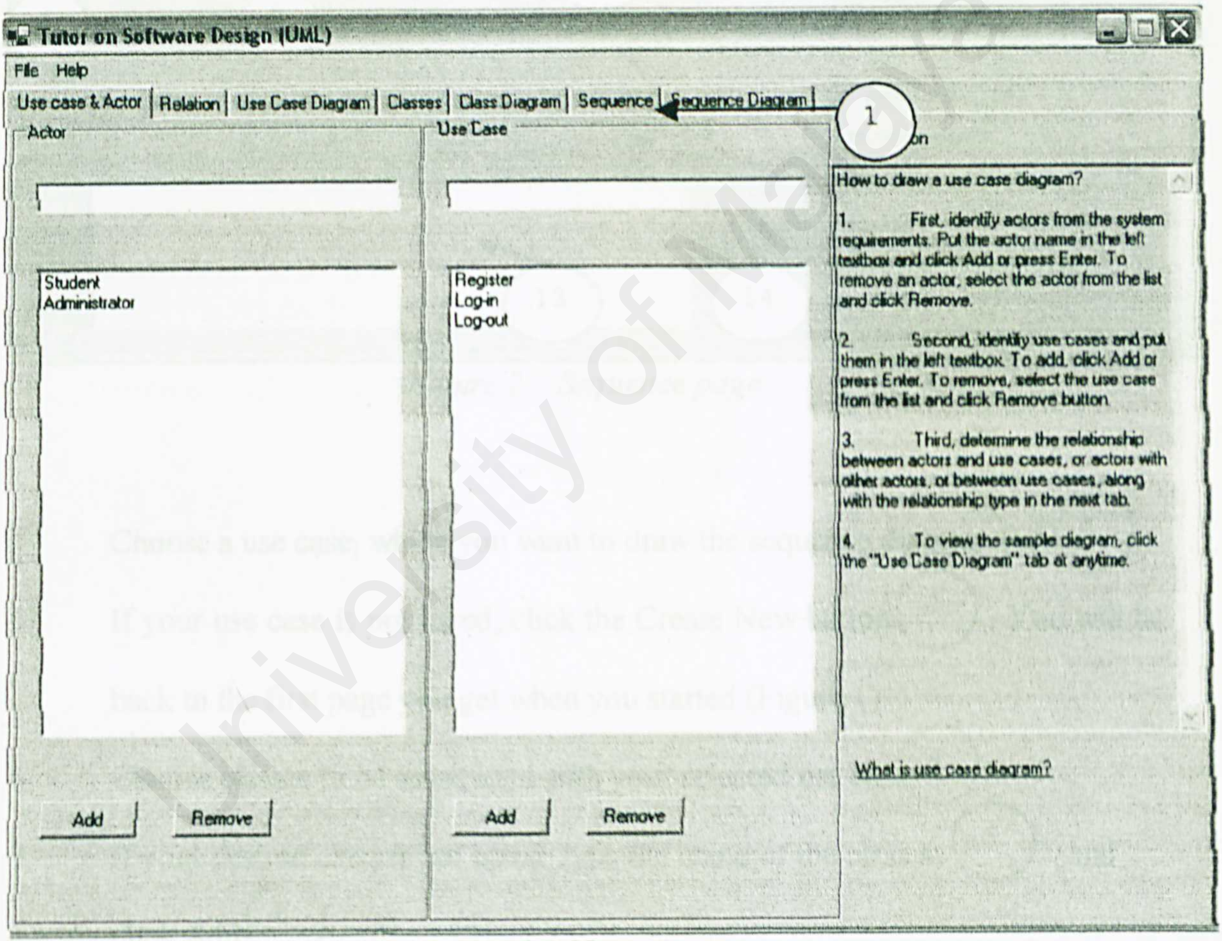
2.2 Software Specification

- ❑ Windows 98/ME/XPoperating system
- ❑ Microsoft Access

3.0 User Guide

This section will show you how to use Tutor on Software Design to learn to draw a sequence diagram for your own system development.

1. Double click on the Tutor on Software Design's icon.
2. You will get the use case page. To draw the sequence diagram, click on the Sequence tab. 1



How to draw a use case diagram?

1. First, identify actors from the system requirements. Put the actor name in the left textbox and click Add or press Enter. To remove an actor, select the actor from the list and click Remove.

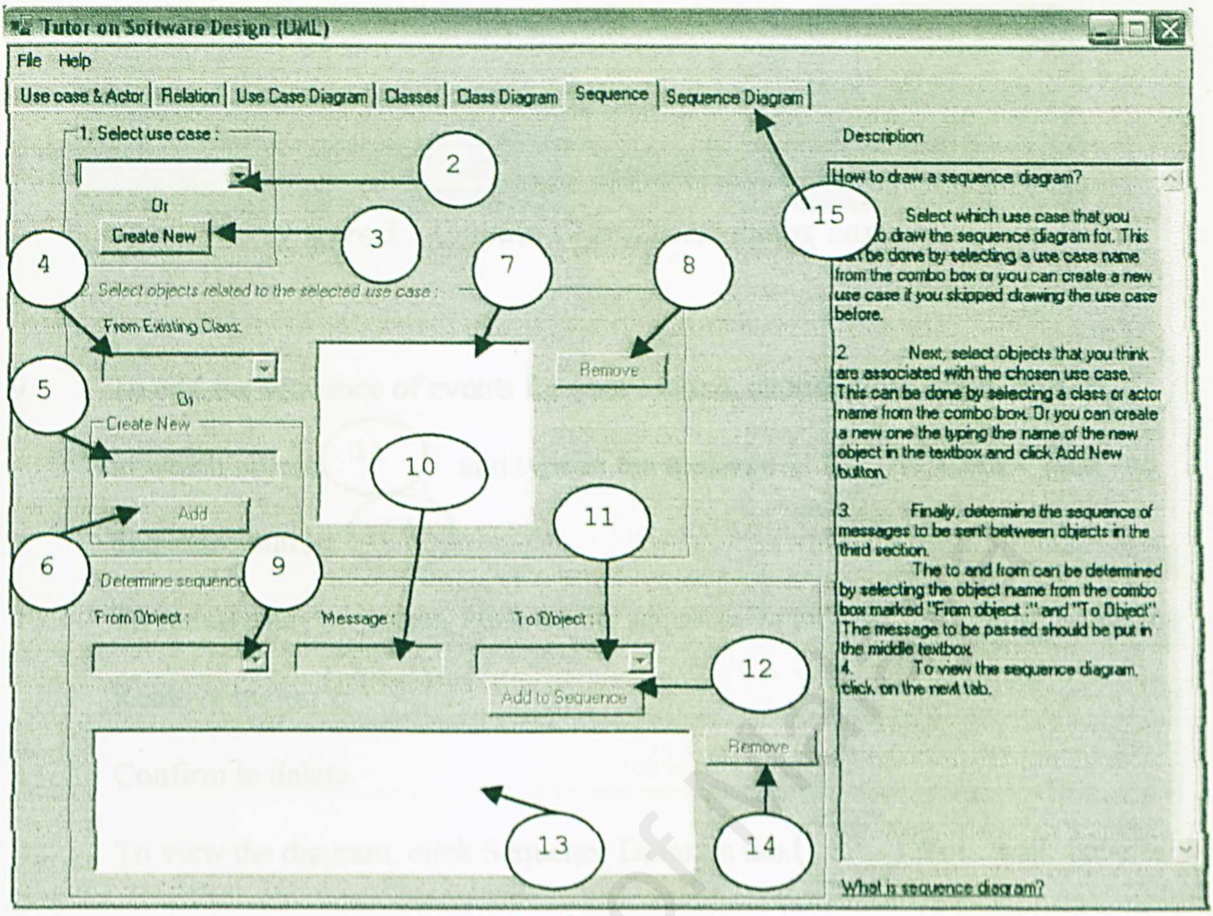
2. Second, identify use cases and put them in the left textbox. To add, click Add or press Enter. To remove, select the use case from the list and click Remove button.

3. Third, determine the relationship between actors and use cases, or actors with other actors, or between use cases, along with the relationship type in the next tab.

4. To view the sample diagram, click the "Use Case Diagram" tab at anytime.

Figure 1 Use case page

3. You will get a page like Figure 2.



How to draw a sequence diagram?

Select which use case that you [illegible] draw the sequence diagram for. This [illegible] be done by selecting a use case name from the combo box or you can create a new use case if you skipped drawing the use case before.

2. Next, select objects that you think are associated with the chosen use case. This can be done by selecting a class or actor name from the combo box. Or you can create a new one the typing the name of the new object in the textbox and click Add New button.

3. Finally, determine the sequence of messages to be sent between objects in the third section.

The to and from can be determined by selecting the object name from the combo box marked "From object :" and "To Object". The message to be passed should be put in the middle textbox.

4. To view the sequence diagram, click on the next tab.

Figure 2 Sequence page

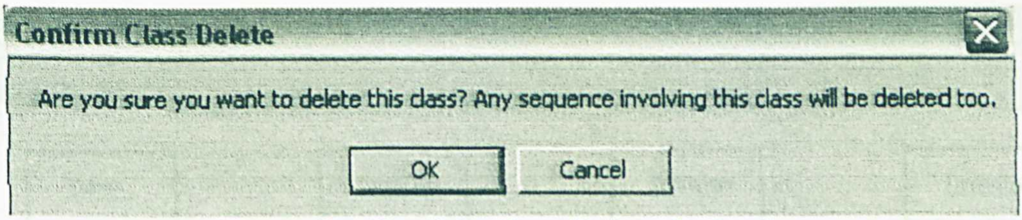
4. Choose a use case, which you want to draw the sequence diagram for. ②

5. If your use case is not listed, click the Create New button ③. You will be back to the first page you get when you started (Figure 1).

6. Choose classes to be associated with your selected use cas④

7. If your desired class is not listed, type the name of the class at ⑤ and click Add.⑥

8. To remove a class, click on the class name in the list ⑦ and click Remove button.⑧ This will affect any sequence involving the removed class. Therefore, you will be asked to confirm your deletion. Click Ok to confirm or Cancel if not to confirm.



Confirm Class Delete

Are you sure you want to delete this class? Any sequence involving this class will be deleted too.

Figure 3 Confirm Class Delete dialog box

9. To add the sequence of events for your system, choose from which object 9 to which object 11 and type in the message 10. Click Add to Sequence button. 12

10. To remove any sequence, click on the sequence in the lis 13 and click Remove button. 14

11. Confirm to delete.

12. To view the diagram, click Sequence Diagram tab. 15 You will have a page like shown in Figure 4.

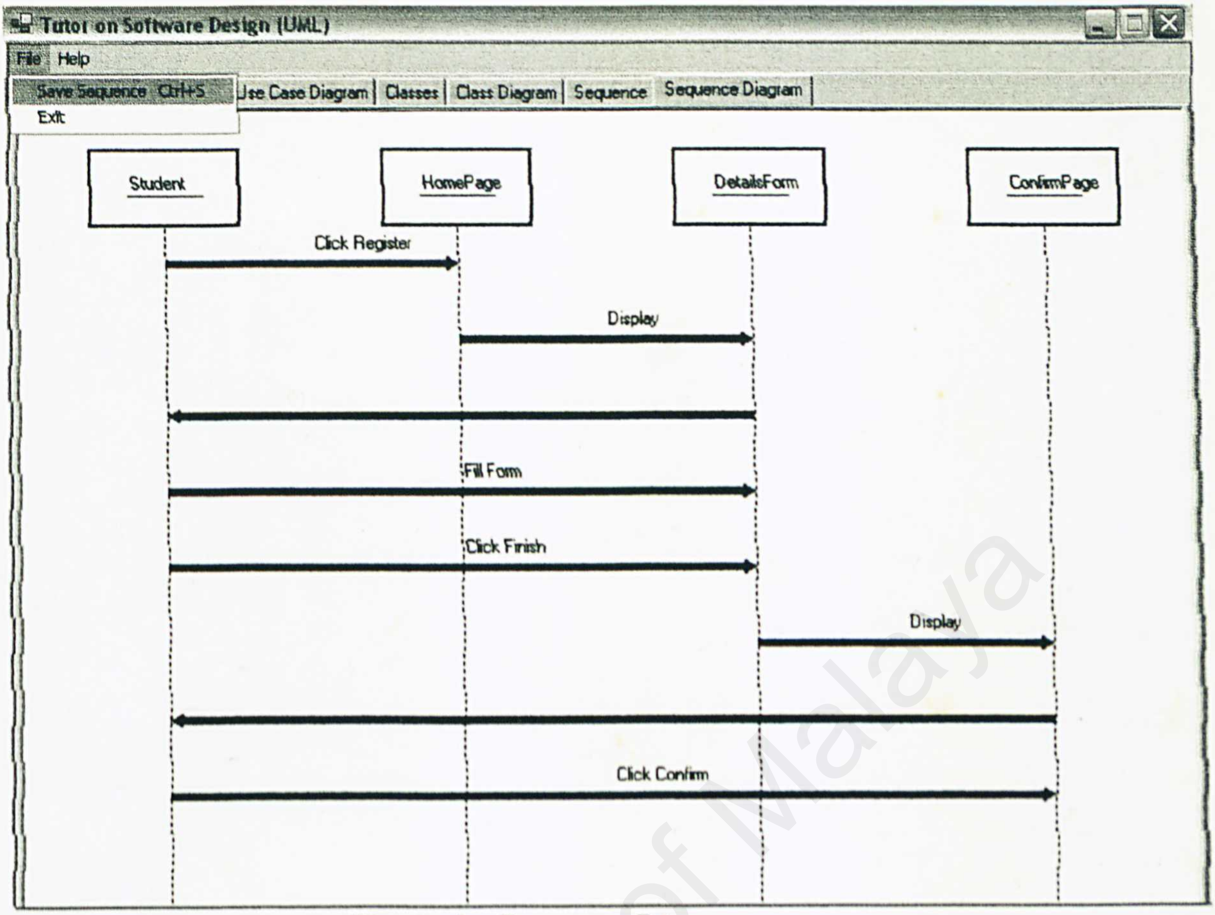


Figure 4 Sequence Diagram page

13. To save the produced diagram, click File -> Save Sequence. Choose a folder, set a name, choose an image format and click save.

14. To exit, click cross at the top right corner or click File -> exit.

4.0 Summary

This user manual is actually not needed, as the system is very easy to understand. However, this document is made for references. Hopefully, Tutor on Software Design has been a great help.