

DEVELOPMENT OF DOE SOFTWARE FOR SINGLE-FACTOR EXPERIMENTS:  
THE ANALYSIS OF VARIANCE;  
CASE OF COMPLETELY RANDOMIZED DESIGN

Oleh

Muzammer bin Zakaria

Disertasi ini dikemukakan kepada  
UNIVERSITI SAINS MALAYSIA

Sebagai memenuhi sebahagian daripada syarat keperluan  
untuk ijazah dengan kepujian

SARJANA MUDA KEJURUTERAAN (KEJURUTERAAN MEKATRONIK)

Pusat Pengajian Kejuruteraan  
Elektrik dan Elektronik  
Universiti Sains Malaysia  
2006

Mei

## **ABSTRACT**

Design of Experiment (DOE) is a process to study the variables that affecting a certain process. Nowadays, there are not many people know about DOE. From my findings in the internet while studying this subject, I have found that DOE is not only be used in manufacturing field, but also used in many other fields.

Since, the scope of applying DOE is becoming wider, it is not impossible if next few years everyone will study about it. As we can see, software about DOE is very few this days, the users always use the statistical software to solve the DOE problems, which is not so relevant or difficult to handle.

Therefore, in this project I try to develop DOE software in case of single factor design of experiment. I have used the C++ Builder program to build this software.

## ABSTRAK

Rekabentuk ujikaji ialah satu cara untuk mengkaji sesuatu proses mengenalpasti pembolehubah-pembolehubah yang mempengaruhi sesuatu proses. Pada masa sekarang tidak ramai yang mengetahui tentang rekabentuk ujikaji ini. Dari penemuan-penemuan saya sepanjang belajar mengenai rekabentuk ujikaji ini, saya telah mendapati bahawa rekabentuk ujikaji ini telah digunakan bukan sahaja dalam bidang perusahaan, malah dalam pelbagai bidang lain lagi. Sebagai contoh, di barat rekabentuk ujikaji telah digunakan untuk mengkaji atau mengenalpasti soalan-soalan yang relevan di tulis di dalam kertas peperiksaan bagi menguji tahap pemikiran pelajar, ujikaji ini telah dijalankan di salah sebuah universiti di sana.

Memandangkan penggunaan rekabentuk ujikaji yang semakin meluas sekarang jadi adalah tidak mustahil jika pada tahun-tahun yang mendatang, setiap orang akan mempelajari tentang rekabentuk ujikaji. Buat masa sekarang perisian mengenai rekabentuk ujikaji adalah terhad atau tidak banyak di pasaran. Kebiasaannya perisian statistik digunakan untuk menyelesaikan masalah tentang rekabentuk ujikaji, yang mana ia adalah kurang relevan dan membebankan.

Jadi di dalam projek ini saya cuba merekabentuk satu perisian mengenai rekabentuk ujikaji dalam kes satu faktor. Saya telah menggunakan perisian C++ Builder dalam merekabentuk perisian ini.

## **ACKNOWLEDGEMENT**

Assalamualaikum w.b.t.

Here, I want to thank to Allah the Almighty because with His blessed then I am able to finish my Final Year Project with success.

First of all, I would like to take this opportunity to thank my supervisor Dr. Zalina Abd. Aziz for all her guide and advices that she has gives to me in order to help me complete this project.

Secondly, I would also like to thank my friends for all their support and ideas while doing this project. Last but not least, I owe the greatest debt to my parents, without whose constant support and cooperation in order to encourage me not to give up.

Finally, I wish to take this opportunity again to thank people who involved and also for their constructive criticisms and helpful comments. I really appreciate for all of that.

## CONTENTS

	PAGE
ABSTRACT	i
ABSTRAK	ii
ACKNOWLEDGEMENT	iii
CHAPTER 1 INTRODUCTION	
1.1 Background	1
1.2 Objectives	2
1.3 Scope Of Research	2
1.4 Methodology	3
1.5 Gantt Chart	5
1.6 Structure of Thesis	6
CHAPTER 2 LITERATURE REVIEW	
2.1 Findings	7
2.1.1 Introduction	7
2.1.2 What is DOE	7
2.1.2.1 Analysis of Variance	7
2.1.2.2 Unbalanced Data	9
2.1.2.3 Residuals Plot	9
2.1.3 Software Process Model (waterfall model)	10
2.1.4 What is C++ Builder	11
2.1.5 What is UML	15
2.2 Questionnaire	18
2.2.1 Software Usability	18
CHAPTER 3 SOFTWARE DESIGN AND TESTING	
3.1 Design	
3.1.1 Introduction	19
3.1.2 Software Design Development	

3.1.2.1	UML Design	20
3.1.2.2	Software Interfacing	23
3.2	Testing	
3.2.1	Introduction	26
3.2.2	Unit Testing	26
3.2.3	Usability Testing	27
3.2.3.1	SUMI	27
CHAPTER 4	RESULTS AND DISCUSSION	
4.1	Introduction	31
4.2	Problems and analysis	31
4.3	Results and analysis	32
4.3.1	Analysis of Feedback Form	32
4.3.2	Analysis from the completed Feedback Form	34
4.4	Discussion	40
CHAPTER 5	CONCLUSION	
5.1	Introduction	41
5.2	Summary of the project	41
5.3	Recommendation on the project	42
5.4	Conclusions of the project	42
REFERENCES		43
APPENDIX A (Mathematical Equations)		44
APPENDIX B ( Gantt Chart)		45
APPENDIX C (Completed Questionnaire)		46

# **CHAPTER 1: INTRODUCTION**

## **1.1 BACKGROUND**

Nowadays, design of experimental (DOE) is recently use in our lives and especially in manufacturing or industry fields, it is either to get the confirmation or exploration (to get the best setting)of the certain processes. Many companies these days realize the importance of DOE in producing quality products. In DOE application, the statistical software's is use to determine the best setting or the factors that control the process in producing a quality or reliable products.

Therefore, this project is to develop software based on the application of design of experimental (DOE). In DOE there are many kind of design of experiment, in this project I am concentrated in experiments with a single factor: the analysis of variance: case of completely randomized design. In completely randomized design, the levels of the factor of the experiment are chosen randomly which is we consider that the population of factor levels is large enough.

Usually for any problem in DOE, we will use many kind of statistical software to solve the problem. Most of this statistical software is not user friendly for DOE users, where the users need to do many things or steps to get the answers for the problem. Therefore, this will make a simple things became hard or difficult. Seeing this kind of problems, I determined to develop DOE software concentrated on the design of experiments with a single factor.

From the completion of the project it is may be possible to develop the DOE software into a commercial product. When this project is complete, it may be possible to develop too many other design of experiment.

## 1.2 **OBJECTIVES**

The aim of this project is to develop DOE software in experiments with a single factor. The objectives in this project are:

- 1) To build and design a user friendly software (easy to use).
- 2) To build software that can analyze single factor experiments based on Completely Randomized Design.
- 3) Make a software that :
  - a) Capable to calculate and build ANOVA Table.
  - b) Capable to calculate Residuals and Plot Residuals versus Fitted value.

## 1.3 **SCOPE OF RESEARCH**

The analysis of the fixed effects models is concentrated for this software. The level of treatments are selected or specifically chosen by experimenter. The number of replicates is also specifically chosen by the experimenter.

In this software, the maximum numbers of levels are 100, which is, the users can only choose up until 100 levels of treatments. While for the replicates, the experimenter can choose until 5 replicates only.

For the calculation of the ANOVA (analysis of variance), this software can calculate and build until get the  $F_o$  value. The users need to see the F-table manually to make a conclusion, since, if



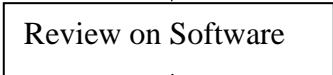
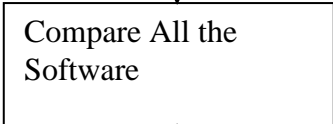
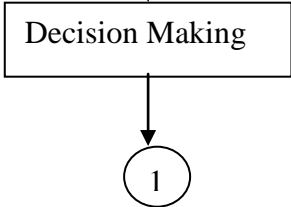
$$F_o > F_{\alpha, \alpha-1, N-a}$$

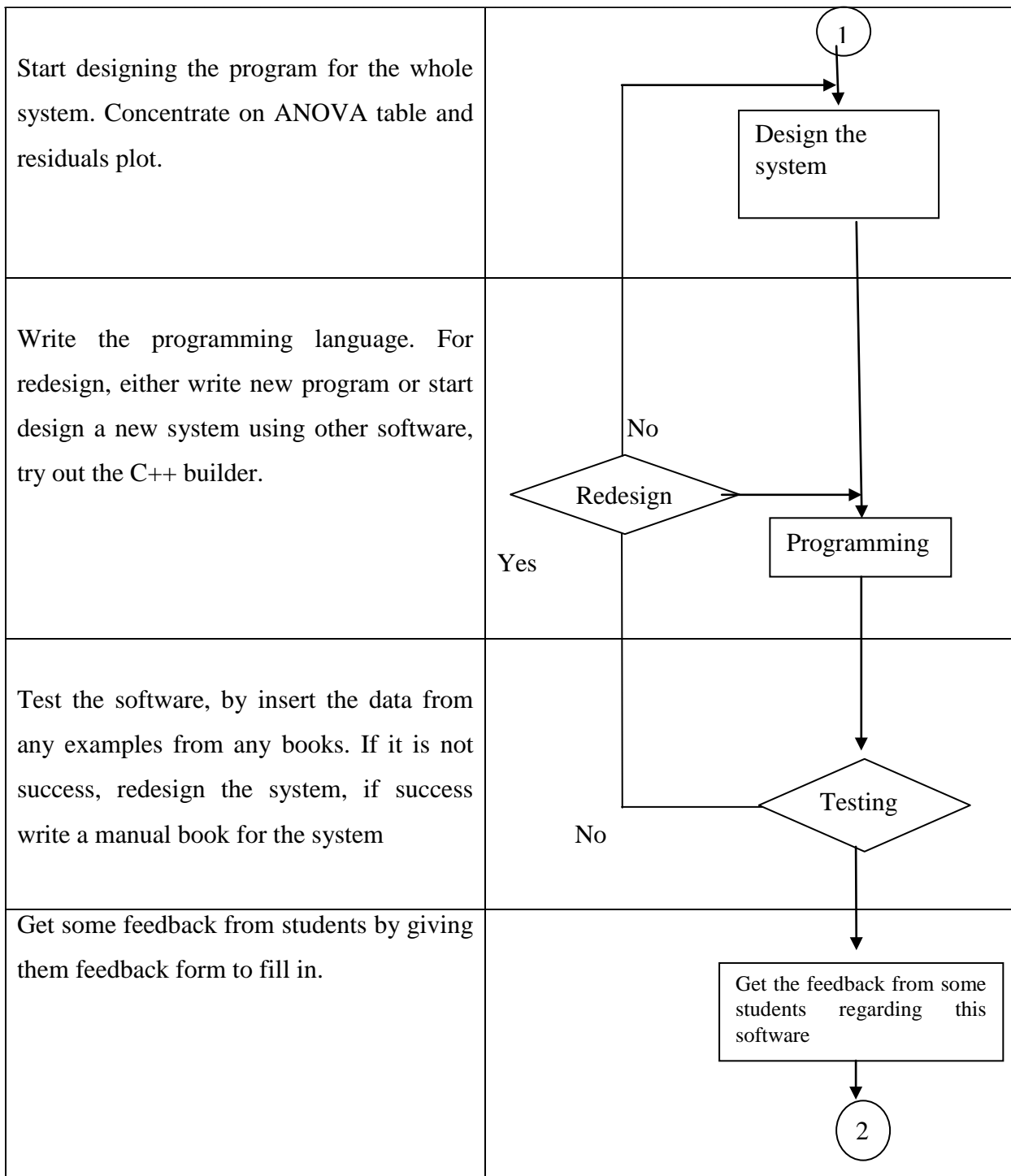
So conclude that the null hypothesis is rejected. Where the  $F_{\alpha, \alpha-1, N-a}$  value is from the F-table. In this case,  $\alpha$ , is the confidence interval,  $a$  is the level of treatments and  $N$  is the number of all data.

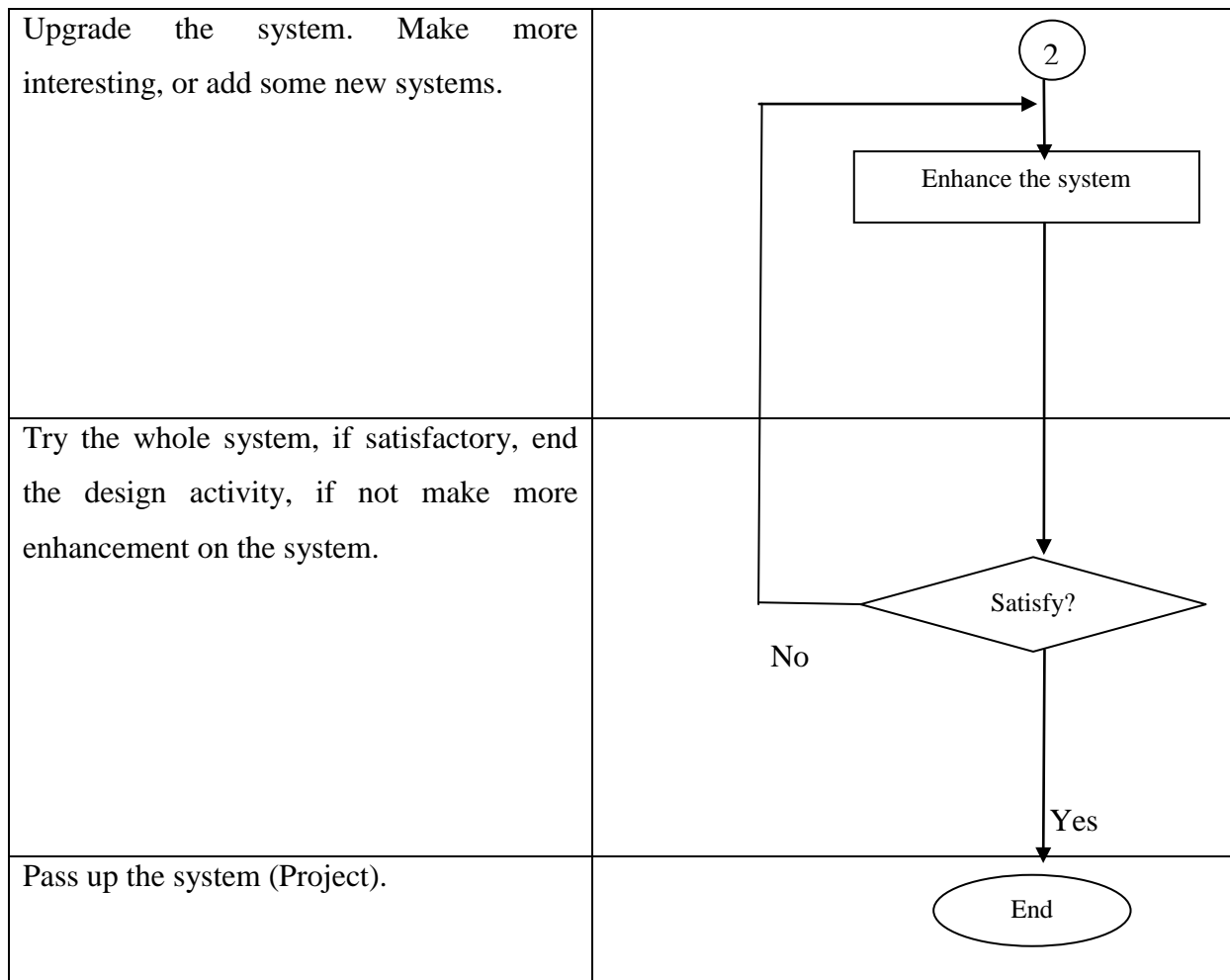
The software can calculate the residuals and plot the residuals versus fitted value. The residual plots that can be plotted is the residuals versus the averages.



## 1.4 METHODOLOGY

Remarks	Flow process
<p>The final year project is given:            'Development of DOE (Design Of Experimental) software for single-factor experiments; The Analysis of Variance; case of completely randomized design'.</p>	 <pre>           graph TD             START([START]) --&gt; PR[Project Review]           </pre>
<p>Study about the completely randomized design for single-factor experiments.</p>	 <pre>           graph TD             PR[Project Review] --&gt; ROS[Review on Software]           </pre>
<p>Search the statistical software that is available, study the manual for each of the software. Software that have been reviewed; Microsoft Excel, Minitab, Matlab and SPSS . I also review on C++ Builder and Visual Basics to make the programming.</p>	 <pre>           graph TD             ROS[Review on Software] --&gt; CAS[Compare All the Software]           </pre>
<p>Compare pros and contras for each software that have been reviewed</p>	 <pre>           graph TD             CAS[Compare All the Software] --&gt; DM[Decision Making]           </pre>
<p>Decide what programming language that I want to use. In this project I like to use Visual Basics or C++ Builder</p>	 <pre>           graph TD             DM[Decision Making] --&gt; END((1))           </pre>





**Figure 1.1:** Block Diagram of The Project.

## 1.5 GANTT CHART

To make sure this project follow the schedule, a Gantt chart have to be built. This is used as a guide to follow and to make sure that the project is finish before the final day. See the appendixes, at the end of this report. See appendix B.

## **1.6 STRUCTURE OF THESIS**

### **Chapter 1**

This chapter introduces the project, which is talk about the background and the objectives of the project. The scope of research also is mention in this chapter, where it notifies the scope that had been cover in the software. The methodology of the project and Gantt Chart also shown in this chapter.

### **Chapter 2**

Chapter 2 is the literature review, which is mention about the DOE, Software process model, C++ Builder, and UML. In this chapter it notify about the DOE, especially, which had been cover for the project. Where, this chapter shows the software process model where the project uses the waterfall model. In addition, chapter 2 tells about the software usability, which is use for make the questionnaire or feedback form.

### **Chapter 3**

Chapter 3 will cover on the software design and testing. For the software design, we can see in this chapter talk about the UML design and design of the software interfacing using the C++ Builder. While, for the software testing, it will tell about the unit has been testing and usability testing. This is to determine the capability of the software.

### **Chapter 4**

The results from the testing are shoe ninth is chapter, where the result from the feedback form is analyzed. The results are analyzed using the quality tools such as check sheet, histogram, and Pareto analysis. This is to trace out the cause of the problems. The result will determine the usability of the software and can determine either the objectives can be achieve or not.

### **Chapter 5**

The last chapter is discussing about the summary of the project. Also, mention about the recommendation to better improvement of the project. In chapter 5, the conclusion of the project is been make.

## **CHAPTER 2: LITERATURE REVIEW**

### **2.1 FINDINGS**

#### **2.1.1 Introduction**

In chapter 2, I will tell about the findings in making this software. The findings that I made in making this software are in DOE which is single factor design of experiment, finding in C++ builder which is to interfacing the software and finding in UML that help me in visualizing, specifying, construction, and documenting of software system.

#### **2.1.2 WHAT IS DOE**

A Design of Experiment (DOE) is a structured, organized method for determining the relationship between factors (Xs) affecting a process and the output of that process (Y).

Other Definitions:

- 1 - Conducting and analyzing controlled tests to evaluate the factors that control the value of a parameter or group of parameters.
- 2- "Design of Experiments" (DOE) refers to experimental methods used to quantify indeterminate measurements of factors and interactions between factors statistically through observance of forced changes made methodically as directed by mathematically systematic tables.

##### **2.1.2.1 Analysis Of Variance**

This software is about the design and analysis of single-factor experiments with  $a$  levels of factor (or  $a$  treatments). We will assume that the experiment has been completely randomized. This randomized test sequence is necessary to prevent the effects of unknown nuisance variables, perhaps varying out of control during experiment, from contaminating the results.

**Table 2.1** The typical data for a single factor experiment

Treatment (level)	Observations				Totals	Averages
1	y11	y12	.....	y1n	y1.	$\hat{y} 1.$
2	y12	y22	.....		y2.	$\hat{y} 2.$
.	y2n				.	.
.	.	.	.....	.	.	.
<i>a</i>	.	.	.....	.	.	.
	ya1	ya2	.....	yan	ya.	$\hat{y} a.$
Total					y..	$\hat{y}..$

From table 2.1, *a* is a different levels of a single factor or treatments. While, *n* is the observations.

For this project, the software of the single-factor analysis of variance for the fixed effects model is developed.

**Table 2.2:** The analysis of variance table for a single factor, fixed effects model.

Source of variation	Sum of squares	Degree of freedom	Mean square	Fo
Between treatments	SS Treatments	a - 1	MS Treatments	Fo = MSTreatments/MSE
Error (within treatments)	SSE = SST-SS Treatments	N - a	MSE	
Total	SST	N - 1		

### 2.1.2.2 Unbalanced data

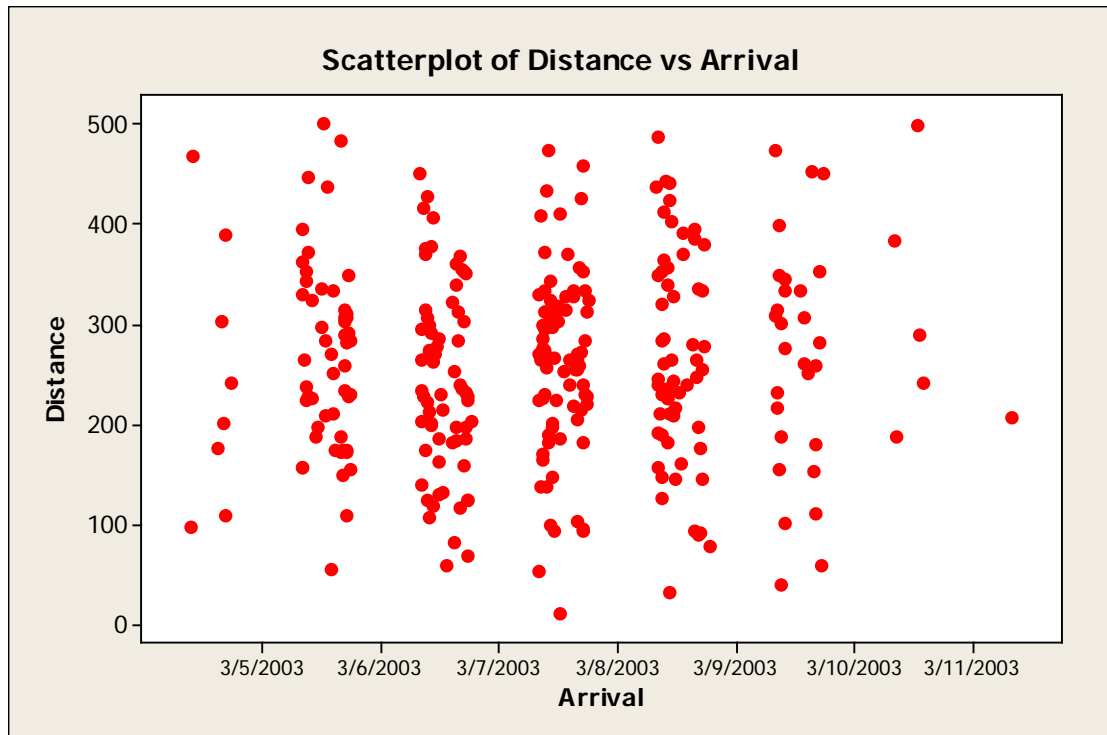
In some special case, the design maybe unbalanced which is the number of observations taken within each treatment maybe different.

### 2.1.2.3 Residuals plot

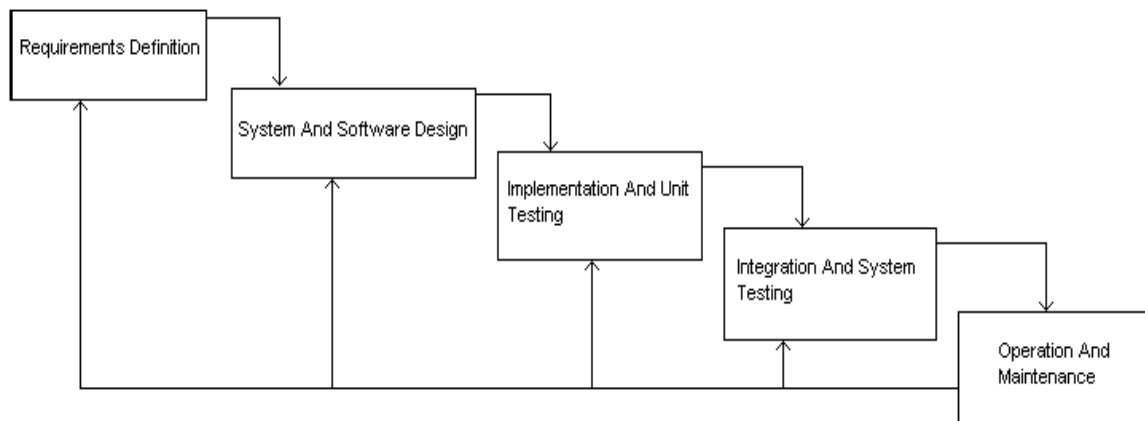
Plotting the residuals in time order of data collection is helpful in detecting correlation between the residuals. A tendency to have runs in positive and negative residuals indicate positive

Plotting residuals versus fitted value is also important to check if the residuals have any relations to any variables. If the residuals are unrelated to any variables, the residuals plot should be structureless.

**Figure 2.1:** The example of residuals plot (DOUGLAS C. MONTGOMERY – 1983)



### 2.1.2 The Waterfall Model:



**Figure 2.2:** The waterfall model

Figure 2.2 above shows the waterfall model in presenting the project flow.

For waterfall model:

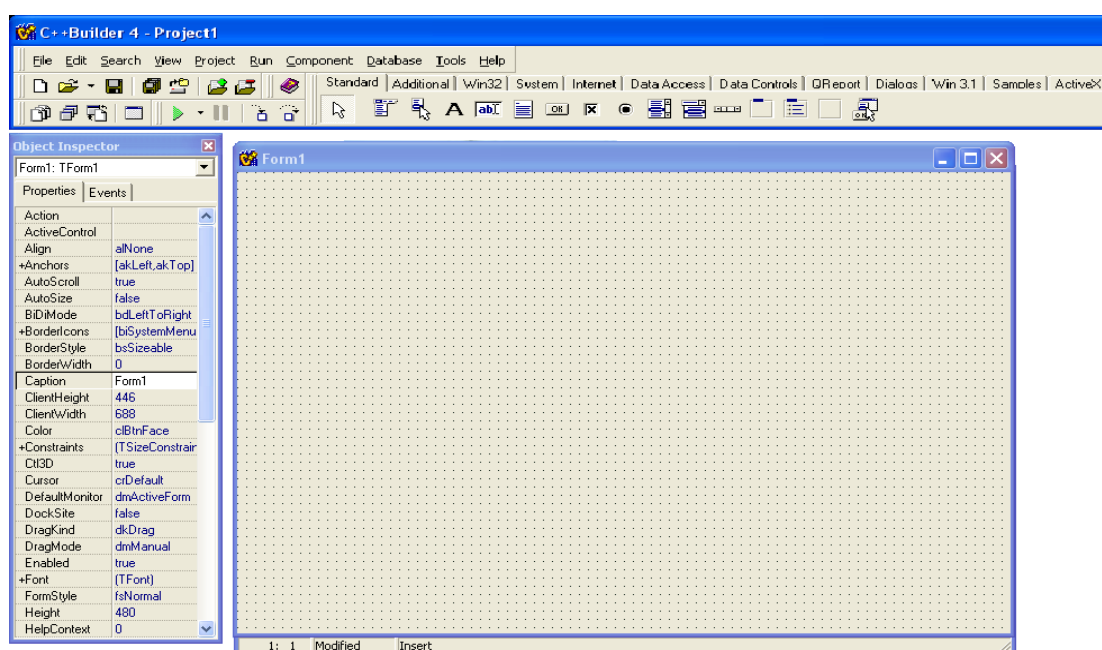
- 1) Many variations exist
- 2) Consists of a number of clear and distinct phases
- 3) One or more documents are produced and “signed off” and after each phase (documentation driven)

The waterfall model is a simple management model.



### 2.1.3 WHAT IS C++ BUILDER

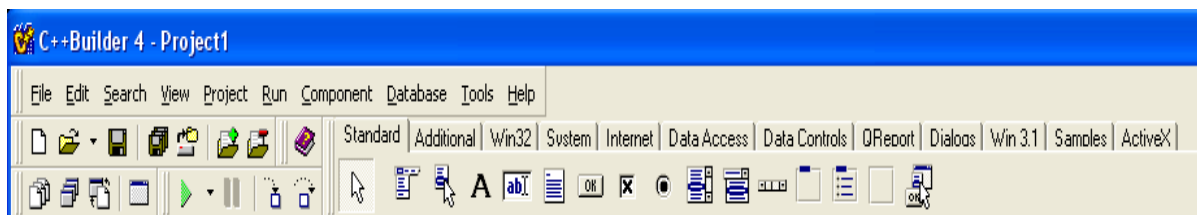
C++ builder is an application on building windows programs. With this software, we can create the user interface to a program (the user interface means the menus, dialog boxes, main window, and so on) using drag-and-drop techniques for true rapid application development. When we start the C++ program, both blank form and the IDE (integrated development environment) are present. See Figure 2.3.



**Figure 2.3:** Main window of C++ builder.

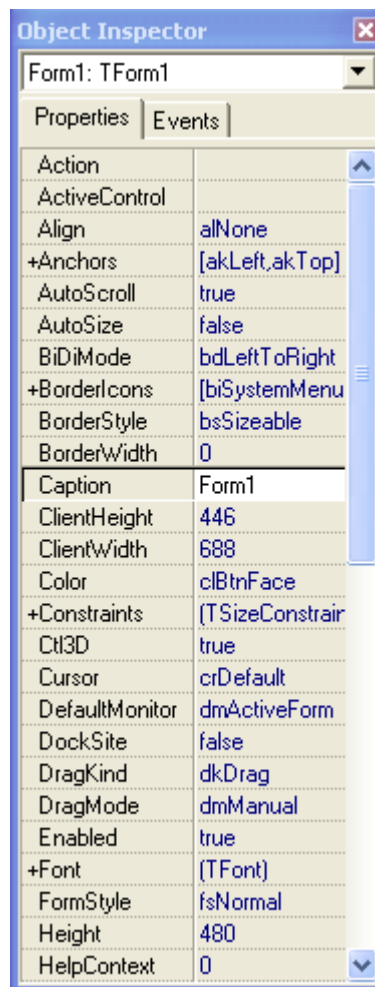
The C++ builder IDE is dividing into three parts. The top window that can be considered as the main window, which is contains the toolbar that give us one-click access to tasks such as opening, saving, and compiling projects. The component palette contains a wide array of components (such as buttons, list boxes, text labels and the like) that you can drop onto the forms. Note that, a component is a self-contained binary piece of software that performs some specific predefined function, such as a text label, an edit control, or a list box. For convenience, the components are dividing into groups, as you can see figure 2.4 below.

**Figure 2.4:** The toolbar



To place the component on the form, we simply click the component's button in the components palette and then click on the form where we want the components to appear.

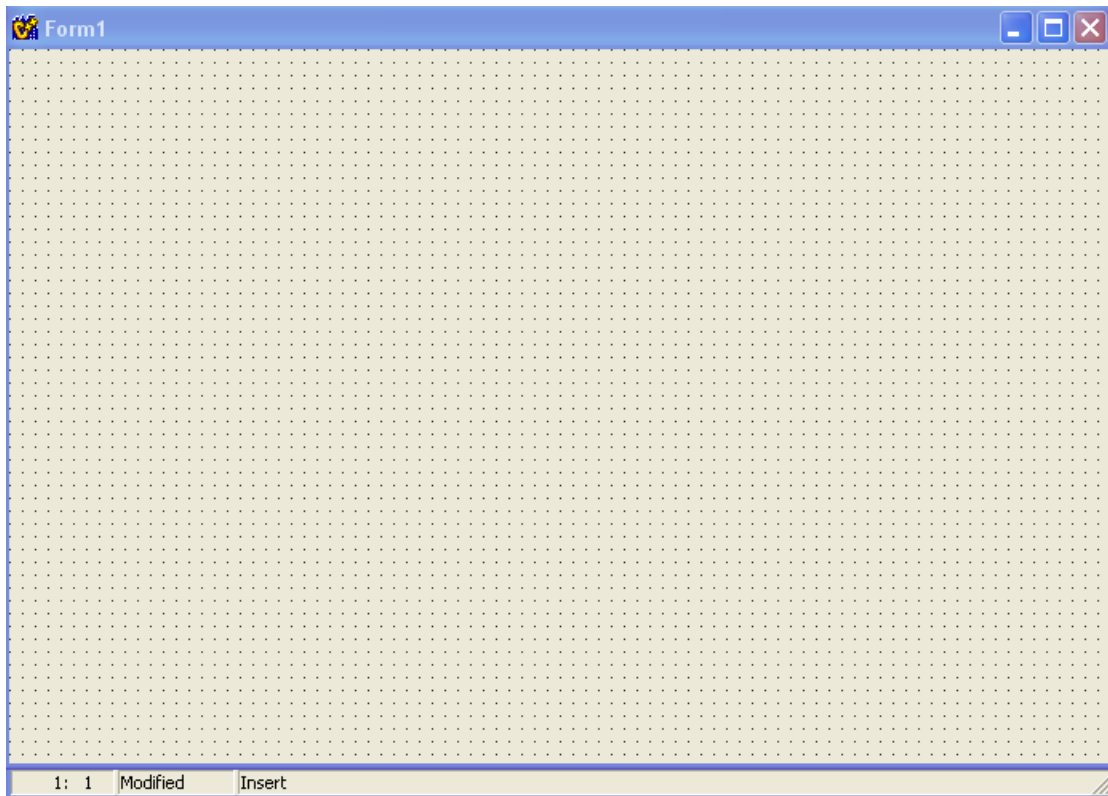
Below the toolbar and the component palette and on the left side of the screen is the object inspector. We modify the component's properties and events through this object inspector. A component's properties control how the component operates.



**Figure 2.5:** Object Inspector

Usually the objects inspector has an events tab in addition to the properties tab. Events occur as the user interacts with a component. For example, when a component is click, an event is generating that tells us that the component was clicked. We can write code that responds to these events, performing specific actions when an event occurs. We can say that an event is something that occurs because of a component’s interaction with the user or with Windows. See figure in the next page.

To the right of the object inspector is the workspace, which is initially, displays the Form Designer. See figure 2.6 below.



**Figure 2.6:** The Workspace

In C++ builder, a form represents a window in the program; the form might be the program's main window, a dialog box or any other type of window. We use the Form Designer to place, move and size the components or part of the form creation process.

(KENT REISDORPH – 1998)

## 2.1.4 WHAT IS UML

UML (unified modeling language) is a visualizing, specifying, constructing, and documenting of software system. It is widely used in the industry to supports the entire software development lifecycle and can be used in diverse application areas. Usually it is based on experience and needs of the users community. UML can be support by many tools and the implementation can be programming language independent.

There are 9 types of UML diagrams:

- 1) Use case diagram.
- 2) Sequence diagram.
- 3) Class diagram.
- 4) Object diagram.
- 5) Collaboration diagram.
- 6) State chart diagram.
- 7) Component diagram.
- 8) Activity diagram.
- 9) Deployment diagram.

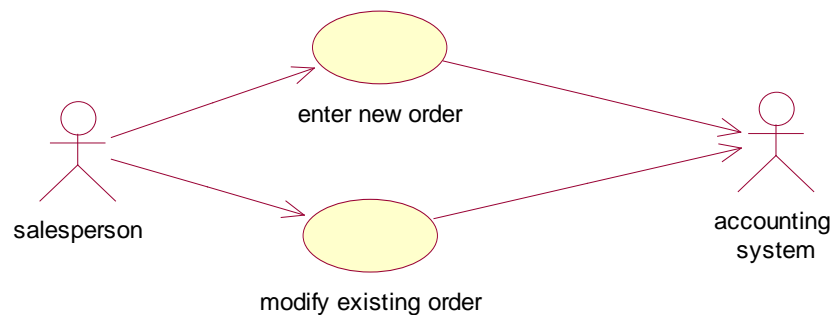
The diagrams describe either the structural or the behavioral of the system;

- 1) Use cases and actors – display the boundary of a system and its major functions.
- 2) Sequence and collaboration diagrams – illustrate specific case realizations.
- 3) Object diagrams – depict object interactions.
- 4) Class diagrams – represent the static structure of a system.
- 5) State transition and activity diagrams – represents the dynamic behavior of objects.
- 6) Component and deployment diagrams – reveal the physical implementation architecture.

Here are some of the examples of the UML diagrams;

## 1. USE CASE DIAGRAMS

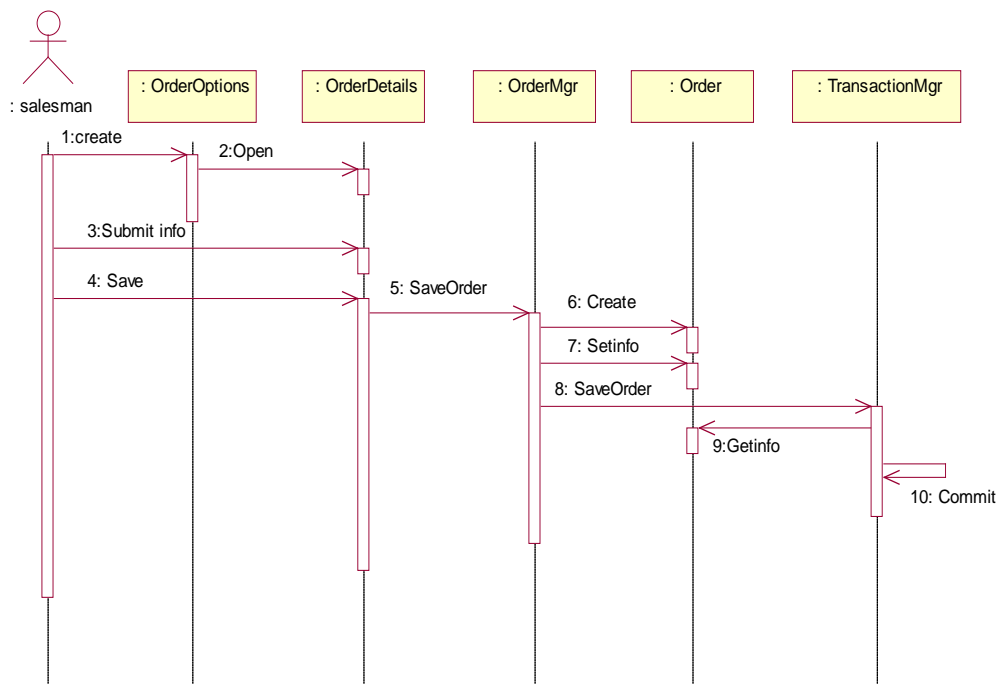
- a) Shows the interaction of the system with the outside world.
- b) Displays the system boundary as well as the relationship among actors other use cases.



**Figure 2.7:** Example of use case diagram

## 2. SEQUENCE DIAGRAM

- a) Shows the interaction amongst classes/objects passing messages.
- b) Depicts what must happen to accomplish a piece of functionality provided by the system.



**Figure 2.8:** Example of sequence diagram

## 2.2 QUESTIONNAIRE

### 2.2.1 SOFTWARE USABILITY

There are several questionnaire based approaches to assessing the usability of the software, and one such usability (**SUMI**) developed by the **HUMAN FACTORS RESEARCH GROUP (HFRG)** at University College Cork is discussed here.

The **SUMI** methodology and the **WAMMI** web based tool are a questionnaire-based approach for assessing usability. Usability is a multi dimensional concept with several properties associated with each dimension of usability.

The **SUMI** methodology list five dimensions of usability to be measured and these dimensions are related to user's expectations and attitudes to the computer system being evaluated.

**Table 2.3:** Five dimensions of usability

<b>Helpfulness</b>	This measures the degree to which the software is self-explanatory as well as adequacy of help facilities and documentation.
<b>Control</b>	This measures the extent to which the user feels in control of the software as opposed to being controlled by the software.
<b>Learn ability</b>	This measures the speed of learning and ease of mastering the software system or new features.
<b>Efficiency</b>	This measures the extend to which the users feel the software assists them in their work
<b>Affect</b>	This measures the degree to which users like the computer system i.e. likeability or the general emotional reaction to the software.