AUTOMATED USECASE DIAGRAM TOOL

**Anfal Ibrahim Al.Rawashdeh**

**Heba Suleiman AL.Qarara'a**

**Nsreen Basam AL.Hamadeen**

**Maram Reyad Joma'a**

**Sara Hassn A'alayan**

**BACHELOR'S DEGREE**

**INFORMATION TECHNOLOGY**

AL-HUSSEIN BIN TALAL UNIVERSITY

2015

**(CERTIFICATION OF PROJECT)**

Permission to Use

In presenting this project in fulfilment of the requirements for a Degree of  
Bachelor of Information Technology from Al-Hussein Bin Talal University, We agree that the University Library may make it freely available for inspection. We further agree that permission for the copying of this project in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the College of Information Technology. It is understood that any copying or publication or use of this project or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Al-Hussein Bin Talal University for any scholarly use which may be made of any material from my project.

Abstract

Software engineering one of the most important Specialties, concerned with principles of the software. Software engineering is a layered technology from pressman's view. The main objective for this project is to developing Automated Use Case Diagram Tool. This tool provides automated or semi-automated support for the process and methods. Moreover, to support engineers to perform their tasks in a systematic and/or automatic manner. Briefly this tool takes the requirement in natural languages from the user then acts as an analyst and parts of the designer through converting the requirement that it was obtained from the user to the UML notation. The incremental methodology was used in order to develop this tool. The tool has been tested and obtained the results with high performance. However, this tool has some limitations such as include and extend relationship.

.

Acknowledgement

**قال الله تعالى:**

**أقرا باسم ربك الذي خلق (1)خلق الإنسان من علق (2)أقرا وربك الأكرم (3)الذي علم بالقلم (4)علم الإنسان مالم يعلم (5) سورة العلق**

There are no words to match my gratitude; Above all else, you've shaped my attitude,

Nurturing me with discipline and light Knowledge is the least of what you taught,

Yet that least at least prepared my head. Out of your heart I've learned the things I ought,

Underscoring words you never said, for you my father which I Carry your name with pride.

Is the candle melts to illuminate paths of others? Is tender, which overflows with no limits? Is the symbol embodies the struggle? Yes, they are our information technology staff.

Let us to write words? Words filled Acknowledgements? Words hesitate on every tongue? You know why? You taught us that the goal of life is not knowledge but action? You taught that we created for success and not failure?

Personally, we would like to thank all those who have helped with their advice and efforts.

We would like also to thank all the information technology staff, especially our supervisor, Dr.Malek Alkasasbeh for all his encouragement, support, and all his efforts. For our parents, friends and everyone, we offer our research.

Table of Contents

CHAPTER 1 Introduction 1

1.1 Project Background 3

1.2 Problem Statement 4

1.3 Project Objectives 4

1.4 ProjectSignificance 5

1.5 Project Ganttchart 5

CHAPTER 2 LITERATURE REVIEW 8

2.1 Definitions 8

2.1.1 Automation 8

2.1.2 Software engineering 9

2.2 Types of automation: 10

2.2.1 Open and closed loop 10

2.2.2 Feedback control 10

2.2.3 Computer control 10

2.3 Automated software engineering tools 11

2.3.1 CRC Cards: 11

CRC Approach – The Process: 12

2.4 Need of automated tools: 17

CHAPTER 3 Methodology 18

3.1 What is incremental methodology 18

3.2 Incremental methodology as a concept 19

3.3 Steps of the incremental methodology 20

3.3.1 Analysis: 20

3.3.2 System Design: 20

3.3.3 Implementation: 21

3.3.4 Testing: 21

CHAPTER 4 ANALYSIS AND DESIGN 22

4.1 Analysis 22

4.1.1 AUDT from technical point of view 22

4.1.2 AUDT from risky point of view 23

4.1.3 AUDT from economical point of view 24

4.1.4 AUDT from SWOT point of view 24

4.1.5 AUDT from Managerial point of view 26

4.1.6 AUDT from programming point of view 26

4.1.7 AUDT from SDLC methodology point of view 28

4.1.8 AUDT from system point of view: 29

4.1.9 Requirements Determination 30

4.1.10 System's Requirements 33

4.2 Design 33

4.2.1 Logical Design 34

4.2.1.1 Use Case Diagram 34

4.2.1.2 Sequence Diagrams 39

4.2.1.3 Context diagram 42

4.2.1.4 Data Flow Diagram 43

4.2.2 Physical Design 44

CHAPTER 5 findingS 46

5.1 Introduction 46

5.2 AUDT Interfaces and their Description 49

5.2.1 Descriptions of the interfaces 50

5.2.1.1 Start 50

5.2.1.2 Type of Requirements 51

5.2.1.3 Tokenization and find error 52

5.2.1.3.1 No error message 55

5.2.1.3.2 Grammatical and syntax error message 56

5.2.1.4 Filter the requirements 58

5.2.1.5 Send to the drawer 64

5.2.1.6 Convert to the use case and display the result 65

5.2.1.7 End the application 73

5.2.1.8 Boss testing 74

5.2.1.9 Help centre 76

5.2.1.10 Upload file 78

5.2.1.11 Save as test 80

5.2.1.12 Add verb to the dictionary 82

5.2.1.13 Ready samples 86

5.2.1.14 Ready template 89

5.2.1.15 Reset the application 108

5.2.1.16 Manual requirements 110

5.2.1.16.1 Arabic natural language 122

5.2.1.16.2 Requirements validation 123

5.2.1.17 Edit massage 126

5.2.1.18 Open notepad 127

5.2.1.19 Note 129

5.2.1.19.1 Use case diagram title 129

5.2.1.19.2 Pre analysis 132

CHAPTER 6 OBSERVE AND EVALUATE 134

6.1 Introduction 134

6.2 Data validation 134

6.3 Software testing 136

6.3.1 Expert testing 137

CHAPTER 7 CONCLUSION 139

7.1 Conclusion 139

7.2 Limitations and future work 140

List of Tables

Table 1.1 Gantt chart 6

Table 1.2 Gantt chart descriptions 7

Table 4.1: functional requirement 33

Table 4.2: non-functional requirement 34

Table 4.3: description for each use case 40

Table 4.4: H/W.S/W Specifications 48

List of Figures

Figure1.1: SDLC phases. 2

Figure2.1: CRC example. 14

Figure 3.1: incremental methodology 18

Figure 3.2: incremental methodology a 19

Figure 3.3: incremental methodology b 19

Figure 3.4: incremental methodology c 19

Figure 3.5: incremental methodology d 20

Figure 4.1: the main process for AUDT 23

Figure 4.2: AUDT from SOWT point of view 25

Figure 4.2: AUDT from system point of view 29

Figure 4.3: AUDT use case 35

Figure 4.4: AUDT use case by using AUDT 36

Figure 4.4: AUDT sequence diagram 40

Figure 4.5: AUDT sequence diagram 41

Figure 4.6: AUDT context diagram 42

Figure 4.7: AUDT data flow diagram 43

Figure 4.8: physical design for AUDT 44

Figure 5.1: The implementation process 46

Figure 5.2: Algorithm of AUDT (steps of implementation) 48

Figure 5.3: start notation 50

Figure 5.4: Home page 50

Figure 5.5 :Data input notation 51

Figure 5.6: requirements typing. 51

Figuer5.7: process notation 52

The code for tokenization and find error shown below 52

Figure5.8: decision notation 55

Figure5.9: no error massage 55

Figure5.10: decision notation 56

Figure5.11: grammatical error massage 56

Figure5.12: syntax error massage 57

Figure5.13: process notation 58

The code for this process shown below: 59

Figure 5.14: Part of speech 63

Figure5.15: process notation 64

The code for this shown below: 64

Figure5.16: process and result notation 65

Figure 5.17: use case diagram. 70

Figure 5.20: stop notation 73

Figure 5.21: Boss testing button 74

The code for this process shown below 74

Figure 5.22: Boss testing 75

Figure 5.23 help button 76

Figure 5.24: help centre 77

Figure 5.25: upload file button 78

The code for this process shown below 78

Figure 5.26: upload file 79

Figure 5.27: save as text button 80

Figure 5.28: save file 81

Figure 5.29: Add verb to the dictionary button 82

Figure 5.30 Add and search 82

The code for this process shown below 82

Figure 5.31: search in the dictionary 83

Figure 5.32: Add verbs to the dictionary 84

Figure 5.33: Sample link 86

Figure 5.34: Samples 88

Figure 5.35: Ready templates button 89

Figure 5.36: Bank system 95

Figure 5.37: General system 96

Figure 5.38: AUDT project 97

Figure 5.39: AHU system 98

Figure 5.40: general example 99

Figure 5.41: AUDT team 100

Figure 5.42: Hospital system 101

Figure 5.43: Appointment system 102

Figure 5.44: face detection system 103

Figure 5.45: Hotel system 104

Figure 5.46: Mobile-learning system 105

Figure 5.47: Registration system 106

Figure 5.48: ATM system 107

Figure 5.49: Clear all button 108

The code for this process shown below 108

Figure 5.50: Clear all (reset) 109

Figure 5.51 manual requirements 110

Figure 5.52: Arrow width and colour 111

The code for this process shown below 111

Figure 5.53: manual use case 121

Figure 5.54: Arabic languages requirements 122

Figure 5.55: invalid requirements 124

Figure 5.56: invalid requirements 125

Figure 5.57: important note 126

Figure 5.58: Edit massages 126

Figure 5.59: Open notepad button 127

Figure5.61: No title massage 129

Figure5.62: appropriate places of title 130

The second place shown in the figure 5.60: 131

Figure5.63: appropriate places of title 131

Figure 5.64: Final result 133

Figure 6.1: disabled button 135

Figure 6.2: expert testing 137

List of Appendices

Appendix A software engineering in details 135

List of Abbreviations

AUDT: AUTOMATED USECASE DIAGRAM TOOL

UML: Unified Modeling Language.

SDLC: system development life cycle,

DFD: data flow diagram.

E-R: Entity relationship diagram.

PLCs: Programmable logic controllers

CRC: Class Responsibility Collaborator

IEEE: Institute of Electrical and Electronic Engineers

SWOT: strength, weakness, opportunist, threats

ASE: Automated Software Engineering

XML: Extensible Markup Language.

# Introduction

Software engineering Used as a theoretical concept from time to time in the late fifties and early sixties of the last century. The first official use of this term was in a conference held by the Scientific Committee of the North Atlantic Treaty Organization in 1968 about the software, this term has been taken to deploy since then and met a growing interest in different ways. Conference address what is known a "crisis of software" which appeared due to non-use of systematic thinking (Software Development Process) when building software, which led to the emergence of many errors during the process of building and maintaining software, during to this software need great time to develop and maintain, and cost more than expected, and lack of efficiency as well as doesn't meet the requirements (peter, nour&brain, 1968)

Software engineering is the branch of information technology aims to develop a set of principles and rules lead to improved methods of design and software development at all levels; in manner that meets the needs of users.

Software engineering is the study and application of engineering to the design, development, and maintenance of software (ACM, 2007)

And we are not out of this (Our project simply takes the requirements of natural language from the user then acts as an analyst and part of the designer through converting requirements obtained from the user to the Use case Diagram.)

The first beneficiary from our project our university and especially our collage, we can make our project as a reference to our teachers and our college student especially in software engineering's subjects, also can be used as an aid to the main software engineering's subjects, and can be used as the way to correct the exams related to the subject of the project.

Our project is also an essential part of problem solving steps, and essential part of the SDLC, which shown in the figure 1.1

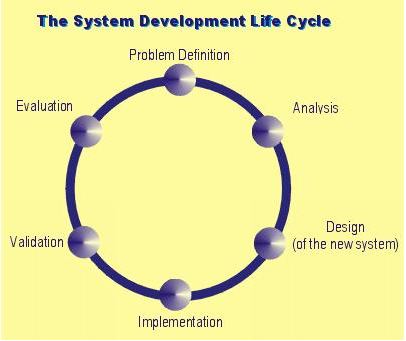


Figure1.1: SDLC phases.

## Project Background

Many tools have been developed to support automation, in narrow domains   
and in broad. To continue to advance the field in Automated Software Engineering, good automation-supporting tools need to be developed and deployed**.** (ASE, 2004)

We are interested in new directions in tool engineering, new approaches, new domains or ways in which to apply tools for requirements engineering. (Our project simply takes the requirements of natural language from the user then acts as an analyst and part of the designer through converting requirements obtained from the user in to UML notation specifically Use case Diagram).

Nowadays, in order to draw "Use Case Diagram" the users should formulate the requirements of a formal way then they enter on application to draw the "Use Case Diagram”. However, when the user typed the requirements by natural language they face problems to find application to draw "Use Case Diagram".

In this project, the programming language is the mean by which the users can type their requirements as is in natural language. Then this programming languages will analysis what users type and extracting possible (Use Case Diagram name, Actors, Functional Requirements and the relation between the Actors and the Functional Requirements), these extracted words will be send to the internal template to be converted to the use case. This template has been stored in the programming language; it can be updated according to the user’s requirement automatically without human interpretation.

## Problem Statement

## Use case modeling, through use case diagrams and use case textual specifications, is commonly applied to structure and document requirements (Krutchen, 2003). In this context, UML Activity diagrams are often used to Visualize use case scenarios to better understand and analyze them which becomes paramount when use cases are large and complex. Therefore, automated support to transform use case scenarios to use case diagram is important. Additionally, automated transformation would enable automated traceability from requirements to activity diagrams. Traceability is important during software development since it allows engineers to understand the connections between various artifacts of a software system (Yue et al., 2010).

## Project Objectives

The main objective for this project is developing Automated Use Case Diagram Tools.

## Project Significance

Our project takes it's important from its idea, such that idea, such that program will make a big difference especially for software engineering specialty, as this program will be supporter for software engineering subject.

The users will type English natural languages then they will have formal notation for "Use Case Diagram" and this make our project unlike to any other program and give it more importance.

## Project Gantt chart

A Gantt chart is a type of bar chart, developed by Henry Gantt in the 1910s, that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project as portrayed in table 1.1 and 1.2.

Table 1.1 Gantt chart

|  |
| --- |
| **1/8 15\8 1/9 17/9 23/9 1/10 13/10 25/10 7/11 30/11 2/12 4/12 8/12** |
| Start  **Analysis**  **Design**  **Implementation**  **Analysis**  **Design**  **Implementation**  **Analysis**  **Design**  **Implementation**  **Analysis**  **Design**  **Implementation**  **Testing**  Finish |

Table 1.2 Gantt chart descriptions

|  |  |
| --- | --- |
| phase name | Description |
| Analysis  Design  Implementation  Testing | All possible requirements of the system to be developed are captured in this phase and documented.  The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.  With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase.  All the units developed in the implementation phase are integrated into a system after testing of each unit. |

# LITERATURE REVIEW

This chapter describes the concepts of (Automation and Software Engineering). Moreover, the types of automation are explained. The needs of automated software engineering tools are described .In addition, current of automated software engineering tools are highlighted.

## Definitions

### Automation

***"Automation*** or automatic control is the use of various control system for operating equipment such as machinery, processes in factories, boilers and heat treating ovens, switching in telephone networks, steering and stabilization of ships, aircraft and other applications with minimal or reduced human intervention. Some processes have been completely automated"(Rifkin & Jeremy, 1995).

***"Automation*** means automatic control, meaning a process is run with minimum operator intervention. Some of the various levels of automation are: mechanical methods, electrical relay, feedback control with a controller and computer control. Automatic speed control is important in many industrial applications, especially in sectional drives, such as found in metal rolling and paper drying. With metal rolling the metal lengthens as it passes through the rollers, necessitating progressively speeding up the rollers. With paper drying the sheet shrinks as it passes around the drying cylinders. The dryers are arranged in groups or sections so as the paper shrinks the dryer sections can turn at lower speeds"(Bennett & s, 1979).

### Software engineering

***Software engineering*** is the branch of information technology aims to develop a set of principles and rules lead to improved methods of design and software development at all levels; in manner that meets the needs of users.(Peter, nour&brain, 1968).

***"Software engineering*** is the study and application of engineering to the design, development, and maintenance of software" (Peter, nour&brain, 1968)

***"Software engineering***: the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"(IEEE, 1990)

***"Software engineering***: An engineering discipline that is concerned with all aspects of software production"(Sommerville, 2007)

## Types of automation:

### Open and closed loop

All the elements constituting the measurement and control of a single variable are called a control loop. Control that uses a measured signal, feeds the signal back and compares it to a set point, calculates and sends a return signal to make a correction, is called closed loop control. If the controller does not incorporate feedback to make a correction then it is open loop. Timers and sequence controllers using logic, such as those on an elevator, are open loop.

### Feedback control

Feedback control is accomplished with a controller. To function properly, a controller must provide correction in a manner that maintains stability. The theoretical basis of feedback control is control theory, which also covers servomechanisms, which are often part of an automated system.

### Computer control

Computers can perform both sequential control and feedback control, and typically a single computer will do both in an industrial application. Programmable logic controllers (PLCs) are a type of special purpose microprocessor that replaced many hardware components such as timers and drum sequencers used in relay logic type systems. General purpose process control computers have increasingly replaced stand-alone controllers, with a single computer able to perform the operations of hundreds of controllers. Process control computers can process data from a network of PLCs, instruments and controllers in order to implement typical control of many individual variables or, in some cases, to implement complex control algorithms using multiple inputs and mathematical manipulations. They can also analyze data and create real time graphical displays for operators and run reports for operators, engineers and management.

## Automated software engineering tools

Automated software engineering tools: is the application of a set of tools and methods to a software system with the desired end result of high-quality, Just to remember the main task for our projects" Our project simply takes the requirements of natural language from the user then acts as an analyst and part of the designer through converting requirements obtained from the user to the UML notation."In this term many projects have been reviewed as a following:

### CRC Cards:

Invented in 1989 by Kent Beck and Ward Cunningham a simple yet powerful object-oriented (analysis/design) technique uses a collection of (standard index) cards that are divided into three sections:

Class

Responsibility

Collaborator

A class represents a collection of similar objects.

A responsibility is anything that a class knows or does.

A collaborator is another class that is used to get information for, or performs actions for the class at hand.

### CRC Approach – The Process:

Exploratory phase

Find classes

Determine operations and knowledge for each class (responsibilities)

Determine how objects collaborate to discharge responsibilities

**In the Analysis phase:**

Use nouns and verbs in requirements as clues Noun phrases leads to objects

Verb phrases lead to responsibilities

Determine how objects collaborate to fulfill their responsibilities to collaborate, objects will play certain roles

Why is this important?

Objects lead to classes, Responsibilities lead to operations or methods, Collaborations and roles lead to associations

**Identifying Objects (Classes):**

Start with requirements specification Look for noun phrases.

Separate into obvious classes, uncertain candidates, and nonsense

**Identifying responsibilities:**

Start with requirements specification Look for verb phrases.

**CRC with our project:**

CRC card example:

Figure2.1: CRC example.

On the back, can write a description of purpose of the class (with its attributes)

The CRC card will support our project as the following:

In the use case:

Noun can be actors, verbs can be Requirements.

In the Sequence diagram:

In the class diagram:

Associations

**Class:** Person

**Responsibilities Collaborations**

Knows name

Knows address Book

Knows phone number Phonebook

…

However, many similar programs to our project such as Rational Rose XDE, an "extended Development Environment" for software developers, integrates with Microsoft Visual Studio .NET and Rational Application Developer. The Rational Rose family of products is a set of UML modeling tools for software design.

CASE stands for Computer Aided Software Engineering which is software that supports one or more software engineering activities within a software development process, and is gradually becoming popular for the development of software as they are improving in the capabilities and functionality and are proving to be beneficial for the development of quality software

## Need of automated tools:

##### The software development process is expensive and as the projects become more complex in nature, the project implementations become more demanding and expensive. That's why software developers always looking for such automated tools that help them in many different ways during the different development stages of software, so that they can understand the software and prepare a good end product that efficiently fulfill the user requirements. Automated tools provide the integrated homogenous environment for the development of complex projects. These tools provide computerized setting to software developers to analyze a problem and then design its system model. The automated tools also provide the environment for monitoring and controlling projects such that team leaders are able to manage the complex projects.

**Basically, the automated tools are used to**

* Reduce the cost as they automate many repetitive manual tasks.
* Reduce development time of the project as they support standardization and avoid   
  repetition and reuse.
* Develop better quality complex projects as they provide greater consistency and   
  coordination.
* Create good quality documentation.
* Create systems that are maintainable because of proper control of configuration item that support traceability requirements.

# Methodology

This chapter describes the incremental methodology, the incremental methodology has adapted and the main stages of the incremental processes are shown in figure 3.1.

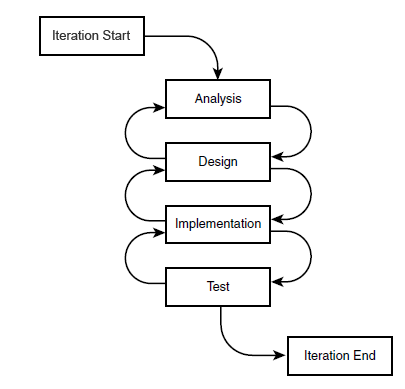


Figure 3.1: incremental methodology

## What is incremental methodology

Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross-discipline work, from the requirements to the deployment

## Incremental methodology as a concept

## 

Figure 3.2: incremental methodology a



Figure 3.3: incremental methodology b

## 

Figure 3.4: incremental methodology c

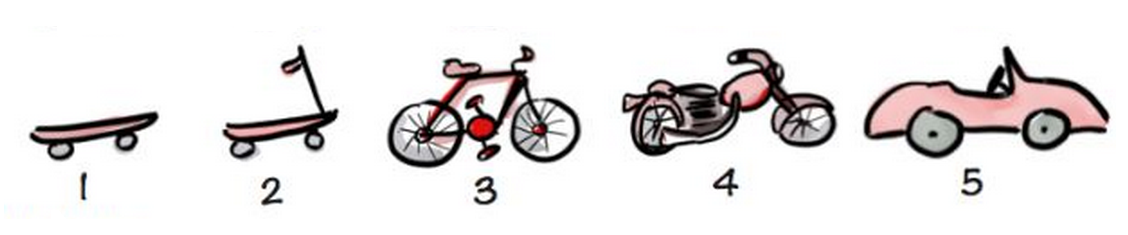


Figure 3.5: incremental methodology d

## Steps of the incremental methodology

### Analysis:

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document

### System Design:

The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture

### Implementation:

With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

### Testing:

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

# ANALYSIS AND DESIGN

In the previous chapter, the methodology of project that was applied in this project has been explained considerably. The methodology consists of four phases which includes analysis, design, implementation and test. In this chapter analysis and design the project will be discussed in a more details elaboration.

## Analysis

The aim of this chapter is to identify what the user would require from the Automated Usecase Diagram Tool (AUDT).

Basically, analyzing the system involved two major activities that are requirements determination and then structuring the determined requirements.

Our project will be analyzed from different point of views:

### AUDT from technical point of view

The programming language is the mean by which the users can type their requirements as is in natural language. Then this programming languages will analysis what users type and extracting possible (use case / actors), these extracted words will be send to the internal template to be converted. This template has been stored in the programming language; it can be updated according to the user’s requirement automatically without human interpretation.

The main process for our project shown in the figure 4.1

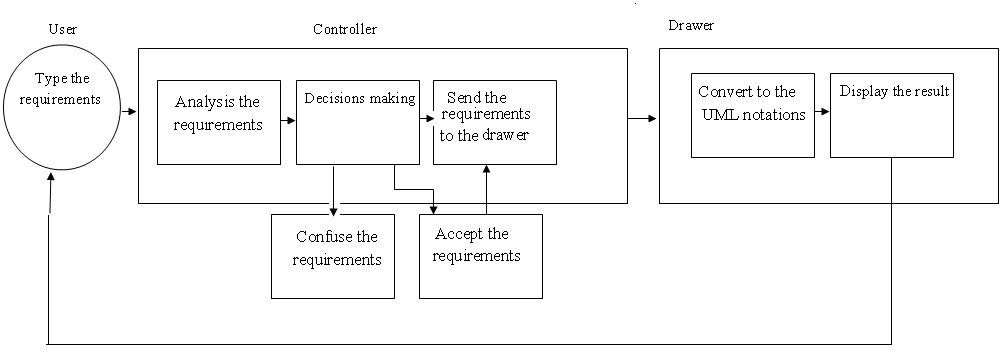


Figure 4.1: the main process for AUDT

### AUDT from risky point of view

Our project will convert what the user input, so the main risks we may face them understandable words from user.

In order to solve this problem we will use famous programming technique which called "exception handling" so our project will not convert any understandable words.

### AUDT from economical point of view

Our project will do the hardest part of any project (analysis and design phases) and if the base was corrected then the result will be corrected and this will achieve the aims of software engineering "cost-effective "and for sure will reduce the cost

### AUDT from SWOT point of view

"A SWOT analysis (alternatively SWOT matrix) is a structured planning method used to evaluate the strengths, weaknesses, opportunities, and threats involved in a project or in a business venture

* Strengths: characteristics of the business or project that give it an advantage over others.
* Weaknesses: characteristics that place the business or project at a disadvantage relative to others
* Opportunities: elements that the project could exploit to its advantage
* Threats: elements in the environment that could cause trouble for the business or project"

(Humphrey, 2005)

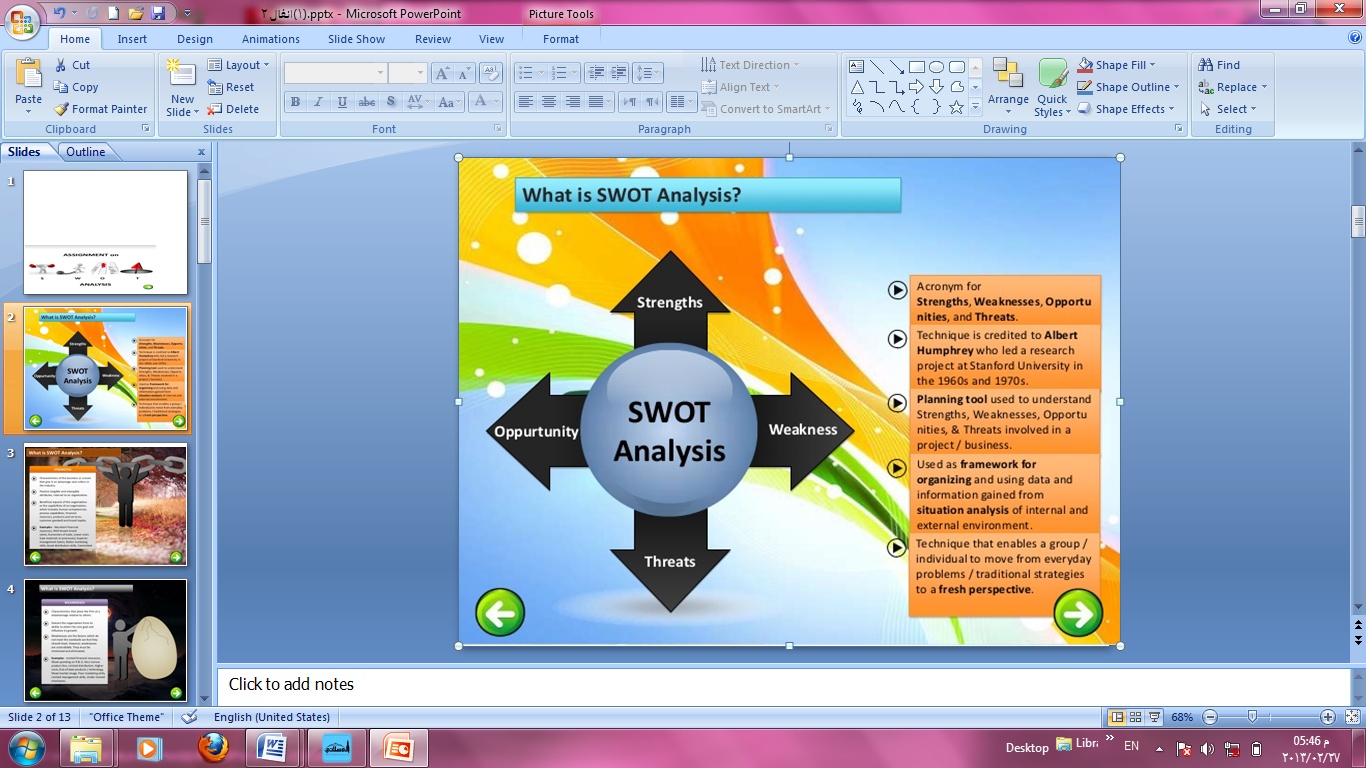


Figure 4.2: AUDT from SOWT point of view

**Strength**:

Our project is unique and there is no similar project for it in our university.

There are experts team to make such that project.

Our project support one of the most important specialties in our university "software engineering".

**Weakness:**

The time only the main problem we may face it

. **Opportunities:**

We can make our project as:

Standard program are referenced when needed.

The way to correct the exams related to the subject of the project.

Certified reference for student and teachers of information technology faculty

**Threat:**

The main threat we may face them understandable words from user.

### AUDT from Managerial point of view

We make sure that we are able to manage projects, resources, risk

### AUDT from programming point of view

Our project will be implemented by using c# programming languages for the following advantages:

C# is a multi-paradigm, object-oriented programming language encompassing strong typing, essential, declarative, efficient, class-based,

And component-oriented programming disciplines. C# is one of the programming languages designed for the Common Language Infrastructure.

C# is a well-designed and type-safe that allows C Sharp developers to build a wide array of secure and robust applications that run on the .NET Framework.

Advantages of C# (C Sharp)

* Friendly user interface
* It is the language which can be easily understood and can be modified.
* Graphics, videos and sounds can also be added
* C# language is projected to be a simple, modern, general-purpose, object-oriented programming language.
* C# helps you to import a namespace and use the classes in a component
* C# supports XML comments to add comments to code. The comments are placed into XML format and can then be used as needed to document your code.
* C# simplifies the syntax to be more constant and more rational while also eliminating some of the more difficult features of C++.
* C# provides operator overloading which is not available in VB.NET
* C# allows you to access memory directly using unsafe code blocks
* C# allows you to implement an interface in a base class and re-implement the interface in the derived class

And provide a new definition for it C# permits nested classes which are not allowed in C and C++

* C# establishes better event management using delegates and supports conditional compilation & cross-language interoperability with any .NET language
* C# is based on reflection mechanism which is biggest advantage of C#
* You need not put much attention on such problems as memory leak, which is troubling problem for C++ programmer.
* Ease-to-development, The rich class library makes many functions easy to be implemented

### AUDT from SDLC methodology point of view

We use incremental methodology for the following advantages:

* Generates working software quickly and early during the software life cycle.
* This model is more flexible – less costly to change scope and requirements.
* It is easier to test and debug during a smaller iteration.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

### AUDT from system point of view:

In this project, natural languages requirement will be entered by using one of the popular programming languages. Moreover, these requirements will be converted to UML Use Case Diagram notations as portrayed in Figure 4.2.

## :

Back of CRC card:

Attributes

Face of CRC card:

Name

Responsibility

Collaboration

Back of CRC card:

Attributes

Feedback\ User satisfaction

Figure 4.2: AUDT from system point of view

### Requirements Determination

**What are requirements?**

It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.This is inevitable as requirements may serve a dual function May be the basis for a bid for a contract - therefore must be open to interpretation May be the basis for the contract itself - therefore must be defined in detail

Both these statements may be called requirements.

**Requirement of AUDT:**

Automated Software engineering tools project consist of the following fundamentals:

Table 4.1: functional requirement

|  |  |  |
| --- | --- | --- |
| Number | Requirement ID | Description of requirement |
| 1. | AUDT\_01 | The user will enter her/his requirements in their appropriate place in our project interface |
| 2. | AUDT\_02 | The controller will analysis the requirements which have been entered by the user and find grammatical and syntax error |
| 3. | AUDT\_03 | Controller must find possible actor, use case, title, tags, and exception. |
| 4. | AUDT\_04 | The controller will send the accepted requirements to the drawer to convert them |
| 5. | AUDT\_05 | The drawer convert the requirement to the use case |
| 6. | AUDT\_06 | The drawer will display the final results to the user. |
|  |  |  |

Table 4.2: non-functional requirement

|  |  |  |
| --- | --- | --- |
| Number | Requirement ID | Description of requirement |
| 1. | AUDT\_01 | Type requirement manual to allow user to type Arabic requirement manually |
| 2. | AUDT\_02 | Help button to make the user understand our project in good way |
| 3. | AUDT\_03 | Upload file to import external requirement written with text file |
| 4. | AUDT\_04 | Save as text button to save written requirement as a text |
| 5. | AUDT\_05 | Add verb to the dictionary button to allow user add and search about verbs |
| 6. | AUDT\_06 | Apply boss test button to identify what is the boss test mean. |
| 7. | AUDT\_07 | Clear all button to remove all written requirement (reset). |

### System's Requirements

Automated Software engineering tools project require the following:

* Our project requires artificial intelligence something, because it deals with natural languages, so it needs to understand and process natural languages in order to deal with it.
* Our project requires programming languages as a mean by which the user can enter his\her requirements.
* The user must able to come out of the application at any time he wants.
* Our project must be able to work in different environments of operating systems.
* Our project must be designed to ensure runs on several environments and on all versions of Windows.

## Design

Theoretically, designing the project system involved two main processes that were categorized into logical design and physical design.

### Logical Design

Havner et al. (2004) explained that logical design is the phase where all functional features that have been chosen for the development of the system are described without regard of any computer platform. Assuming that the developed system could be implemented on any hardware or system software, the aim of this phase is actually to make sure that the system can really functions as it should be.

#### Use Case Diagram

**Use case scenario:**

User enters the requirements then the controller find possible grammatical and syntax errors in the typed requirements. Then it will process them by extracting possible actors, use case, tags, and title. After that send those to the drawer to convert them to the use case diagram then the final result will be displayed to the user.

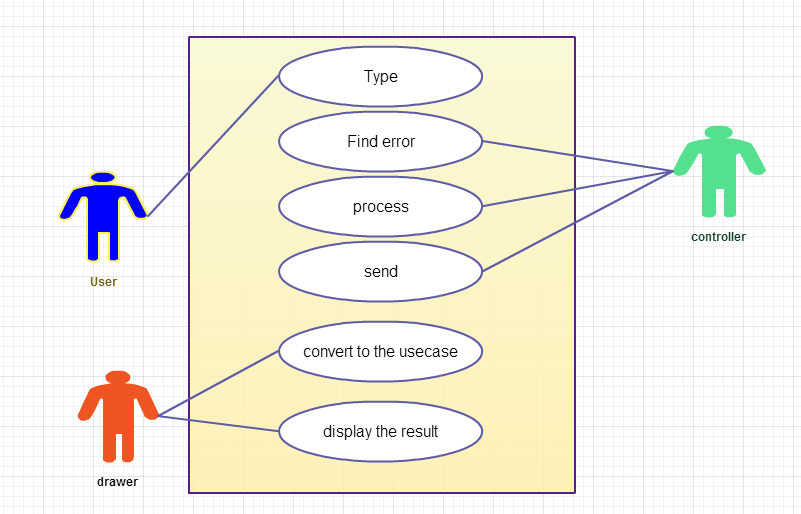
****

Figure 4.3: AUDT use case

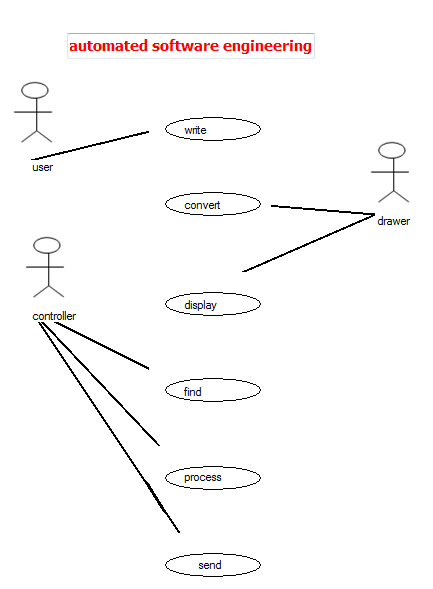


Figure 4.4: AUDT use case by using AUDT

**AUDT Actor list**

* User: will input the natural languages requirements.
* Stored template (drawer): which will convert the natural languages requirements in to UML notations
* Programming languages (controller): which act as an intermediate between the user and drawer accept the natural languages requirements from the user then send them to the drawer.

**AUDT use case list:**

* Typing: the user will input the natural languages requirements.
* Find error: if the requirements written with grammatical and syntax error the controller will inform user.
* Send to the drawer: if the controller accepts what the user write it will send them to the drawer to convert them.
* Convert to the UML notations: the drawer will convert the requirements to the UML notations.
* Display the result: display the final result to the user

Table 4.3: description for each use case

|  |  |  |  |
| --- | --- | --- | --- |
| Use case name | Actors | Preconditions | Description |
| Typing | User | Have our project interface | The user will enter her/his requirements in their appropriate place in our project interface |
| Find error | Controller | User's requirements | The controller will analysis the requirements which have been entered by the user and find grammatical and syntax error |
| Process | Controller | Analyzed and understood requirements | Controller must find possible actor, use case, title, tags, and exception. |
| Send the requirements to the drawer | Controller | Accepted requirements | the controller will send the accepted requirements to the drawer to convert them |
| Convert to the use case  Display the results | Drawer  Drawer | Processed requirements  Converted requirements to the UML notations | The drawer will convert the requirement to the use case  The drawer will display the final results to the user. So the user can see her /his requirements have been converted to the use case diagram. |
|  |  |  |  |

#### Sequence Diagrams

Sequence diagram scenario:

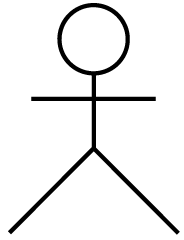
User enters the requirements then if the controller accepts and understand the requirements (find no grammatical and syntax errors) it will process them by extracting possible actors, use case, tags, title. Then send them to the drawer to convert them to the use case diagram then the final result will be displayed to the user.

Face of CRC card:

Name

Responsibility

Collaboration



Back of CRC card:

Attributes

Face of CRC card:

Name

Responsibility

Collaboration

Name

Attributes

Responsibilities

Input:

Natural languages requirement by using one of the popular programming languages

Processes:

Convert these inputs to the UML notations

Output:

UML notations such as:

Use case diagram

Controller

Drawer

Typing

Find error

Accepted

Process

Send

Convert

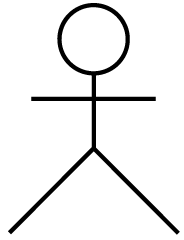
Display the result

Figure 4.4: AUDT sequence diagram

Sequence diagram scenario:

User enters the requirements then if the controller can't understand the requirements (find grammatical and syntax errors) it will confuse them.

User



Controller

Drawer

Figure 4.5: AUDT sequence diagram

#### Context diagram

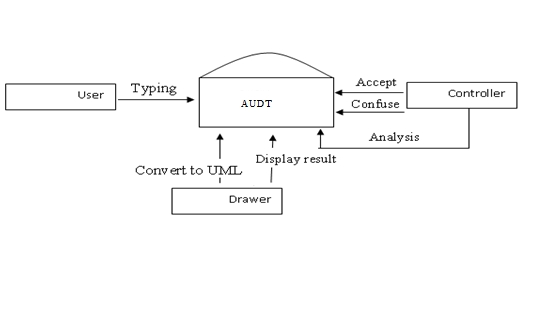


Figure 4.6: AUDT context diagram

#### Data Flow Diagram

Figure 4.7: AUDT data flow diagram

### Physical Design

Meanwhile, physical design deals with the process of converting the logical design into a more technical specification of the system development. In designing the physical part of the system, all diagrams that were produced in the logical design were turned into a structured systems design.

Typing

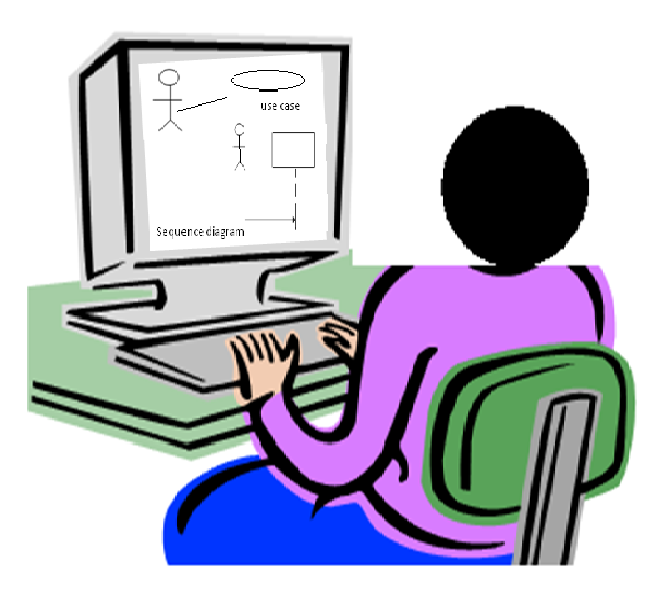


Figure 4.8: physical design for AUDT

During physical design, we determined which programming language system will be used as well as the determination of which hardware platform, operating system and network environment the system will run under. The specifications are portrayed in table 4.4

Table 4.4: H/W.S/W Specifications

|  |  |
| --- | --- |
| Purpose | H/W.S/W Requirements |
| Programming Language | C# |
| Operating System | Windows 7 |
| Hardware | Computers |

# findingS

## Introduction

Implementation is the realization of an application, or execution of a plan, idea, model, design, specification, standard, algorithm, or policy.



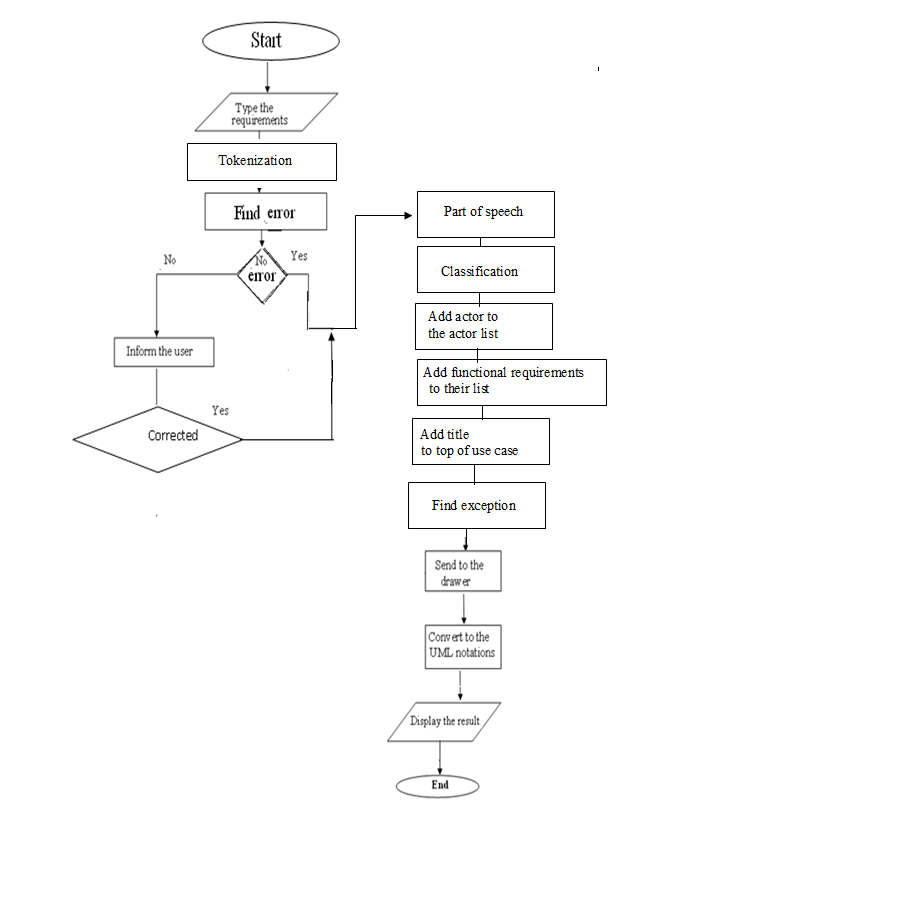
Figure 5.1: The implementation process

In computer science, an implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment.

In the IT Industry, implementation refers to post-sales process of guiding a client from purchase to use of the software or hardware that was purchased. This includes requirements analysis, scope analysis, customizations, systems integrations, user policies, user training and delivery. These steps are often overseen by a project manager using project management methodologies. Software Implementations involve several professionals that are relatively new to the knowledge based economy such as business analysts, technical analysts, solutions architects, and project managers.

The c# programming languages has been used to develop the **AUTOMATED USECASE DIAGRAM TOOL**

In this chapter we convert all the diagrams which have been drawn in to executable computer program by using the following algorithm which shown in the figure 5.2

Figure 5.2: Algorithm of AUDT (steps of implementation)

## AUDT Interfaces and their Description

The interface is the form of the system that the user deals with and form should have the following properties:

1. User friendly: it means to be easy to understand.
2. Understandable: the languages easy so that all kind of people can deal with it.
3. Integrity: it covers all the operations.

### Descriptions of the interfaces

In this section we will display the interfaces with their code in addition to the algorithm notations which shown in the figure 5.2

#### Start

As shown in figure 5.2 and 5.3, the system beginning of start notation and implemented as shown figure 5.4 as a home page.



Figure 5.3: start notation

Figure 5.4: Home page

**Description**: This page represents AUDT home page.

#### Type of Requirements

As shown in figure 5.2 and 5.5, type of requirements in this system represent as input notation and implemented as shown figure 5.6 as requirements typing..



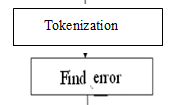
Figure 5.5 :Data input notation

Figure 5.6: requirements typing.

**Description:** This page allows users to type their natural languages requirements in order to be converted to the use case diagram.

#### Tokenization and find error

As shown in figure 5.2 and 5.7, tokenization and find error in this system represent as process notation and implemented as shown figure 5.9, 5.11, 5.13 as error message type



Figuer5.7: process notation

The code for tokenization and find error shown below

int verb\_count = 0;

int i = 0;

int falg = 0;

int brake = 0;

int correct = 0;

string[] words = richTextBox1.Text.Split('.');

foreach (string word in words)

{

string[] show = words[i].Split(' ');

foreach (string wordss in show)

{

if (listBox1.Items.Contains(wordss) || listBox3.Items.Contains(wordss))

{

falg = 1;

}

if (wordss == "all"||wordss=="same")

brake = 1;

if (listBox1.Items.Contains(show[0]) || listBox3.Items.Contains(show[0]))

{

verb\_count = 1;

break;

}

else

{

for (int f = 0; f < show.Length-1; f++)

{

if (f == 2)

break;

if (listBox10.Items.Contains(show[f]))

{

if (listBox1.Items.Contains(show[f + 1]) || listBox3.Items.Contains(show[f + 1]))

{

verb\_count = 1;

break;

}

}

else

{

if (!listBox10.Items.Contains(show[f]) && !listBox12.Items.Contains(show[f]))

verb\_count = 0;

}

}

}

}

if(brake==1)

break;

if (i < words.Length - 1)

{

{

if (falg != 1)

MessageBox.Show("undefined verb at statement" + " " + (i + 1).ToString());

}

}

falg = 0;

if (i < words.Length - 1)

{

if (verb\_count == 1)

{

MessageBox.Show("wrong at your statement" + " " + (i + 1).ToString() + " " + " the verb can not came befor subject");

correct = 1;

}

}

i++;

}

if (correct == 0)

{

button5.Enabled = true;

MessageBox.Show("there is no grammatical error at your statements");

}

correct = 0;

}

##### No error message

As shown in figure 5.2 and 5.8, no error decision in this system represent as decision notation and implemented as shown figure 5.9 as no error message.

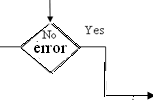


Figure5.8: decision notation

**Description:** If there is no grammatical or syntax error the following messages will be appeared:

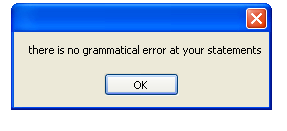


Figure5.9: no error massage

##### Grammatical and syntax error message

As shown in figure 5.2 and 5.10, grammatical and syntax error decision in this system represent as decision notation and implemented as shown figure 5.11 as grammatical error message and figure 5.12 as syntax error message

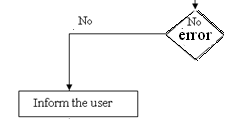


Figure5.10: decision notation

**Description:** If there is grammatical error the following messages will be appeared:

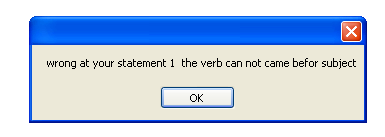


Figure5.11: grammatical error massage

**Description:** If there is r syntax error the following messages will be appeared:

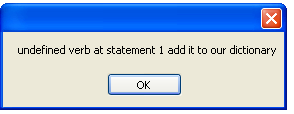


Figure5.12: syntax error massage

**Description:** The pervious messages will be displayed according to the type of errors may be happen (grammatical, syntax).

#### Filter the requirements

As shown in figure 5.2 and 5.13, part of speech, classification, add actor and functional requirements to their lists, find title of the system and exception in this system represent as process notation and implemented as shown figure 5.14 as filter the requirements

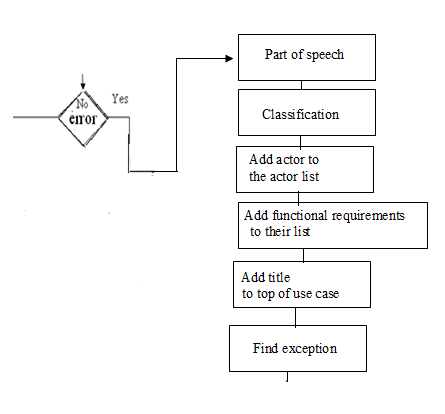


Figure5.13: process notation

The code for this process shown below:

int i = 0;

string comb = "";

Boolean ex = false;

string[] words = richTextBox1.Text.Split('.');

foreach (string word in words)

{

listBox11.Items.Add(word);

string[] show = words[i].Split(' ');

foreach (string wordss in show)

{

listBox9.Items.Add(wordss);

if (listBox1.Items.Contains(wordss) || listBox3.Items.Contains(wordss))

verb.Add(wordss);

if (!listBox10.Items.Contains(show[0])&&show[0]!=" ")

{

for (int n = 1; n < show.Length-1; n++)

{

if (!listBox10.Items.Contains(show[n]) && !listBox1.Items.Contains(show[n]) && !listBox12.Items.Contains(show[n])&& !listBox3.Items.Contains(show[n]))

{

comb = show[0] + " " + show[n];

if (!listBox4.Items.Contains(comb))

listBox4.Items.Add(comb);

continue;

}

else

{

if (!listBox4.Items.Contains(show[0])&&!comb.Contains(show[0]))

listBox4.Items.Add(show[0]);

}

break;

}

for (int y = 2; y < show.Length-1; y++)

{

if (listBox12.Items.Contains(show[y]) || listBox1.Items.Contains(show[y]))

{

break;

}

if (!listBox10.Items.Contains(show[y + 1]) && !listBox1.Items.Contains(show[y + 1]) && !listBox3.Items.Contains(show[y + 1]) && !listBox12.Items.Contains(show[y + 1]) && !listBox10.Items.Contains(show[y]) && !listBox1.Items.Contains(show[y]) && !listBox12.Items.Contains(show[y]) && !listBox3.Items.Contains(show[y]))

{

comb = show[y] + " " + show[y + 1];

if (!listBox4.Items.Contains(comb))

listBox4.Items.Add(comb);

continue;

}

else

{

if (!listBox10.Items.Contains(show[y]) && !comb.Contains(show[y]))

if (!listBox4.Items.Contains(show[y]))

listBox4.Items.Add(show[y]);

}

}

}

else

{

for (int y = 0; y < show.Length-1; y++)

{

if (listBox1.Items.Contains(show[y]) || listBox12.Items.Contains(show[y]) || listBox3.Items.Contains(show[y]))

{

break;

}

if (!listBox10.Items.Contains(show[y]))

if (!listBox10.Items.Contains(show[y + 1]) && !listBox1.Items.Contains(show[y + 1]) && !listBox12.Items.Contains(show[y + 1]) && !listBox3.Items.Contains(show[y + 1]))

{

comb = show[y] + " " + show[y + 1];

if (!listBox4.Items.Contains(comb))

listBox4.Items.Add(comb);

continue;

}

else

{

if (!listBox4.Items.Contains(show[y])&&!comb.Contains(show[y]))

listBox4.Items.Add(show[y]);

}

}

}

if (listBox1.Items.Contains(wordss) || listBox3.Items.Contains(wordss))

listBox4.Items.Add(wordss);

if (listBox7.Items.Contains(wordss))

{

all = true;

ex = true;

}

if (ex == true)

{

for (int s = 0; s < verb.Count; s++)

listBox4.Items.Add(verb[s]);

ex = false;

}

}

i++;

}

for (int f = 0; f < listBox9.Items.Count; f++)

{

if (listBox9.Items[f].ToString() == "system" || listBox9.Items[f].ToString() == "system.")

{

if (listBox9.Items[f + 1].ToString() == "of " || listBox9.Items[f + 1].ToString() == "of")

{

textBox2.Text = listBox9.Items[f + 2].ToString();

title = textBox2.Text;

break;

}

else

{

textBox2.Text = listBox9.Items[f - 1].ToString();

title = textBox2.Text;

break;

}}}

string[] remove = textBox1.Text.Split(' ');

foreach (string rem in remove)

{

for (int p = 0; p < listBox4.Items.Count; p++)

if (listBox4.Items[p].ToString().Contains(rem))

listBox4.Items.Remove(listBox4.Items[p]);

}

Figure 5.14: filter the requirements

**Description:** When the user press filter the requirements button, the system will show user possible actors, use case, title, tags.

#### Send to the drawer

As shown in figure 5.2 and 5.15, send to the drawer in this system represent as process notation.

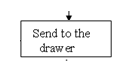


Figure5.15: process notation

The code for this process shown below:

if (textBox2.Text == "")

MessageBox.Show("This usecase has no title");

for (int f = 0; f < listBox4.Items.Count; f++)

{

listBox4.Items.Remove(" ");

}

for (int i = 0; i < listBox4.Items.Count; i++)

Form4.draw.Add(listBox4.Items[i].ToString());

for (int r = 0; r < verb.Count; r++)

if (!Form4.verb.Contains(verb[r]))

{

Form4.verb.Add(verb[r]);

}

foreach (string none in listBox4.Items)

{

if (!verb.Contains(none))

Form4.none.Add(none);

}

Form4 f4 = newForm4();

f4.Show();

this.Hide();

}

#### Convert to the use case and display the result

As shown in figure 5.2 and 5.16, convert to the use case in this system represent as process notation and display the result notation in this system represent as output notation and implemented as shown figure 5.17, 5.18, 5.19 as use case diagram

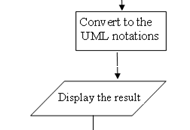


Figure5.16: process and result notation

The code for this process shown below

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

}

public int i = 0;

public int j = 0;

public int s = 0;

public int w = 0;

public int t = 7;

int z = 7;

System.Drawing.Point nl;

System.Drawing.Point vl;

Pen pen;

System.Drawing.Graphics g;

public static List<string> verb = new List<string>();

public static List<string> none = new List<string>();

public static List<string> draw = new List<string>();

public static List<Point> nloc = new List<Point>();

public static List<Point> vloc = new List<Point>();

private void drawingat()

{

int x = 0;

int j = 0;

int v = 0;

int n = 0;

while (j < draw.Count)

{

pen = new Pen(Color.Black, 2);

if (n < nloc.Count)

nl = nloc[n];

if (v < vloc.Count)

vl = vloc[v];

g = this.CreateGraphics();

g.DrawLine(pen, vl, nl);

i = 0;

while (i < none.Count)

{

if (draw[x] == none[i])

{

n++;

}

i++;

}

v++;

x++;

j++;

}

}

private void drawingats()

{

int x = 0;

int j = 0;

int v = 0;

int n = 0;

while (j < verb.Count)

{

pen = new Pen(Color.Black, 2);

if (n < nloc.Count)

nl = nloc[none.Count-1];

if (v < vloc.Count)

vl = vloc[v];

g = this.CreateGraphics();

g.DrawLine(pen, vl, nl);

if (v < vloc.Count - 1)

v++;

j++;

}

}

private void Form4\_Load(object sender, EventArgs e)

{

textBox1.Text = Form3.title;

foreach (Control lab in Controls)

{

{

if (lab is PictureBox)

if (j < none.Count)

{

string name = lab.Name;

string ser = "no";

if (name.Contains(ser))

{

lab.Visible = true;

j++;

}

}

}

}

foreach (Control lab in Controls)

{

if (lab is Label)

{

if (i < none.Count)

{

string name = lab.Name;

string ser = "n";

if (name.Contains(ser))

{

lab.Visible = true;

lab.Text = none[i].ToString();

nloc.Add(lab.Location);

i++;

}

}

}

}

foreach (Control crt in Controls)

{

if (crt is PictureBox)

{

if (s < verb.Count)

{

string name = crt.Name;

string ser = "ve";

if (name.Contains(ser))

{

crt.Visible = true;

s++;

}

}

}

}

foreach (Control crtw in Controls)

{

if (crtw is Label)

{

if (w < verb.Count)

{

string name = crtw.Name;

string ser = "v";

if (name.Contains(ser))

{

crtw.Visible = true;

crtw.Text = verb[w].ToString();

vloc.Add(crtw.Location);

crtw.ForeColor = Color.Black;

w++;

}

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

verb.Clear();

none.Clear();

draw.Clear();

vloc.Clear();

nloc.Clear();

Form3 f1 = new Form3();

f1.Show();

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

drawingat();

if (Form3.all == true)

drawingats();

Form3.all = false;

}

private void button3\_Click(object sender, EventArgs e)

{

Form7 f7 = new Form7();

this.Close();

f7.Show();

}

private void button4\_Click(object sender, EventArgs e)

{

Form2 f2 = new Form2();

f2.ShowDialog();

}

private void v0\_Click(object sender, EventArgs e)

{

}

}

}

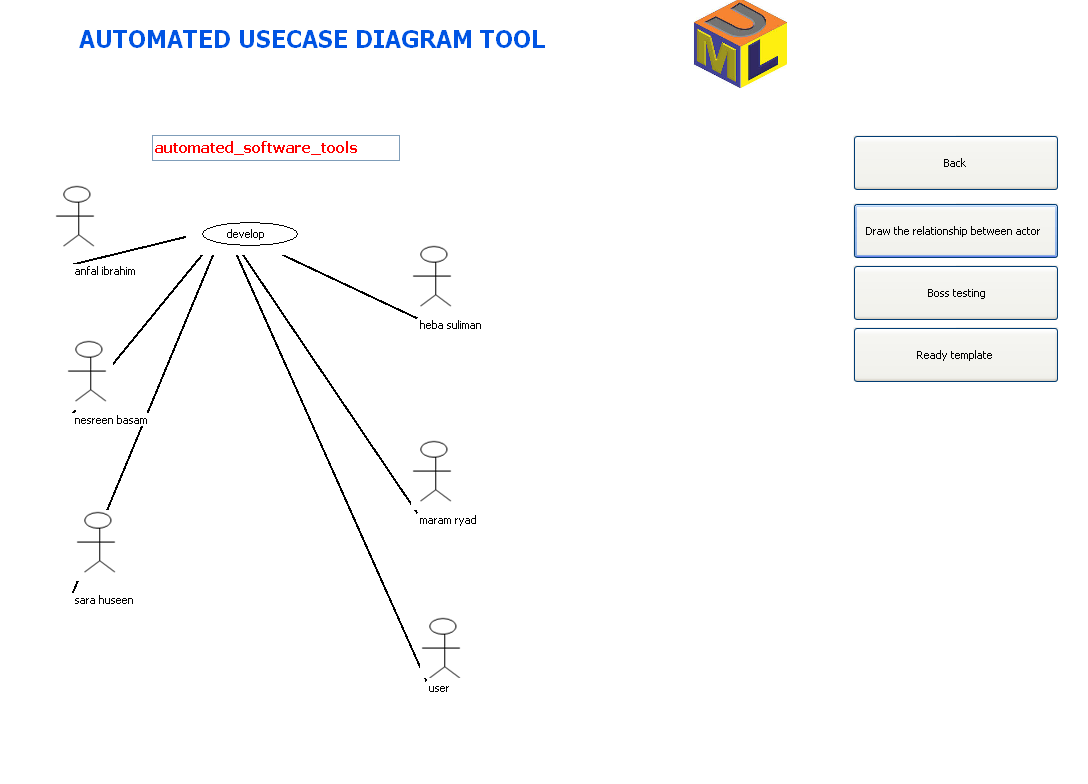
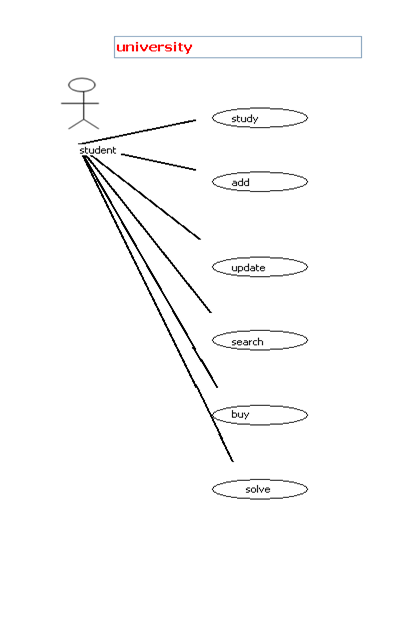


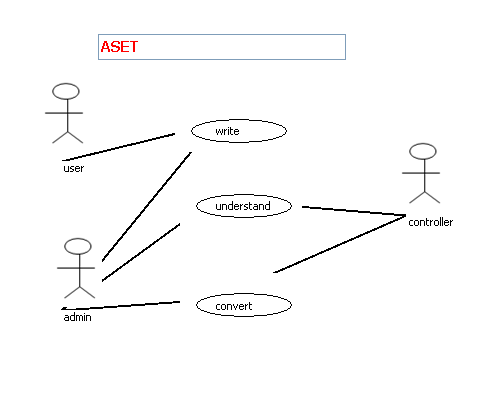
Figure 5.17: use case diagram.

**Description:** In this page the user can see their requirements have been converted to the use case.

**

**Description:** Figure 5.18: use case diagram.

In this page the user can see their requirements have been converted to the use case.

**

**Description:** Figure 5.19: use case diagram.

In this page the user can see their requirements have been converted to the use case.

#### End the application

As shown in figure 5.2 and 5.20, end the application in this system represent as stop notation.

**

Figure 5.20: stop notation

Application.Exit();

#### Boss testing

As shown in figure 5.21 boss test button and when the user press it he will move to the figure 5.22



Figure 5.21: Boss testing button

The code for this process shown below

Form7 f7 = newForm7();

this.Close();

f7.Show();

privatevoid Form7\_Load(object sender, EventArgs e)

{

string boss = "";

boss = "Imagin you are the mananger of the company and you are asked your employee .....what are you doing? and the answer was your functional requirements......would you agree on that or not.........thats what boss test mean ";

richTextBox1.Text = boss;

for (int i = 0; i <Form4.verb.Count; i++)

listBox1.Items.Add(Form4.verb[i]);

}

privatevoid button1\_Click(object sender, EventArgs e)

{

Form4 f4 = newForm4();

f4.Show();

this.Close();

}

Figure 5.22: Boss testing

**Description:** In this page the users can apply the boss test to their requirements.

#### Help centre

As shown in figure 5.23, help button and when the user press it he will move to the figure 5.24

**

Figure 5.23 help button

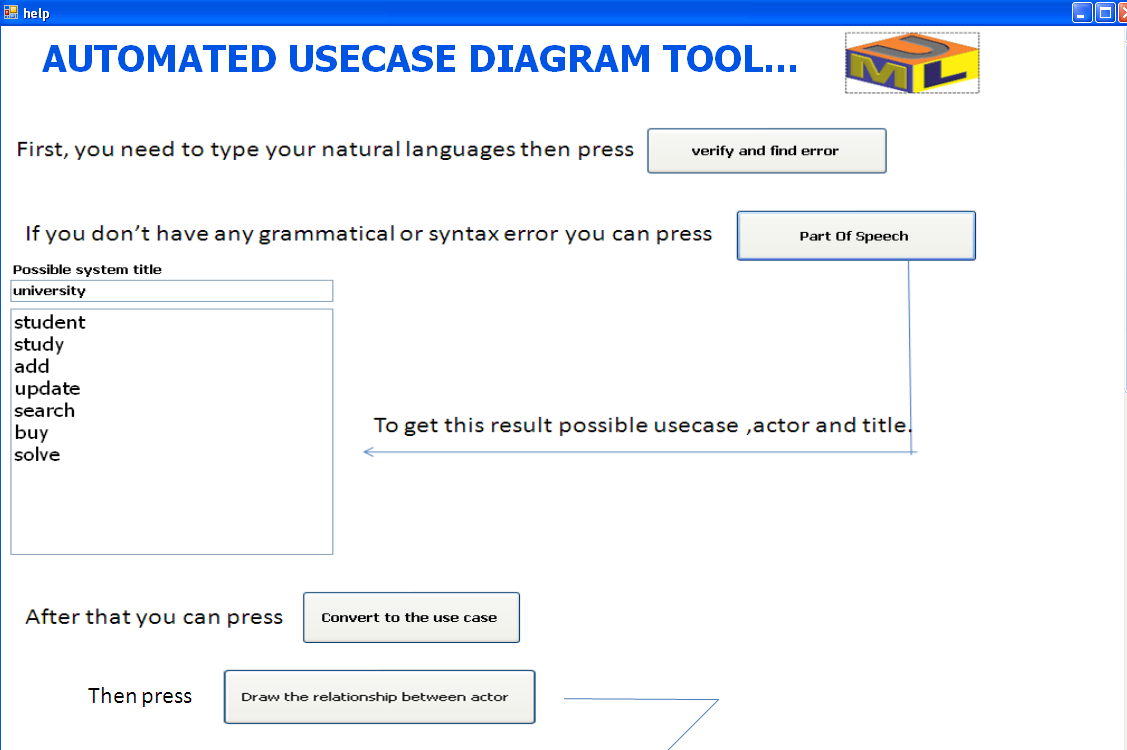
**

Figure 5.24: help centre

**Description:** This page help user to understand our project in good way, by providing full example.

#### Upload file

As shown in figure 5.25, upload file button and when the user press it he will move to the figure 5.26

**

Figure 5.25: upload file button

The code for this process shown below

OpenFileDialog openFile1 = newOpenFileDialog();

openFile1.Filter = "Text Files|\*.txt";

if (openFile1.ShowDialog() == System.Windows.Forms.DialogResult.OK)

richTextBox1.LoadFile(openFile1.FileName,

RichTextBoxStreamType.PlainText);

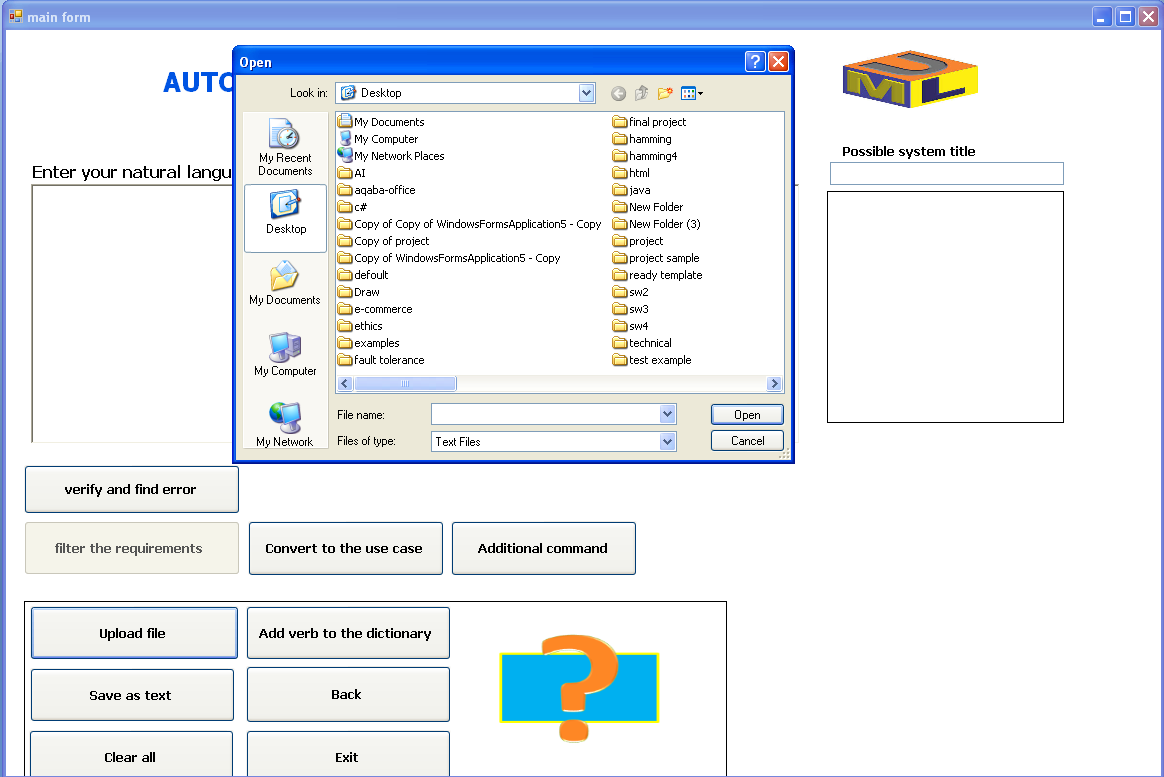
**

Figure 5.26: upload file

**Description:** This page allow user to import external requirements that written with text file to our project.

#### Save as test

As shown in figure 5.27 save as text button and when the user press it he will move to the figure 5.28

**

Figure 5.27: save as text button

The code for this process shown below

saveFileDialog1.Filter = "txt files (\*.txt)|\*.txt";

if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK

&& saveFileDialog1.FileName.Length > 0)

{

richTextBox1.SaveFile(saveFileDialog1.FileName,

RichTextBoxStreamType.PlainText);

}

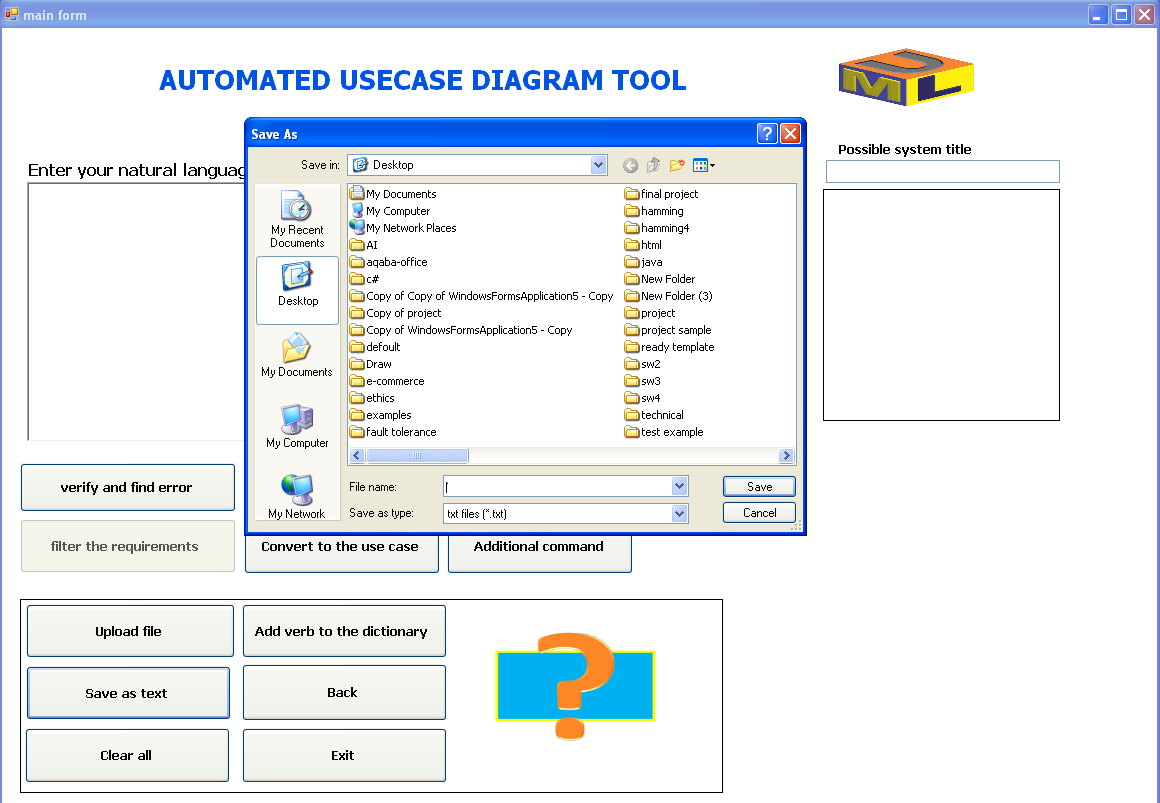
**

Figure 5.28: save file

**Description:** This page allow user to save the requirements as a text file.

#### Add verb to the dictionary

As shown in figure 5.29 add verb to the dictionary button and when the user press it he will move to the figure 5.30

**

Figure 5.29: Add verb to the dictionary button

panel3.Visible = true;

**

Figure 5.30 Add and search

The code for this process shown below

if (textBox5.Text != "")

if (listBox1.Items.Contains(textBox5.Text))

MessageBox.Show(" the verb is found");

else

{

listBox1.Items.Add(textBox5.Text);

MessageBox.Show("your verb have been added thank you");

}

File.AppendAllText("anfal.txt", textBox5.Text + " ");

TextReader tr = newStreamReader("anfal.txt");

listBox1.Items.Add(tr.ReadLine());

tr.Close();

textBox5.Text = "";

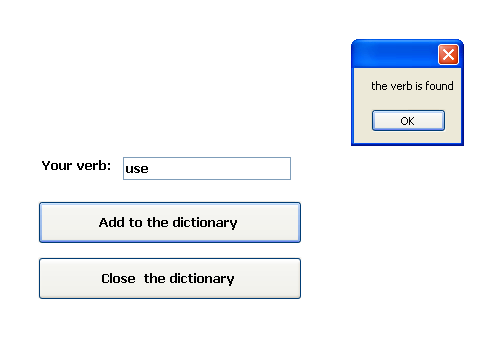
**

Figure 5.31: search in the dictionary

**Description:** when try to search about exiting verb in AUDT dictionary the above message will be displayed.

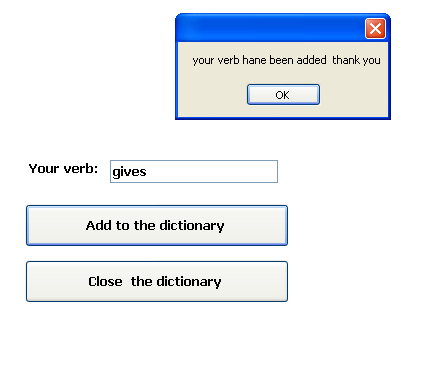
**

Figure 5.32: Add verbs to the dictionary

**Description:** This page allow user to add verbs to the dictionary, or search about verb in our dictionary. Actually the user adds verbs to the intermediate text file then the dictionary read this verb from this intermediate text file. As shown in this example:

And when the main form of our project have been loaded the content of this file will be add to our project. By using the following code:

richTextBox2.LoadFile("anfal.txt",

RichTextBoxStreamType.PlainText);

string[] add = richTextBox2.Text.Split(' ');

int f = 0;

foreach (string ad in add)

{

if(f<add.Length-1)

listBox1.Items.Add(ad);

f++;

}

#### Ready samples

As shown in figure 5.33 sample link label and when the user press it he will move to the figure 5.34

**

Figure 5.33: Sample link

The code for this process shown below

Form6 f6 = newForm6();

f6.ShowDialog();

richTextBox1.Text = alll;

publicstaticstring sample = "";

privatevoid linkLabel4\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

sample = "the student can study , add , update the university system. also they can search and buy books from the library.also the student can solve any problem.";

richTextBox1.Text = sample;

}

privatevoid linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

sample = "anfal ibrahim , heba suliman , nesreen basam , maram ryad and sara huseen develop a system of automated\_software\_tools.also the user do the same . ";

richTextBox1.Text = sample;

}

privatevoid linkLabel3\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

sample = "the user will write her/his requirements.controller can understand typed requirements and convert it to the usecase diagram.";

richTextBox2.Text = sample;

}

privatevoid linkLabel2\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

sample = "the teacher can teach the class.the student can study the exam.an user can use , add , update.admin do all and lead. ";

richTextBox2.Text = sample;

}

privatevoid linkLabel5\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

sample = "the user will write her/his requirements.controller can understandy typed requirements.";

richTextBox3.Text = sample;

}

privatevoid linkLabel6\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

sample = "play is player.study is student.";

richTextBox3.Text = sample;

}

privatevoid linkLabel7\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Form3.alll = sample;

}

privatevoid linkLabel8\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Form3.alll = sample;

}

privatevoid linkLabel9\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Form3.alll = sample;

}

privatevoid Form6\_Leave(object sender, EventArgs e)

{

}

privatevoid button1\_Click(object sender, EventArgs e)

{

this.Close();

}

Figure 5.34: Samples

**Description:** This page allow user to try different samples that have been written and stored in our project.

#### Ready template

As shown in figure 5.35 ready template button and when the user press it he will move to the figure 5.36 to chooses of the stored template.

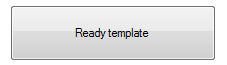
****

Figure 5.35: Ready templates button

The code for this process shown below

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

publicpartialclassForm2 : Form

{

public Form2()

{

InitializeComponent();

}

privatevoid radioButton1\_Click(object sender, EventArgs e)

{

pictureBox1.Visible = true;

pictureBox2.Visible = false;

pictureBox3.Visible = false;

pictureBox4.Visible = false;

pictureBox5.Visible= false;

pictureBox6.Visible = false;

pictureBox7.Visible = true;

pictureBox8.Visible = false;

pictureBox9.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton2\_Click(object sender, EventArgs e)

{

pictureBox3.Visible = true;

pictureBox1.Visible = false;

pictureBox2.Visible = false;

pictureBox4.Visible = false;

pictureBox5.Visible = false;

pictureBox6.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox9.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton3\_Click(object sender, EventArgs e)

{

pictureBox5.Visible = true;

pictureBox2.Visible = false;

pictureBox3.Visible = false;

pictureBox4.Visible = false;

pictureBox1.Visible = false;

pictureBox6.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox9.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton4\_Click(object sender, EventArgs e)

{

pictureBox4.Visible = true;

pictureBox2.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible= false;

pictureBox6.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox9.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton5\_Click(object sender, EventArgs e)

{

pictureBox2.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox6.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox9.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton2\_CheckedChanged(object sender, EventArgs e)

{

}

privatevoid radioButton6\_Click(object sender, EventArgs e)

{

pictureBox6.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox9.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton7\_Click(object sender, EventArgs e)

{

pictureBox8.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox9.Visible = false;

pictureBox6.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton8\_Click(object sender, EventArgs e)

{

pictureBox9.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox6.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton1\_CheckedChanged(object sender, EventArgs e)

{

}

privatevoid radioButton9\_Click(object sender, EventArgs e)

{

pictureBox10.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox6.Visible = false;

pictureBox9.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton10\_Click(object sender, EventArgs e)

{

pictureBox11.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox6.Visible = false;

pictureBox10.Visible = false;

pictureBox9.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton11\_Click(object sender, EventArgs e)

{

pictureBox12.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox6.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox9.Visible = false;

pictureBox13.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton12\_Click(object sender, EventArgs e)

{

pictureBox13.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox6.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox9.Visible = false;

pictureBox14.Visible = false;

}

privatevoid radioButton13\_Click(object sender, EventArgs e)

{

pictureBox14.Visible = true;

pictureBox4.Visible = false;

pictureBox3.Visible = false;

pictureBox1.Visible = false;

pictureBox5.Visible = false;

pictureBox2.Visible = false;

pictureBox7.Visible = false;

pictureBox8.Visible = false;

pictureBox6.Visible = false;

pictureBox10.Visible = false;

pictureBox11.Visible = false;

pictureBox12.Visible = false;

pictureBox13.Visible = false;

pictureBox9.Visible = false;

}

}

}

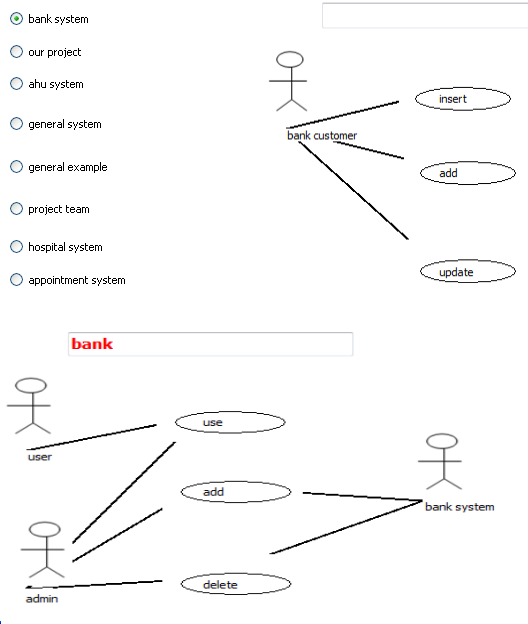
****

Figure 5.36: Bank system

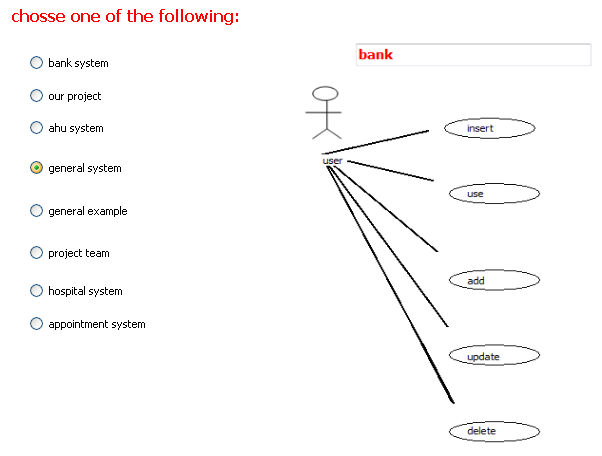
****

Figure 5.37: General system

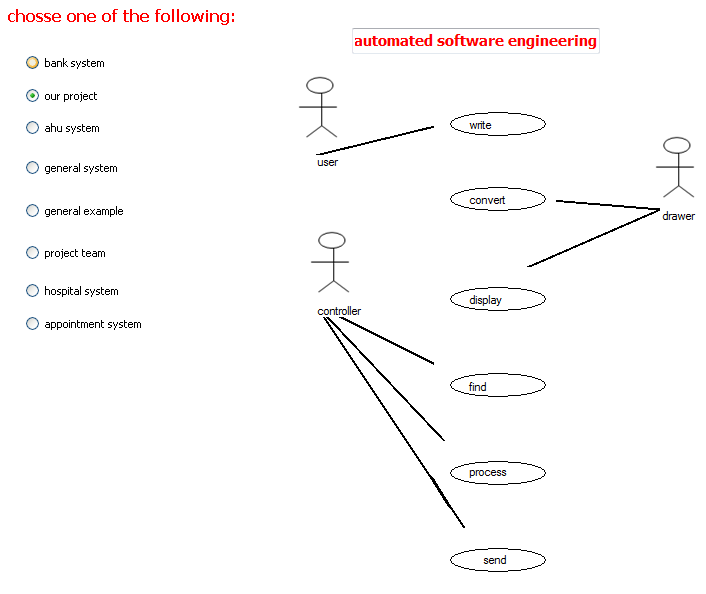
****

Figure 5.38: AUDT project

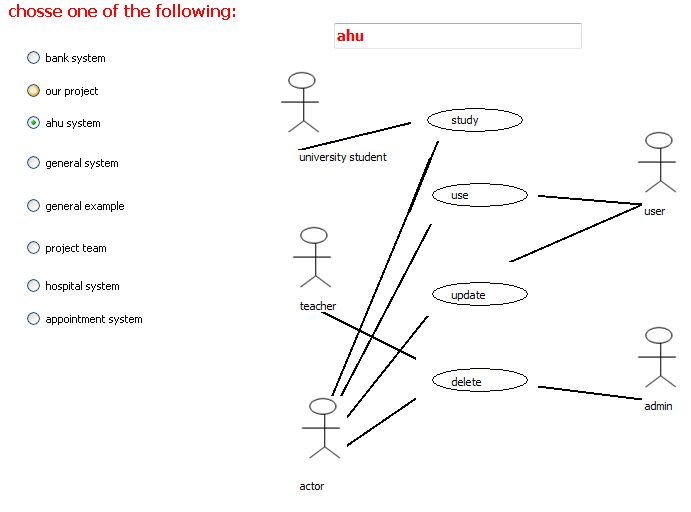
****

Figure 5.39: AHU system

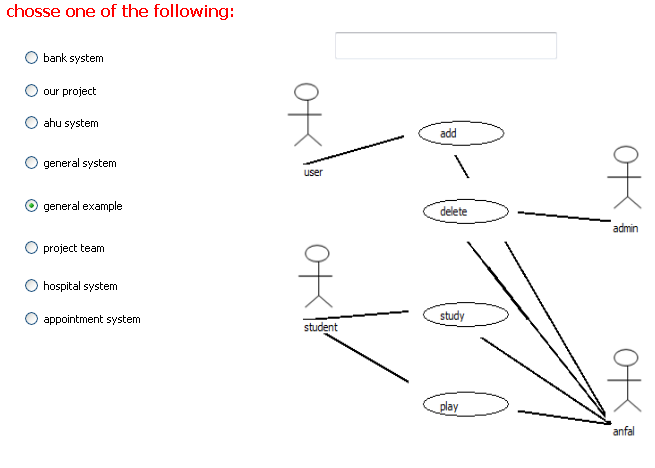
****

Figure 5.40: general example

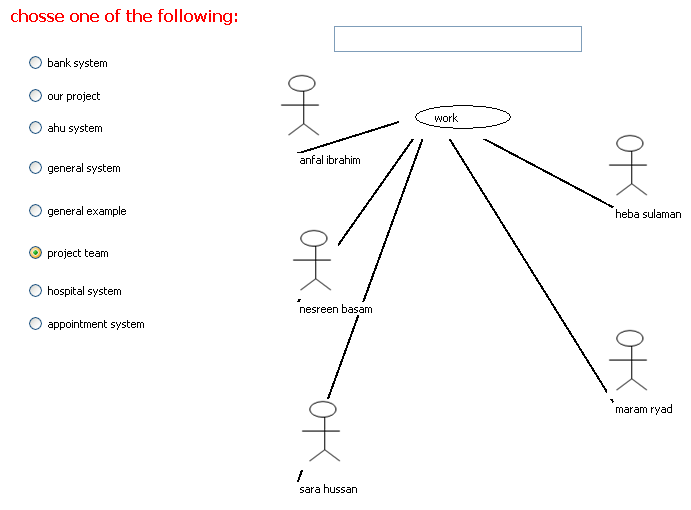
****

Figure 5.41: AUDT team

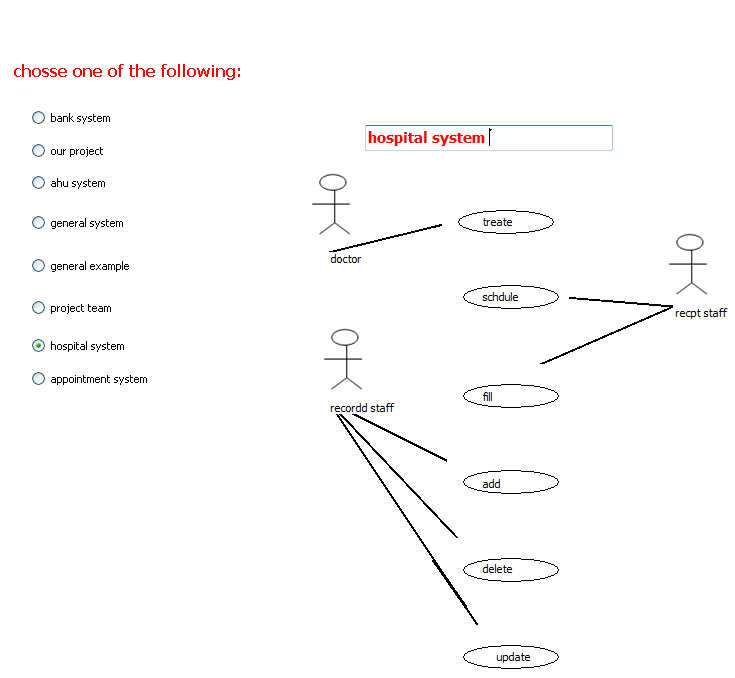
****

Figure 5.42: Hospital system

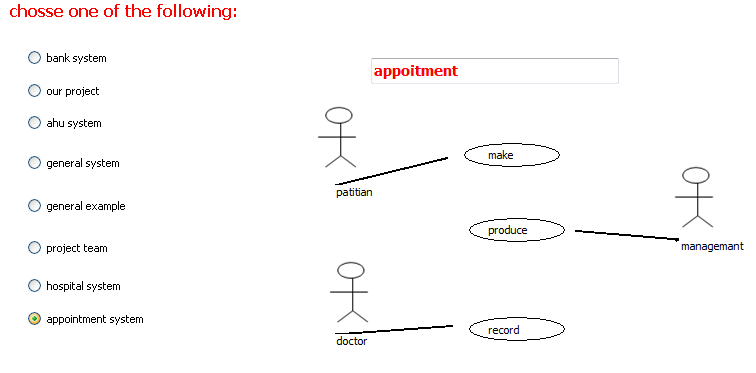
****

Figure 5.43: Appointment system

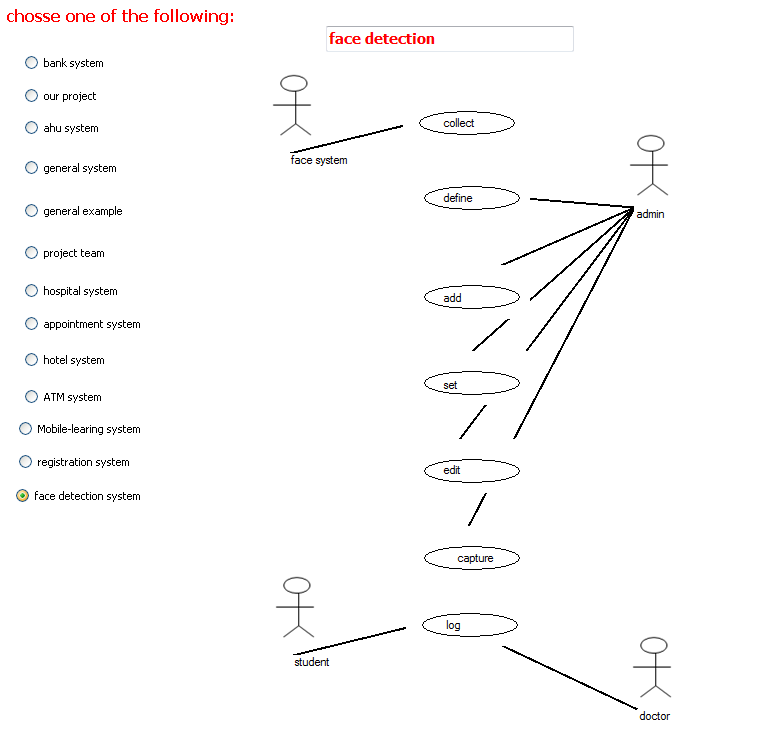
**

Figure 5.44: face detection system

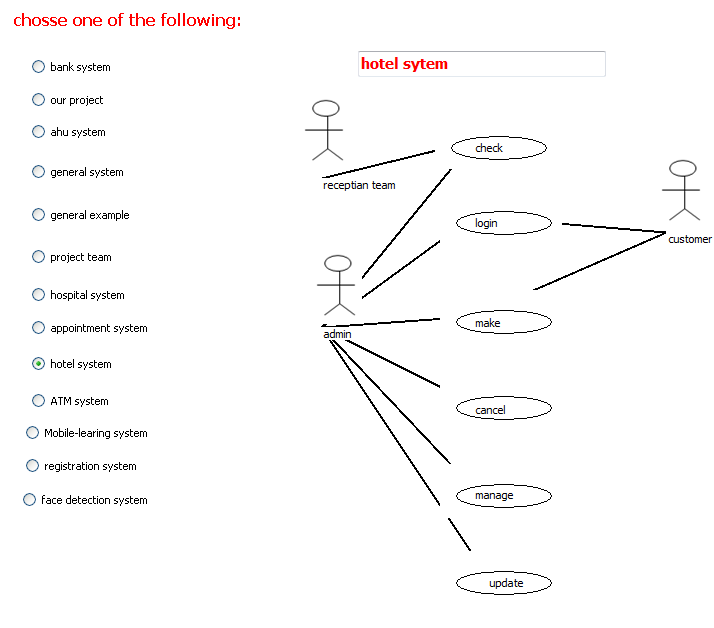
**

Figure 5.45: Hotel system

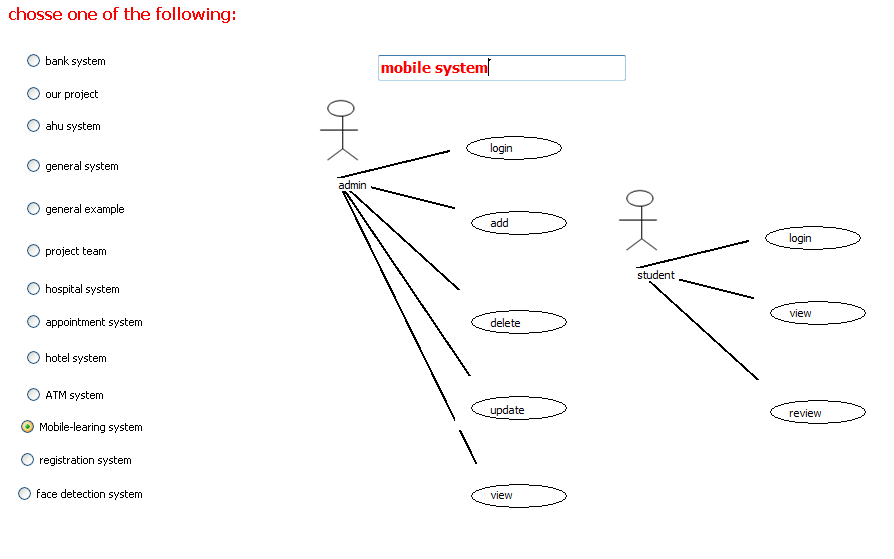
**

Figure 5.46: Mobile-learning system

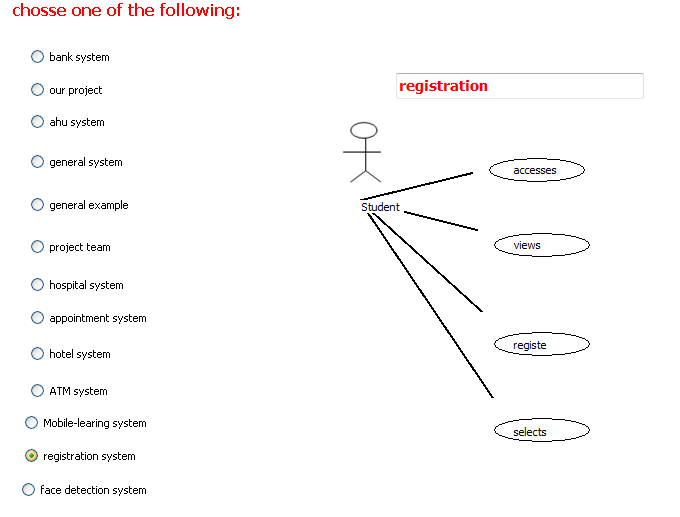
**

Figure 5.47: Registration system

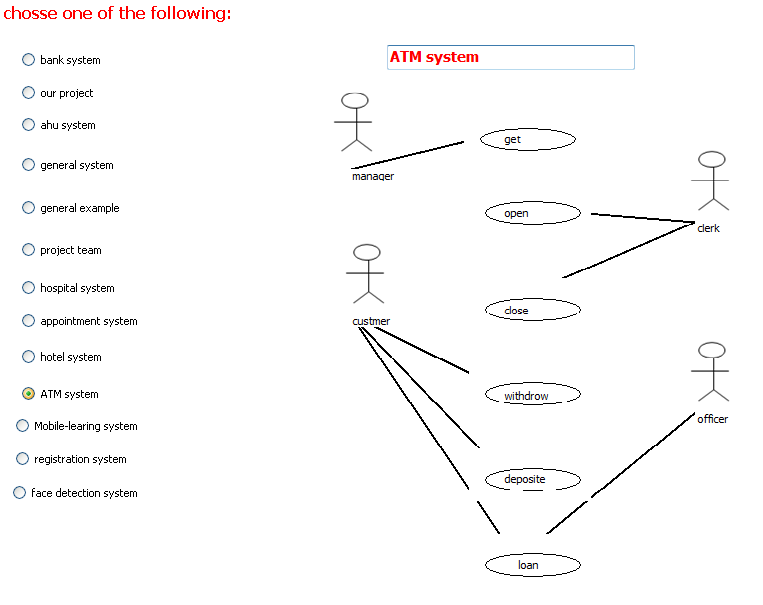
**

Figure 5.48: ATM system

**Description:** The figures from 5.53 to 5.47 ready sample that have been stored in the project.

#### Reset the application

As shown in figure 5.49 clear all button and when the user press it he will move to the figure 5.50

****

Figure 5.49: Clear all button

The code for this process shown below

listBox4.Items.Clear();

listBox11.Items.Clear();

richTextBox1.Text = " ";

textBox2.Text = " ";

listBox9.Items.Clear();

panel3.Visible = false;

Figure 5.50: Clear all (reset)

**Description:** The user can reset the system when he press clear all button.

#### Manual requirements

As shown in figure 5.51 manual requirements button and when the user press it he will move to the figure 5.52 to determine arrow width and color

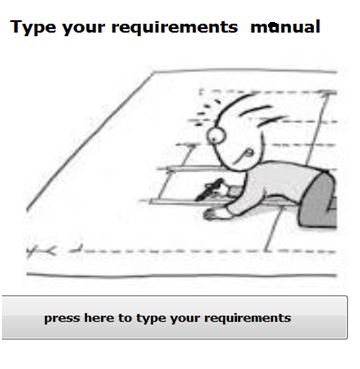


Figure 5.51 manual requirements

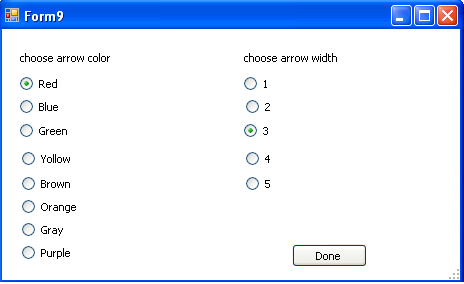


Figure 5.52: Arrow width and colour

The code for this process shown below

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

publicpartialclassForm9 : Form

{

public Form9()

{

InitializeComponent();

}

publicstatic Color c = Color.Black;

publicstaticint width = 2;

privatevoid radioButton1\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Red;

}

privatevoid radioButton2\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Blue;

}

privatevoid radioButton3\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Green;

}

privatevoid radioButton4\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Yellow;

}

privatevoid radioButton5\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Brown;

}

privatevoid Form9\_Load(object sender, EventArgs e)

{

}

privatevoid radioButton8\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Orange;

}

privatevoid radioButton7\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Gray;

}

privatevoid radioButton6\_CheckedChanged(object sender, EventArgs e)

{

c = Color.Purple;

}

privatevoid radioButton16\_CheckedChanged(object sender, EventArgs e)

{

width = 1;

}

privatevoid radioButton15\_CheckedChanged(object sender, EventArgs e)

{

width = 2;

}

privatevoid radioButton14\_CheckedChanged(object sender, EventArgs e)

{

width = 3;

}

privatevoid radioButton13\_CheckedChanged(object sender, EventArgs e)

{

width = 4;

}

privatevoid radioButton12\_CheckedChanged(object sender, EventArgs e)

{

width = 5;

}

privatevoid button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

publicpartialclassForm5 : Form

{

public Form5()

{

InitializeComponent();

}

publicint i = 0;

publicint j = 0;

publicint s = 0;

publicint w = 0;

publicint t = 7;

int z = 7;

publicstaticList<string> verb = newList<string>();

publicstaticList<string> none = newList<string>();

Boolean dra;

Graphics g;

Pen p;

int prevX;

int prevY;

privatevoid button3\_Click(object sender, EventArgs e)

{

MessageBox.Show("use the mouse to make relationship between actors");

for (int i = 0; i < listBox1.Items.Count; i++)

{

if(!none.Contains(listBox1.Items[i].ToString()))

none.Add(listBox1.Items[i].ToString());

}

for (int i = 0; i < listBox2.Items.Count; i++)

{

if (!verb.Contains(listBox2.Items[i].ToString()))

verb.Add(listBox2.Items[i].ToString());

}

foreach (Control lab in Controls)

{

{

if (lab is PictureBox)

if (j < none.Count)

{

string name = lab.Name;

string ser = "no";

if (name.Contains(ser))

{

lab.Visible = true;

j++;

}

}

}

}

foreach (Control lab in Controls)

{

if (lab is Label)

{

if (i < none.Count)

{

string name = lab.Name;

string ser = "n";

if (name.Contains(ser))

{

lab.Visible = true;

lab.Text = none[i].ToString();

i++;

}

}

}

}

foreach (Control crtw in Controls)

{

if (crtw is Label)

{

if (w < verb.Count)

{

string name = crtw.Name;

string ser = "ve";

if (name.Contains(ser))

{

crtw.Visible = true;

crtw.Text = verb[w].ToString();

w++;

}

}

}

}

foreach (Control crt in Controls)

{

if (crt is PictureBox)

{

if (s < verb.Count)

{

string name = crt.Name;

string ser = "ve";

if (name.Contains(ser))

{

crt.Visible = true;

s++;

}

}

}

}

privatevoid button1\_Click(object sender, EventArgs e)

{

if (textBox4.Text != "")

{

listBox1.Items.Add(textBox4.Text);

textBox4.Text = "";

}

else

MessageBox.Show("enter the none first");

}

privatevoid button6\_Click(object sender, EventArgs e)

{

if (textBox3.Text != "")

{

listBox2.Items.Add(textBox3.Text);

textBox3.Text = "";

}

else

MessageBox.Show("enter the verb first");

}

privatevoid button7\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedIndex > -1)

listBox1.Items.RemoveAt(listBox1.SelectedIndex);

else

MessageBox.Show("please choose the word first");

}

privatevoid button8\_Click(object sender, EventArgs e)

{

if (listBox2.SelectedIndex > -1)

listBox2.Items.RemoveAt(listBox2.SelectedIndex);

else

MessageBox.Show("please choose the word first");

}

privatevoid button2\_Click(object sender, EventArgs e)

{

Form1 f1 = new Form1();

f1.Show();

this.Close();

}

privatevoid Form5\_Load(object sender, EventArgs e)

{

dra = false;

g = CreateGraphics();

p = new Pen(Color.Black, 2);

}

privatevoid Form5\_Paint(object sender, PaintEventArgs e)

{

}

privatevoid Form5\_MouseUp(object sender, MouseEventArgs e)

{

dra= false;

}

privatevoid Form5\_MouseDown(object sender, MouseEventArgs e)

{

dra = true;

prevX = e.X;

prevY = e.Y;

}

privatevoid Form5\_MouseMove(object sender, MouseEventArgs e)

{

if (dra)

{

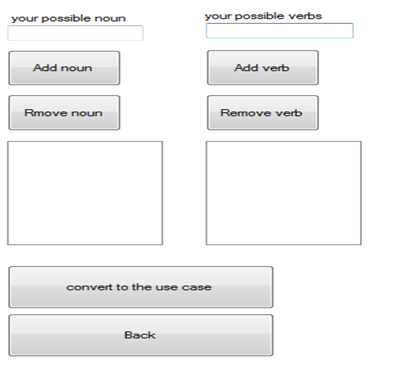
g.DrawLine(p, prevX, prevY, e.X, e.Y);

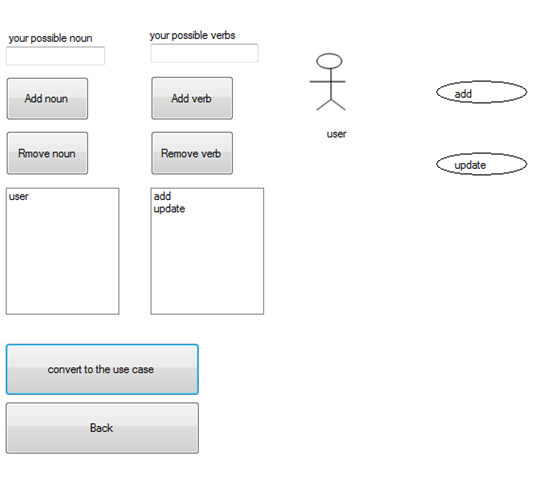
prevX = e.X;

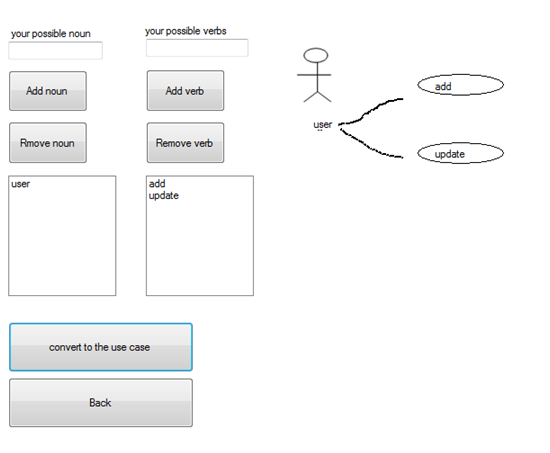
prevY = e.Y;

}

}



**



**

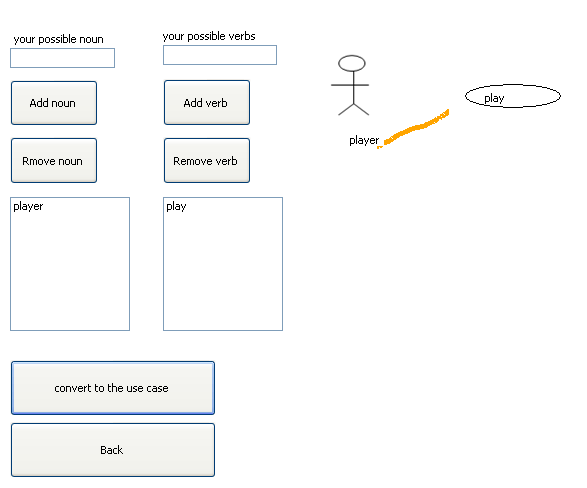
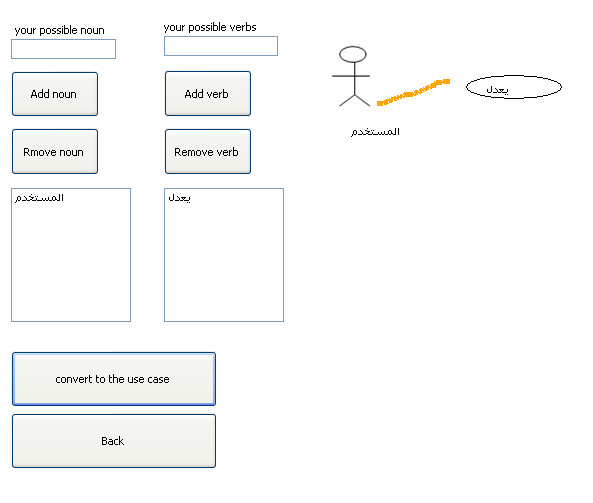
**

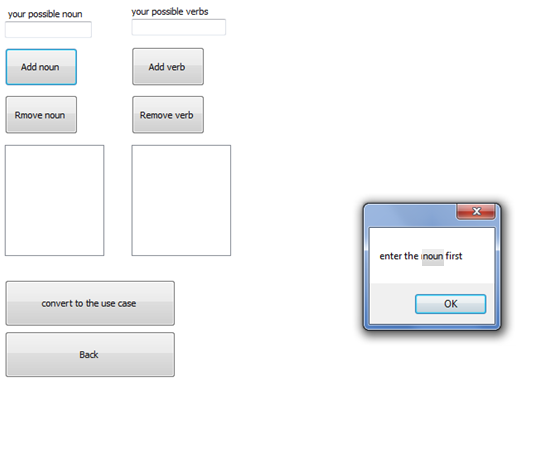
Figure 5.53: manual use case

**Description:** In this page you can type your requirements manual by typing just noun and verb then using mouse to make relationship between actors so the user can determine the arrow width and colour. Also the user can type Arabic languages as in the figure 5.54

##### Arabic natural language

Figure 5.54: Arabic languages requirements

##### Requirements validation

**

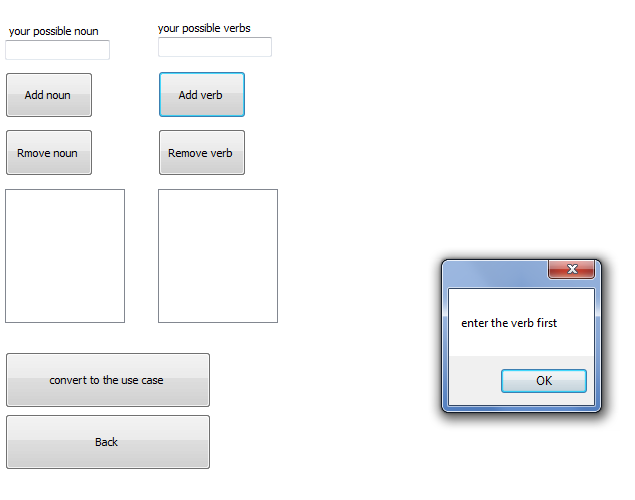
**

Figure 5.55: invalid requirements

**Description:** The pervious messages will be displayed if the user press add button with no requirements.

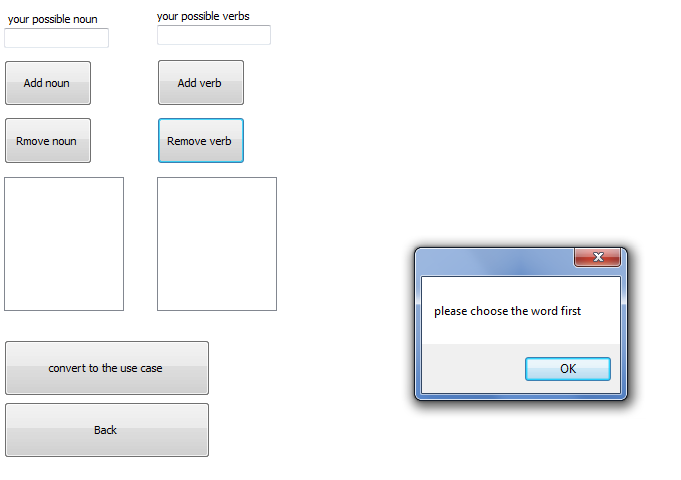
**

Figure 5.56: invalid requirements

**Description:** The previous message will be displayed if the user tries to press remove button with no selected items from the list to be removed.

#### Edit massage

As shown in figure 5.57 important note link label and when the user press it he will move to the figure 5.58



Figure 5.57: important note

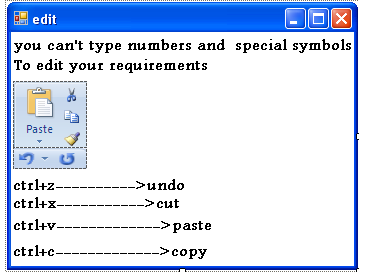


Figure 5.58: Edit massages

**Description:** This massage show to user how can edit his\her requirements using keyboard.

#### Open notepad

As shown in figure 5.59 open notepad button and when the user press it he will move to the figure 5.60

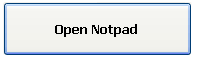


Figure 5.59: Open notepad button

The code for this process shown below

System.Diagnostics.Process p;

p = System.Diagnostics.Process.Start("Notepad");

Figure 5.60: Open notepad

**Description:** In this page the user can use and write the requirements on the notepad program.

#### Note

##### Use case diagram title

**NOTE**

When the user forget the use case title the following massage will be displayed

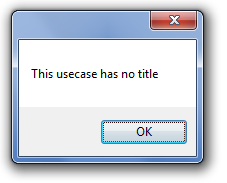


Figure5.61: No title massage

In addition to that the user can enter it manually in two appropriate places

The first one shown in the figure 5.58

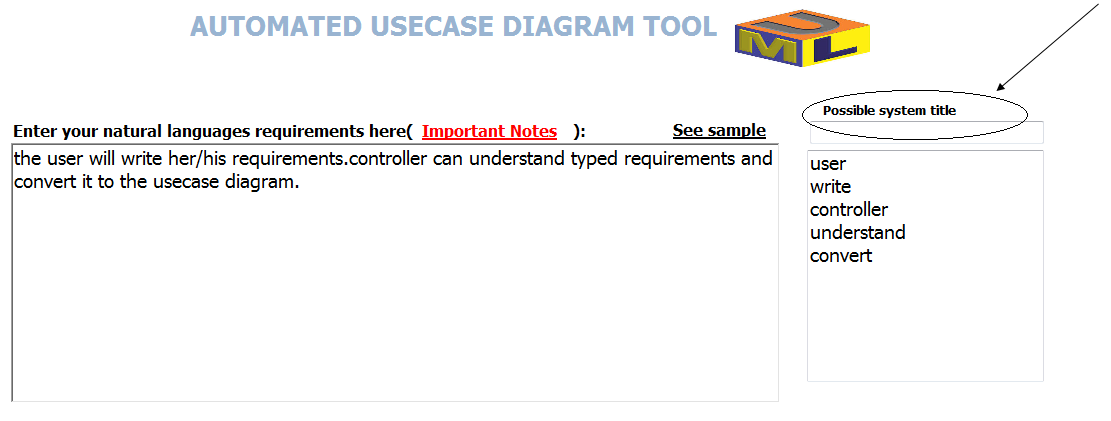


Figure5.62: appropriate places of title

The second place shown in the figure 5.60:

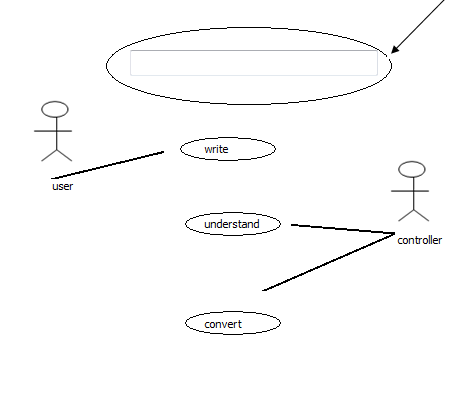


Figure5.63: appropriate places of title

##### Pre analysis

In order to prevent user from writing Arabic languages requirements, special symbols and numbers by using the following code:

private void richTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

String valid\_chr = "";

valid\_chr = valid\_chr + "abcdefghijklmnopqrstuvwxyz";

valid\_chr = valid\_chr + ".,\_";

valid\_chr = valid\_chr + "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

valid\_chr = valid\_chr + (char)Keys.Back + (char)Keys.Space;

if (!valid\_chr.Contains(e.KeyChar))

{

e.Handled = true;

}

}

Figure 5.64: Final result

# OBSERVE AND EVALUATE

## Introduction

***“Software testing*** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test**“**

**(**Cem Kaner, 2006)

It involves the execution of a software component or system to evaluate one or more properties of interest

## Data validation

***Data validation***s important because it ensures that the information input into an application to be processed falls within the accepted boundaries of that application or that particular function**,** without validation, the information processed by an application or system would be inaccurate, unreliable.

Data validation is the process of verifying that all the data that an application uses is valid.

Data validation is intended to provide certain well-defined guarantees for fitness, accuracy, and consistency for any of various kinds of user input into an application or automated system.

In computer science, data validations the process of ensuring that a program operates on clean, correct and useful data

In our project we can ensure that the data is valid by using this code:

privatevoid richTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

String valid\_chr = "";

valid\_chr = valid\_chr + "abcdefghijklmnopqrstuvwxyz";

valid\_chr = valid\_chr + ".,\_";

valid\_chr = valid\_chr + "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

valid\_chr = valid\_chr + (char)Keys.Back + (char)Keys.Space;

if (!valid\_chr.Contains(e.KeyChar))

{

e.Handled = true;

}

}

Which prevent user from writing Arabic languages requirements, special symbols and numbers.

In addition to that our project checks for the grammarian and syntax errors and don’t allow user to filter the requirements if such that errors are found.



Figure 6.1: disabled button

As you see in the figure 6.1 filter the requirements button will be disabled and the user can’t press it if there are any grammarian or syntax errors.

## Software testing

***“Software testing*** is an investigation conducted to provide stakeholders with information about the quality of the product or service under test**“**

**(**Cem Kaner, 2006)

It involves the execution of a software component or system to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

* meets the requirements that guided its design and development
* responds correctly to all kinds of inputs
* performs its functions within an acceptable time
* is sufficiently usable
* can be installed and run in its intended environment
* Achieves the general result its stakeholder's desire.

### Expert testing

****

Figure 6.2: expert testing

The first thing we do it make our English doctors as a reference in order to check for English grammarian and syntax……and to take in to account every possible states that the user can type for exception statements.

Then we test our project in order to check if it achieves its main goal "convert natural languages to the use case" and our project pass this test in excellent way.

# CONCLUSION

## Conclusion

The main aim for AUDT project is to develop automated use case diagram tools with distinct feathers that make it unlike to any other programs, AUDT accept requirements in the natural languages then convert them to the use case diagram by using c# programming languages.

AUDT will help and give support for our specialty "software engineering  
and that's our dream.

So in the end of our project we provide our thanks to everyone who has helped us….has given advices for us……

## Limitations and future work

Our future dream makes our project as a powerful tools to build use case from natural languages requirements………To include and extend relationship….apply boss testing …….take into account exception statements such as else. Take into account (NNN) when three nouns expressed as one actor (first name, middle name, last name).

We also hope if we can make automated software engineering tools to process Arabic languages requirements.

references

القران الكريم

Peter, Naur; Brian Randell (7–11 October 1968). "Software Engineering: Report of a

Conference sponsored by the NATO Science Committee" Garmisch, Germany: Scientific Affairs Division, NATO. Retrieved 2008-12-26)

ACM (2007) "Computing Degrees & Careers" ACM Retrieved 2010-11-23)

(Laplante, Phillip (2007). What Every Engineer Should Know about Software Engineering. Boca Raton: CRC. ISBN 978-0-8493-7228-5 Retrieved 2011-01-21

Krutchen, P. (2003) The Rational Unified Process: An Introduction (third Ed.)Addison-Wesley, Boston

Yue, T., Briand, L.C., Labiche, Y. (2010). An Automated approach to Transform Use Cases into Activity diagrams. In: 6th European Conference on Modeling Foundations and Applications, Paris, France, LCNS 6138, Springer.

The ASE annually International Conference on Automated Software Engineering, sponsored by IEEE and ACM (visit http://www.ase-conference.org)

Rifkin, Jeremy (1995). The End of Work: The Decline of the Global Labor Force and the Dawn of the Post-Market Era. Putnam Publishing Group. pp 66, 75.ISBN 0-87477-779-8

Bennett, S. (1979). A History of Control Engineering 1800-1930 London: Peter Peregrinus Ltd. ISBN 0-86341-047

IEEE Standard Glossary of Software Engineering Terminology,” IEEE STD 610.12-1990, 1990

(Sommerville, Ian (2007) [1982] "1.1.2what is software engineering?" Software engineering (8th Ed) Harlow, England: Pearson Education. p7 ISBN 0-321-31379-8

Software Engineering Information Processing (North-Holland Publishing Co.) 71: 530–538. 1972)

Humphrey, Albert (December 2005). "SWOT Analysis for Management Consulting.SRI Alumni Newsletter (SRI International)

Albert Humphrey the "Father" of TAM". TAM UKRetrieved 2012-06-03

[Exploratory Testing](http://www.kaner.com/pdfs/ETatQAI.pdf), Cem Kaner, Florida Institute of Technology, *Quality Assurance Institute Worldwide Annual Software Testing Conference*, Orlando, FL, November 2006

http://www.tutorialspoint.com

http://www.buzzle.com

http://Wikipedia, the free encyclopedia.

http://Hire programmers

http://Microsoft developer network

http://www.ask.com

1. software engineering in details

Software engineering is concerned with theories, methods and tools for professional software development.

Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.

Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.

Software engineering is concerned with cost-effective software development

**Software products**

Generic products

Stand-alone systems that are marketed and sold to any customer who wishes to buy them.

Examples – PC software such as graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

Customized products

Software that is commissioned by a specific customer to meet their own needs

Examples – embedded control systems, air traffic control software, traffic monitoring systems.

**Frequently asked questions about software engineering**:

|  |  |
| --- | --- |
| **Question** | **Answer** |
| What is software? | Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market. |
| What are the attributes of good software? | Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable. |
| What is software engineering? | Software engineering is an engineering discipline that is concerned with all aspects of software production. |
| What are the fundamental software engineering activities? | Software specification, software development, software validation and software evolution. |
| What is the difference between software engineering and computer science? | Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software. |
| What is the difference between software engineering and system engineering? | System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process. |

**Essential attributes of good software**:

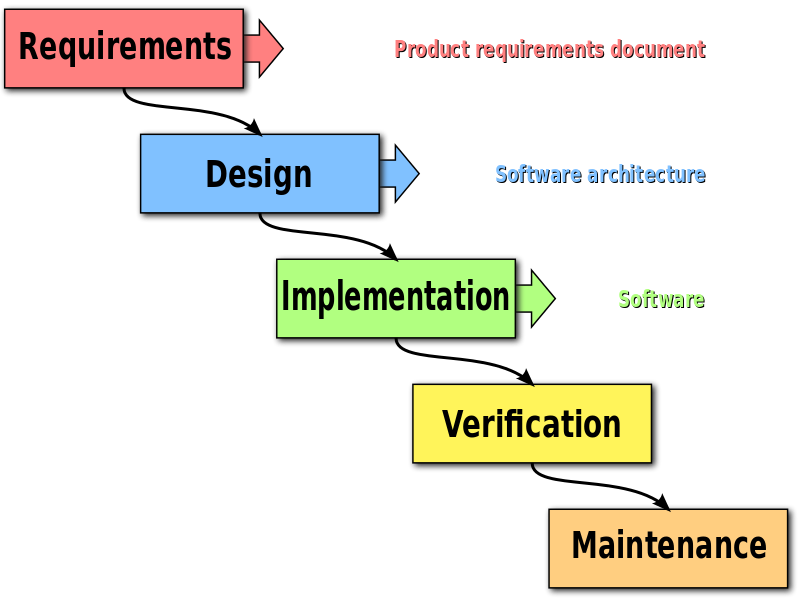
|  |  |
| --- | --- |
| **Product characteristic** | **Description** |
| Maintainability | Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment. |
| Dependability and security | Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system. |
| Efficiency | Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc. |
| Acceptability | Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use. |

**Importance of software engineering**:

More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.

It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

**Software development activities:**

****