

RE-ENGINEERING OF LEGACY SOFTWARE SYSTEMS

Ph.D. Thesis

Submitted to

**SAM HIGGINBOTTOM INSTITUTE OF AGRICULTURE,
TECHNOLOGY AND SCIENCES**

(Formerly Allahabad Agriculture Institute)

(Deemed to be University)

DOCTOR OF PHILOSOPHY

In

Computer Science and Information Technology

By

AHMED SALEEM ABBAS

(ID.NO. 11PHCSIT101)



2014

**Department of Computer Science & I. T.
Shepherd School of Engineering & Technology
ALLAHABAD-211007, INDIA**

SELF ATTESTATION

I hereby declare that the thesis entitled “**Re-engineering of Legacy Software Systems**” is my own work and that to the best of my Knowledge and belief. It contains no material previously submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Date:

Place: Allahabad

Ahmed Saleem



सैम हिगिगनबॉटम इन्स्टीट्यूट ऑफ एग्रीकल्चर, टेक्नालॉजी एण्ड साइंसेज
Sam Higginbottom Institute of Agriculture, Technology & Sciences
(Formerly Allahabad Agricultural Institute)
(Deemed to be University)
Allahabad - 211 007, India

ISO 9001:2008 Certified

Office : 91-532-2684281, 2684781
Fax : 91-532-2684394
Website : www.shiats.edu.in
E-mail : info@shiats.edu.in

CERTIFICATE OF ORIGINAL WORK

It is certified that the thesis entitled “**Re-Engineering Of Legacy Software Systems**” submitted to Sam Higginbottom Institute of Agriculture, Technology and Sciences, Deemed to be University, Allahabad, Uttar Pradesh in partial fulfillment of the requirement for the degree of Doctor of Philosophy in Computer Science and Information Technology is a bonafide record of research carried out by Mr. Ahmed Saleem Abbas under my guidance and supervision. The thesis is recommended for acceptance.

Associate Prof. (Dr.) W. Jeberson

Advisor

Head, Department of Computer Science and
Information Technology

Sam Higginbottom Institute of Agriculture,
Technology and Sciences,

Deemed to be University,

Allahabad, Uttar Pradesh-211007

Date:

Place: Allahabad



सैम हिगिनबॉटम इंस्टीट्यूट ऑफ एग्रीकल्चर, टेक्नालॉजी एण्ड साइंसेज
Sam Higginbottom Institute of Agriculture, Technology & Sciences
(Formerly Allahabad Agricultural Institute)
(Deemed to be University)
Allahabad - 211 007, India

ISO 9001:2008 Certified

Office : 91-532-2684281, 2684781
Fax : 91-532-2684394
Website : www.shiats.edu.in
E-mail : info@shiats.edu.in

THE CERTIFICATE OF ACCEPTANCE OF STUDENT ADVISORY COMMITTEE

The Thesis entitled “” has been prepared and submitted by Mr. Ahmed Saleem Abbas (ID No. 11PHCSIT101) in fulfillment of the requirement for the award of degree of Doctor of Philosophy of the Department of Computer Science and Information Technology.

Name, Designation	Evaluation	Signature
1. Associate Prof. (Dr.) W. Jeberson Advisor Secretary, SAC Committee Head, Department of Computer Science & I.T.	Satisfactory Non-satisfactory	-----
2. Prof. (Col.) Gurmit Singh Member, SAC Committee	Satisfactory Non-satisfactory	-----
3. Associate prof. Dr. Deva Raj Badugu, Member, SAC Committee	Satisfactory Non-satisfactory	-----

This thesis has been examined by student advisory committee and found acceptable..

Place: Allahabad
Date:

Prof (Dr.)
(Chairman)



सैम हिगिन्बॉटम इन्स्टीट्यूट ऑफ एग्रीकल्चर, टेक्नालॉजी एण्ड साइंसेज
Sam Higginbottom Institute of Agriculture, Technology & Sciences
(Formerly Allahabad Agricultural Institute)
(Deemed to be University)
Allahabad - 211 007, India

ISO 9001:2008 Certified

Office : 91-532-2684281, 2684781
Fax : 91-532-2684394
Website : www.shiats.edu.in
E-mail : info@shiats.edu.in

**DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION
TECHNOLOGY**

**EVALUATION REPORT OF Ph. D. VIVA-VOCE EXAMINATION OF MR.
AHMED SALEEM ABBAS**

This is to certify that **Ahmed Saleem Abbas** presented the oral defense of his Ph.D. thesis entitled **“Re-Engineering of Legacy Software Systems”**. His presentation on the topic was very impressive und practical oriented.

We the members of the evaluation committee (Board of Examiners) agree and recommend to the University for the award of Degree of Doctor of Philosophy in Computer Science and Information Technology to **Ahmed Saleem Abbas** (ID No. 11PHCSIT101)

Prof. (Dr.)
External Examiner

Associate Prof. (Dr.) W. Jeberson
Advisor
Head, Department of Computer Science and
Information Technology
Shepherd School of Engineering and
Technology SHIATS, Allahabad

Prof. (Dr.)
Chairman
Viva-Voce Exam Board

CONTENTS

TITLE	PAGE NO.
ACKNOWLEDGEMENT	i-i
ABSTRACT	ii-ii
LIST OF FIGURE	iii-ix
LIST OF TABLES	x-xi
LIST OF ABBREVIATIONS	xii-xiii
CHAPTER 1 INTRODUCTION	1-19
1.1 Back ground	1
1.2 Business process re-engineering	2
1.2.1 Business processes	2
1.2.2 Principles of business process re-engineering	2
1.2.3 A BPR model	4
1.3 Software re-engineering	6
1.3.1 Software maintenance	7
1.3.2 Software re-engineering process model	8
1.3.3 Why software re-engineering?	12
1.3.4 The benefits of software re-engineering	12
1.3.4.1 Increased maintainability	12
1.3.4.2 Improved performance	13
1.3.4.3 Increased interoperability	15
1.3.4.4 Decreased personal dependency	16
1.3.4.5 Improved testability	17
1.8 Research objectives	19
1.9 Organization of the thesis	19
CHAPTER 2 REVIEW OF LITERATURE	20-80
2.1 Introduction	20
2.2 Classification of software engineering research	21
2.3 Software development process	24
2.4 Software management	28
2.5 Software re-engineering	32

2.6	Software engineering techniques	37
2.7	Software applications	40
2.8	The gap in software engineering	43
2.9	Literature review on software re-engineering	44
2.10	Conclusion	80
CHAPTER 3	MATERIALS AND METHODS	81-106
3.1	Hardware and software tools	81
3.1.1	Tools and platform used	81
3.1.1.1	Supermarket management system (legacy system)	81
3.1.1.2	Visustin v7	81
3.1.1.3	ASP.NET	82
3.1.2	Operating system	83
3.1.3	Database tool	83
3.1.4	Front-end tool	84
3.1.5	Hardware and software configuration	85
3.1.5.1	Hardware used	85
3.1.5.2	Software configuration	85
3.2	FUSION re-engineering model	85
3.3	The main activities of the FUSION model	86
3.4	Legacy software system	91
3.4.1	The legacy supermarket management system	92
3.4.2	The objective of the legacy supermarket management system	92
3.4.3	The inputs to the legacy supermarket software are the following:	93
3.4.4	Output of the legacy supermarket software	94
3.4.5	Process logic of the legacy supermarket software	94
3.5	Implementation of FUSION model on legacy supermarket management system.	94
3.6	Applying FUSION model	106
CHAPTER 4	RESULT AND DISCUSSION	107-121
4.1	Comparison between legacy supermarket management system and the re-engineered system by using FUSION model	107

4.2	Differences between FUSION Re-engineering Model and other Models	114
4.2.1	Comparison between FUSION Re-engineering Model and General Model for Software Re-engineering	115
4.2.2	Comparison between FUSION Re-engineering Model and Iterative re-engineering process	117
4.2.3	Comparison between FUSION Re-engineering Model and Dual-Spiral Re-engineering Model	118
4.2.4	Comparison between FUSION Re-engineering Model and Hybrid Re-Engineering Model	119
4.3	The diagnostic features of FUSION Re-engineering Model	121
CHAPTER 5	SUMMARY AND CONCLUSION	122-122
5.1	Conclusion	122
5.2	Future works	122
REFERENCES		123-131
APPENDICES		132-252
A1.	The UML activity diagrams of legacy supermarket system	132
A2.	The DFD, ERD, Use case diagrams, execution flowchart and database of legacy supermarket system	154
A3.	The DFD, ERD, Use case diagrams, execution flowchart and database of re-engineered supermarket system	181
A4.	The UML activity diagrams of the re-engineered supermarket system	221

ACKNOWLEDGMENT

This thesis is a representation of the combined efforts and invaluable contributions of a large number of individuals and it is my great pleasure to acknowledge them with due obligations.

First of all, I would thank the God who has always blessed me by granting His kindness to complete this task,

I acknowledge my sincere thanks to Karbala University, for granting me an opportunity to obtain doctorate degree for the betterment of my academic career.

I am heartily thankful to my research Advisor, Associate Prof. (Dr.) W. Jeberson Head of Computer Science and Information Technology Department, Shepherd School of Engineering and Technology SHIATS, Allahabad, for providing invaluable guidance, constant supervision and helpful criticism at various computational stages of research work,

I express my appreciation for kind help provide by Assistant Prof. V.V. Kfinsega, and also all my colleagues and obedient technical staff of the Computer Science and Information Technology department for their kind help.

I would like to express my heartily regards to my great father and beloved mother, who gave me inspiration and perennial encouragement during my Education.

I would like to express my sincere thanks to my beloved wife Nada Khazal for her sacrifice and valuable assistance and my lovely kids, Saleem, and Lemar for giving encouragement.

(Ahmed Saleem Abbas)

RE-ENGINEERING OF LEGACY SOFTWARE SYSTEMS

ABSTRACT

(By Ahmed Saleem Abbas under the supervision of Associate Prof. (Dr.) W. Jeberson)

The various traditional software re-engineering processes were analyzed and improved new software re-engineering process model is proposed. The proposed process incorporates both the reverse engineering process and forward engineering process integrated with spiral model where reverse engineering can be applied to existing system code to extract design & requirements, although this is often used as means to mitigate risks & reduced costs of operation and maintaining the software system. The study represents how quality is going to be effect with the help of given software re-engineering process. After the proposed software re-engineering model was suggested and proved as a useful tool to convert old, obsolete systems to more efficient, streamlined systems, This model was used practically to convert the legacy supermarket management system to the target system according to the feedback of the customer, where the legacy system was used as a base line for this re-engineering process. Evaluation of the proposed software re-engineering model (FUSION re-engineering model) was made and the comparison with other model was made. The proposed model of re-engineering is successfully applied on a legacy supermarket management system. Through the analysis of FUSION model, it was found that the FUSION Software re-engineering process model reduces risk level, cost, Cyclomatic complexity, and effort. The use of this re-engineering approach provides a way to increase testability, portability, modifiability, functionality, reliability, performance, and security of legacy software. So it is concluded that the FUSION model is an effective software engineering process to perform the re-engineering of legacy software.

LIST OF FIGURES

FIG. NO.	TITLE	PAGE NO.
1.1	A BPR model	5
1.2	General Model for Software Re-engineering	7
1.3	A software re-engineering process model	11
1.4	The system deteriorates (measured in maintainability) as changes are implemented	13
1.5	Change C' (a re-structuring of the system) causes maintainability to (temporarily) increase	13
1.6	The effect of SRP on system performance	15
1.7	A Process for Interoperability	16
1.8	Parallel Projects	18
2.1	Classification of research papers selected for literature review	22
2.2	Distribution of research paper with respect to categories	22
2.3	Papers analyzed and year of publication	23
2.4	Papers published in various journals	24
2.5	software development process sub-categories	27
2.6	Number of papers published in the software development process based on the year of publishing.	28
2.7	Software management sub-categories	31
2.8	Number of papers published in the software management based on the year of publishing	32
2.9	Software Re-engineering sub-categories	35
2.10	Papers published in the software Re-engineering.	36
2.11	Software engineering techniques sub-categories	39
2.12	Papers published in the software engineering techniques.	40
2.13	Software engineering techniques sub-categories	42
2.14	Papers published in the software applications category	43
2.15	Papers published in each category from the year (2008-2012, March)	44
2.16	Iterative re-engineering process	45
2.17	Decremental Legacy system Pattern and Incremental Target system Pattern in Dual-Spiral Model	47
2.18	Hybrid Re-Engineering Model	79
3.1	Main window of the legacy supermarket management system	81
3.2	Main window of Visustin v7 analysis tool	82
3.3	Main window of ASP.NET of visual studio 2008	83
3.4	The main window of Microsoft SQL server 2008	84
3.5	The main window of Visual Basic	85
3.6	FUSION model	86
4.1	The comparison between the size (LOC) of the legacy supermarket system and the re-engineered system using FUSION model.	110

4.2	The comparison between the number of source code modules of the legacy supermarket system and the re-engineered system using FUSION model.	110
4.3	The comparison between the Cyclomatic complexity of the legacy supermarket system and the re-engineered system using FUSION model.	111
4.4	The comparison between the Mean Cyclomatic complexity of the legacy supermarket system and the re-engineered system using FUSION model.	111
4.5	The comparison between the decision density of the legacy supermarket system and the re-engineered system using FUSION model.	112
A1.1.a	UML activity diagram of the structures [about_Click(), billing_Click(), billrep_Click(), customer_Click()] in “fmEmployee.frm”	132
A1.1.b	UML activity diagram of the structures [custrep_Click(), dept_Click(), employee_Click(), emprep_Click()] in “fmEmployee.frm”	132
A1.1.c	UML activity diagram of the structures [exit_Click(), itrm_Click(), vender_Click()] in “fmEmployee.frm”	133
A1.1.d	UML activity diagram of the structure of Toolbar1_ButtonClick() in “fmEmployee.frm”	134
A1.2.a	UML activity diagram of the structures [cmdCancel_Click(), Calendar1_Click()] in “fmEmployee.frm”	135
A1.2.b	UML activity diagram of the structures of cmdAdd_Click() in “fmEmployee.frm”	135
A1.2.c	UML activity diagram of the structures [cmdedit_Click(), cmdexit_Click(), combDeptid_Click(), combDeptid_Click()] in “fmEmployee.frm”	136
A1.2.d	UML activity diagram of the structures of cmddelete_Click() in “fmEmployee.frm”	136
A1.2.e	UML activity diagram of the structures [cmdfirst_Click(), cmdLast_Click()] in “fmEmployee.frm”	137
A1.2.f	UML activity diagram of the structures [cmdnext_Click(), cmdprev_Click()] in “fmEmployee.frm”	137
A1.2.g	UML activity diagram of the structures [txttempname_KeyPress(KeyAscii As Integer), Picture1_Click(), txtcity_KeyPress(KeyAscii As Integer), txtdesign_KeyPress(KeyAscii As Integer), txttempid_KeyPress(KeyAscii As Integer)] in “fmEmployee.frm”	138
A1.2.h	UML activity diagram of the structures [txtstreet_KeyPress(KeyAscii As Integer), Form_Unload(Cancel As Integer), txtphone_KeyPress(KeyAscii As Integer), txtqulf_KeyPress(KeyAscii As Integer), txtsalary_KeyPress(KeyAscii As Integer)] in “fmEmployee.frm”	138
A1.2.i	UML activity diagram of the structures [Form_Load(), activecmd(toggle As Boolean), ClearRec()] in “fmEmployee.frm”	139
A1.2.j	UML activity diagram of the structures [txtzip_KeyPress(KeyAscii As Integer), ActiveRec (toggle As Boolean), txtstate_KeyPress(KeyAscii As Integer)] in “fmEmployee.frm”	140

A1.2.k	UML activity diagram of the structures of cmdSave_Click()in “fmEmployee.frm”	140
A1.2.l	UML activity diagram of the structures of cmdshow_Click()in “fmEmployee.frm”	141
A1.2.m	UML activity diagram of the structures of combDeptid_Click() in “fmEmployee.frm”	142
A1.2.n	UML activity diagram of the structures of ListView1_Click() in “fmEmployee.frm”	142
A1.2.o	UML activity diagram of the structures of Istempid_Click() in “fmEmployee.frm”	143
A1.2.p	UML activity diagram of the structures of WriteRec() in “fmEmployee.frm”	144
A1.2.q	UML activity diagram of the structures of ReadRec() in “fmEmployee.frm”,	145
A1.3.a	UML activity diagram of the structures of [init (),Timer1_Timer (), grid1show()] in “frmBilling”	146
A1.3.b	UML activity diagram of the structures of [puttbuffer (), grid2show()] in “frmBilling”	146
A1.3.c	UML activity diagram of the structures of [getccode (), getdepartment()] in “frmBilling”	147
A1.3.d	UML activity diagram of the structures of [Form_Load (), cmdPurchase_Click()] in “frmBilling”	147
A1.3.e	UML activity diagram of the structures of billno () in “frmBilling”	148
A1.3.f	UML activity diagram of the structures of combitemcode_Click () in “frmBilling”	149
A1.3.g	UML activity diagram of the structures of combDepartment_Click () in “frmBilling”	149
A1.3.h	UML activity diagram of the structures of txtBillno_Validate (Cancel as Boolean) in “frmBilling”	150
A1.3.i	UML activity diagram of the structures of cmdbuy_Click () in “frmBilling”	151
A1.3.j	UML activity diagram of the structures of txtqty_KeyPress (KeyAscii As Integer) in “frmBilling”	152
A2.1	Zero level DFD of legacy supermarket system	154
A2.2	The 1st Level Data Flow Diagram of legacy supermarket system	156
A2.3	2nd Level Data Flow Diagram of legacy supermarket system	160
A2.4.a	The Flow Chart to Add New employee	161
A2.4.b	The Flow Chart to edit employee information	162
A2.4.c	The Flow Chart to delete employee	163
A2.5.a	The Flow Chart to Add New vendor	164
A2.5.b	The Flow Chart to edit vendor information	165
A2.5.c	The Flow Chart to delete vendor	166
A2.6.a	The Flow Chart to Add New customer	167

A2.6.b	The Flow Chart to edit customer information	168
A2.6.c	The Flow Chart to delete customer	169
A2.7.a	The Flow Chart to Add New item	170
A2.7.b	The Flow Chart to delete item	171
A2.8	The Flow Chart for selling an item	172
A2.9	The use case diagram of the legacy supermarket management system	174
A2.10	The ERD diagram of the legacy supermarket management system	175
A3.1	Zero level DFD of the re-engineered Supermarket system	181
A3.2	1st Level Data Flow Diagram of the re-engineered system	183
A3.3	2'nd Level Data Flow Diagram of supermarket re-engineered system	189
A3.4.a	The Flow Chart to Add New operator	191
A3.4.b	The Flow Chart to edit operator information	192
A3.4.c	The Flow Chart to delete operator	193
A3.5.a	The Flow Chart to Add New supplier	194
A3.5.b	The Flow Chart to edit supplier information	195
A3.5.c	The Flow Chart to delete supplier	196
A3.6.a	The Flow Chart to Add New customer	197
A3.6.b	The Flow Chart to edit customer information	198
A3.6.c	The Flow Chart to delete customer	199
A3.7.a	The Flow Chart to Add New product	200
A3.7.b	The Flow Chart to edit product	201
A3.7.c	The Flow Chart to delete product	202
A3.8.a	The Flow Chart to Add New user	203
A3.8.b	The Flow Chart to edit user	204
A3.8.c	The Flow Chart to delete user	205
A3.9	The Flow Chart to update stock master	206
A3.10	The Flow Chart for selling an item	207
A3.11	The Flow Chart for Purchase an item	208
A3.12.a	The Flow Chart to Add New department	209
A3.12.b	The flowchart to edit department	210
A3.12.a	The Flow Chart to delete department	211
A3.13	The Flow Chart to Login process	212
A3.14	The use case diagram of the re-engineered supermarket management system	213
A3.15	The ERD diagram of the re-engineered supermarket management system	214
A4.1	UML activity diagram of the structures [using system, public partial class about : System.Web.UI.Page] in "about.aspx.cs"	221
A4.2	UML activity diagram of the structures [using System, public partial class admin : System.Web.UI.Page] in "admin.aspx.cs"	223
A4.3.a	UML activity diagram of the structures [using System, public partial class admin : System.Web.UI.MasterPage] in "admin master.cs"	224
A4.3.b	UML activity diagram of the structures [protected void Page_Load(object sender, EventArgs e)] in "admin master.cs"	225

A4.4.a	UML activity diagram of the structures of [using System, public partial class employee : System.Web.UI.Page] in " employee.aspx.cs"	225
A4.4.b	UML activity diagram of the structures of [protected void ProductMasBtm_Click (object sender, EventArgs e), protected void StockMasterBtm_Click (object sender, EventArgs e), protected void CustomerInfoBtm_Click (object sender, EventArgs e), protected void OperatorInfoB_Click (object sender, EventArgs e)] in "employee.aspx.cs"	226
A4.4.c:	UML activity diagram of the structures of [protected void SaleBtm_Click (object sender, EventArgs e), protected void PurchaseBtm_Click (object sender, EventArgs e), protected void LogOutB_Click (object sender, EventArgs e), protected void Page_Load (object sender, EventArgs e)] in "employee.aspx.cs"	226
A4.5.a	UML activity diagram of the structures of [using System, public partial class employee: System.Web.UI.MasterPage] in "employee.master.cs"	227
A4.5.b	UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e)] in "employee.master.cs"	227
A4.6.a	UML activity diagram of the structures of [using System, public partial class EmployeeRe : System.Web.UI.Page] in "EmployeeRe.aspx.cs"	228
A4.6.b	UML activity diagram of the structure of [using System, protected void Page_Load (object sender, EventArgs e)] in "EmployeeRe.aspx.cs"	228
A4.7.a	UML activity diagram of the structures of [using System, public partial class feedback : System.Web.UI.Page] in "feedback.aspx.cs"	229
A4.7.b	UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void ResetCmd_Click(object sender, EventArgs e)] in "feedback.aspx.cs"	229
A4.7.c	UML activity diagram of the structures of [protected void SubmitCmd_Click(object sender, EventArgs e)] in "feedback.aspx.cs"	230
A4.8.a	UML activity diagram of the structures of [using System, public partial class feedbackRe : System.Web.UI.Page] in "feedbackRe.aspx.cs"	230
A4.8.b	UML activity diagram of the structures of [using System, protected void Page_Load(object sender, EventArgs e)] in "feedbackRe.aspx.cs"	231
A4.9.a	UML activity diagram of the structures of [using System, public partial class _Default : System.Web.UI.Page] in "Login.aspx.cs"	231
A4.9.b	UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void Button3_Click1(object sender, EventArgs e)] in "Login.aspx.cs"	232
A4.9.c	UML activity diagram of the structures of [protected void LoginBt_Click1(object sender, EventArgs e)] in "Login.aspx.cs"	232
A4.10.a	UML activity diagram of the structures of [using System, public partial class regs : System.Web.UI.Page] in "regs.aspx.cs"	233
A4.10.b	UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void ResetBt_Click(object sender, EventArgs e)] in "regs.aspx.cs"	233

A4.10.c	UML activity diagram of the structures of [protected void RegBt_Click(object sender, EventArgs e)] in "regs.aspx.cs"	234
A4.11.a	UML activity diagram of the structures of [using System, public partial class edituser : System.Web.UI.Page] in "edituser.aspx.cs"	235
A4.11.b	UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e), protected void ReserCmd_Click (object sender, EventArgs e), protected void userid_SelectedIndexChang ed (object sender, EventArgs e)] in "edituser.aspx.cs"	235
A4.11.c	UML activity diagram of the structures of [protected void UpdateCmd_Click (object sender, EventArgs e)] in "edituser.aspx.cs"	236
A4.12.a	UML activity diagram of the structures of [using System, public partial class deleteuser: System.Web.UI.Page, protected void Page_Load (object sender, EventArgs e)] in " deleteuser.aspx.cs"	237
	UML activity diagram of the structure of [protected void SaveCmd_Click (object sender, EventArgs e)] in "ecustomer.aspx.cs"	237
A4.12.b	UML activity diagram of the structures of [protected void Button1_Click(object sender, EventArgs e)] in " deleteuser.aspx.cs"	237
A4.13.a	UML activity diagram of the structures of [using System, protected void Button5_Click(object sender, EventArgs e)] in "sale.aspx.cs"	238
A4.13.b	UML activity diagram of the structures of [public partial class Default3: System.Web.UI.Page] in "sale.aspx.cs"	238
A4.13.c	UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void ClearBt_Click(object sender, EventArgs e)] in "sale.aspx.cs"	239
A4.13.d	UML activity diagram of the structures of [protected void pcode_SelectedIndexChanged(object sender, EventArgs e)] in "sale.aspx.cs"	239
A4.13.e	UML activity diagram of the structures of [protected void PrintBt_Click(object sender, EventArgs e)] in "sale.aspx.cs"	240
A4.13.f	UML activity diagram of the structures of [protected void Calendar1_SelectionChanged(object sender, EventArgs e)] in "sale.aspx.cs"	241
A4.13.g	UML activity diagram of the structures of [protected void AddNewBt_Click(object sender, EventArgs e)] in "sale.aspx.cs"	241
A4.13.h	UML activity diagram of the structures of [protected void RefroBillBt_Click (object sender, EventArgs e)] in "sale.aspx.cs"	243
A4.13.i	UML activity diagram of the structures of [protected void AddTbilBt_Click1 (object sender, EventArgs e)] in "sale.aspx.cs"	245
A4.14.a	UML activity diagram of the structures of [using System, public partial class SaleRe : System.Web.UI.Page] in "SaleRe.aspx.cs"	246
A4.14.b	UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e)] in "SaleRe.aspx.cs"	246
A4.15.a	UML activity diagram of the structures of [using System, public partial	247

	class BillReport: System.Web.UI.Page] in “BillReport.aspx.cs”	
A4.15.b	UML activity diagram of the structures of [protected void Page_load(Object sender, EventArgs e)] in “BillReport.aspx.cs”	247
A4.16.a	UML activity diagram of the structures of [using System, protected void Page_Load (object sender, EventArgs e)] in “stock master.aspx.cs”	248
A4.16.b	UML activity diagram of the structures of [public partial class Default2: System.Web.UI.Page, protected void ClearBt_Click (object sender, EventArgs e)] in “stock master.aspx.cs”	248
A4.16.c	UML activity diagram of the structures of [protected void UpdateBt_Click(object sender, EventArgs e)] in “stock master.aspx.cs”	249
A4.16.d	UML activity diagram of the structures of [protected void FindBt_Click (object sender, EventArgs e)] in “stock master.aspx.cs”	250
A4.17	UML activity diagram of the structures of [using System, public partial class Stock_view: System.Web.UI.Page] in “Stock view.aspx.cs”	251
A4.18.a	UML activity diagram of the structures of [using System, public partial class StockRe : System.Web.UI.Page] in “StockRe.aspx.cs”	251
A4.18.b	UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e)] in “StockRe.aspx.cs”	252

LIST OF TABLES

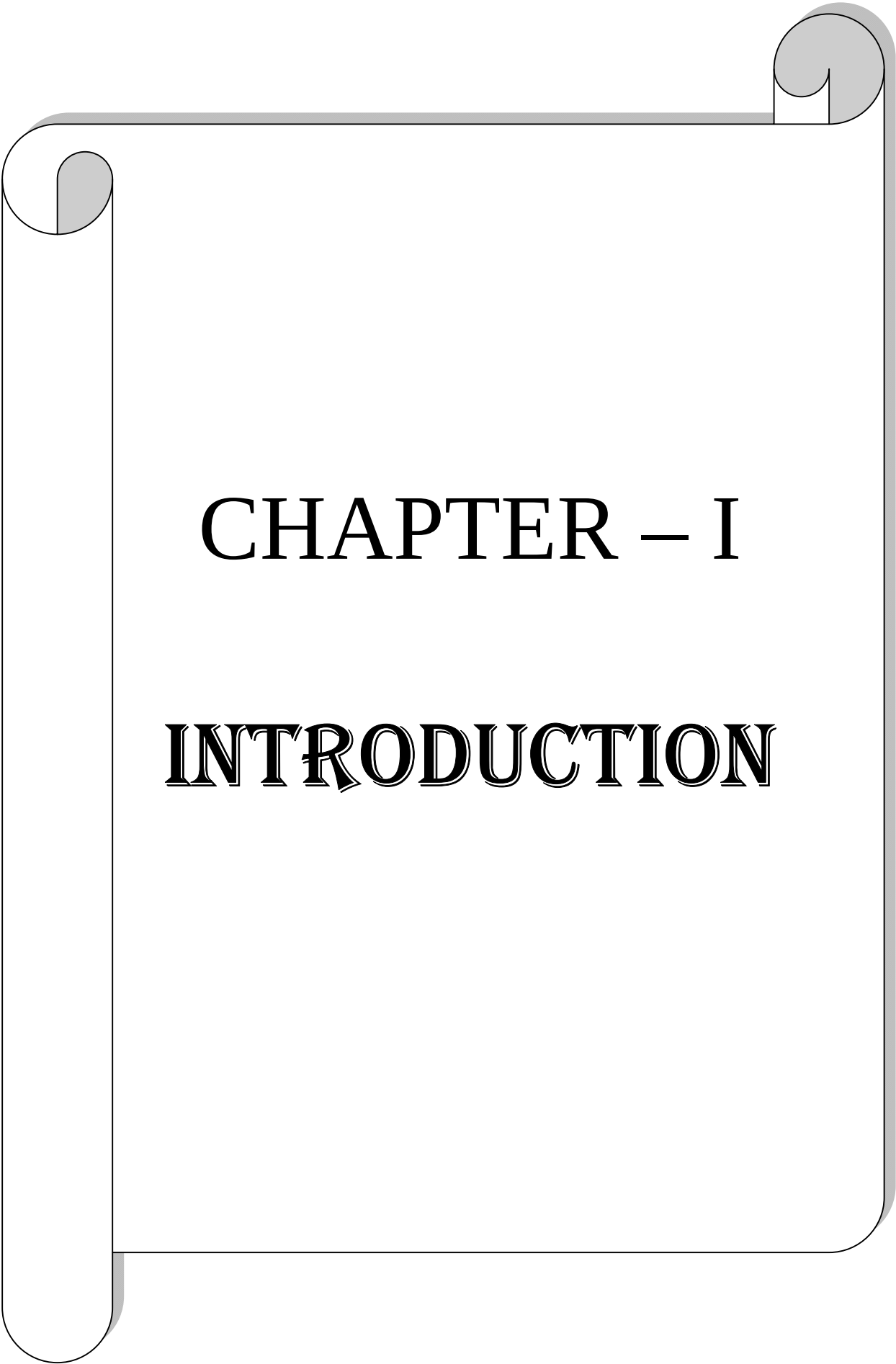
TABLE NO.	TITLE	PAGE NO.
2.1	Number of published papers in each category	23
2.2	Number of published papers in each year	23
2.3	Number of papers published in various Journals	24
2.4	Number of papers in each sub-category of Software Development Process	27
2.5	Number of papers in each sub-category of software management	32
2.6	Number of papers in each sub-category of Software Re-engineering	36
2.7	Number of papers published in each sub-category of Software engineering techniques	40
2.8	Numbers of papers published in each sub-category of Software application	42
3.1	Risk Evaluation according to Mean Cyclomatic Complexity	89
3.2	comparison between the code of bill generation that belong to legacy software and the re-structured code of re-engineered software	99
3.3	Testing and result of testing	100
4.1	Metrics Comparision between legacy supermarket management system and the re-engineered system using FUSION model	107
4.2	Technical comparison between legacy supermarket management system and the re-engineered system using FUSION model	108
4.3	Comparison between FUSION and General Software Re-engineering Model	115
4.4	Comparison between FUSION and Iterative Software Re-engineering Model	117
4.5	Comparison between FUSION and Dual-Spiral Software Re-engineering Model	118
4.6	Comparison between FUSION and Hybrid Software Re-engineering Model	120
A2.1	Employees information	176
A2.2	Vendors information	177
A2.3	Customers information	177
A2.4	Items information	178
A2.5	Customer bill information	179
A2.6	Transactions information	179

A2.7	Department information	180
A3.1	Operators information	213
A3.2	Suppliers information	216
A3.3	Customers information of re-engineered system	216
A3.4	Products information	217
A3.5	Sale information	217
A3.6	Purchase information	218
A3.7	Department information of re-engineered system	218
A3.8	Feedback information	219
A3.9	Stock information	219
A3.10	User login information	220

LIST OF ABBREVIATIONS

NOTATION	DESCRIPTION
AHF	Attribute Hiding Factor
BPEL	Business Process Execution Language
Ca/Ce	Afferent/Efferent coupling
CBD	Component-Based Development
CBSE	Component-based software engineering
CIF	Common Interchange/Intermediate Format
CORE	Component Oriented Reverse Engineering
COTS	common-off-the-shelf
CRUD	Create Read Update Delete
DCOM	Distributed Component Object Model
DPs	Design patterns
DSM	Dependency Structure Matrix
DSM	Dependency Structure Matrix
eDSM	enriched DSM
EIS	Enterprise Information System
ESIDE	An Integrated Development Environment for Component-Based Embedded Systems
FAMIX	FAMOOS Information Exchange Model
Gralab	timer
GReQL	Graph Repository Query Language
IBM's RSA	IBM Rational Software Architect
JEAs	Java Enterprise Applications
KOGGE	Kartographie-Orientiertes Graphisches Geschichte- Erkundungssystem
LISI	Levels of Information Systems Interoperability
MDE	Math Description Engineering
MHF	Method Hiding Factor
MPR	Medical Process Re-engineering
MSDN	Microsoft Developer Network

MTTMR _i	Mean Time To Maintenance Request for the i-th component
NAC	number of ancestor classes
OMG	Object Management Group
OO	Object Oriented
OOAD	Object Oriented Analysis and Design
OOP	Object oriented programming
ORBs	Object Request Brokers
OSS	Open-source software
PC	Personal Computer
PDF	Portable Document Format
PDS	product definition statement
PrF	Private Factor
PuF	Public Factor
QoS	Quality of Service
RACI	Responsible, Accountable, Consulted, and Informed chart
RAD	Rapid application development
RDSEFF	Resource Demanding Service Effect Specifications
ReDSeeDS	Requirements Driven Software Development System
RT _i	Re-engineering time for the i-th component,
SA	Software Architecture
SDLC	software development life cycle
SLOC	Source Line of Code
SOA	Service-Oriented Architecture
SOSR	Service-Oriented Software Re-engineering
SPE	Software performance engineering
SRE	Software Re-Engineering
SRS	Software Requirement Specification
TEL	Technology Enhanced Learning
TGraphs	Graph Technology
W3C	World Wide Web Consortium
WCCF	Weighted Class Complexity Factor
XSLT	Extensible Stylesheet Language Transformations



CHAPTER – I

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 Back ground

In a seminal article written for the Harvard Business Review, **Hammer (1990)** laid the foundation for a revolution in management thinking about business processes and computing:

The re-engineering should be done to the businesses by using the power of modern information technology to radically re-design the business processes in order to achieve dramatics improvements in their performance. Every company operates according to a great many unarticulated rules. Re-engineering strives to break away from the old rules about how to organize and conduct the business.

Like all revolutions, Hammer's call to arms resulted in both positive and negative changes. During the 1990s, some companies made a legitimate effort to re-engineer legacy software, and the results led to improved competitiveness. Other companies relied solely on downsizing and outsourcing (instead of re-engineering) to improve their business, where downsizing is reducing the number of employees on the operating payroll, and outsourcing is an arrangement in which one company provides services for another company, **DeMarco (1995)**.

During the first decade of the twenty-first century, the hype associated with re-engineering has waned, but the process itself continues in companies large and small. The nexus between business re-engineering and software engineering lies in a "system view". Software is often the result of the business rules that Hammer discusses. As these rules change, software must also change. Today, major companies have thousands of computer programs that support the old business rules. As managers work to change the bases to achieve greater effectiveness and competitiveness, software must keep pace. In some cases, this means the creation of major new computer-based systems. But in many others, it means the modification or rebuilding of existing applications, **Pressman (2005)**.

The word 're-engineering' is composed of two parts: 're-' and '-engineering'. It means that something is to be done again, and that engineering principles are to be followed in this task.

1.2 Business process re-engineering

Business process re-engineering (BPR) is the process that search for radical change in business process and then implements these change to achieve breakthrough results. This definition has been suggested in the Fortune magazine, **Stewart (1993)**.

1.2.1 BUSINESS PROCESSES

A business process is a set of logically related tasks performed to achieve a defined business outcome, **Davenport et al. (1990)**. Within the business process, peoples, equipment, material resources, and business procedures are united to produce a specified result. Examples of business processes include designing a modern product, purchasing services and supplies, hiring a new employee, and paying suppliers. Each demands a set of tasks and each draws on diverse resources within the business. Every business process has a defined customer - a person or group that receives the outcome (e.g., an idea, a report, a design, a product). In addition, business processes cross organizational boundaries. They require that different organizational groups participate in the “logically related tasks” that define the process. In reality every system is actually a hierarchy of subsystems. A business is no exception. The overall business is segmented in the following manner:

The business

 Business systems

 Business processes

 Business sub-processes

Each business system (also called business function) is composed of one or more business processes, and each business process is defined by a set of sub-processes. BPR (Business Process Re-engineering) can be applied at any level of the hierarchy, but as the scope of BPR broadens (i.e., as move upward in the hierarchy), the risks associated with BPR grow dramatically. For this reason, most BPR efforts focus on individual processes or sub-processes.

1.2.2 Principles of business process re-engineering

In many ways, BPR is identical in focus and scope to business process engineering. In an ideal setting, BPR should occur in a top-down manner, beginning with the identification of major business objectives and goals and culminating with a much more detailed specification of the tasks that define a specific business process. **Hammer (1990)** suggests

a number of principles that guide BPR activities when they begin at the top (business) level:

- **Organize around outcomes, not tasks.** Many companies have compartmentalized business activities so that no single person (or organization) has responsibility (or control) of a business outcome. In such cases, it is difficult to determine the status of work and even more difficult to debug process problems if they do occur. BPR should design processes that avoid this problem.
- **Have those who use the output of the process perform the process.** The intent of this recommendation is to allow those who need business output to control all of the variables that allow them to get the output in a timely manner. The fewer separate constituencies involved in a process, the smoother is the road to a rapid outcome.
- **Incorporate information processing work into the real work that produces the raw information.** As IT becomes more distributed, it is possible to locate most information processing within the organization that produces the raw data. This localizes control, reduces the time of communication, and puts computing power in the hands of those that have a vested interest in the information that is produced.
- **Treat geographically dispersed resources as though they were centralized.** Computer-based communications have become so sophisticated that geographically diverse groups can be placed in the same “virtual office.” For example, instead of running three engineering shifts at a single location, a global company can run one shift in Europe, a second shift in North America, and a third shift in Asia. In each case, engineers will work during day time hours and communicate via high-bandwidth networks.
- **Link parallel activities instead of integrating their results.** When different constituencies perform work in parallel, it is essential to design a process that demands continuing communication and coordination. Otherwise, integration problems are sure to result.
- **Put the decision point where the work is performed, and build control into the process.** Using software design jargon, this principle suggests a flatter organizational architecture with reduced factoring.
- **Capture data once, at its source.** Data should be stored on-line so that once collected it need never be re-entered.

Each of these principles represents a big picture view of BPR. Guided by these rules, business planners and process designers must begin process re-design.

1.2.3 A BPR MODEL

Like most engineering activities, business process re-engineering is iterative. Business goals and the processes that achieve them must be adapted to a changing business environment. For this reason, there is no start and end to BPR: it is an evolutionary process. A model for business process re-engineering is depicted in Fig.1.1, The model defines six activities:

- **Business definition.** Business goals are identified within the context of four key drivers: reduction of the cost, reduction of the time, improvement of the quality, and personnel development and empowerment. Goals may be defined at the business level or for a specific component of the business.
- **Process identification.** Processes that are critical to achieving the goals defined in the business definition are identified. They may then be ranked according to importance, need for modification, or according to any other way that is appropriate for the re-engineering activity.
- **Process evaluation.** The existing process is thoroughly analyzed and measured. Process tasks are identified; the costs and time consumed by process tasks are noted; and quality/performance problems are isolated.
- **Process specification and design.** Based on information gained during the first three BPR activities, use-cases are made for each process that is to be re-designed. Within the context of BPR, use-cases identify a scenario that delivers some outcome to a customer. With the use-case that represent the specification of the process, a new set of tasks are designed for the process.
- **Prototyping.** A re-designed business process must be prototyped before it is fully integrated into the business. This activity “tests” the process so that refinements can be made.
- **Refinement and instantiation.** This activity depends on the feedback that resulted from the prototype, the business process is refined to be better and then this refined business process will be instantiated within a business system.

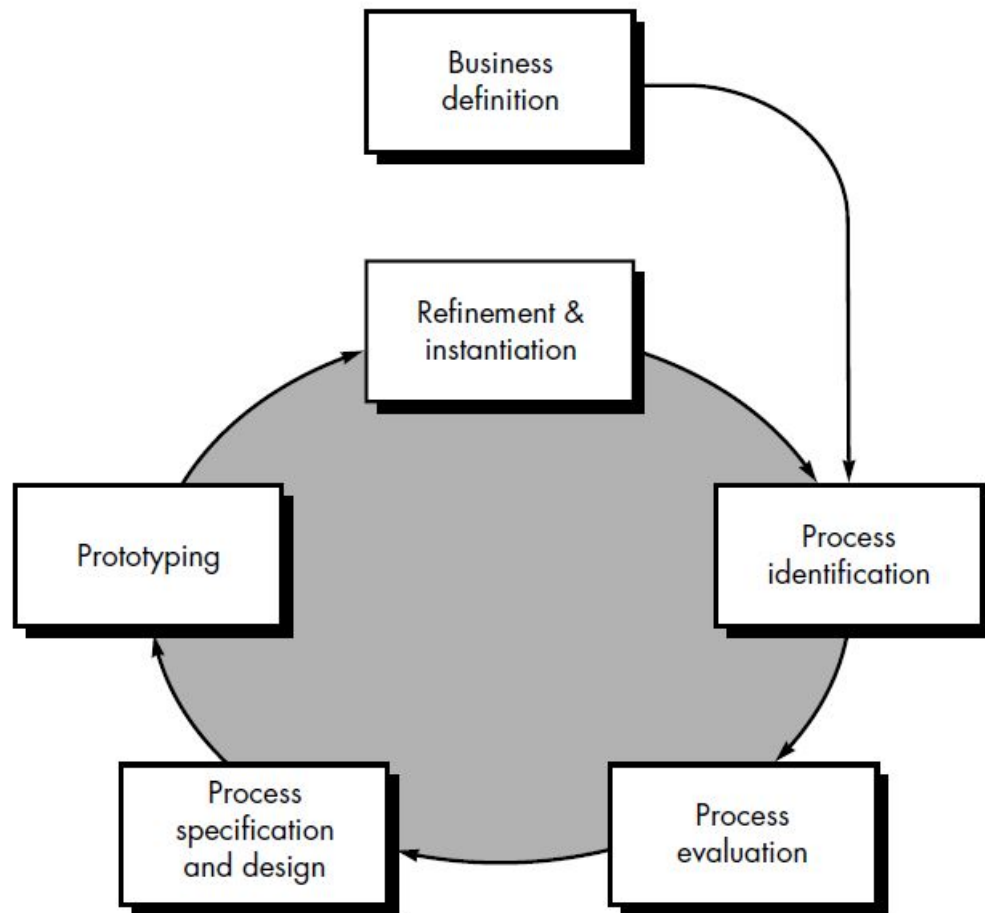


Fig.1.1: A BPR model.

These BPR activities are sometimes used in conjunction with workflow analysis tools. The intent of these tools is to build a model of existing workflow in an effort to analyze the existing processes. In addition, the modeling techniques commonly associated with business process engineering activities such as information strategy planning and business area analysis which can be used to implement the first four activities described in the process model of business process re-engineering, **Pressman (2005)**.

BPR can work, if it is used by motivated, trained people who recognize that process re-engineering; it is a continuous activity. If BPR is conducted effectively, information systems are better integrated into the business process. Re-engineering older applications can be examined in the context of a broad-based business strategy (broad-based is something with a wide or comprehensive basis, an example of broad-based is a television show that appeals to a very large audience), and priorities for software re-engineering can be established intelligently. But even if business re-engineering is a strategy that is rejected by a company, software re-engineering is something that must be done. Thousands of

legacy systems are in dire need of refurbishing or rebuilding, where legacy systems are the applications that are crucial to the success of businesses, **Pressman (2005)**.

1.3 SOFTWARE RE-ENGINEERING

Software Re-engineering is the examination, analysis and changing of an existing software system to re-constitute it in a modern form, and the subsequent implementation of the modern form. The process typically encompasses a combination of other processes such as reverse engineering, re-documentation, re-structuring, translation, and forward engineering. The goal is to understand the legacy software (specification, design, and implementation) and then to re-implement it to improve the system's functionality, performance or implementation, **Linda (2010)**.

The scenario of the life of an application is too common: An application has served the business needs of a company for 10 or 15 years. During that time this application has been corrected, adapted, and enhanced many times. The specialist people approached this work with good intention, but good software engineering practices were always handled to the side. Now the status of the application is unstable. It still works, but all the time a change is attempted, unexpected and many side effects occur. Still the application must continue to evolve. This problem of instability occurs to most of the applications, so what has to be done to overcome it?

Unmaintainable software is not a new problem. In fact, the broadening emphasis on software re-engineering has been spawned by a software maintenance “iceberg” that has been building for more than three decades.

There are four re-engineering objectives. They are: Preparation for functional enhancement, improve maintainability, migration, and improve reliability, **Linda (2010)**. Fig.1.2 refers to the general model of software re-engineering and illustrates the reverse engineering and forwarded engineering.

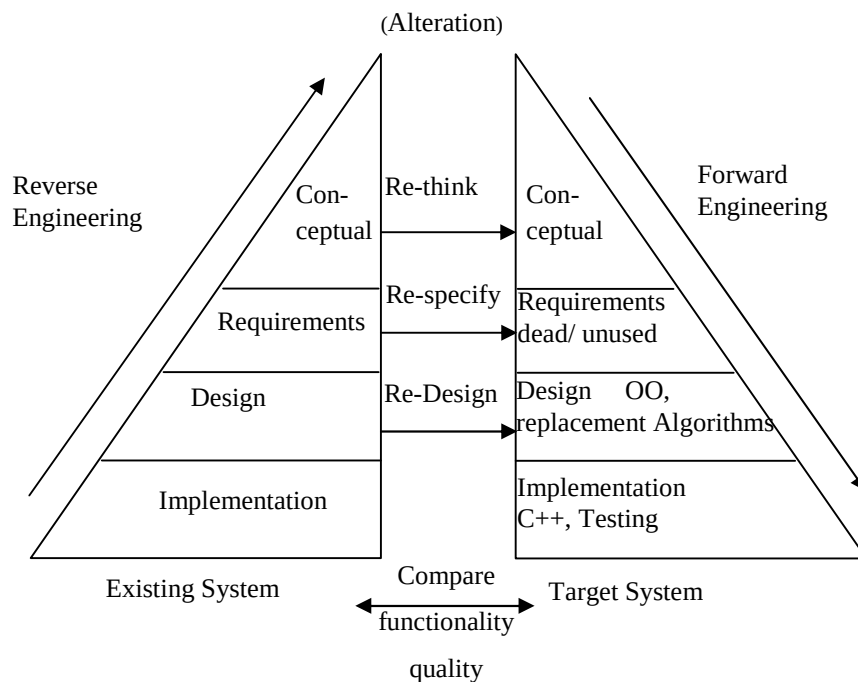


Fig.1.2: General Model for Software Re-engineering.

For example when one would wish to re-engineer code could be the modification of a date field (Y2K) or possibly re-modularization of code to facilitate later maintenance, where (Y2K) is another name for the year 2000, it also mean $[Y(\text{ear}) + 2 + K]$, where K is the symbol for the number 1000. In a typical (Y2K) scenario, at the midnight on (December 31, 1999) most programs could only handle 20th-century dates, when dates turned to '00', computer would default to 1900 that produce erroneous results which affected dates and times on and after (January 1, 2000), early program design represented years with two digit placeholders (MM/DD/YY format, i.e. 08/30/99) instead of four. This problem was solved by re-engineering the old system and doing transformation to the size of the date field in a record, and other transformations that are concerned with code that is used for outputting dates.

1.3.1 Software maintenance

Thirty years ago, software maintenance was characterized as an "iceberg". There is an enormous mass of potential problems and cost lies under the surface of software maintenance. In the early 1970s, the maintenance iceberg was big enough to sink an aircraft carrier. But today, it could easily sink the entire navy!, **Canning (1972)**.

The maintenance of existing software can account for over 60 percent of all effort expended by a development organization, and the percentage continues to increase as more software is produced, **Manna (1993)**. Uninitiated readers may ask why so much maintenance is required and why so much effort is expended. **Osborne and Chikofsky (1990)** provide a partial answer:

Much of the software that business depends on today is on average 10 to 15 years old. Also these programs were created using the best design and coding techniques known at that time. They were developed when program size and storage space were principle concerns. They were then migrated to modern platforms, adjusted for changes in machine and operating system technology and enhanced to meet new user requirement, all without enough regard to overall architecture. The result is the poorly designed structures, and poor in coding, logic, and documentation of the software systems. This answer is given by **Osborne and Chikofsky (1990)**.

Software maintenance is far more than fixing mistakes. The maintenance was defined by describing four activities that are undertaken after a program is released for use; maintenance activities: corrective, adaptive, perfective, and preventive maintenance or re-engineering. Only about 20 percent of all maintenance work is spent on fixing mistakes, the remaining 80 percent is spent on adapting existing systems to changes in their external environment, making improvements requested by users, and re-engineering an application for future use. When maintenance is considered to include all of these activities, it needs so much effort **Swanson (1976)**.

1.3.2 Software re-engineering process model

Re-engineering takes time; it costs significant amounts of money; and it absorbs resources that might be otherwise occupied on immediate concerns. For all of these reasons, re-engineering is not accomplished in a few months or even in a few years. Re-engineering of information systems is an activity that will absorb information technology resources for many years. That's why every organization needs a pragmatic strategy for software re-engineering, **Pressman (2005)**.

Re-engineering is a re-building activity, and the software re-engineering process model defines six activities shown in Fig.1.3. In some cases, these activities occur in a linear sequence, but this is not always the case. For example, it may be that reverse engineering (understanding the internal workings of a program) may have to occur before document re-

structuring can commence. The re-engineering paradigm shown in Fig.1.3 is a cyclical model. This means that each of the activities presented as a part of the paradigm may be revisited. For any particular cycle, the process can terminate after any one of these activities, **Pressman (2005)**.

1. **Inventory analysis.** Every software organization should have an inventory of all applications. The inventory can be simple thing like a spreadsheet model containing information that provides a detailed description (e.g., size, age, business criticality) of every active application. By sorting this information according to the criticality of business, longevity, current maintainability, and other locally important criteria, candidates for re-engineering appear. Resources can then be allocated to candidate applications for re-engineering work. It is important to note that the inventory should be revisited on a regular cycle. The status of applications (e.g., business criticality) can change as a function of time, and as a result, priorities for re-engineering will shift, **Pressman (2005)**.
2. **Document restructuring.** Weak documentation is the trademark of many legacy systems. One of the following steps should be taken to avoid it:
 - Creating documentation is far too time consuming. If the system works, the user lives with what he has. In some cases, this is the correct approach. It is not possible to re-create documentation for hundreds of computer programs. If a program is relatively static, is coming to the end of its useful life, and is unlikely to undergo significant change.
 - Documentation must be updated, but limited resources are available. So “document when touched” approach will be used. It may not be necessary to fully re-document an application. Rather, those portions of the system that are currently undergoing change are fully documented. Over time, a collection of useful and relevant documentation will evolve.
 - The system is business critical and must be fully re-documented. Even in this case, an intelligent approach is to pare documentation to an essential minimum.

Each of these options is viable. A software organization must choose the one that is most appropriate for each case, **Pressman (2005)**.

3. **Reverse engineering.** The term reverse engineering has its origins in the hardware world. A company disassembles a hardware product of competitive company in an

effort to understand its competitor's design and manufacturing "secrets." These secrets could be easily understood if the competitor's design and manufacturing specifications were obtained. But these documents are proprietary and unavailable to the company doing the reverse engineering. In essence, successful reverse engineering derives one or more design and manufacturing specifications for a product by examining actual specimens of the product. Reverse engineering for software is quite similar. In most cases, however, the program to be reverse engineered is not a competitor's. Rather, it is the company's own work (often done many years earlier). The "secrets" to be understood are obscure because no specification was ever developed. Therefore, reverse engineering for software is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code. Reverse engineering is a process to recover the design of the legacy software. Reverse engineering tools extract design information about data, architecture, and procedure from an existing program, **Pressman (2005)**.

4. **Code re-structuring.** The most common type of re-engineering (actually, the use of the term re-engineering is questionable in this case) is code re-structuring. Some legacy systems have relatively solid program architecture, but individual modules were coded in a manner that makes them difficult to understand, test, and maintain. In these cases, the code within the suspicious modules can be re-structured. To accomplish this activity, the source code is analyzed using a re-structuring tool. Violations of structured programming constructs are noted and code is then re-structured (this can be done automatically). The resultant re-structured code is reviewed and tested to ensure that no anomalies have been introduced, also the internal code documentation is updated, **Pressman (2005)**.
5. **Data re-structuring.** A program with weak data architecture will be difficult to adapt and enhance. In fact, for many applications, data architecture has more to do with the long-term viability of a program than the source code itself. Unlike code re-structuring, which occurs at a relatively low level of abstraction, data structuring is a full-scale re-engineering activity. In most cases, data re-structuring begins with the activity of reverse engineering. Current data architecture is analyzed and necessary data models are specified. Data objects and attributes are identified, and the quality of existing data structures are reviewed. When data structure is weak (e.g., flat files are currently implemented, when a relational method would very simplify processing), the data are

re-engineered. Because data architecture has a strong influence on program architecture and the algorithms that manipulate it, changes to the data will invariably result in either architectural or code-level changes, **Pressman (2005)**.

6. **Forward engineering.** In a perfect world, applications would be re-built using an automated re-engineering engine. The old program would be fed into the engine, analyzed, re-structured, and then re-generated in a form that exhibited the best aspects of software quality. In short term, it is unlikely that such an “engine” will appear, but CASE vendors have introduced tools that provide a limited subset of these capabilities that addresses specific application domains (e.g., applications that are implemented using a specific database system). More important, these re-engineering tools are becoming increasingly more developed. Forward engineering, also named as renovation or reclamation, which not just recovers design information from existing software, but uses this information to alter or re-constitute the existing system in an effort to improve its overall quality, **Chikofsky and Cross (1990)**. In most cases, re-engineered software re-implements the function of the existing system and also adds new functions and/or improves overall performance.

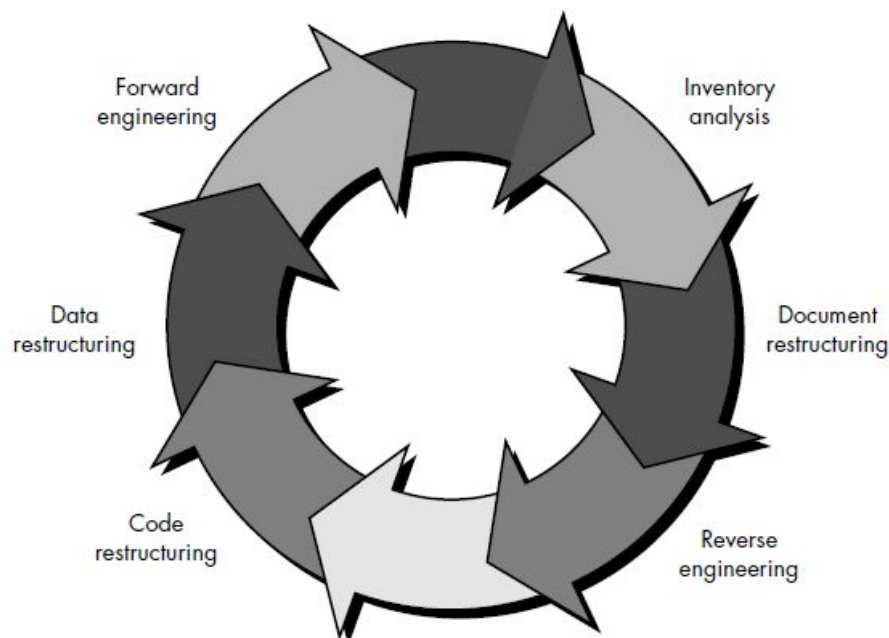


Fig.1.3: A software re-engineering process model.

1.3.3 Why software re-engineering?

The need of software re-engineering comes from the following:

- Increasing numbers of legacy software.
- New emerging technologies.
- Decrease in the ratio of Successful projects.
- Increasing number of competing Companies.
- More demand for quality attributes.
- Changing attitude of people.
- Insistence for Software maintenance.

1.3.4 The benefits of software re-engineering

The benefits of software re-engineering are as follows:

- Increased maintainability, **Fowler (1999)**.
- Improved performance, **Sneed (1998)**.
- Increased interoperability, **Sneed and Nyary (1994)**.
- Decreased personal dependency, **Sneed et al. (2006)**.
- Improved testability, **Sneed et al. (2006)**.

1.3.4.1 Increased maintainability

Many resources are spent on software maintenance. Thus, producing software that is easy to maintain may potentially save large costs. The problem of maintaining software is widely acknowledged in industry, and much has been written on how maintainability can be facilitated by e.g. tools and processes. However, no one can control what he cannot measure, and there is yet no universal measure of maintainability. Some proposals have indeed been presented, but the very idea of measuring maintainability has inherent problems, **Land (2002)**.

Maintainability: The facility with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment, **IEEE Standard Glossary (1990)**.

In practice, when measuring maintainability of a system during a long sequence of changes, if it is true that the system decays continuously, the resulted graph will be similar to the one in Fig.1.4, **Land (2002)**.

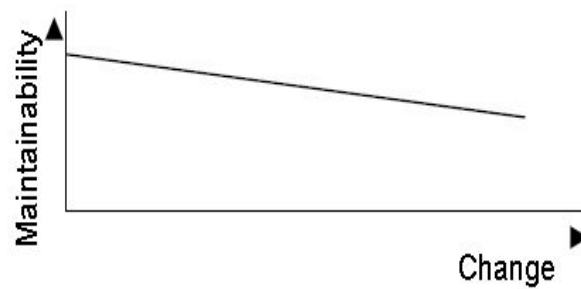


Fig.1.4: The system deteriorates (measured in maintainability) as changes are implemented.

The maintainability sometimes increases and sometimes decreases, that was expected. For example, the logical change “system re-structuring” should cause the maintainability measure to increase, as Fig.1.5 describes, **Land (2002)**.

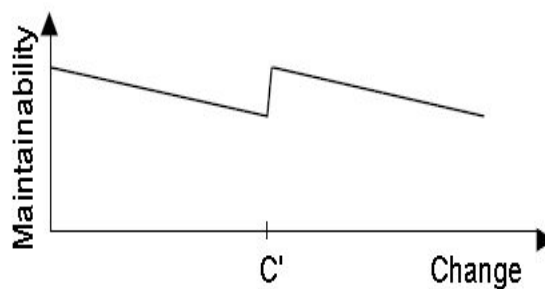


Fig.1.5: Change C' (a re-structuring of the system) causes maintainability to (temporarily) increase.

The software re-engineering has many activities and the re-structuring activity is one of them. So the maintainability will increase when the SRE (Software Re-Engineering) is applied.

1.3.4.2 Improved performance

Performance (responsiveness and scalability) is a make-or-break quality for software. Software industry annually has lost revenue, which reaches to millions of dollars because of poor performance, decreased productivity, increased development and hardware costs, and destroyed customer relations. When performance problems occur, these problems must be fixed immediately. In response, the project often goes into “crisis mode” in an attempt

to tune or even re-design the software to meet performance goals. In these situations, it is vital to maximize the performance and capacity payoff of the tuning efforts, **Williams and Smith (2002)**.

There is a systematic, quantitative approach to performance tuning that helps the developer to quickly detect problems, identify possible solutions, and prioritize the efforts to achieve the greatest enhancement with the least effort. The following steps can be taken:

- Figure out where developer needs to be.
- Determine where developer is now.
- Decide whether developer can achieve his objectives.
- Develop a plan for doing the objectives.
- Doing an economic analysis of the project.

Using this approach one can provide a high payoff for a relatively small cost. Once developer runs into trouble, tuning the software is likely to be his only choice. However, it's necessary to realize that a tuned system will little, if ever, exhibit the level of performance that developer could have achieved by re-engineering that system. The key to achieving optimum performance is to adopt a proactive approach to performance management that anticipates potential performance problems and includes techniques for identifying and responding to those problems early in the process. With a proactive approach, developer produce software that meets performance objectives and is delivered on time and within budget, and avoid the project crisis brought about by the need for tuning at the end of the project. Software performance engineering (SPE) provides a systematic, quantitative approach to manage software performance proactively, **Williams and Smith (2002)**.

Fig.1.6 illustrates how the software re-engineering process increases the performance of the system, **Willis and Brown (2002)**.

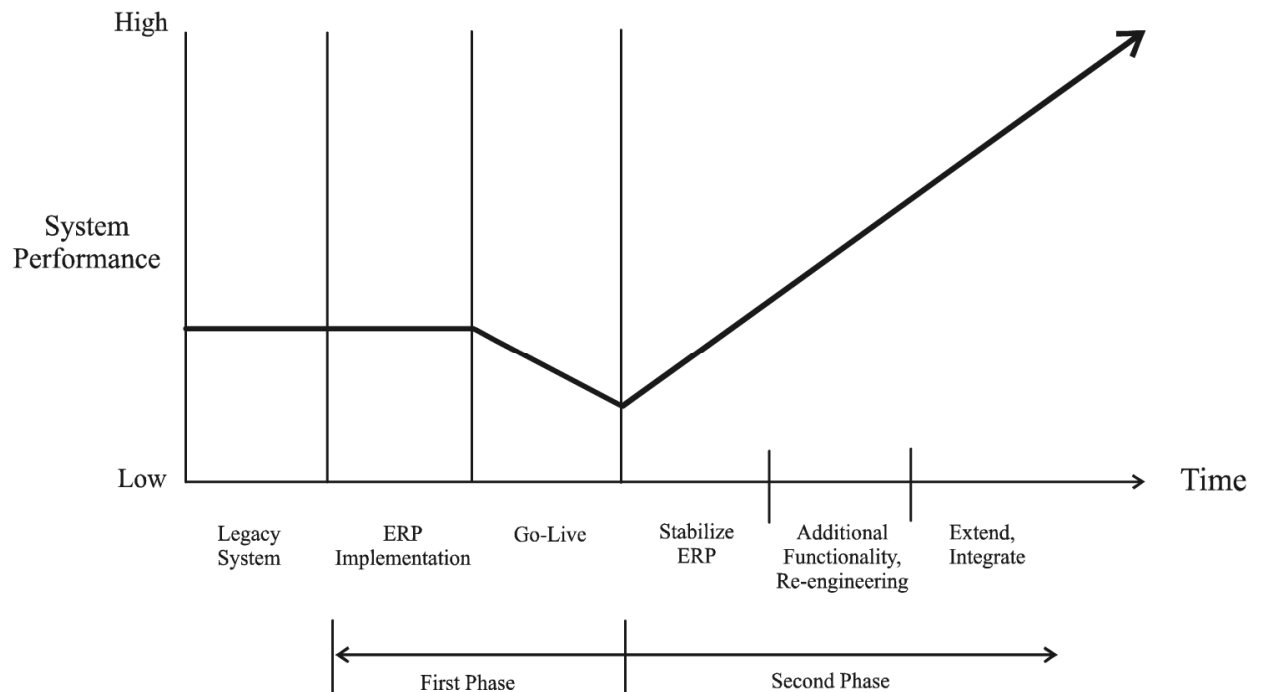


Fig.1.6: The effect of SRP on system performance.

1.3.4.3 Increased interoperability

Interoperability was defined as the capability of two or more systems or components to exchange information and to use the information that has been exchanged. This definition is according to IEEE Standard Glossary of Software Engineering Terminology, **Sanders and Hamilton (2002)**.

The Department of Defense Directive defines interoperability as “the ability of systems, units or forces to provide data, information, materials, and services, to accept the same from other systems, units or forces and to use the data, information, materials, and services, so exchanged to enable them to operate effectively together”. Fig.1.7 refers to the Levels of Information Systems Interoperability (LISI), **Sanders and Hamilton (2002)**.

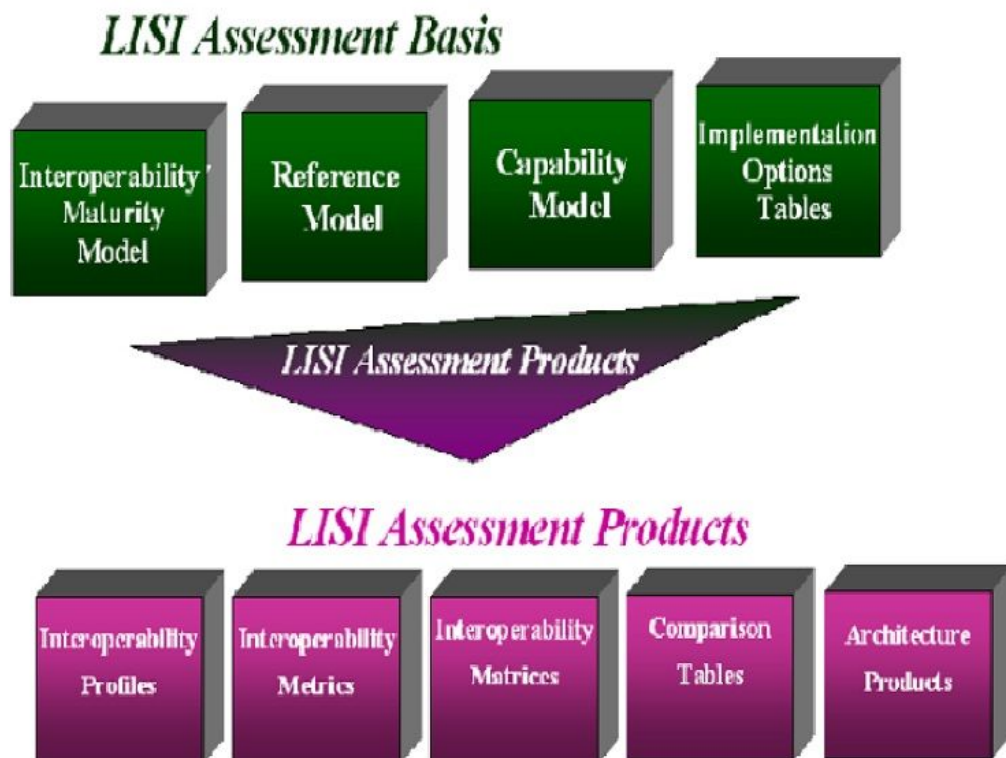


Fig.1.7: A Process for Interoperability.

System interoperability involves the efficiency of producing and refining software to interact with new and existing hardware to reliably exchange information between systems. More specifically, system interoperability encompasses the idea of software process and software modeling involving legacy systems and new hardware/software that must interact to provide reliable exchange of information between systems. Dynamic systems engineering requires that both processes and models be flexible to allow re-engineering and re-evaluation of the processes and models themselves to improve the interoperability, **Sanders and Hamilton (2002)**.

1.3.4.4 Decreased personal dependency

Coordination was required in Large-scale software development, that coordination within and between very large engineering teams which may be located in different buildings, on different company campuses, and in different time zones.

Coordination between software development teams is one of the most difficult-to-improve aspects of software engineering. The software industry has been in crisis mode for

its entire existence, and a root cause is the difficulty in coordinating work between teams of developers. Researchers have studied professional software development teams empirically to gain greater understanding of how software development processes, tools, and people impact coordination. The importance of intra- and inter-team coordination is a foremost concern as software development increasingly becomes globally distributed, and remains a persistent challenge in other disciplines as well.

Coordination is a decision-making that required communication, capacity and cooperation. These three components of coordination are necessary, but coordination to take place requires all these components. Communication is necessary because person A needs to communicate to person B, in some form, and B needs to understand the communication. Capacity is important because B needs to be able to do what is required of him. Cooperation is necessary because B needs to be willing to do what is required of him. If any of the three necessary components are lacking, the outcome will be less than ideal, **Poile et al. (2009)**.

From this, one can say that the personal dependency is important and take a lot of effort to coordinate the software development team work, so that, Software re-engineering is used to decrease personal dependency by re-structuring the legacy system in a manner that reduce the dependency of person A on person B where both A & B are members in the development team, **Sneed et al. (2006)**.

1.3.4.5 Improved testability

Test costs are driven by the size and complexity of the software. Size can be measured in terms of the number of elements making up the system. These elements could be statements, methods, classes, components, interfaces, files, database tables and GUIs. Complexity is measured in terms of the number of interactions between the elements. These could be associations, calls, messages, file transfers, database accesses, import, exports and events from outside. The less there are, the less there is to test, **Sneed et al. (2006)**.

Another factor which influences testability is the visibility of the data interfaces. Data passed between components can be encoded in internal data formats or it can be passed as readable character strings, **Sneed et al. (2006)**.

A very critical factor in testability is the separation of the user interface from the processing logic. This separation of presentation from processing is a prerequisite to testing the processing, i.e. the business logic, without having to enter the data in the user interface, which requires a lot of time and is difficult to automate. A final factor in reducing test effort is the separation of the data access operations from the data processing, **Sneed et al. (2006)**.

The goal of re-engineering for testability is to re-structure the software in such a way that testability criteria can be met while at the same time reducing the size and the complexity of the system. Re-engineering software for testability is definitely a worthwhile effort. Identifying and removing clones, re-factoring deeply nested code and re-structuring the architecture are tasks that can be automated. Several tools exist which support that. By using them, re-engineering costs can be minimized. Other tasks such as algorithm optimization, merging data accesses and simplifying user interfaces can be done manually at a rather low cost. In view of the potential savings in testing costs, it is well worth to invest in a re-engineering project running parallel to the development project. Re-engineering for testability can still be worthwhile before going into system testing. It might also mean that the development process is being complemented by a parallel re-engineering process, intended to raise the quality of the software, including testability. Fig.1.8 refers to parallel projects, **Sneed et al. (2006)**.

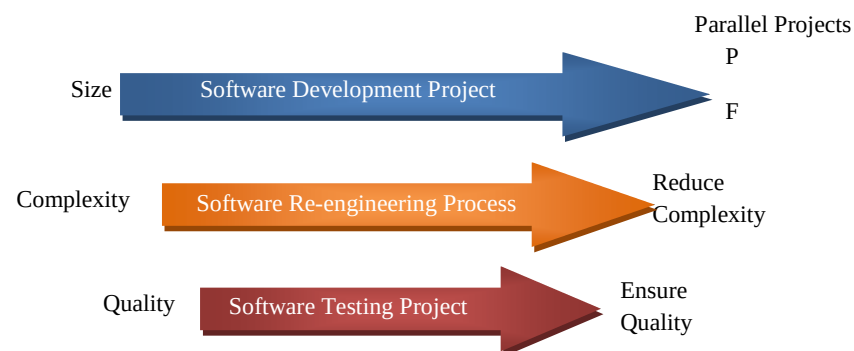


Fig.1.8: Parallel Projects.

Many modern software design methods have been developed to improve the re-usability and maintainability of software and to reduce the time required for the maintenance and development operations, but many companies have old or legacy software systems. These

companies spend a lot of money to maintain their old systems. These systems cannot be replaced by new systems because they contain implicit information and decisions cannot be lost. For these purposes, re-engineering becomes a useful tool to convert old, neglected systems to more functional, streamlined systems.

1.8 Research objectives

- To design a software re-engineering process model.
- To analyze a selected legacy software.
- To implement the proposed software re-engineering process in the selected legacy system.
- To compare the proposed software re-engineering process with other legacy software re-engineering processes.

1.9 Organization of the thesis

The thesis contains five chapters and contents of these chapters are briefly outlined as follows:

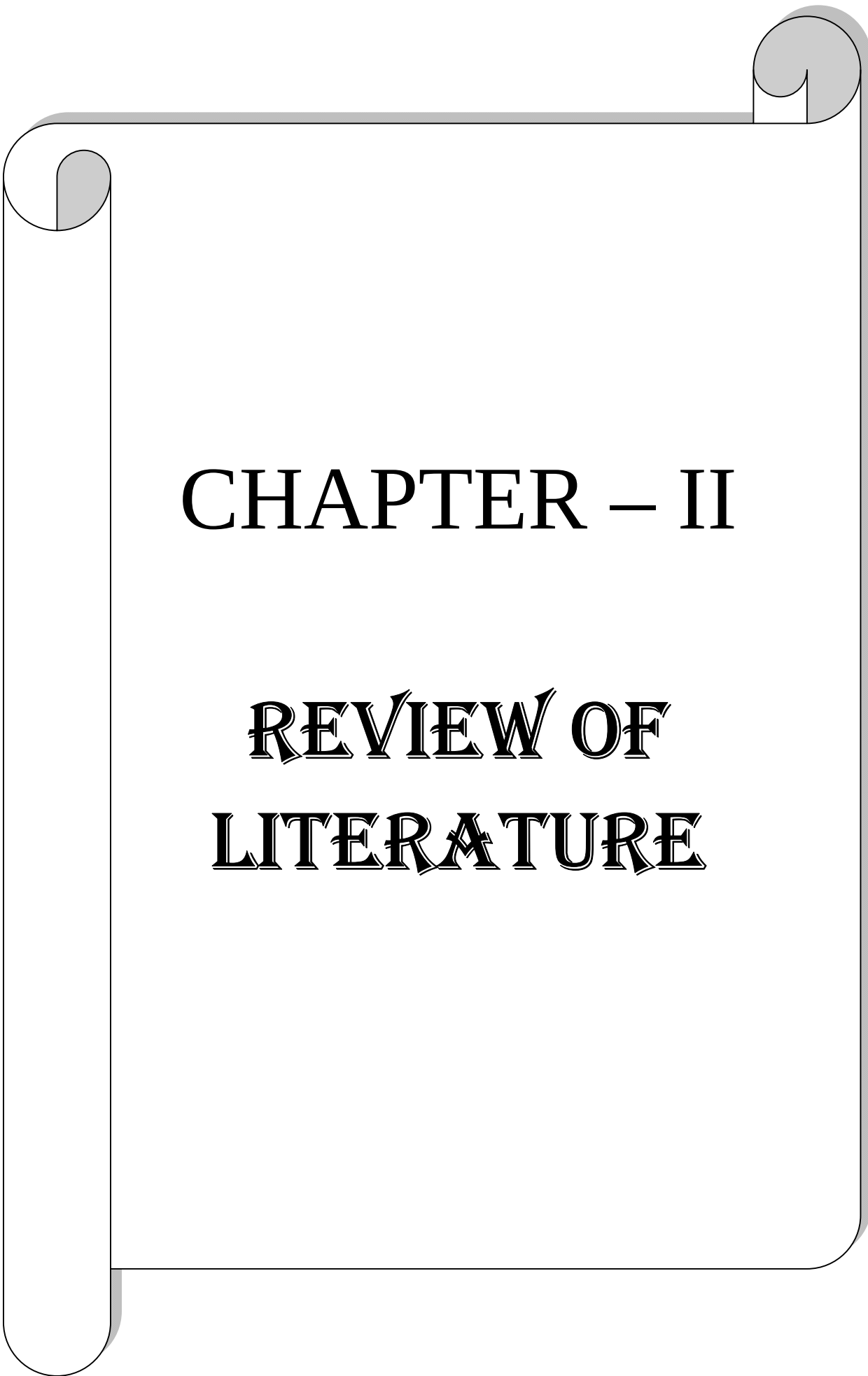
Chapter 1 is a general introduction of software re-engineering and presents the organization of the thesis.

Chapter 2 deals with the literature survey in the related area. Literature survey pertaining to software re-engineering process models has been presented. Various types of process models have been discussed in this chapter.

Chapter 3 presents the suggested software re-engineering process that named as FUSION model, and then discusses all the activities of this model in details.

Chapter 4 presents how the FUSION model can be implemented on the legacy supermarket management system to re-engineer it, then the result of comparison between the legacy software and the re-engineered software will be presented. The diagnostic features of the FUSION model are also presented in this chapter.

Chapter 5 summarizes the results obtained in the thesis. Scope for future work has also been suggested in this chapter.



CHAPTER – II

REVIEW OF LITERATURE

CHAPTER 2

REVIEW OF LITERATURE

2.1 Introduction

In this chapter, a literature review and classification scheme for selected software engineering researches is presented. The study shows that an increasing number of software engineering researches have been conducted in diverse scope of areas. The articles are classified and results of these are presented based on classification scheme that consist of five main categories: software development process, software engineering technique, software management, software re-engineering, and software application. Analyses of the chosen researches are carried out and gaps in the research are identified. An inclusive list of references is presented. This review is intended to increase motivation in research and help to determine the area of the work in this thesis.

More than 250 papers, which were published between the time period 2008-2012 in conferences and international journals like Springer, IEEE, IJSEA, IJCSI, IJCSE, IJCA, IJEST, famoosR, COMATS, and Elsevier were collected, analyzed and classified into a number of categories and subcategories. The study guided to the identification of gap in software engineering researches and enabled the author to recommend the area where there is a lot of scope for future research.

- **Software engineering**

Software engineering can be defined as the application of a systematic, disciplined, quantifiable process to the software development, software operation, and software maintenance, and the study of these processes; which is, the application of engineering to software, **Bourque and Dupuis (2004)**. Software engineering is the application of engineering to software because it integrates significant mathematics, computer science and practices whose sources are in engineering. Software engineering is also defined as a systematic approach to analysis, design, assessment, implementation, testing, maintenance and engineering of software, **Phillip (2007)**. The term software engineering first appeared in the 1968 NATO Software Engineering Conference, and was meant to excite thought regarding the perceived software crisis at that time, **Naur and Randell (1968), Randell (2001)**.

2.2 Classification of software engineering research

The classification framework shown in Fig.2.1 is based on the literature review and the nature of software engineering researches, which meant to give an understanding of how the subject has evolved and progressing. Since software engineering is an important area it is very essential to be clear of its definition. There are several researchers working in this area and trying to define and specify the universally accepted list of software engineering principles and concepts and trying to apply all these principles and concepts on a real project but until now they could not do that completely and perfectly. The software engineering researches can be classified into five categories, which are; (I) software development process which account to 35 % (90 papers), (II) software management which counts to 30 % (75 papers), (III) software re-engineering which shares (36 papers) of the distribution 14 %, (IV) software engineering techniques which shares (34 papers) of the distribution 13 %, and (V) software applications which accounts to 8 % (19 papers). The proportion of research carried out in each category is shown in Fig.2.2.

These main categories are further divided into a number of sub categories as in Fig.2.1. The percentage distribution of research in the main categories is shown in Fig.2.2 and according to the year of publishing, these papers can be grouped as shown in Fig.2.3. The papers are further grouped based on the journal in which they are published as shown in Fig.2.4, and the number of published papers in each category shown in Table 2.1, number of published papers in each year shown in Table 2.2 and the number of papers published in each journal is shown in Table 2.3. All the categories and their sub-categories will be discussed in detail in the later sections with graphs and tables.

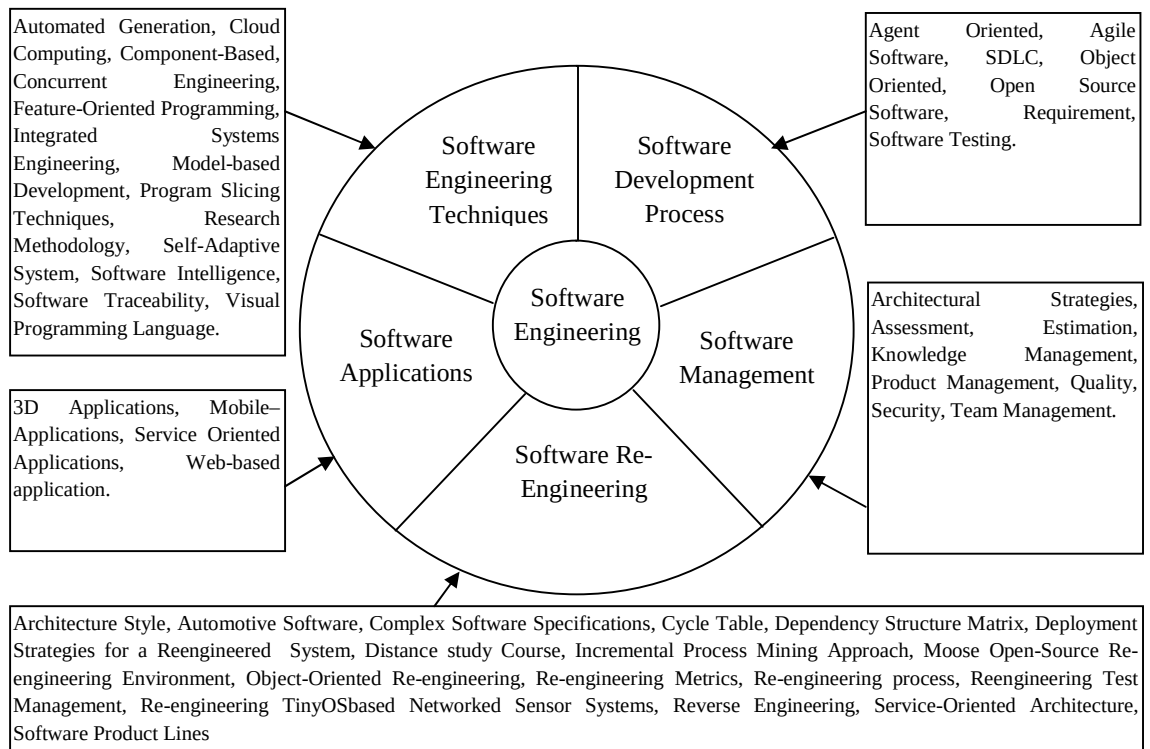


Fig. 2.1: Classification of research papers selected for literature review

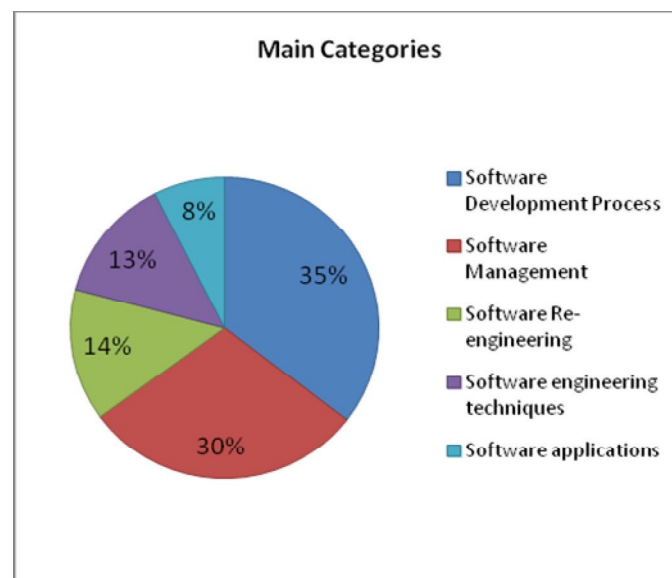


Fig. 2.2: Distribution of research paper with respect to categories

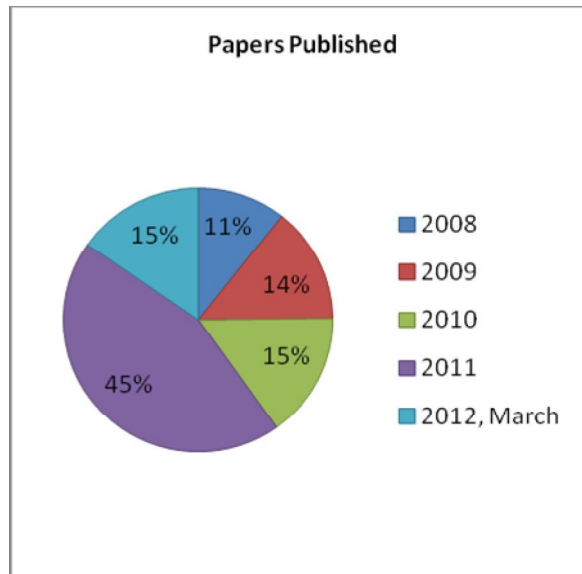


Fig. 2.3: Papers analyzed and year of publication

Table 2.1: Number of published papers in each category

Main Categories	No. of papers
Software Development Process	90
Software Management	75
Software Re-engineering	36
Software engineering techniques	34
Software applications	19
Total	254

Table 2.2: Number of published papers in each year

Year	No. of published papers
2008	27
2009	36
2010	39
2011	113
2012	39
Total	254

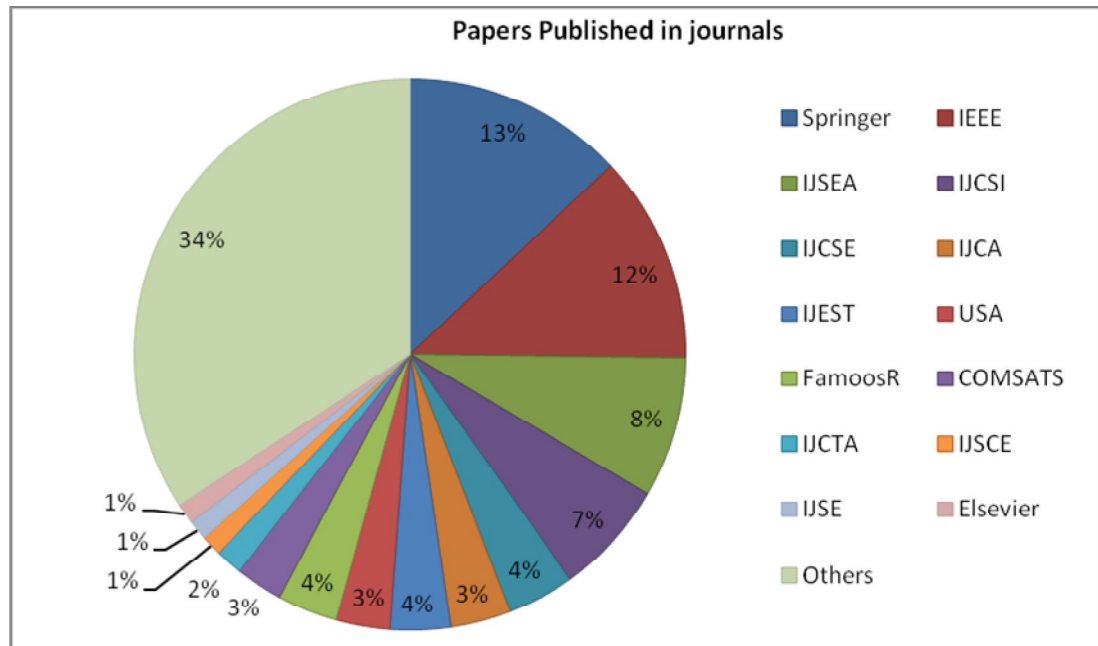


Fig. 2.4: Papers published in various journals

Table 2.3: Number of papers published in various Journals

Journals / Institute	No. of published papers	Journals / Institute	No. of published papers
Springer	33	FamoosR	9
IEEE	31	COMSATS	7
IJSEA	21	IJCTA	4
IJCSI	17	IJSCE	3
IJCSE	10	IJSE	3
IJCA	9	Elsevier	3
IJEST	9	Others	87
USA	8	Total	254

2.3 Software development process

A software development process, also known as a software development life cycle (SDLC) is a structure imposed on the development of a software product. This includes software life cycle and software process. Software development process is often considered as a subset of systems development life cycle. There are many models for such processes, all of them

describing approaches to a variety of tasks or activities that take place during the process. A life-cycle model is considered as a more general term and a software development process is considered as a more specific term. For example, there are several specific software development processes that 'fit' the spiral model. ISO/IEC 12207 is an international standard for software life-cycle processes. ISO/IEC 12207 aims to the standard that defines all the tasks required for developing and maintaining software, **Wiegers and Karl (2003)**.

There are several activities involved in software development called as “Software development activities”; these activities include planning (requirements analysis), design, implementation, testing and documenting, deployment and maintenance.

In this research area there are 90 papers, which are analyzed and classified into number of sub-categories, which are described briefly as follows:

- a. SDLC: It is a software development life cycle and there are many models like Waterfall model, V model, spiral model, incremental model, and evolutionary model, **Wiegers and Karl (2003)**. In this subcategory there are 32 papers and their percentage is 34%.
- b. Requirement engineering: Requirement is a singular documented physical and functional need that a particular product or service must be or perform. Requirement is most commonly used in a formal sense in systems engineering, enterprise engineering, or software engineering. Also, it is a statement that identifies a required attribute, capability, characteristic, or quality of a system for it to have value and utility to a user, **Wiegers and Karl (2003)**. In this subcategory there are 15 papers and their percentage is 16%.
- c. Software testing: Software testing can be stated as the process of validating and verifying that a software program/application/product meets the requirements that guided its design and development; works as expected; and can be implemented with the same characteristics, **Pan (1999)**. In this subcategory there are 15 papers and their percentage is 16%.
- d. Agile Software development process: Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions generated through collaboration between self-organizing, cross-functional teams. It supports adaptive planning, growth development and delivery, a time-boxed iterative approach, and supports rapid and flexible response to change. Also, it is a conceptual framework that promotes foreseen interactions throughout the development

cycle, **Beck et al. (2001)**. In this subcategory there are 12 papers and their percentage is 13%.

- e. Open Source Software development process: Open-source software (OSS) is a computer software that is available in source code form: the source code and certain other rights normally reserved for copyright holders are provided under an open-source license that permits users to study, modify, improve and distribute the software. OSS is often developed in a public, collaborative manner. OSS is the most prominent example of open-source development and often compared to (technically defined) user-generated content or (legally defined) open content movements, **Verts and William (2008)**. In this subcategory there are 7 papers and their percentage is 8%.
- f. Object oriented software development process: It is a programming paradigm using "objects" – data structures consisting of data fields and methods together with their interactions – to design applications and computer programs. Programming techniques may have features such as data abstraction, messaging, encapsulation, modularity, polymorphism and inheritance. Many new programming languages now support OOP, at least as an option, **McCarthy et al. (1960)**. In this subcategory there are 6 papers and their percentage is 6%.
- g. Agent oriented software development process: It is a programming paradigm where the construction of the software is centered on the concept of software agents. It has externally-specified agents (with interfaces and messaging capabilities) at its core. They can be as abstractions of objects. Exchanged messages are translated by receiving "agents", in a way specific to its class of agents, **Shoham (1990)**. In this subcategory there are 3 papers and their percentage is 3%.

All these sub-categories are shown in Fig. 2.5 and the numbers of papers in each sub-category is shown in Table 2.4. The number of published paper in this category with respect to the year of publishing is shown in Fig. 2.6.

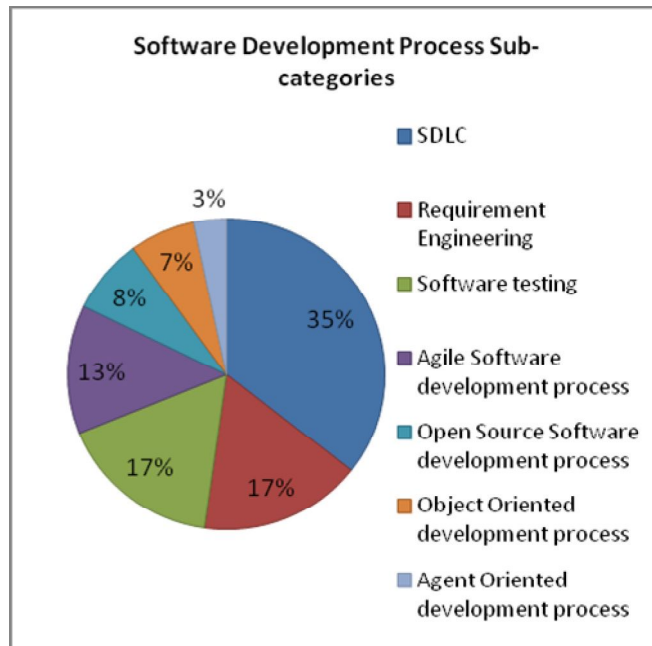


Fig. 2.5: software development process sub-categories

Table 2.4: Number of papers in each sub-category of Software Development Process

Software Development Process Sub-categories	No. of paper in each sub-category
SDLC	32
Requirement Engineering	15
Software testing	15
Agile Software Engineering	12
Open Source Software Engineering	7
Object Oriented Engineering	6
Agent Oriented Engineering	3

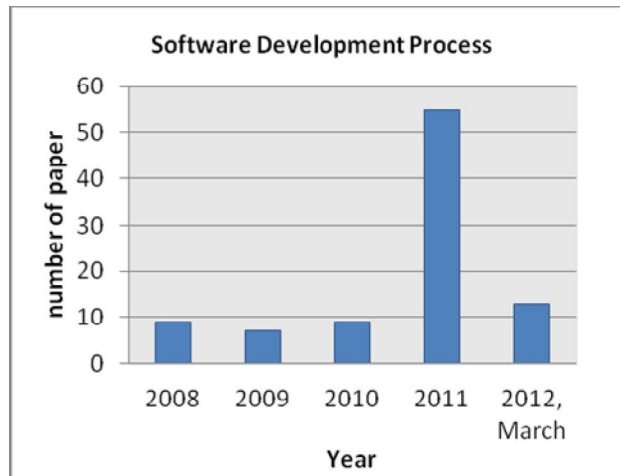


Fig. 2.6: Number of papers published in the software development process based on the year of publishing.

2.4 Software management:

Software project management begins with a set of activities that are collectively called project planning. Before the project can start, the manager and the software team must estimate the work to be accomplished, the resources that will be required, and the time that will be consumed from start to finish. Whenever estimates are made, the team looks into the future and accepts some degree of uncertainty as a matter of course, **Pressman (2005)**.

In this research area there are 75 papers, which are classified into number of sub-categories, brief description of each one is given below:

- a. **Assessment:** Several methods have been created to define an assessment process for software. Some focus on some aspects like the maturity, the strength and the strategy of the organization around the project itself. Other methods add functional aspects to the assessment process, **Pressman (2005)**. In this subcategory there are 18 papers and their percentage is 24%.
- b. **Quality:** In the context of software engineering, software quality indicates to two related but distinct concepts that exist wherever quality is defined in a business context. Software functional quality reflects how well it complies with or conforms to a given design, depending on the functional requirements or specifications. That attribute can be described as the fitness for purpose of a piece of software or how it compares to competitors in the

marketplace as a worthwhile product; Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, as robustness or maintainability, the degree to which the software was created correctly, **Pressman (2005)**. In this subcategory there are 16 papers and their percentage is 21%.

- c. Architectural Strategies: As organizations enhance their architecture competency, architects will come to play a more active role in the strategy setting process, in addition to the critical role they already play in business strategy implementation. However, this strategy is not the part of the average computer science program, nor is it a competency that companies necessarily foster in the technical community. The goal of Architectural Strategies is to provide the developer with a strategy for the intelligent introduction to state-of-the-art business strategy thinking and models. The aim is to help in developing the language, tools, and sensibilities that will enable the developer to quickly position himself to make a useful contribution to the strategy process. There more attention is given to competency because it is emerging as a critical avenue for competitive variance. The trend has been for technology to become more essential to basic business capabilities as well as to differentiate value propositions that distinguish the business in the marketplace. At the same time, technology is complicated and fast moving enough that it requires its own kind of attention, and it falls to the architects to fetch this insight to the strategy table. With this base in the formulation of the business strategy, architects can also be a more efficient bridge between business strategy and the technical strategy that implements it, **Bredemeyer et al. (2003)**. In this subcategory there are 12 papers and their percentage is 16%.
- d. Estimation: Estimation is responsive to the widespread problems the software industry has experienced in creating meaningful cost and schedule estimates. Software estimation is not as difficult or mysterious as many people think, but the knowledge of how to create efficient estimates has not been well publicized. Software Estimation provides an inclusive set of tips and heuristics that software developers, project managers, and technical leads can apply to create more accurate estimates. Software Estimation is fundamental estimation techniques and addresses specific estimation challenges. It explains how to avoid common mistakes. Software estimation does not avoid hairy mathematical methods, but the non-mathematical reader will find plenty of useful guidelines without getting

bogged down in complex formulas, **Steve (2006)**. In this subcategory there are 11 papers and their percentage is 15%.

- e. Team Management: It refers to techniques, processes and tools for organizing and coordinating a group of individuals working towards a common goal. The 'Team Development Model', identified by Bruce Tuckman, offers a definition of the stages teams go through during their lifecycle. The activities of team management are not new, and many of the tools used by team managers. Organizational Development-oriented practitioners often use interview-based analysis and provide reports and insights that team leaders and their management may use to adapt team practices for higher performance. Teams can be developed by team building activities - which can also be used simply to build relationships where team members lack cohesion due to physical distance or organizational structure. Project managers may approach team management with a focus on structure, standardized practices and communications, **Pressman (2005)**. In this subcategory there are 9 papers and their percentage is 12%.
- f. Security: It is the degree of protection against damage, danger, loss, and crime. Security as a form of protection is structures and processes that provide or improve security as a condition. Security must be compared to related concepts: safety, reliability, continuity. The key difference between security and reliability is that security must take into account the actions of people attempting to cause destruction, **Schneier and Fear (2003)**. In this subcategory there are 7 papers and their percentage is 9%.
- g. Knowledge Management: It comprises a range of strategies and practices used in an organization to identify, create, represent, distribute, and activate adoption of acumen and experiences. Such insights and experiences formalize knowledge, either embedded in organizations or embodied in individuals as practices or processes. Knowledge management efforts typically focus on objectives of organization such as enhanced performance, innovation, competitive feature, the sharing of the learned lessons, integration and continuous enhancement of the organization. Knowledge management efforts interfere with organizational learning, and may be recognized from that by a greater focus on the management of knowledge as a strategic asset and a focus on encouraging the sharing of knowledge, **Rachael et al. (2006)**. In this subcategory there is only 1 paper and the percentage is 1%.

- h. **Product Management:** It is an organizational lifecycle function within a company dealing with the planning or marketing of a product or products at all stages of the product lifecycle. Product management (inbound focused) and product marketing (outbound focused) are different yet complementary efforts with the objective of maximizing sales incomes, market share, and profit margins. The main role of product management spans many activities from strategic to tactical and varies based on the organizational structure of the company, **Misra (2009)**. In this subcategory there is one paper and the percentage is 1%.

All these sub-categories are shown in Fig. 2.7 and the number of papers published in each sub-category is shown in Table 2.5. The number of papers published in this category with respect to the year of publishing is shown in Fig. 2.8.

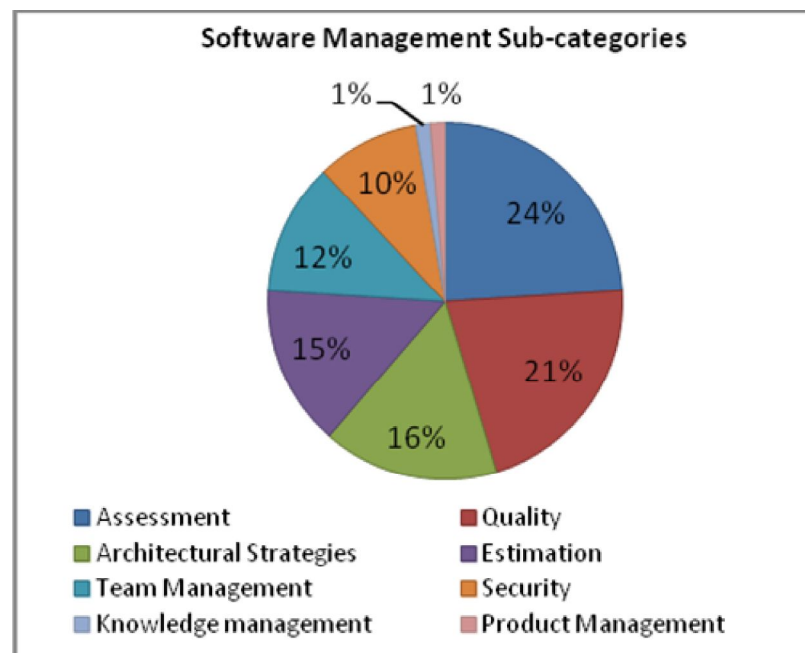


Fig. 2.7: Software management sub-categories

Table 2.5: Number of papers in each sub-category of software management

Software Management Sub-categories	No. of paper in each sub-category
Assessment	18
Quality	16
Architectural Strategies	12
Estimation	11
Team Management	9
Security	7
Knowledge management	1
Product Management	1

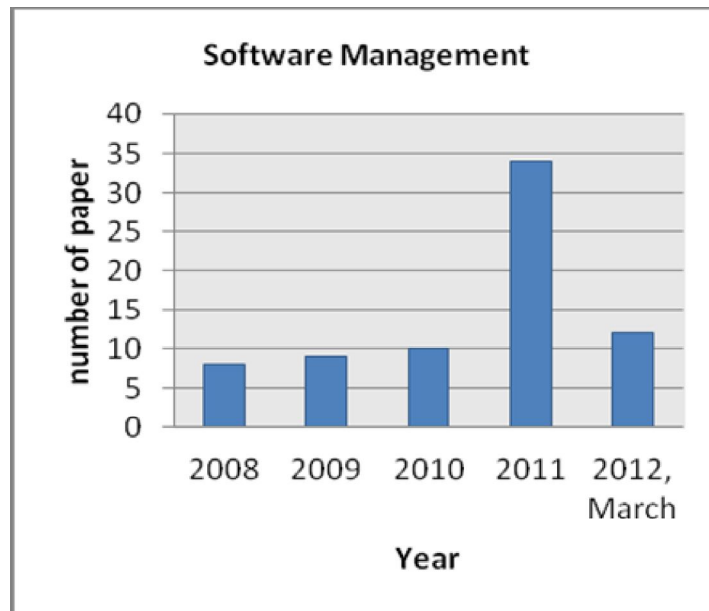


Fig. 2.8: Number of papers published in the software management based on the year of publishing

2.5 Software re-engineering:

The re-engineering of software was described by Chikofsky and Cross in their 1990 paper **Chikofsky and Cross (1990)** as the examination and alteration of a system to reconstitute it in a new form. Also, re-engineering is the modification of a software system that takes place after the system has been reverse engineered, generally to add more functionality, or to correct faults. The entire process is often erroneously referred to as reverse engineering, however, it is

more precise to say that reverse engineering is the initial examination of the system, and re-engineering is the subsequent modification. Re-engineering is foremost used in the context where a legacy system is involved, **Gahalaut and Khandnor (2010)**. Software systems are evolving on high rate because there more research to make them better; therefore legacy software needs to operate on a new computing platform through re-engineering. The re-engineering is a set of activities that are carried out to re-structure a legacy system to a new system with better functionalities and conform to the hardware and software quality constraint.

In this research area there are 36 papers. These papers are classified into number of sub-categories, and they are the following:

- a. Reverse Engineering: it is the process of discovering the technological principles of a device, system, or object through analysis of its structure, operation, and function. It often involves taking something (e.g., a mechanical device, electronic component, software program, biological, chemical, organic matter) apart and analyzing its workings in detail to be used in maintenance, or to try to make a new program or device that does the same thing without using or simply duplicating (without understanding) the original, **Eldad and Elliot (2007)**. In this sub-category there are 12 papers and their percentage is 33%.
- b. Object-Oriented Re-engineering: Object oriented development has proved its worth in today's system because its design and development is preferable, reliable and easier to access than the traditional methodologies. Because of updated requirements and lack of documentation in old systems has provided a motivation to revamp the systems. Re-building or re-designing the same system is highly expensive. To solve this problem reverse engineering of the system is used as most suitable alternative, **Carchiolo et al. (2010)**. In this subcategory there are 3 papers and their percentage is 8%.
- c. Service-Oriented Architecture: In software engineering, a Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are good-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes. SOA design principles are used through the phases of systems development and integration, **Bell and Michael (2008)**. In this subcategory there are 3 papers and their percentage is 8%.

- d. Automotive Software: It is different from the kind of software that is usually addressed by current re-engineering research. These papers give an overview of the particular conditions and challenges in the automotive domain, **Coerne et al. (2010)**. In this subcategory there are 2 papers and their percentage is 6%.
- e. Cycle Table: Identifying and understanding cycles in large applications is a challenging task. In these papers CycleTable which displays both direct and indirect references was presented. It shows how minimal cycles are intertwined through shared edges and allows the re-engineer to devise simple strategies to remove them, **Laval et al. (2009)**. In this subcategory there are 2 papers and their percentage is 6%.
- f. Re-engineering Metrics: A metric is nothing more than a standard measure to assess your performance in a particular area. Metrics are at the core of a good, customer focused process management system and any program directed at continuous improvement. The focus on customers and performance standards show up in the form of metrics that assess the ability to meet customers' needs and business objectives, **Wesner et al. (1995)**. In this subcategory there are 2 papers and their percentage is 6%.
- g. Re-engineering process: It includes all the activities from the legacy software analyzing to the generation of new software. In this subcategory there are 2 papers and their percentage is 6%.
- b. There are other researches in the software re-engineering field, about one paper in each sub-categories, these sub-categories are: Complex Software Specifications, Dependency Structure Matrix, Deployment Strategies for a Re-engineered System, Distance study Course, Incremental Process Mining Approach, Moose Open-Source Re-engineering Environment, Re-engineering Test Management, Re-engineering TinyOSbased Networked Sensor Systems, Software Product Lines, Architecture Style.

All these sub-categories are shown in Fig. 2.9 and the number of papers in each sub-category is shown in Table 2.6. The number of papers published in this category with respect to the year of publishing is shown in Fig. 2.10.

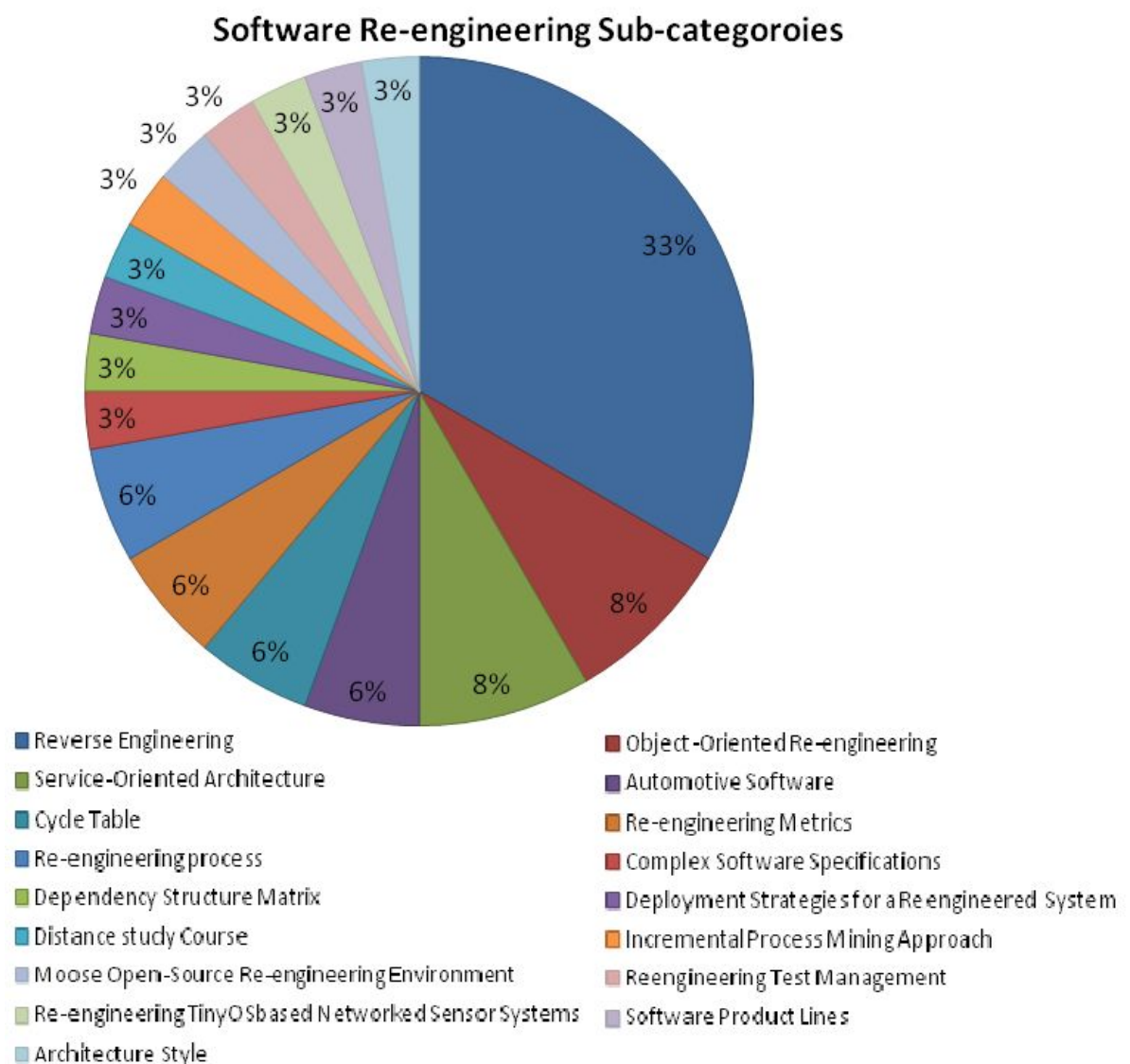


Fig. 2.9: Software Re-engineering sub-categories

Table 2.6: Number of papers in each sub-category of Software Re-engineering

Software Re-engineering Sub-categories	No. of paper in each sub-category
Reverse Engineering	12
Object-Oriented Re-engineering	3
Service-Oriented Architecture	3
Automotive Software	2
Cycle Table	2
Re-engineering Metrics	2
Re-engineering process	2
Complex Software Specifications	1
Dependency Structure Matrix	1
Deployment Strategies for a Reengineered System	1
Distance study Course	1
Incremental Process Mining Approach	1
Moose Open-Source Re-engineering Environment	1
Reengineering Test Management	1
Re-engineering TinyOSbased Networked Sensor Systems	1
Software Product Lines	1
Architecture Style	1

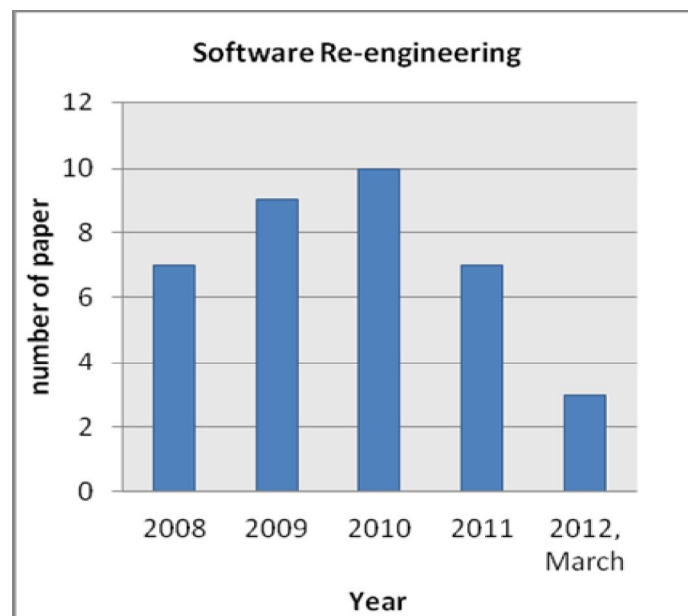


Fig. 2.10: Papers published in the software Re-engineering.

2.6 Software engineering techniques:

Today, software is becoming ever more complicated, developed with a very short time to market, and required to dealing with changing requirements. This scenario calls for high levels of flexibility and agility, both in the used technologies and the processes followed for engineering software. So many techniques were provided that can meet these challenges and these techniques are known as Software engineering techniques.

In this research area there are 33 papers, which are classified into number of sub-categories. They are the following:

- a. **Component-Based:** Component-based software engineering (CBSE) (also known as component-based development (CBD)) is a branch of software engineering that emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. CBSE is a reuse-based method to defining, implementing and composing loosely coupled independent components into systems. This practice aims to reach an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software, **Bruce (2010)**. In this sub-category there are 10 papers and their percentage is 29%.
- b. **Automated Generation:** It is a style of computer programming that uses automated source code creation through general frames, classes, prototypes, aspects, templates, and code generators to improve programmer productivity. Automated Generation is often related to code-reuse topics such as component-based software engineering and product family engineering, **James Wilcox (2011)**. In this sub-category there are 7 papers and their percentage is 20%.
- c. **Concurrent Engineering:** It is a work methodology based on the parallelization of tasks (i.e. performing tasks concurrently). Concurrent Engineering refers to an approach used in product development in which number of functions are integrated to reduce the elapsed time required to bring a new product to the market, these functions as manufacturing engineering, design engineering and others, **Chen and Thimm (2008)**. In this sub-category there are 3 papers and their percentage is 9%.

- d. **Research Methodology:** It is a way to systematically solve the research problem. It may be understood as a science of studying how research is done scientifically. There are various steps that are generally adopted by a researcher in studying the research problem along with the logic behind them. It is necessary for the researcher to know the research methods/techniques and the methodology. Researchers need to know how to develop certain indices or tests, how to compute the mean, the mode, the standard deviation or the median or chi-square, how to implement particular research techniques, also they need to know which of these methods or techniques are relevant and which are not, and what would they mean and the indication and why. Researchers also need to recognize the assumptions underlying various techniques and they need to know the criteria by which they can decide certain techniques and procedures will be applicable to certain problems. All these techniques and procedures which are requisite for the researcher to design his methodology for his problem as the same may differ from problem to problem, **Kothari (2004)**. In this sub-category there are 3 papers and their percentage is 9%.
- e. **Cloud Computing:** It is a metaphor used by Technology or IT Services companies for the delivery of computing requirements as a service to a heterogeneous community of end-recipients. The term cloud theoretically means abstraction of technology, resources and the location of technology, that are very vital in building integrated computing infrastructure (as networks, systems and applications). All Cloud computing models rely heavily on sharing of resources to achieve coherence and economies of scale similar to a utility (like the electricity grid) over a network (typically the Internet), **NIST (2011)**. In this sub-category there are 2 papers and their percentage is 6%.
- f. **Self-Adaptive System:** The self-adaptability of systems has been studied in a wide range of disciplines, from biology to robotics, just recently has the software engineering community recognized its key role in enabling the development of future software systems that are able to self-adapt to changes that may occur in the system, the requirements of that system, or the environment in which it is deployed, **Cheng et al. (2009)**.
- g. There are other researches in the field of software engineering techniques, about one paper in each sub-categories, these sub-categories are: Feature-Oriented Programming, Integrated Systems Engineering, Model-Based Development, Program Slicing Techniques, Software Intelligence, Software Trace-ability, and Visual Programming Language

All these sub-categories are shown in Fig. 2.11 and the number of papers in each sub-category is shown in Table 2.7. The number of papers published in this category with respect to the year of publishing is shown in Fig. 2.12.

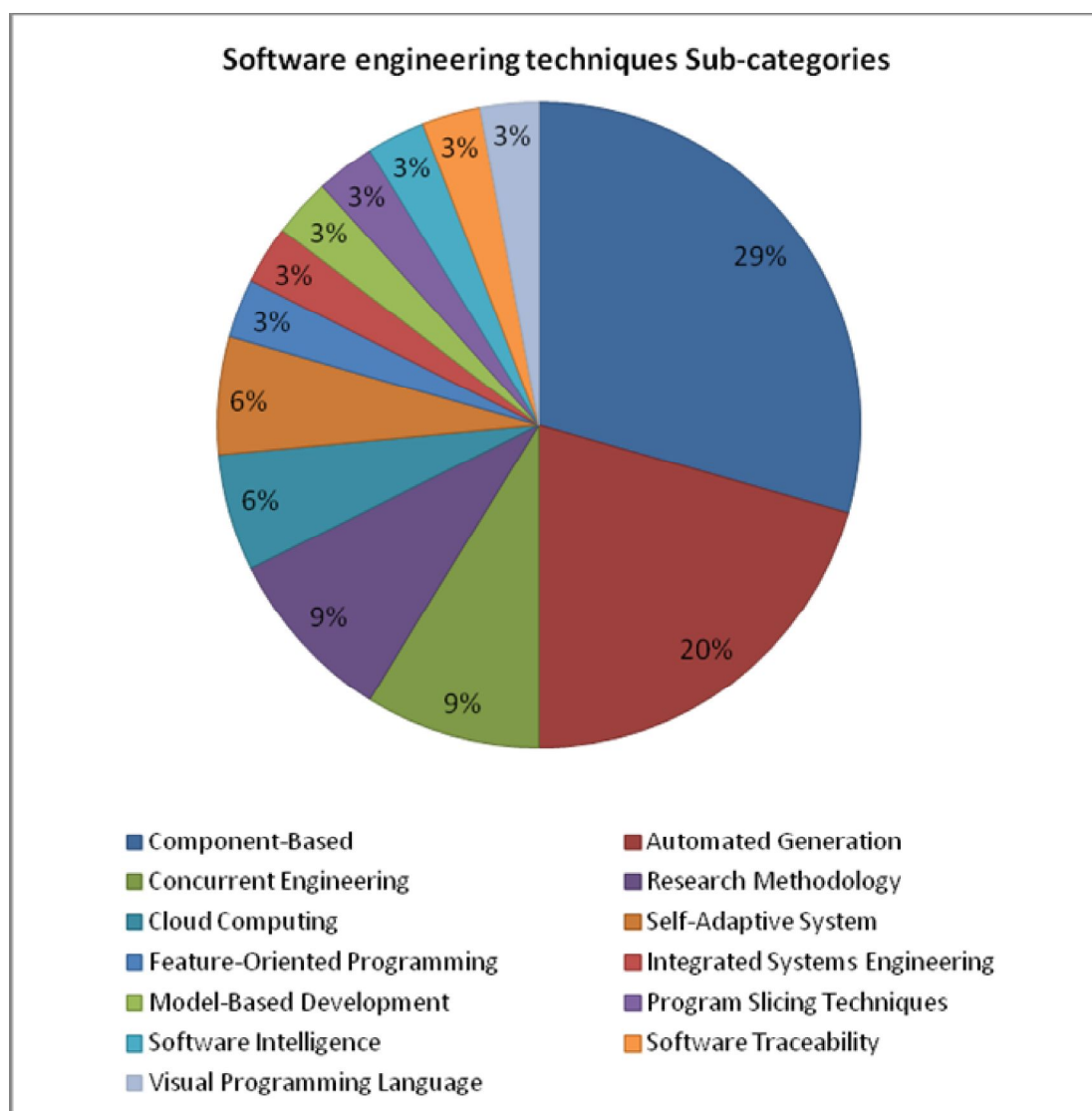


Fig. 2.11: Software engineering techniques sub-categories

Table 2.7: Number of papers published in each sub-category of Software engineering techniques

Software engineering techniques Sub-categories	No. of paper in each sub-category
Component-Based	10
Automated Generation	7
Concurrent Engineering	3
Research Methodology	3
Cloud Computing	2
Self-Adaptive System	2
Feature-Oriented Programming	1
Integrated Systems Engineering	1
Model-Based Development	1
Program Slicing Techniques	1
Software Intelligence	1
Software Traceability	1
Visual Programming Language	1

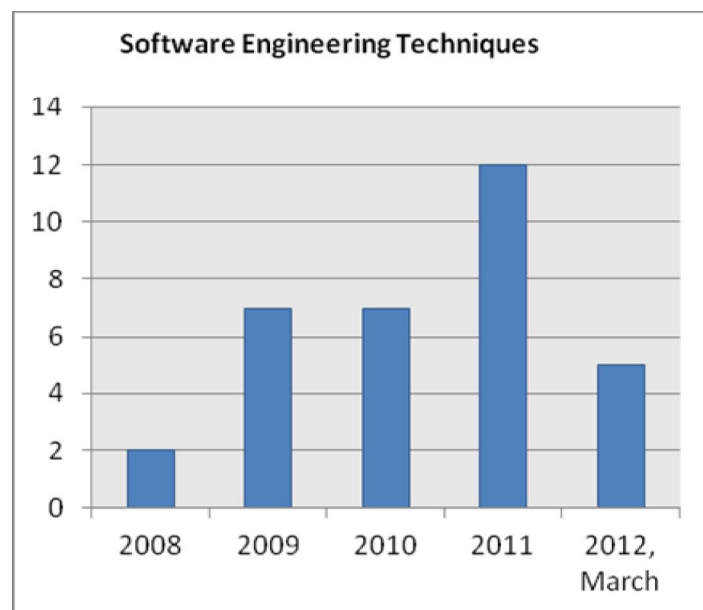


Fig. 2.12: Papers published in the software engineering techniques.

2.7 Software applications:

Also known as an application or an "app" it is a computer software designed to help the user to perform particular tasks. Examples include enterprise, accounting, graphics, and office suites software and media players. Many application programs deal basically with documents. Apps may be packed with the computer and its system software, or can be published separately. Some users are satisfied with the packed apps and need never install

one. Application software is contrary with system software and middleware, which integrate and manage a computer's capabilities, but usually do not directly apply in the performance of tasks that benefit the user. The system software works as services provider for the application, which in turn serves the user, **Ceruzzi and Paul (2000)**.

In this research area there are 19 papers; these papers are classified into number of sub-categories, as the following:

- a. Web-based application: A web application is an application that is accessed over a network such as the Internet or an intranet. The web application term may also represent a computer software application that is coded in a browser-supported language (such as JavaScript, integrated with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable. Web applications are popular because of the ubiquity of web browsers, and the suitability of using a web browser as a client, sometimes named as a thin client. The capability to maintain and update web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, like the inherent support for cross-platform compatibility. Popular web applications include webmail, online retail, , online sales and public sale, and many other functions, **Pressman (2005)**. In this sub-category there are 12 papers and their percentage is 63%.
- b. Mobile – Application: It is an application software that developed for small low-power handheld devices such as mobile phones, and digital assistants for personal or enterprise purposes. These applications can be pre-installed on phones during manufacture, downloaded by customers from varied mobile software distribution platforms, or web applications delivered via HTTP which use server-side or client-side processing (e.g. JavaScript) to provide an "application-like" experience within a Web browser, **Wasserman (2010)**. In this sub-category there are 3 papers and their percentage is 16%.
- c. Service Oriented Application: A service-oriented architecture is essentially a collection of services. These services connect with each other. The connection can involve either simple data passing or it could involve two or more services coordinating some activity. Some means to connect services to each other is needed. The architectures of SOA are not a new thing. The first architecture for many people in the past was with

the use Distributed Component Object Model (DCOM) or Object Request Brokers (ORBs) based on the Common Object Request Broker Architecture (CORBA) specification, **Thomas Erl (2005)**. In this sub-category there are 3 papers and their percentage is 16%.

- d. 3D Application: It refers to programs used to create 3D computer-generated imagery. There are many software like 3D Studio MAX, 3D modeling application, Animation Master, and etc. **Mathworks (2007)**. In this sub-category there is only 1 papers and the percentage is 5%.

These sub-categories are shown in Fig. 2.13 and the numbers of papers in each sub-category is shown in Table 2.8. The number of papers published in this category with respect to the year of publishing is shown in Fig. 2.14.

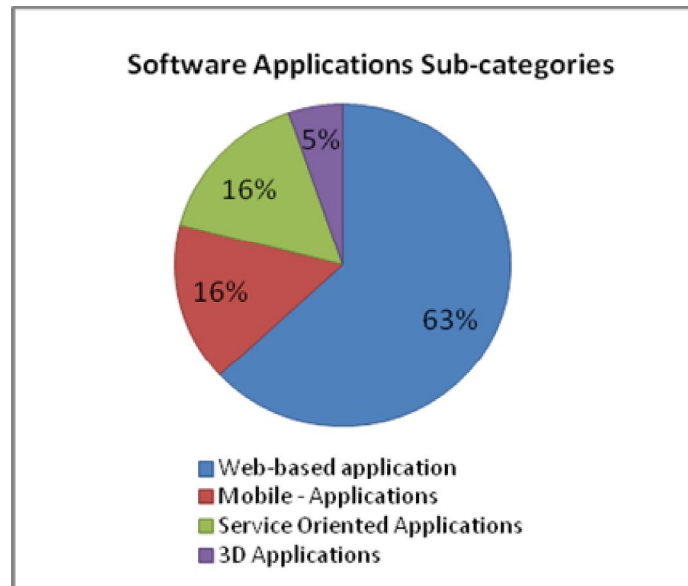


Fig. 2.13: Software engineering techniques sub-categories

Table 2.8: Number of papers published in each sub-category of Software application

Software applications Sub-categories	No. of paper in each sub-category
Web-based application	12
Mobile - Applications	3
Service Oriented Applications	3
3D Applications	1

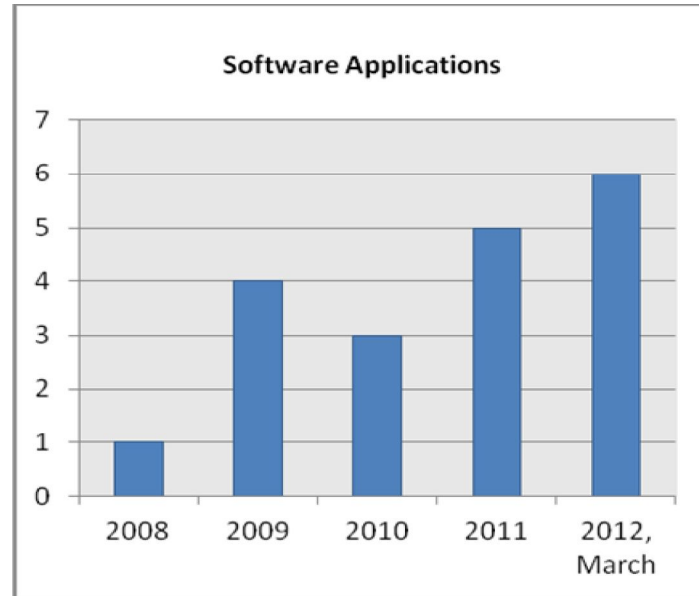


Fig. 2.14: Papers published in the software applications category

2.8 The gap in software engineering

On classification of the researches published in the last five years and analysis of data, it gives a concrete idea that research conducted in software re-engineering, software engineering techniques and software applications is meager while a higher percent of research is carried out in rest of the two areas as shown in Fig. 2.2 and Table 2.1. The comparison among the number of papers published in all five categories with respect to the year of publishing is shown in Fig. 2.15. From the figure, it is understood that very less research is conducted in the area of software re-engineering in the current year (March 2012), therefore, research gap exists in the area of software re-engineering which means more research is possible in the same.

On reviewing more than 250 papers related to software engineering published in various reputed journals, it was realized that software engineering is a very important research area and it can be classified into various sub-areas like software development process, software management, software re-engineering, software engineering techniques and software applications. Research was carried out to a lesser in all five categories in the year 2008, 2009 and 2010, where as there is substantial rise in research conducted in the year 2011 specifically in the first two categories, which is again followed by a sudden drop in the year 2012 until March. Further, the study also provides an insight on the insufficient

research carried out in software re-engineering and hence the author recommend work on this particular area to improve and enhance the software re-engineering process.

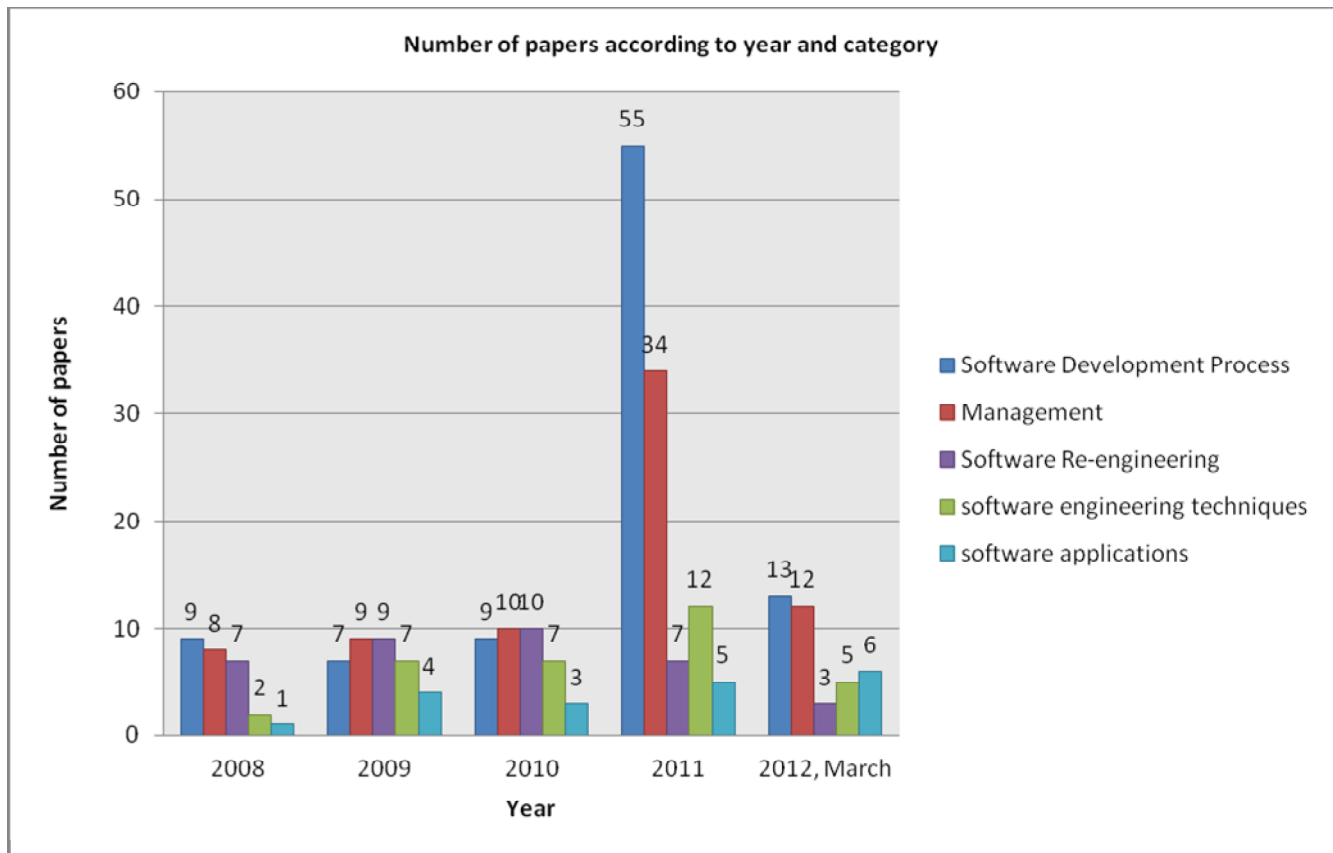


Fig. 2.15: Papers published in each category from the year (2008 - 2012, March).

2.9 Literature review on software re-engineering

Yang et al. (2001), published that due to the improper software maintenance practice in the past decades, the documentation of a software product often becomes inconsistent or fails to be up-to-date and therefore is no longer a reliable map indicating the linkage between high-abstract level knowledge and source code. The elicitation of information from source code is therefore important for many software engineering activities such as program understanding, software maintenance and software evolution. In this paper, the background, definitions, processes, coverage, techniques and tools for information elicitation from source code, and its future opportunities are introduced.

In 2001, **Bianchi et al. (2001)**, published a paper titled “Iterative Re-engineering of Legacy Functions”, this paper explain a process of gradual re-engineering of the

procedural components of a legacy system. The proposed method enables the legacy system to be gradually emptied into the re-engineered system, without needing to either duplicate the legacy system or freeze it. The process made up of evolving the legacy system components toward firstly a restored system and then toward the re-engineered system. Meanwhile, the legacy system can coexist with both the re-stored and the re-engineered parts. By the end of the process, a single system will be in existence: the re-engineered one. The method has been applied to re-engineer a real system and demonstrated its ability to: support gradual re-engineering, maintain the system at work during the process, and minimize the need to freeze maintenance requests, renew the operative environment of the re-engineered system with respect to the legacy system and, finally, remove all the system's aging symptoms. This re-engineering process is shown in Fig. 2.16, the various phases each being enclosed in a box, while the label on the arrow linking two successive phases indicates the output produced by the relative phase, **Bianchi et al. (2001)**.

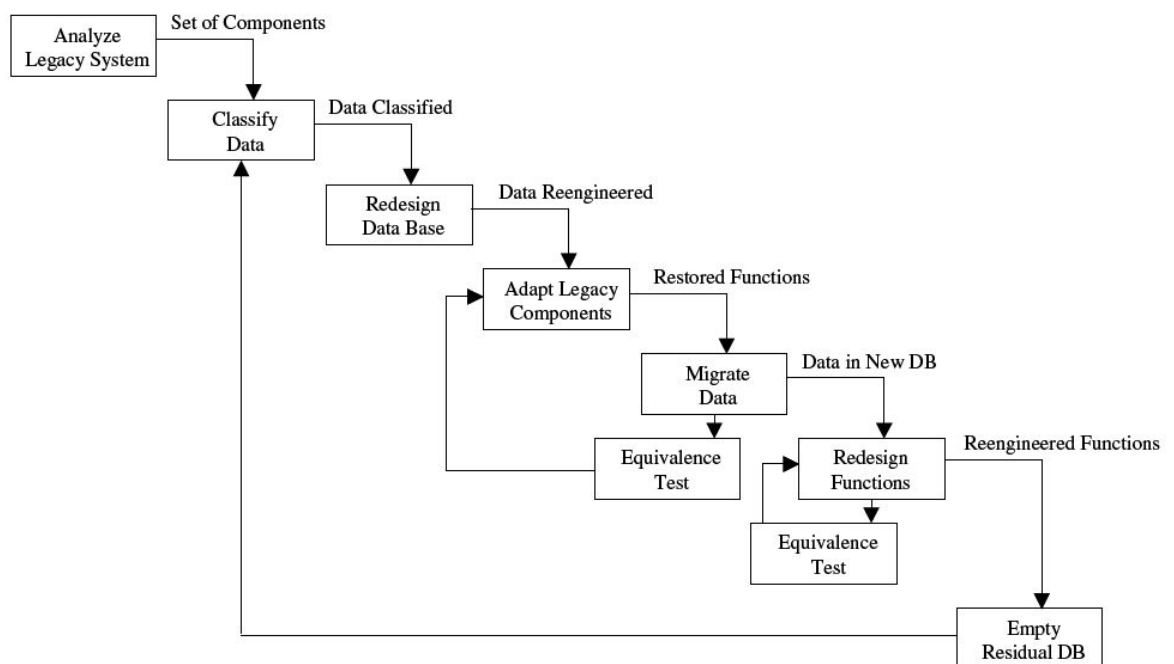


Fig. 2.16: Iterative re-engineering process.

William and Lu (2002), found that most of the legacy systems were designed in traditional way with inflexible architecture. Due to improper representations or designs, they are very difficult to understand and maintain. As a result, maintaining these legacy systems is very costly. The objective of software re-engineering is to re-structure or re-write legacy systems into new systems with better maintainability, flexibility, and scalability. Design patterns (DPs) have integrated successful standard design practices and

expert experiences into a set of components that exhibit known behaviors with better structures. Re-engineering legacy systems into pattern-based systems not only decreases the cost of further software evolution but also increases their re-usability, **William and Lu (2002)**.

Ying (2003), carried out his thesis that represent a re-engineering framework, that allows for quality requirements of the target system to be modeled as soft-goals, and transformations to be applied in selective manner towards achieving specific quality requirements for the target system. In this regard, the migration process is denoted by a sequence of transformations each one of which alters the state of the system being migrated. A state transition system and the Viterbi algorithm are used to identify the optimal sequence of transformations that can be applied at any given state of the migration process. The re-engineering framework can be applied incrementally after a large system has been decomposed into a collection of work areas. Furthermore, a technique that allows for the integration of the migrant systems with other third party applications in a Web enabled environment is presented. A re-engineering toolkit to automatically migrate a set of open source procedural systems, such as Apache, Bash and Clips to an object oriented platform has been developed using the theory presented in this thesis. The obtained results demonstrate the effectiveness and usefulness of the proposed incremental quality driven migration technique, **Ying (2003)**.

Zhuopeng et al. (2005), found that the adoption to Web service technologies and existing non-service-oriented software systems turn to be legacy systems. In this paper, an architecture-based service-oriented approach was used to support service-oriented re-engineering. It integrates and re-uses software components as services over Internet by wrapping underlying computing models with Extensible Markup Language (XML). Service packaging processes and service identification are described in details, which are the keys to service-oriented re-engineering. Target services are defined according to service identification, which is depend on a comparison of analyses in both problem domain and legacy systems. In service identification process, clustering techniques are applied to analyze re-covered architectural information and to identify related modules. Service packaging is take advantage of the architecture-based development and re-use, **Zhuopeng et al. (2005)**.

Yang et al. (2005), have published paper titled "A Dual-Spiral Re-engineering Model for Legacy System". This paper presents a dual-spiral re-engineering model, which performs as a cyclic approach. The main workflow in Dual-Spiral Re-engineering Model requires that the two systems (legacy one and target one) work together, and move the functionality (not the modules) from the legacy system to the target system step by step, as in the spiral model. During the entire process, the active functionality in the legacy system is in a decremented pattern, and the active functionality in the new target system is in an incremental pattern, as Fig. 2.17, **Yang et al. (2005)**.

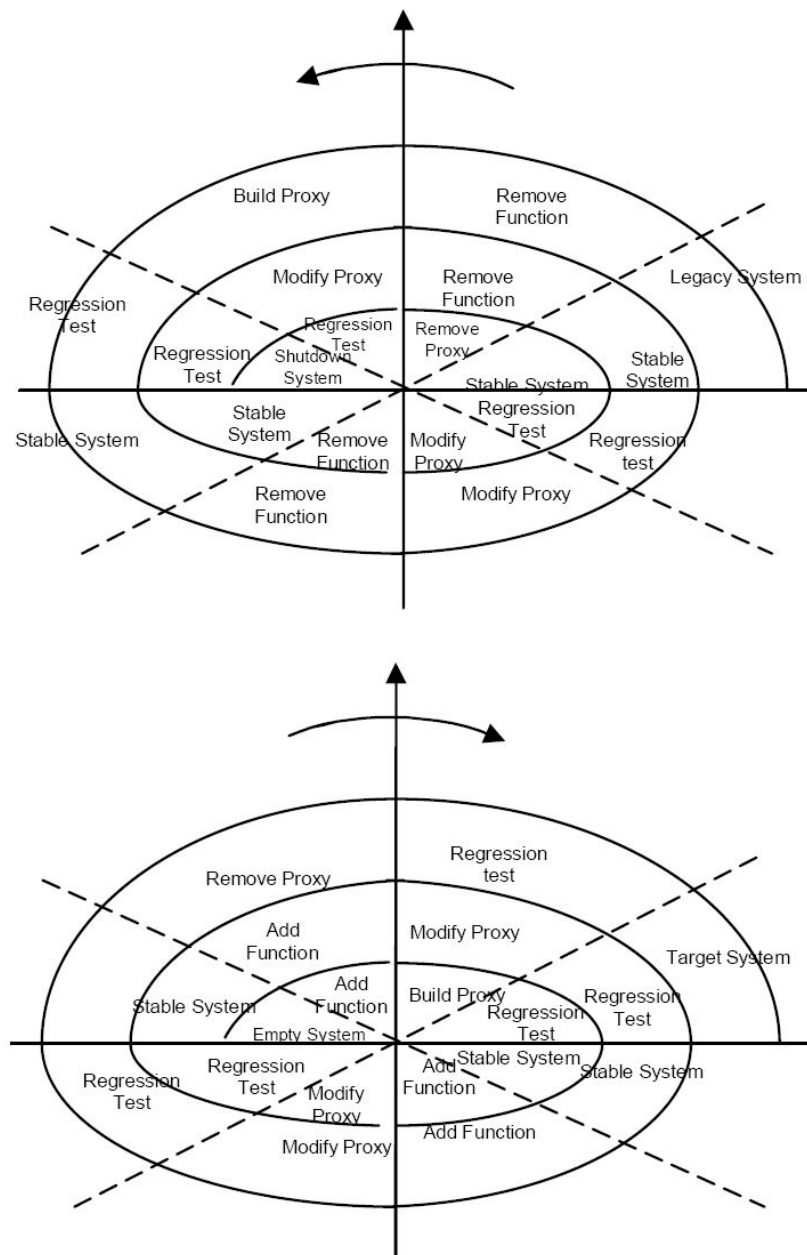


Fig. 2.17: Decremental Legacy system Pattern and Incremental Target system Pattern in Dual-Spiral Model.

Christophe and Corbiere (2006), have identified the Specifications recently proposed as standards in the domain of Technology Enhanced Learning (TEL), questioned the designers of TEL systems on how to put them into practice. Modern studies in Model Driven Engineering have highlighted the need for a framework which could formalize the use of these specifications as well as enhance the quality of the developments. This paper contend with the opportunity for the TEL community to adopt such a model to express and formalize the design organization and the engineering process of a TEL system. This type of model could provide a set of concepts allowing the description of these tasks: specifying, modeling and analyzing, which required to perform for defining the negotiation and the communication between actors in such a community. In the first part, they stress the need for the instantiation of this framework so as to integrate recent results as well as taking into account the evolution of software re-engineering. In the second part, they propose two instances of this framework: the first concerns the reverse engineering of a TEL system and the second its re-engineering, **Christophe and Corbiere (2006)**.

Krogmann (2007), has written the paper titled “Re-engineering of Software Component Models to Enable Architectural Quality of Service Predictions” and he said that: Predicting extra-functional properties of software systems requires knowledge about their architectures. For component-based software, it also needs for understanding extra-functional properties of each individual component. However, black-box components, as most popular provided by current component models, do not provide enough details to predict QoS. Two architectural design scenarios were sketched and described, these scenarios have information about a component that needed to enable relevant analyses. Depending on these considerations a component model is presented which defines such relevant information. They discuss re-engineering approaches, namely static analysis and analysis integrated with evolutionary algorithms to learn from monitored data to yield such information about a component. They conclude with a more refined view on the black-box-nature of components. For many, the black-box principle of component usage means that a component does not expose any information on its internal. Accordingly, components are used at design- and run-time solely via their interfaces. In this paper they have shown that for the analysis of extra-functional properties, one needs information on the internal of a component which is not given in the interfaces. Since this trade-off between the black-box nature of a component on the one hand and the architectural and QoS analysis on the other

hand seems to be unavoidable, they can ask for the reasons and benefits of the black-box nature of components. From their point of view, there are two reasons why the black-box nature of components should be kept in high esteem:

- 1) The re-use of a component at design-time should be as easy as possible. In particular, one should not have to understand the component's internal to compose and deploy the component (information hiding). This is one of the major benefits of components compared to other software re-use approaches.
- 2) The business knowledge of the component creator should be protected. Unlike open source software, the creator of a re-usable component may base his or her business model on the assumption that the know-how which was used to create the component is protected and not "lost" when the component is released.

However, they take the position that these requirements do not conflict with the retrieval of additional information on a component beyond its interfaces. Therefore, they presented a component re-engineering approach supporting information in addition to interfaces to enable the prediction of extra-functional properties. These models of components, Resource Demanding Service Effect Specifications (RDSEFF) are going to be extracted using an approach combining static and dynamic analysis with an evolutionary algorithm, **Krogmann (2007)**.

Denier et al. (2008), have written the paper titled "Reverse Engineering the Literature on Design Patterns and Reverse-Engineering" and they said that: Since their beginning in 1994, design patterns have been the subject of many papers. In the reverse-engineering community, several authors have proposed approaches to consider design patterns during reverse- and re-engineering. However, it has been recently put forward in the community that it is difficult to compare previous approaches due to the diversity of vocabulary and the lack of a general framework to map and relate these approaches. Consequently, they study 59 papers related to design patterns in the software engineering community at large (1) to identify and define common terms related to design patterns, (2) to determine recurring themes in the papers, and (3) to further characterize approaches for design pattern detection along several categories. Recurring themes allow them to provide the portrait of the typical "paper on design patterns while categories draw the portrait of the typical" approach in design pattern detection. They proposed to the community to use a x

vocabulary, to diversify the approaches, and to build a common benchmark to assess the reverse engineering of design patterns. They presented an extensive study of the literature on design patterns triggered by the loose definitions of design patterns, which prevent comparison among approaches, particularly in the reverse-engineering community. Firstly they performed a domain analysis of the literature on design pattern to propose a consistent vocabulary. The retained terms allowed them to cast previous papers in a common framework and to compare their approaches from the linguistic point of view. Secondly using the previous vocabulary, they identified six themes categorizing the papers on design patterns. The themes allow them to classify approaches and to help readers focus on approach of interest. Thirdly they refined the design pattern detection sub-theme into seven categories to further classify related approaches. Finally, they discussed the difficulty of measuring the precision and recall of the detection approaches due to the lack of common vocabulary and of agreed-upon benchmarks. They drew the portraits of the typical paper and approach related to design patterns and their detection. They also suggested research directions for the community to strengthen the field of design pattern detection in reverse-engineering, **Denier et al. (2008)**.

Mongan et al. (2008), have written the paper titled “Re-engineering a Reverse Engineering Portal to a Distributed SOA” and they said that: REportal is an existing web-based reverse engineering portal web site that provides access to a suite of reverse engineering and program comprehension tools via a web browser. This extraction was intended to allow ease of system maintenance by adding and upgrading tools without involving the end user. Anywise, the software tools and server technologies used became deprecated so quickly that it was not possible to take full advantage of the architectural vision. Using a service-oriented architecture, they abstract the process flow of the system from the underlying tools, enabling a wizard-style approach of adding services to the system, and facilitating more “hands-off” maintenance. In this paper, they describe the challenges and benefits of this architectural migration. In this paper, they also discussed the original spirit of the REportal system. they detailed the challenges they faced in maintaining REportal and its underlying tools as their technologies evolved. They re-engineered REportal by taking its underlying tools and creating service wrappers around them, and then integrating those services using a JSP-based web service client. This new architecture allows us to maintain REportal by using abstract XML queries for

representations of data, facilitates adding new tools to the system by insulating them in a decoupled service layer, and permits heterogeneous REportal client instances that independently invoke the portal's services. Using REportal as a case study, a great deal of potential in the area of SOA research was also provided, including a business process model for reverse engineering and an XML representation for program understanding, **Mongan *et al.* (2008)**.

Nojournian and Lethbridge (2008), have written the paper titled "Re-engineering PDF-Based Documents Targeting Complex Software Specifications", The goals of this paper at document engineering of PDF-based complex software specifications, where the specifications of Object Management Group (OMG) are the initial targets. Their motivation is that such specifications are dense and intricate to use, and tend to have complex structures. The objective is therefore to create an approach that allowed them to re-engineer PDF-based documents, and to illustrate how to make more applicable versions of electronic documents such as specifications, technical books, conference proceedings, etc so that end-users to have a best experience with them. The first step was to elicit the logical structure of the document in a meaningful XML format for subsequent processing. Their initial assumption was that, several key concepts of a document are expressed in this structure. In the next phase, they created the multi-layer hypertext version of the document to facilitate document browsing and navigating. Although they initially focused on OMG software specifications, they chose a general approach for various phases of their work including format conversions, logical structure elicitation, text extraction, multi-layer hypertext generation and concept exploration; therefore, they can process other complex documents to achieve mentioned goals. In this research, they have described an approach for taking a raw PDF version of a published software specification, and conversion this into a multi-layer hypertext document that will be much more useful to end users. As an intermediate step, they generated a clean XML document with meaningful tags, and then constructed from this a series of hypertext pages constituting the final system. The key contributions of the research are: (a) to illustrate methods for re-engineering a PDF-based specification in a general way, and (b) to demonstrate how to make a more usable hypertext version of a document so that end-users to have a better experience with software specifications. The major goals were to make a complex document more usable by allowing navigation of both its structure, and also of semantics illustrated by the document.

The first phase involved document analysis to better understand the structure of the document and establish a good infrastructure for their later objectives. They experimented with processing using a variety of tools and formats for transformation and extracted the logical structure of various OMG software specifications in XML formats. Text Extraction by XSLT and XPath Technologies Transforming the Doc by Adobe Extracting Logical Structure: XML Producing Multiple Hypertext Pages, Producing Table of Contents, Extracting Document, Concept Generating, Heading, Numbers Generating Headings and Their Corresponding Hyperlinks Filtering Headings Linking Hypertext Pages: Linker() Cross Referencing CrossRef(), PDF Document with Bookmarks, Multi-Layer Hypertext Version: HTML Extracting Doc Body without Headings, Extracting Document Headings, Document's Data Analysis. In the second phase, they generated multiple hypertext pages for end users, to facilitate document browsing, processing, navigating, and concept exploration. they used the latest World Wide Web Consortium (W3C) technologies such as XSLT and XPath expressions and learned that although using these technologies, they can parse every XML document, it would be more usable if the created XML documents have strong logical relationships among their elements and attributes similar to the XML documents they produced (e.g. if the first child of a <Section> element contains the Class Descriptions string then developer can detect UML classes and packages in grandchildren of that <Section> element and so on, That is because of the excellent logical capabilities that XPath and XQuery expressions provide for processing XML documents while other kinds of parsers may not supply such a valuable ability. As a consequence, they do believe that people in charge of software specifications can enrich their documents by embedding such a logical relationship and meaningful keywords inside headings, paragraphs, etc. To conclude, since they considered the generality of every module that was generated, except the concept extractions which needs further research, they ended up with the idea of creating a new Document Engineering Framework for complex software specifications and documents, **Nojournian and Lethbridge (2008)**.

Bergel et al. (2008), have written the paper titled "Enhanced Dependency Structure Matrix for Moose". The Dependency Structure Matrix (DSM) is an approach developed in the context of process optimization, has been applied successfully to identify software dependencies among packages and subsystems. It exists a couple of algorithms to help organizing the matrix in a form that reflects the architecture and highlights patterns and

problematic dependencies between subsystems. However, the existing DSM implementations often miss some important information in their visualization to fully support a re-engineering effort. In this paper, they enhanced DSM with enriched cell contextual information by showing information (i) about the kinds of references made (inheritance, class accesses), (ii) the rate of entities (classes/methods) doing references, (iii) the rate of entities been the target of the references. They distinguish independent cycles and stress the cycles using coloring information. This work has been implemented on top of the Moose open-source re-engineering environment and Mondrian. It has been implemented to non-trivial case studies such as the Morphic UI frameworks available in Squeak, an open-source Smalltalk. This paper enhances Dependency Structure Matrix (DSM), an approach which identifies dependencies between software packages. A DSM renders references between packages in a matrix-like table. In addition, several algorithms already exist to reorder the matrix and so reveal the software architecture. However, a traditional DSM does not provide enough information about the references between the packages and about the cycles which can exist between them. This paper addresses such limitations by improving the information contained in a DSM and enhancing the way a DSM is rendered. First, the cell contents is enriched with the kinds and spread of the references as well as with the number of classes involved. Second, several colors are used to distinguish direct and indirect cycles, **Bergel et al. (2008)**.

Niiyama et al. (2008), have written the paper titled “Service-Oriented Software Re-engineering Methodology for Composite Services”, The aim of it is double: 1) to apply a current re-engineering methodology called Service-Oriented Software Re-engineering (SOSR) to the modernization of a legacy application into a service-oriented application, and 2) to extend the SOSR method to contain composite services. There is a real need for new software methodologies to address the effects of composite web services to software re-engineering projects. One of the methodologies called SOSR is still in its early stages and has yet to be fully tested against applications designed and implemented with a business process engine. It has not been applied to applications using composite web services. For this purpose, the reverse software engineering portion of the current SOSR is applied to a service-oriented system in which composite services are heavily employed. Based upon the results of reverse engineering, the SOSR methodology is extended to include guidelines for a service broker who composes a set of services, deploys, and

executes the composite services. And then, they verify the extended SOSR methodology by re-engineering a legacy system not using services by following the methodology. The guidelines of a service broker are described in terms of roles of a service broker over time, tasks within 4+1 views, the descriptions of the tasks in UML (Unified Modeling Language), and the relationships between roles and tasks, which are represented by an RACI (Responsible, Accountable, Consulted, and Informed) chart. The results of this research will help a software developer to re-engineer a legacy system to a target system using composite services. Software design composed of loosely coupled web service components is an emerging software development paradigm. Business Process Execution Language (BPEL) engines provide an environment to develop software as a set of Business Processes comprised of web service components. This type of loosely coupled architecture provides a flexible, efficient, and time effective way of software development. Business process needs drive business decisions and BPEL process engines meet these needs head on with the help of Extended SOSR. In this paper, they first applied the SOSR methodology to the modernization of a legacy application that does not use web services and gained some key experiences. Based upon those experiences, they extend the SOSR methodology to include guidelines for a service broker who composes a set of services as a composite service(s), deploys, and executes the composite services. The results of this research will help a software developer to re-engineer a legacy system to a target system using composite services. The Extended SOSR has several strengths compared to the current SOSR. First of all, it proposes Service-Oriented Reverse Software Engineering in terms of both 4+1 views and three service stakeholders. Secondly, the sequence of views in reverse and forward engineering processes is clearly defined. Therefore, the UML modeling sequence is also very clear. Thirdly, guidelines of modeling for a service broker are also presented. Lastly, the sub-roles of the service broker and their tasks are managed through a proposed RACI chart, **Niiyama *et al.* (2008)**.

Riediger and Winter (2008), have written the paper titled “Graph Technology in Reverse Engineering, The TGraph Approach”. The TGraphs are defined as directed graphs together with attributed, typed, ordered edges and nodes. These characteristics leverage the use of graphs as models for all kinds of artifacts in the context of software re-engineering. TGraphs are accompanied by meta-model based technologies to define, visualize, manipulate, analyze, query, and transform graphs. This paper summarizes the work

accomplished around TGraphs and graph based modeling over the last decade. They introduce TGraphs as a versatile and expressive formalism and demonstrate their well-suitedness for solving reverse engineering problems. TGraph technology provides a versatile graph model and comes with elaborated grUML meta-modeling facilities to enable extensive adaptability. Its implementation and analysis support by GraLab and Graph Repository Query Language (GReQL) make it practically available for the development of reverse engineering tools. Next to their application in reverse engineering, TGraphs and their associated techniques were also successfully applied in defining meta-models for visual languages and creating and using the KOGGE-MetaCase-Tool. Currently, they are applied as central fact repository in the Requirements Driven Software Development System (ReDSeeDS). The TGraph related development reported in this paper evolved over the years and was driven by varying requirements from the different project contexts. Since seamlessness has been the main goal, all design decisions were done in the light of keeping the approach consistent and well-balanced. There are still some open requests for extending the approach, the most relevant being the claim for distributed graphs, hierarchical graphs and/or hypergraphs. The extension of the approach in these directions is the main challenge for the future, **Riediger and Winter (2008)**.

Tankeleviciene (2008), has written the paper titled “Using Re-engineering Techniques for Distance Study Course Improvement”. This paper analyses and discusses the concept of re-engineering as it comes from software engineering and management fields. Two viewpoints were integrated and applied to solve a problem of re-engineering of a distance study system, in general, and the learning unit, in particular. She propose a framework for re-engineering of unit of learning, based on general model of software re-engineering, and present a case study, in which she describes how one topic of distance study course was re-engineered, considering triple consistency principle and requirements for computer science. The suggested framework contributes to increasing quality, efficiency and systematization of delivering distance studies. The proposed framework contributes to increasing quality and effectiveness in delivering distance studies. Organization must follow common framework for distance study course re-engineering, in order to gain minimum expenditures for overall e-Learning system maintenance. The systematic approach and the use of software engineering methodologies in instructional engineering allow them:

- 1) To decline expenditures of time for direct (design and development) and indirect (re-design and re-development due to misunderstanding) processes;
- 2) To accept and manage complexity;
- 3) To facilitate re-use;
- 4) To prepare the best for adaptation and personalization.

The case study shows re-engineering of one topic, but the framework can be used while re-engineering ULs of different granularity. Also, it can be extended to provide clarity and systematization in higher layers of e-Learning system. Further, the results of analysis of empirical student evaluation of the course and course log files can be employed in this framework, **Lina Tankeleviciene (2008)**.

Mishra *et al.* (2009), have written the paper titled “Creating Reusable Software Component from Object-Oriented Legacy System through Reverse Engineering”. Because of the constant change in technology, shortage of standardization, difficulties of changes and absence of distributed architecture, the legacy system's business value have become weaker. The importance of legacy systems could not be undermined because some of their functions are too valuable to be discarded and too expensive to reproduce. The researchers and software industry have recently paid more attention towards the component based software development to enhance the productivity and accelerate time to market. Re-implementing the business critical applications within time and resource constraints is not the best option, the best option is Software Re-engineering (SRE) with effective design and architecture which can make the system better for re-usability and maintainability. The main motive behind the re-engineering is integrating the legacy system with emerging technologies. To achieve these goals, they have proposed a systematic and concrete model named as Component Oriented Reverse Engineering (CORE). It aims to identify and develop re-usable software components. By using the techniques of reverse engineering; they can extract architectural information and services from legacy system and later on convert these services into components. They have proposed a model named Component Oriented Reverse Engineering (CORE) for identifying and creating re-usable software component. It handles reverse engineering methodology to expose components from the object oriented legacy system. CORE makes use of repository to store and manage the tested components and re-structures the new system that finally integrates the new system

with the re-usable components. The re-usability of the software component is most popular way to enhance the productivity and improve the quality and reliability of the new software systems by reducing the development costs. Because of these reasons, it is very important to identify independent components having low coupling and high cohesion. They have also proposed a systematic approach to identify re-usable component from object oriented legacy system through Object Oriented Analysis and Design (OOAD) models namely use case, class diagrams and communication diagram. Accordingly, the technique of component identification may identify re-usable components more effectively based on facility of CRUD (Create Read Update Delete) matrix, class clustering. They have validated this approach by using a case-study named Library Management System and also test the components for re-usability and the result shows that the approach supports all the 5 major functionalities, **Mishra *et al.* (2009)**.

Laval *et al.* (2009), have written the paper titled “Identifying cycle causes with CycleTable”. The identifying and understanding cycles in large applications are a challenging task. In this paper they present CycleTable which displays both direct and indirect references. It depicts how minimal cycles are intertwined through shared edges and allows the re-engineer to devise simple strategies to remove them. In this paper they presented CycleTable. This visualization shows cycles between packages in a system. Each cycle is presented in a separated column. A colored frame show which edge is shared by several cycles. To complete the visualization, enriched cells have been adapted and integrated to represent each edge. This visualization is a good complement of DSM visualization. This visualization allows solving cycles separately or conjointly, **Laval *et al.* (2009)**.

Ratiu (2009), has written the paper titled “Challenges for Domain Knowledge Driven Program Analysis”. Programming is a knowledge intensive activity. In order to write and evolve programs, programmers make use of knowledge of different kinds like the business domain, programming technologies or design. Most of the current reverse engineering approaches do not explicitly take into consideration the domain knowledge implemented in programs when the analyses are performed and thereby they lose important information. This situation is due to a series of difficulties that need to be overcome when performing analyses that interpret the code from the point of view of domain knowledge. In this paper they advocate the need for the explicit use of domain knowledge in program analyses and

they discuss the most important challenges that have to be addressed to realize such analyses in the practice: making the knowledge explicit, obtaining the knowledge, defining appropriate interpretations, and recovering the interpretations automatically. They present islands of solutions to approach these challenges and they argue that to systematically overcome them is required a community effort. They advocate that in order to systematically address these challenges and to use domain knowledge in automatic program analyses, they need a community effort. Systematic building of knowledge bases about programming technologies that are frequently used annotations of standard APIs with the conceptual information, definition of interpretations adequate for performing certain analyses require efforts that can be addressed only in a community, **Ratiu (2009)**.

Perin (2009), has written the paper titled “Driving the refactoring of Java Enterprise Applications by evaluating the distance between application elements”. Java Enterprise Applications (JEAs) are complex systems composed using various technologies that in turn rely on languages other than Java, such as XML or SQL. Given the complexity of these applications, the need to reverse engineer them in order to support further development becomes critical. In this paper they show how it is possible to split a system into layers and how it is possible to interpret the distance between application elements in order to support the refactoring of JEAs. The purpose of this paper is to explore ways to provide suggestions about the refactoring operations to perform on the code by evaluating the distance between layers and elements belong to those layers. They split JEAs into layers by considering the kinds and the purposes of the elements composing the application. They measure distance between elements by using the notion of the shortest path in a graph. Also they present how to enrich the interpretation of the distance value with enterprise pattern detection in order to refine the suggestion about modifications to perform on the code. In this paper they summarized their proposal to drive the refactoring of JEAs by comparing the distance between the application’s elements to that of the data source architectural patterns. In particular they presented the layering scheme that they adopt to re-group different elements of a JEAs. They also explore how to apply the concept of method distance to create an index that can be used to detect the presence of a wrong application structure. Finally they relate the concept of distance to the data source architectural patterns in order to modify the interpretation of the distances. In this case they propose to compile a catalogue of patterns and distances together with heuristics to drive

the refactoring. The basic idea is that every service has to be implemented touching every layer starting from the Presentation one. The catalog will be able to indicate what is wrong in the implementation and how the code should be modified in order to have a right structure. The approach could also be used to expose code that is too complex, i.e., if the distance is too high, **Perin (2009)**.

Rothlisberger (2009), has written the paper titled “Why and How to Substantiate the Good of our Reverse Engineering Tools?”, and he said that: Practitioners and researchers are usually eager to develop, test and experiment with new ideas and new techniques to analyze software systems and/or to present results of such analyzes, for instance new type of visualizations or analysis tools. However, usually these novel and definitely promising ideas are never properly and seriously evaluated through empirical observation. Instead their inventors simply resort to anecdotal proof to substantiate their beliefs and claims that their ideas and also the realizations there from are literally helpful in theory and practice. The main reason why proper validation is often neglected is that serious evaluation of any newly tool, realized technique, or concept in reverse engineering is time-consuming, laborious, and usually tedious. Furthermore, he suppose that there is also a lack of knowledge or experience concerning empirical evaluation in the community. This paper hence sketches some ideas and discusses best practices of how could be still, with moderate expenses, come up with at least some empirical validation of the next project in the field of reverse engineering. In this paper he first motivated the need for conducting empirical research in the reverse engineering and re-engineering field by raising some important questions that remain unanswered as long as only very little empirical validation on techniques and tools is performed. In a second step, he introduced several types of empirical studies such as surveys, case studies, or controlled experiments that could be employed to actually conduct empirical studies. Moreover, he gave some hints how to concretely conduct which kind of study in which context and referred to important work on empirical software engineering done by other researchers, **Rothlisberger (2009)**.

Jasmine and Vasantha (2009), have found that software reuse is widely considered as a way to increase the productivity and improve the quality and reliability of new software systems. Identifying, eliciting and re-engineering software components, which implement abstractions within existing systems is a promising cost-effective way to create re-usable test assets. In the present scenario, one of the major problems in building large-scale

enterprise systems is anticipating the performance of the eventual solution before it has been built. Testing is a significant and important part of the software development lifecycle to ensure a high quality product with a minimum number of faults. But most organizations don't have a standard process for defining, managing, organizing, and documenting their testing efforts. Usually testing is conducted as an ad hoc activity, and it changes with every new product. Without a standard foundation for test planning, development, execution, and error tracking, testing efforts are non-repeatable, non-re-usable, and difficult to measure. Re-engineering the test management process can solve the problems due to unstructured, decentralized test management. This paper explains the goals of re-engineering test management and how to achieve it and the approach as demonstrated, constructs useful models that act as predictors of testing effectiveness in component based enterprise applications, **Jasmine and Vasantha (2009)**.

Garg and Jindal (2009), have written the paper titled "Reverse Engineering – Roadmap to Effective software Design" and they said that: The process of duplicating an existing component, product, or subassembly, without the assistance of drawings, documentation, or computer model is known as reverse engineering. So reverse engineering is the process of learning the design of an object by studying its implementation. It is the process of discovering the technological principles of a device, system or object through analysis of its structure, function and operation. The role of reverse engineering in software evolution is important. It provides the means for re-capturing lost information, re-structuring complex systems or transforming old systems to a new, better architecture and more maintainable. Reverse engineering software aids researchers to examine the strength of systems and identify their weaknesses in terms of performance, security, and interoperability. The goal of reverse engineering itself is the improved documentation of how the original product works by uncovering the underlying design. In this paper various reverse engineering tools are discussed which help in developing software design better by the use of existing source code. It is discussed in this paper that reverse engineering begins with the product and works through the design process in the opposite direction to arrive at a product definition statement (PDS). The creation of better designs and the interoperability of existing products often begin with reverse engineering. In doing so, it uncovers as much information as possible about the design ideas that were used to produce a particular product. It generally involves developing a set of functional specifications for a

product, system or piece of equipment, based on an analysis of existing product, **Garg and Jindal (2009)**.

Laval et al. (2009), have written the paper titled “Identifying cycle causes with Enriched Dependency Structural Matrix”, the Dependency Structure Matrix (DSM) has been successfully applied to identify software dependencies among packages and subsystems. A number of algorithms were suggested to compute the matrix so that it highlights patterns and problematic dependencies between subsystems. However, existing DSM implementations usually miss important information to fully support re-engineering effort. For example, they do not clearly qualify and quantify problematic relationships, information which is critical to support remediation tasks. In this paper they present enriched DSM (eDSM) where cells are enriched with contextual information about (i) the kind of dependencies (inheritance, class reference), (ii) the rate of referencing entities, (iii) the rate of referenced entities. They distinguish independent cycles and stress potentially simple fixes for cycles using coloring information. This task is language independent and has been implemented on top of the Moose re-engineering environment. It has been applied to non-trivial case studies among which , Morphic and ArgoUML the UI framework available in two open-source Smalltalks, Pharo and Squeak. Solution to problems identified by eDSM have been performed and retrofitted in Pharo main distribution. This paper enhances Dependency Structure Matrix (DSM) using small multiples. Firstly, colors are used to distinguish direct and indirect cycles. Secondly, cell contents are enriched with the nature and strength of the dependencies as well as with the classes involved. They applied the eDSM on a complex system and systematically checked and tried to fix the cycles. Out of 45 direct cycles, they could propose 25 solutions to break the cycles. 18 got accepted and implemented by the maintainers of the Pharo open-source Smalltalk, **Laval et al. (2009)**.

Coerne et al. (2010), have written the paper titled “Automotive Software: Characteristics and Re-engineering Challenges”. The automotive software is different from the kind of software that is usually addressed by current re-engineering research. This paper gives an overview of the particular conditions and challenges faced in the automotive domain. In the last few years, re-engineering was introduced to Robert Bosch software development. Within the existing programming paradigm, re-structuring techniques were successfully applied and resulted in improved maintainability and reduced resource usage. However,

this was largely done manually, which is too laborious as a developer's daily work, **Coerne et al. (2010)**.

Wirsing et al. (2010), have written the paper titled "SENSORIA Patterns: Augmenting Service Engineering with Formal Analysis, Transformation and Dynamicity" and they said that: The IST-FET Integrated Project SENSORIA is developing a novel comprehensive approach to the engineering of service-oriented software systems where foundational theories, methods and techniques are fully integrated into pragmatic software engineering processes. The tools and techniques of SENSORIA encompass the whole software development cycle, from architectural design and business, to quantitative analysis and qualitative analysis of system properties, and to code generation and transformation. The SENSORIA method takes also into account re-configuration of service-oriented architectures (SOAs) and re-engineering of legacy systems. In this paper they give first a short overview of SENSORIA and then present a pattern language for augmenting service engineering with formal analysis, dynamicity and transformation. The patterns are designed to aid software developers choose appropriate tools and techniques to develop service-oriented systems with support from formal methods. They support the all stages of development process, from the first stage of modeling to deployment activities and give an overview of many of the research areas pursued in the SENSORIA project. In this paper, they have presented some results of the IST-FET EU project SENSORIA, in the form of a pattern language. The patterns address a wide range of issues, such as modeling, specification, analysis, verification, synchronization, and deployment of services. They are currently working on systematizing and extending the collection of patterns in these areas, and they will also be developing patterns for areas which are not currently addressed, e.g., business process analysis and modeling. This pattern catalogue is a useful guide to the research results of the SENSORIA project, they are investigating a broad range of subjects and without some guidance it may not be easy for software developers to find the appropriate tools or techniques, **Wirsing et al. (2010)**.

Pilkington et al. (2010), have written the paper titled "ESIDE: An Integrated Development Environment for Component-Based Embedded Systems". In this paper they present ESIDE, a combined development environment for component-based embedded systems. The benefit of component-based software engineering principles is that the helps to facilitate efficient, robust hardware/ software, and scalable, co-design, co-simulation, co-

verification, and their seamless integration. They first describe the architecture and features of ESIDE, then discuss several design decisions that they faced in developing ESIDE and the trade-offs in making these decisions. To provide perspective, they report their experiences in re-engineering TinyOS based networked sensor systems into complete component based designs that cover both hardware and software. In this paper, they have presented ESIDE, an integrated development environment for component-based embedded systems. Case studies on re-engineering networked sensor systems have demonstrated that component-based development using ESIDE facilitates effective co-simulation and co-verification while preserving the efficiencies needed by embedded systems, **Pilkington et al. (2010)**.

Gahalaut and Khandnor (2010), have written the paper titled “Reverse Engineering: An Essence For Software Re-engineering and Program Analysis”. The object oriented software developers now admit that thinking about object-oriented program understanding and comprehension to be relatively easier is not that easy. Programs are even difficult and more complex to comprehend, unless strictly documented. What if the documentation is inappropriate? To affect change management, even a simpler upgrade may become cumbersome then. This is the reason that cause eminent development houses now focusing on advanced documentation support. Re-engineering code environment then largely affect the problem issues regarding program comprehension when the software size grows enormously. Reverse Engineering is a method that greatly reduces the time, complexity and effort involved in solving these issues providing efficient program understanding as an integral constituent of re-engineering paradigm. This paper talks about reverse engineering of java code. It also concludes about the efficiency of some java reverse engineering tools and sufficiency of static analysis over runtime dynamic analysis for revelation of code structure from byte-code. Testing de-compiler tools with the current java features revealed the areas where these tools failed to produce correct results. UML reverse engineering outcome from the UML tools they have used in their study, ALTOVA U-Model and Enterprise Architect from Sparx Systems reveal same structural information about java code as the compiled byte-code with input once taken as source decompiled form the de-compiler tools and original Test programs next, which more ascertains the sufficiency of static byte-code analysis for the original code structure recovery by using reverse engineering tools, **Gahalaut and Khandnor (2010)**.

Singh et al. (2010), have written the paper titled “Re-engineering to Analyze and Measure Object Oriented Paradigms”, the object oriented development has proved its worth in today’s system because its design and development is reliable, better, and easier to access than the traditional methodologies as the updated requirements and lack of documentation in old systems has provided a motivation to revamp the systems. Re-designing or re-building the same system is highly expensive. To solve this problem reverse engineering of the system is used as most suitable alternative. Field of reverse engineering is extending its horizon day by day; it requires re-usability not only at code level but also at a higher level which can measure the analysis results and original system. Reverse engineering, process has been developed to analyze and modeling the OO files by designing the translator. It models and measures the object oriented by using traditional metrics and new encapsulation metrics (Public Factor (PuF) & Private Factor (PrF)) essential for developing the good software. In this work they tried to refine metrics especially for object-oriented programming and set of these metrics has been defined. Model proposed in this paper is performing the static analysis of CPP file. While analyzing it analyses two things first the structure of the program file (modeling) and then it measure the quality of the program with help of some new and traditional metrics. Firstly, structure of the program file was analyzed by translator by abstracting the packages used, classes and their relationship with other classes, functions and their definition, and variables encapsulated in functions, in classes and declared globally. While analyzing a file with translator, rules were devised to notate the analysis of files. In this efforts were made to model and structure an OO file so that it should be easy to understand. Secondly, quality of programs was analyzed with the help of metrics. In this paper effort has been made to measure the OO paradigms by set of metrics which include some pre-defined and new metrics. OO paradigms like inheritance was measured by Common Interchange/Intermediate Format (CIF) and number of ancestor classes (NAC), Information Hiding by Method Hiding Factor (MHF) and Attribute Hiding Factor (AHF), Class weight in program was measured by Weighted Class Complexity Factor (WCCF) where as Encapsulation and Abstraction was measured by new defined metrics i.e. PuF and PrF. From this discussion they want to conclude that MHF & AHF only measures the invisibility of the system but not the integrity or unity, which is also to be measured along with the invisibility as per the definition of encapsulation. This analyzes model helps in the way that programmer need not to start the work from scrap. Model gives a block model of the files from which devised concept and quality was revealed. If

concept is known then it is very easy to re-design the program and with quality factors (metrics) better quality software can be designed, but concept or quality revealed is not all about the OO. Object Oriented is much more than that. Another aspect which is not measured is design pattern which can help to prepare a better quality design. Another problem is it does not tell where the problem is and how to re-design it. Question can also be raise that is there any place for the software metrics in future? It has future but as long as standard metrics are not devised and threshold values are not defined. The above translator can be implemented to other types of OO languages also, to make it universal analysis tool. Translator has some limitations too, it does not support multiple source files which should make it more active and it understands well formed gaps only, **Singh et al. (2010)**.

Kalsing et al. (2010), have written the paper titled “An Incremental Process Mining Approach to Extract Knowledge from Legacy Systems”. Several approaches have already been proposed to extract both business processes and business rules from a legacy source code. These approaches contemplate static source code analysis for the extraction procedure. However, business processes have elements that cannot be directly extracted by static analysis (i.e., participants, responsibilities, and coincidental activities). As, well-known static analysis algorithms do not support the progressive extraction of data from the legacy code, massive legacy systems will have the benefit of a progressive analysis strategy so as to supply repetitious data extraction. This paper discusses a new approach for business knowledge extraction from legacy systems. The method considers an incremental process mining technique to extract business process structures and the business rules associated to it. Discovery results can be used in multiple ways by business analysts and software architects, for example, documentation of legacy systems or for re-engineering purposes. This paper proposed an incremental process mining approach for the extraction of business process models from event logs of legacy systems. This approach used the Incremental Miner algorithm, an incremental process mining algorithm that extracts business process structures from execution logs. The execution logs have shown important information of business process like task participants and concurrent activities, which were not present in the models identified in previous research. Other important aspect of using process mining is the reduction of source code syntactic dependency. That is, control flow structures (e.g. if-then-else, switch, return, goto, etc) and parallelism (e.g.

use of threads by languages like C++, Java, etc) are easily processed by these techniques. Moreover, they have shown that an incremental approach can be useful during the identification of business processes in large legacy systems, allowing the extraction of partial results (i.e. analysis by module) and significantly reducing the total processing time. Altogether, the main contribution of this work is the use of incremental process mining techniques to support incremental learning of business process models. The use of process mining has also enabled the extraction of important process structures (i.e. participants and concurrent activities), available only in the dynamic behavior of system. They are currently working on some improvements of their approach. The first one concerns the possibility of extracting collaborative process from the legacy. Collaborative processes contain tasks associated to different human participants, where every participant collaborates in the accomplishment of work. The second improvement concerns the static analysis of legacy source code. They are studying additional techniques to improve the extraction of business rules, as the extraction of the data flows related to it (i.e. all input and output data related to a business rule). The third improvement is the extraction and integration of decision conditions (i.e. control flow conditions) to the final process model. This data can be obtained using decision mining techniques, **Kalsing et al. (2010)**.

Anquetil et al. (2010), have written the paper titled “Legacy Software Re-structuring: Analyzing a Concrete Case”. The software re-modularization is an old preoccupation of reverse engineering research. The benefits of a well structured or modularized system are well known. Yet after so much efforts and time, the field seems incapable to come up with solutions that make a clear difference in practice. Lately, some researchers started to question whether some basic assumptions of the field were not overrated. The important one consists in evaluating the high-cohesion/low-coupling dogma with metrics of unknown relevance. In this paper, they study a real structuring case (on the Eclipse platform) to try to better understand if (some) existing metrics would have helped the software engineers in the task. Results show that the cohesion metrics and coupling metrics used in the experiment did not behave as expected and would probably not have helped the maintainers reach their goal. They also measured another possible re-structuring which is to decrease the number of cyclic dependencies between modules. Again, the results did not meet anticipations. In these experiments, they studied the evolution of different metrics that purport to evaluate the quality of a modularization. The traditional view of the

problem is that re-structuring a system should increase the cohesion of the modules, and decrease their coupling. Another valid goal for a re-structuring is to remove cyclic dependencies between modules which would tend to also reduce their coupling (cyclic dependencies are removed by breaking some dependency, thus reducing the coupling of at least one module. they measured coupling between Java packages and Eclipse plugins with two different coupling metrics: Bunch's cohesion/ coupling and Afferent/Efferent coupling (resp. Ca/Ce). All metrics gave results that did not match expectations: After each of two re-structurings, the Bunch cohesion did not increase and rather decreased, and efferent coupling, Ce, increased. If they accept that the restructurings studied were reasonably successful, which is a plausible hypothesis given the continuing success of the Eclipse platform six years later, they remain with various possible explanations:

- The metrics themselves are flawed, or they do not adequately measure the modularity of systems;
- The cohesion/coupling dogma is flawed, re-structuring does not generally improve modularity in terms of cohesion/ coupling;
- There may be different types of re-structuring that would have different properties, and here they would be in the presence of re-structuring that did not improve the modularity of the system.

They already showed that the decreased in Bunch's cohesion and coupling was caused mechanically by a property of the metrics' formula rather than by a property of the system's modules themselves. Therefore these particular metrics did present some kind of flaw. However that does not apply for Ce that did not either give the expected results. Because Ce is a very simple metric, it is difficult to imagine that it has some other hidden flaw. Other measurement with other coupling metrics could still be conducted to bring more light on this issue. A second unexpected result was that cohesion and coupling (as measured with the Bunch metrics) do not evolve in opposition (one increasing when the other decreases), but presented the same trend (decreasing in their experiments). The third unexpected result, was that restructurings did not either improved the situation in regard to the number of cyclic dependencies between the Java packages, and finally, a fourth unexpected result was that they could find no clear differences between re-structuring evolutions and "normal" ones. They favor the hypothesis that the cohesion/coupling

dogma, as measured by the existing cohesion/coupling metrics, is largely over-rated. This does not automatically mean one should not zeal for high cohesion/low coupling when developing or re-structuring systems, but rather that they don't have any automated metrics that can measure the kind of cohesion and coupling the software engineers try to optimize in their development and re-structuring work. One must consider that software engineers work with higher level concepts that cannot be measured by the simple, existing, cohesion/coupling metrics, **Anquetil et al. (2010)**.

Carchiolo et al. (2010), have written the paper titled "Object-oriented re-engineering of manufacturing models: A case study". According to this paper, the current manufacturing systems have a very structured production model, particularly when high complexity and precision is required, as in the manufacturing of semiconductor devices. In addition, rapid changes in both production and market requirements may occur, hence great flexibility is essential. These goals, together with the capability of adding new features as well as removing some model limitations, usually impose to re-engineer periodically existing models, decreasing as much as possible the time-to-market of a new product. Updating a model in general also requires so as to improve existing applications, both adding new features to existing components as well as re-writing software components. Currently the model used inside STMicroelectronics facilities to define production flow (the sequence of operations to be performed in order to make products) is considered. The model at first is described highlighting its limitations, and then an improved object-oriented model is introduced, in order to provide both a best management of all entities through aggregational and constitutional hierarchies and a significant time reduction of production flow definitions through a flexible inheritance mechanism. Also, required improvements on existing applications are considered, producing an enhanced environment capable of supporting the enhanced model while preserving both technical and economic investments thus avoiding radical changes in existing environment. In this paper, they presented the re-engineering of the production flow model used inside STMicroelectronics facilities using an object-oriented approach, also introducing related environment enhancements. They first introduced current model used for specification of the production flow for Integrated Circuit (IC) manufacturing, starting from an overview of the entire ST production model, then focusing on specific phases concerning with the production flow specification, introducing the facility-products-lots aggregational hierarchy, as well as products

classifiers (P&L family, technology and process families) and product operation, events constitutional hierarchy. They then highlighted main limitations of current model, moving towards an object-oriented model, where a new aggregational hierarchy with explicit products classification is introduced, together with an object-oriented operation management, in which script object management is improved through removing the current separation between operation and script. Then core objects, i.e. process, family and product, have been considered in detail. In particular, they introduced a powerful inheritance mechanism between process family and product through which product setup time and costs are significantly reduced. They also outlined three categories of operations, production, optionality and monitoring, in order to provide the production flow specification with a more powerful expressiveness (e.g. if...then...else construct can be defined). A three steps specification (type, template, default) for production operation was introduced, improving the flexibility of the inheritance between process families and products, since the degree of inheritance for production operation can be set (i.e. what part must be inherited). Type, template and default also significantly improve operations setup, since events have not to be separately specified, each time a new operation is defined, as it is currently required, rather event groups can be selected from type-template default libraries and then used to rapidly specify a script inside an operation. They then introduced conditions and expressions support for optionality operation, defining off-line, on-line, and human-dependent expressions. Such expression types are needed to model existing environment (current model does not support such features). Moreover, a conditional inheritance mechanism between process families and products is provided through off-line expressions, allowing to define more and more production flows in a single design phase. The description of the enhanced model was completed by examining finite state machines for process family and product objects to show their management. In particular, conflicts issues are considered when a new object is created by copy, then the updating state was analyzed, starting from defining modifications from low-level (events) to high-level (process families and products) objects, also considering online questions, introducing lots sensitivity and temporary state needed to manage updating actions while production is running. They introduced the architecture of an enhanced environment, placed on top of the existing one, and consisting of a Java-based application and a support database. Such framework is capable of supporting the enhanced model, offering a user-friendly interface, and preserving at the same time both technical and economic investments since no radical

changes are required in existing environment. Their re-engineering project is currently being implemented in Catania STMicroelectronics facility and they are collecting response data in order to quantify how productivity is improved through relieving users of current errorprone and wasting-time works. Evaluation of productivity improvement requires a medium-long term analysis, since a relevant factor lies in personnel training, so migration towards new model may require a certain time to reach a stable state in which time and costs are really reduced, **Carchiolo et al. (2010)**.

Dugerdil and Repcnd (2010), have written the paper titled “Automatic Generation of Abstract Views for Legacy Software Comprehension”. One of the main motivations for the reverse engineering of software programs is to help with software comprehension. Although multiple techniques have been presented in the literature to reverse-architect software, the corresponding views often do not help much. In reality, most of the published techniques recover the architecture of the software by focusing on the abstract properties of the components such as coupling and coherence. They claim that the recovered components should rather represent abstract functional entities whose behavior could be understood independently from the others. Then, an abstract point of view of the system would represent the interactions between such functional entities. In this paper they present a technique and a tool able to generate abstract sequence diagrams to represent the global working of legacy programs. This displays the main interactions between abstract functional components. When matching the automatically generated sequence diagrams to the one a developer would produce by hand. They realized that the representation were very close. Their work could then be considered as a first step to the automatic generation of human-understandable abstract views of the working of legacy programs. In this paper they have showed that their software clustering technique, which is based on the dynamic analysis of the methods called when executing a scenario, allow them to automatically build an "abstract" view of the working of a system. In particular they are able to show that the main steps of a complex processing have been automatically revealed in a sequence diagram. Moreover, the generated sequence diagram seems to be close to the one they would actually draw by hand to explain the working of the system. Although this important feature needs be confirmed by more experiments it is very encouraging. This result has been obtained because of their very dynamic analysis technique coupled with

the automatic generation of sequence diagrams. Here again are the main steps of their approach:

- An execution trace is generated when executing some scenario of business value (use-case). They only keep the events between application classes.
- The clusters that are identified are strongly linked to business functions. In a sense they represent chunks of elementary functionality;
- The working of the system is abstracted through displaying only the method called between the clusters.
- The resulting set of event is further compressed by using event patterns repetition compression.

Their cluster identification technique is to be contrasted with the ones that try to identify the components based on some software engineering metrics such as high cohesion and low coupling. They are not targeting the recovery of any structural view of the software. Rather, they wish to generate a view that would support the understanding of software behavior. Therefore the conceptual grouping of classes as well as the analysis technique cannot rest on static analysis techniques. With the latter, there is no hope that the component will represent steps of some complex behavior related to the business function. The comparison of the dynamic clusters with the static structure of the program (for example its packages) is useful to show where the responsibilities pertaining to some behavior are actually implemented. The criteria that lead to the static structure of the code are usually not based on the working of scenarios; therefore, there little hope to end up with a set of functional components that would match the packages of the source code. Their technique has been integrated in the IBM Rational Software Architect (IBM's RSA) environment as an Eclipse plugin. The key contribution of this paper is to show that the automatic generation of an abstract view of a legacy system is indeed possible, **Dugerdil and Repcnd (2010)**.

Laval et al. (2010), have written the paper titled "Supporting Simultaneous Versions for Software Evolution Assessment", when re-engineering software systems, maintainers should be capable to assess and compare multiple change scenarios for a given goal, in order to choose the most pertinent one. Because they tacitly consider one single working

copy, refinement control systems do not scale up well to perform simultaneous analyses of multiple versions of systems. They designed Orion, an interactive prototyping tool for re-engineering, to simulate changes and compare their impact on multiple versions of software source code models. They deal with requirements such as: interactive simulation of changes, re-use of existing assessment tools, and capability to hold multiple and branching versions simultaneously in memory. Specifically, they devise an infrastructure which optimizes memory usage of multiple versions for large models. This infrastructure uses an expansion of the FAMOOS Information Exchange Model (FAMIX) source code meta-model but it is not limited to source code analysis tools since it can be applied to models in general. In this paper, they validate their approach by running benchmarks on memory usage and computation time of model queries on large models. Their benchmarks show that the Orion approach scales up well in terms of memory usage, while the current implementation needs enhancement to lower computation time. They also report on two large case studies on which they have started to work with Orion. They present a novel approach to re-engineering through simulation of changes in source code model. In particular, they claim that software maintainers should be able to assess and compare multiple change scenarios. They define user and technical requirements for an infrastructure supporting their vision, ranging from re-use of existing tools to handling simultaneous versions in memory. They implemented the requirements in Orion, their current infrastructure. The main concern in this paper is the efficient manipulation of simultaneous versions in memory of large source code models. Copy process does not scale up in memory for such models. In Orion, only modified entities are copied between versions, unmodified entities being effectively shared. Then, basic queries look after retrieving a consistent view of entities in the analyzed version. Their benchmarks report the large gain in memory for their approach with an acceptable overhead in query running time. Overall, it lets Orion to scale up and be usable. They have started to use Orion on two large case studies, **Laval et al. (2010)**.

Nagy et al. (2011), have written the paper titled “Solutions for Reverse Engineering 4GL Applications, Recovering the Design of a Logistical Wholesale System” and they said that: Re-engineering a legacy software system to support modern, new technologies instead of old ones is not an easy task, particularly for large systems with a complex architecture. The usage of reverse engineering tools is crucial for different subtasks of the full process, such

as re-documenting the old code or re-covering its design. There are many tools available to assist developers, but majority of these tools were designed to deal with third generation languages (e.g. Java, C, C++, C#). However, many large systems are developed in higher level languages (e.g. Magic, Informix, ABAP) and current tools are not able to support all the arising problems during re-engineering systems written in fourth generation languages. In this paper they present a project whose main goal is the development of a technologically and functionally renewed medicinal wholesale system. the medicinal wholesale system is developed in Magic 4GL, and its development is depend on re-engineering an old Magic (version 5) system to uniPaaS, which is the current version of Magic. In the early phases of this project they developed a reverse engineering toolset for Magic 4GL to support reverse engineering, re-covering the design of the legacy system, and to support some forward engineering tasks too. Here they present a report on this project that was carried out in cooperation with SZEGED Software Zrt and the Department of Software Engineering at the University of Szeged. The project was partly funded from the Economic Development Operational Programme, New Hungary Development Plan. their primary goal was to re-engineer a complex Magic 4GL application. Besides the full re-engineering process, in this paper they focused on the reverse engineering steps. In these steps they made the following contributions. They have successfully adapted and tailored for 4GLs the Columbus RE technology, which was originally invented for object oriented languages. Using this technology they developed a methodology for re-covering the design of legacy 4GL applications. This methodology defines architectural views which reflect the program structure, semantics and its relations to the underlying databases. Their methodology was considered useful by assisting program comprehension and re-engineering tasks of Magic 4GL applications in a successful project which was partly funded by EU. A remarkable outcome of their work is the tool-chain, which they plan to use for offering services in further projects such as migration, maintenance and program comprehension of Magic programs, **Nagy *et al.* (2011)**.

Bartholdt and Becker (2011), have written the paper titled "Re-Engineering of a Hierarchical Product Line" and they said that: Transition from a classic platform re-use approach towards a product line engineering paradigm for existing products is demanding. Existing software design, a large amount of code, a determined development process, organizational setup of teams and the mindset of involved people must be considered. All

of these factors make the transition challenging compared to a green-field approach, where re-usable assets are already developed in a product line spirit. They describe their experience during transitioning to a product line approach in the context of a healthcare imaging platform from the viewpoint of responsible domain engineering architects. The primary focus is on re-engineering activities to introduce product line engineering practices to an existing legacy system. Common product-line challenges that are not related to re-engineering are omitted in this paper. They recognize additional benefits when applying a hierarchical product line approach in their domain. They conclude with the factors that impact a successful transition to a product-line approach. Applying product-line engineering techniques for their platform seems to be promising. Explicit management of variation points, a collaborative development strategy and consistent governance across platform and product development help eliminate inefficiencies in both projects: platform and product. Still, general software engineering activities are necessary – they have focused here on PLERelated topics only. The benefits in reverse engineering in many situations beyond product line engineering aspects. It uncovers the interdependencies of decisions taken over the project lifetime. These decisions should finally be revalidated in the light of today's system usage. Accurate architecture specifications shorten the collection of decisions and associated forces, but they cannot substitute this reflection step. During transition the product-line must continuously be running and new releases must be delivered. Therefore a migration path, realizing smaller pieces with immediate benefit, must be defined. Re-engineering efforts will compete with feature requests, thus a balance between long-term and short-term benefits must be found, **Bartholdt and Becker (2011)**.

Banodha and Saxena (2011), have written the paper titled “A Software Architecture Style for Medical Process Re-engineering” and they said that: Software Architecture has been recognized both by Academia and software industries as the most promising approach to tackle the problems in the era of Finance, Medical, Communications, Run-time applications etc. Software architecture style is a classification of SA. Software architecture style made the communication more precise and convenient at the level of SA. Different style has different system characteristics. In this paper they propose to model the architecture of medical process Re-engineering represented as instance of UML class diagram based on Service-Oriented Architectural style which is still lacking in the field of medical Domain. They describe architecture of medical process re-engineering model

which provides patients, clinical staff and other individuals with knowledge and person – specific information, filtered intelligently and presented at appropriate times to enhance health and health care with the concept of using software re-engineering. The Re-engineering process helped to improve understanding about the processes and led to the conclusions with deletion of subsequent improvements to those processes. They discuss the properties that are interesting to be analyzed with the process of analysis and re-engineering. The paper presents the impact of architectural styles on the medical domain with the re-engineering concepts. The current work is on experimenting different solutions to express constraints, rules, and control mechanisms depend on the knowledge base to find the right balance between expressiveness and analyzability with the applicability of re-engineering on various Medical Process Re-engineering (MPR) components. Rules can be extended to address adaptability and the capability of full or partial automatic recovery in the complete process. If they consider modern scenarios where applications are ubiquitous and they must adapt their behavior to the context in which they are executed, a re-engineering approach to modeling these aspects is essential. Rules offer a clean and neat way to specify how the architecture should react to the different scenario with the analysis capabilities as model checking and simulation; complement the design with the capability of automatic reasoning and predicting the behavior of specified architectures. In a similar way, rules can specify the self-healing capabilities of MPR components which are associated with the way in a specific style or family of architectures styles, **Banodha and Saxena (2011)**.

Nguyen (2011), has written the paper titled “The Study and Approach of Software Re-engineering” and he said that: The nature of software re-engineering is to improve or transform existing software so it can be understood, controlled and re-used as new software. The importance of re-engineering software has greatly increased. The system software has been neglected and no longer used in architecture, platform they are running, consistent and stable they support the development and support needs change. Software re-engineering is necessary to restore and re-use the things inherent in the existing software, put the software maintenance cost to the lowest in the control and establish a basis for the development of software in the future. As the software industry moves to a new millennium, many software designers in developing new methods, improved software and reduce development time and maintenance. However, most companies have legacy

systems that are overdue and save costs to maintain. The system cannot only be replaced by the new system, which contains company information and implied that would be lost. They are an investment, and too expensive to develop and evolve just to remove. For re-engineering to become a useful tool that can convert old, outdated systems to the more efficient, logical system. However, development projects often short of time and money, making the need to look at the replacement necessary. The use of common-off-the-shelf (COTS) package is considered a way to increase reliability while reducing development and testing time. Translation as a means of reducing the time and cost. This results in a combination of measures to develop a form of re-engineering. Re-engineering is structured in software development and different approaches and specific stages and tasks. However, extensive work is still needed. Although it is noted that the system administrator is very important, no method for incorporating them. There is not an approach for evaluating the quality of the new COTS system. Data are also needed during the re-engineering, but it is not clear exactly how the concept can be considered and expected that the actual data, evaluate what they are proposing to define quantify. Paper is a summary of the strategies and techniques in re-engineering, to serve as a basis for future work, **Nguyen (2011)**.

Abdul-Bari and Ahamad (2011), have written the paper titled “Process of Reverse Engineering of Enterprise Information System Architecture”, in which they discuss about the availability of computer-aided systems-engineering environments has re-defined the organization system development. Pace of change accelerates in the twenty-first century as a result of technological opportunities, liberalization of the world markets, demands for innovation and continuous decreasing life cycle of software. Organizations have to continuously re-adjust and re-align their operation to meet all these challenges. This pace of changes has increasingly forced organization to be more outward looking, market-oriented and knowledge driven. This paper present integrated framework of how reverse engineering process work goes in Enterprise Information System (EIS).The essential functions in reverse engineering, how they are associated with EIS, what will be the impact on the organization which is using reverse engineering. The field of IT support system has moved away from stand-alone, dedicated solution with localized impact to more integrated, flexible enterprise-wide system, a fresh approach was needed. In essence enterprise architecture brings with it. Not only it addresses organizational business change perspective but also support the software configuration. This paper has provided a frame

work for examining reverse engineering process with the help of Enterprise architecture. Reverse engineering is rapidly becoming a recognized and important component of the future software environments. It provides a major link in the overall process of software development and maintenance. Reverse engineering, used with evolving software development will provide significant enhancement to new software product. The proposed architecture of reverse engineering (which has taken enterprise architecture in to account also) human expertise is used on well defined occasion to overcome statement situation because of hybrid nature and goal oriented approach. The most promising direction in this area is the continuous software understanding approach. The premise the software reverse engineering need to applied continuously throughout the lifetime of the software and it is important to understand the potential reconstruct the early design and the architectural decision. For the future it is critical that they can effectively answer questions such as: how much knowledge, what level of abstraction do they need to extract from the software, to make informed decision about re-engineering. It will be not ever able to predict all needs of the reverse engineering and their fore, must developed tools are the end user programmable. Software architecture is now established in many computer science course but the topics such as software evolution, reverse engineering and software migration are rare, **Abdul-Bari and Ahamad (2011)**.

Behera and Tiwari (2011), have written the paper titled “Analysis and Design of Software Visualization Tool for the Behavior of Object Oriented Programming”, in which they say that the software visualization tools are highly required by the industry and research centers. In past, many specific tools have been developed, whose proportion of development does not match the rate of requirement of industry. In this paper, they discuss the analysis of the similar tools available and design & development of their Software Visualization Tool, which focuses the class diagram with attributes and methods of class of object oriented programming. They design the common algorithm for visualization of object oriented program. The aim of their tool is to simplify the design of complex code, which can be easily maintainable and usable. This paper presents a visualization design of object oriented software with focus on the inheritance of properties. This provides actual design of program, flow of data and methods from one class to the other. This paper describes the concept of the tool and the techniques used in it. The goal of these systems is to provide multiple object-oriented environments. This tool is already implemented in their

research lab. This is mostly helpful for research students and industries used C++ programs. As per the testing concern, it can extract small and medium scale programs and provide the required output. They will further enhance this tool for large scale programs, **Behera and Tiwari (2011)**.

Fontana and Zanoni (2011), have written the paper titled “A tool for design pattern detection and software architecture reconstruction” and they said that: The Replacement of a legacy system with newly re-engineered information system has always been a challenging task. The end-users of the system and the integrated software environment of legacy system make things harder for the deployment team. The legacy data migration face complicates the very important deployment phase. In this paper, they have discussed and analyzed three different strategies to deploy a newly re-engineered information system and replace the legacy system in a homogeneous environment. This paper fundamentally addresses the issues concerning minimum disturbance to the legacy environment and smooth deployment of the new information system. It is relevant to mention that the issues regarding the development of re-engineered information system or the legacy data migration have not been discussed here. At the end they have provided a detailed analysis and finally recommended the most optimal strategy among all. The rationale behind the proposed solutions is an extensive study conducted at their research and development Lab. In this study three deployment strategies of re-engineered information system in the context of legacy systems have been discussed in detail. The outcomes of all the strategies were explained in sufficient details stating the pros and cons of each deployment strategy. At the end, two applicable strategies were evaluated with respect to three common core criteria parameters. They also tried to present a rationale that could help in selecting and adopting the more adequate and suitable strategy between the two. After a detailed evaluation work they recommend the strategy B as the most optimum and stable strategy, because it not only causes minimum disturbance to the legacy environment but also requires relatively little effort to implement. Here they also propose that adopting a hybrid strategy means the amalgamation of the two strategies being discussed could be an area of potential research. A new strategy could be introduced in which both the strategies reinforce each other by using the advantages of one another, **Fontana and Zanoni (2011)**.

Tarar and Kumar (2012), have designed the Hybrid re-engineering paradigm. Hybrid re-engineering is a re-engineering process that uses not just a single, but a collection of

abstraction levels and alteration methods to transition of an existing system to a target system. In order to do the hybrid re-engineering, the re-engineering model is mapped on with the software component library that can be Commercial off the Shelf (COTS) or Math Description Engineering (MDE). Because hybrid re-engineering uses COTS or MDE libraries through which design and requirements can be specified much faster thereby reducing effort, time and increasing reliability. COTS have some risk associated with it. In other words, a package will not perform an anticipated or advertised, or that will be unreliable, incomplete or immature. Also COTS product may limit further enhancements to the system because changes in COTS provided functions may not be possible due to legal or contractual issues. All the main principles of hybrid re-engineering approach namely re-specify, re-design, re-code & re-test the main task remains of integrating all the work of these principles together to make an effective model or system. Any specific principle is useless unless there is no integration with the other. The Hybrid model is shown in Fig. 2.18, **Tarar and Kumar (2012)**.

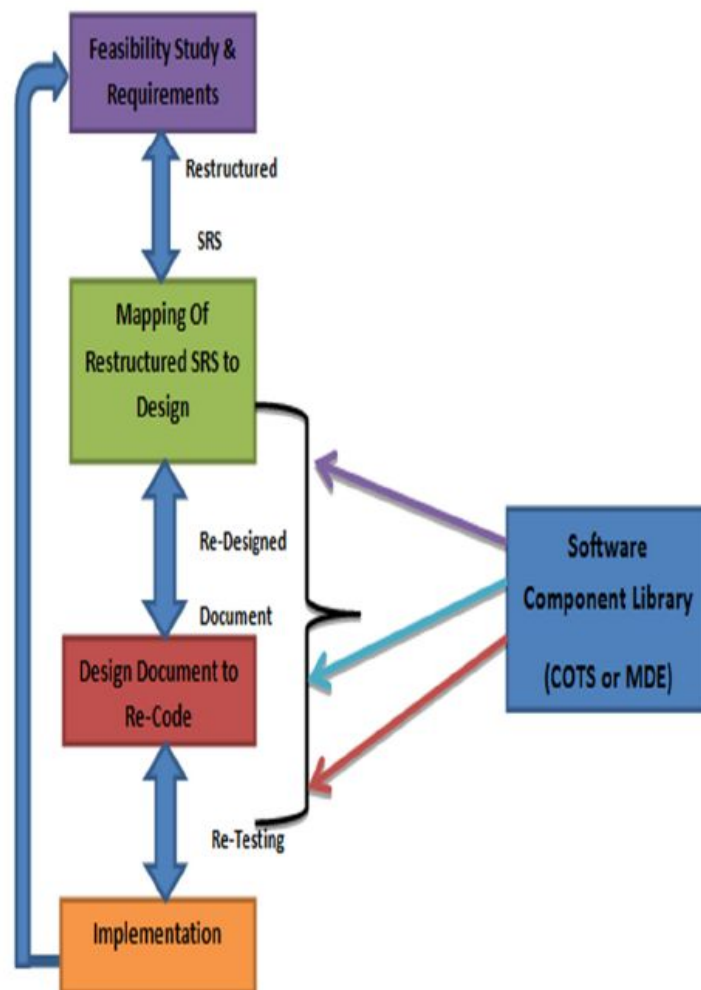
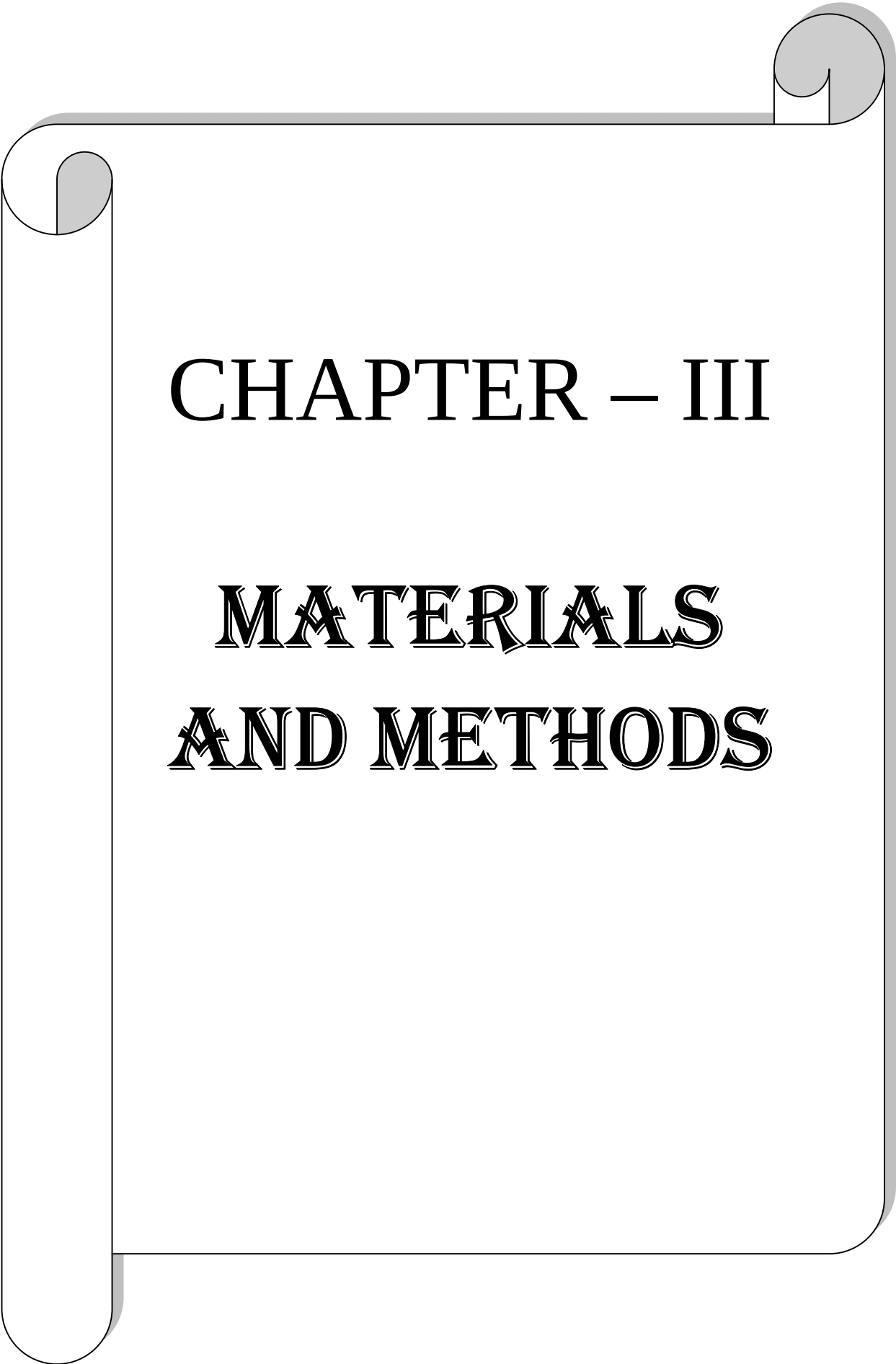


Fig. 2.18: Hybrid Re-Engineering Model.

2.10 Conclusion

From this literature review the need to suggest new better model for re-engineering has been identified.



CHAPTER – III

MATERIALS AND METHODS

CHAPTER 3

MATERIALS AND METHODS

3.1 Hardware and software tools

The proposed software re-engineering model that named as FUSION model is the re-engineering model that consists of set of activities to convert a legacy system to the target system with high quality and more functionality. The required tools to apply FUSION model on the legacy software are as follows:

3.1.1 Tools and platform used

3.1.1.1 Supermarket management system (Legacy system)

Legacy supermarket management system is a window-based application. It was developed using visual basic 6 programming language. The main window of the legacy supermarket management system is shown in Fig. 3.1.

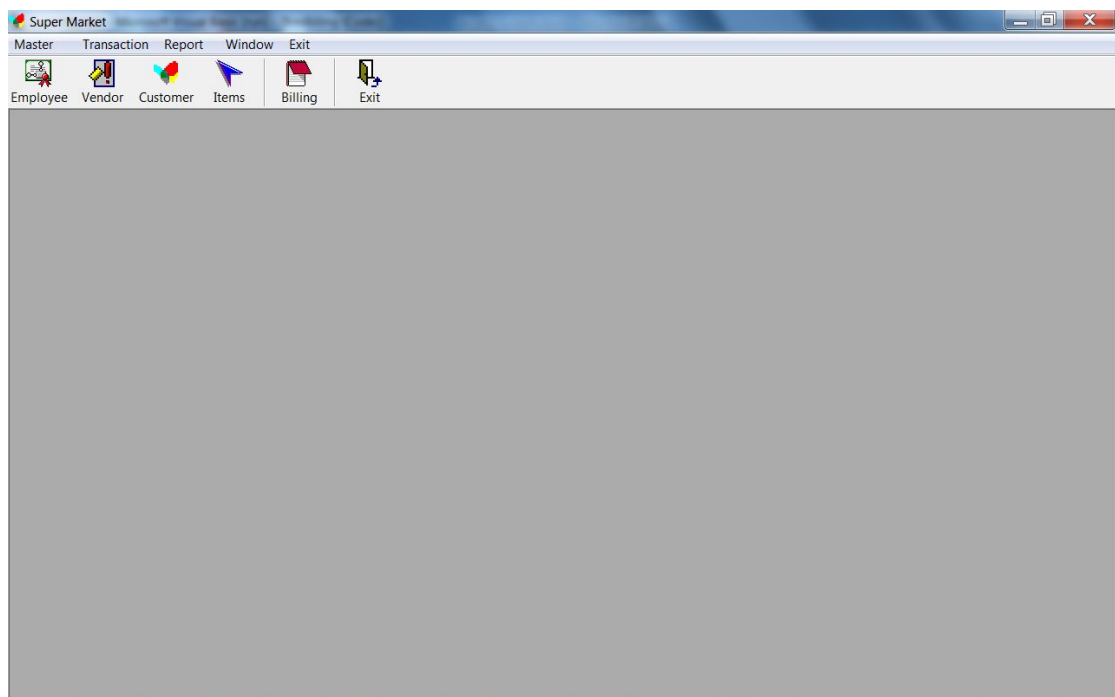


Fig. 3.1: Main window of the legacy supermarket management system

3.1.1.2 Visustin v7

Visustin v7 is designed to visualize procedural code of any complexity. Visustin works even with true spaghetti code and unstructured jumps. It works well with executable code,

but Visustin can also consume non-executable declarations. Visustin generates flow diagrams for the many programming languages as: ABAP, ActionScript/MXML, Assembly languages (MASM, NASM, IAR/MSP430), BASIC: Visual Basic, VB.NET, VBA, VBScript (+ ASP), BASIC: LotusScript, PowerScript, PureBasic, QuickBASIC, REALbasic , Batch files, C/C++, C#, Java, JSP, PHP, Clipper and Visual FoxPro, COBOL, ColdFusion, Fortran, JavaScript (+ASP), JCL (MVS), Matlab, Pascal/Delph, Perl , PL/I and SAS, PL/SQL and Ada, Python, Rexx, Ruby, Shell script: sh, csh/tcsh, ksh, bash, Tcl, T-SQL, and XSLT, Pseudocode.

Visustin v7 is an analysis tool which has been used to analyze the legacy supermarket management system. Visustin converts source code to flow charts and UML activity diagrams. This tool is used to understand existing code, debug algorithms and review, verify program logic, document complex functions, and to re-structure incomprehensible code. The main window of Visustin v7 analysis tool is shown in Fig. 3.2. This tool can run on Windows 2000/XP/2003/Vista/7/8.

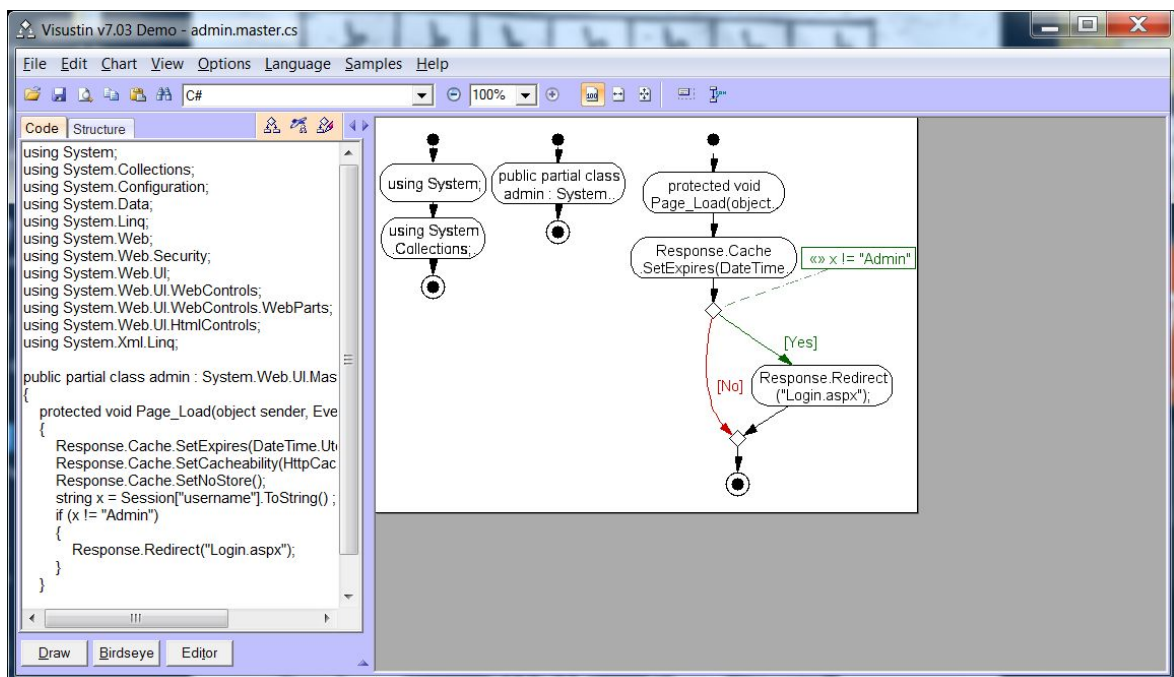


Fig. 3.2: Main window of Visustin v7 analysis tool

3.1.1.3 ASP.NET

ASP.NET with C# is used to develop the re-engineered supermarket management system, where ASP.NET is a reworking of the original Active Server Pages technology that was introduced about ten years ago. ASP.NET is Microsoft's web server technology, now based

on the .NET framework and up to its 4th incarnation. Old releases were 1.0, 1.1, 2.0 and now 3.5. Usually a web server delivers html pages. Web server technology as ASP.NET actually executes code on the server, code that can use databases and then produce html to the browser. It also keeps the code and HTML in separate files so that designers can work on the look and feel and developers on the functionality without tripping each other up. ASP.NET lets a developer to create complex and sophisticated web server controls that speed up development of web sites. It is probably the most advanced web server development technology that currently exists and certainly the cheapest but it's not as easy to learn. The main window ASP.NET of visual studio 2008 is shown in Fig.3.3.

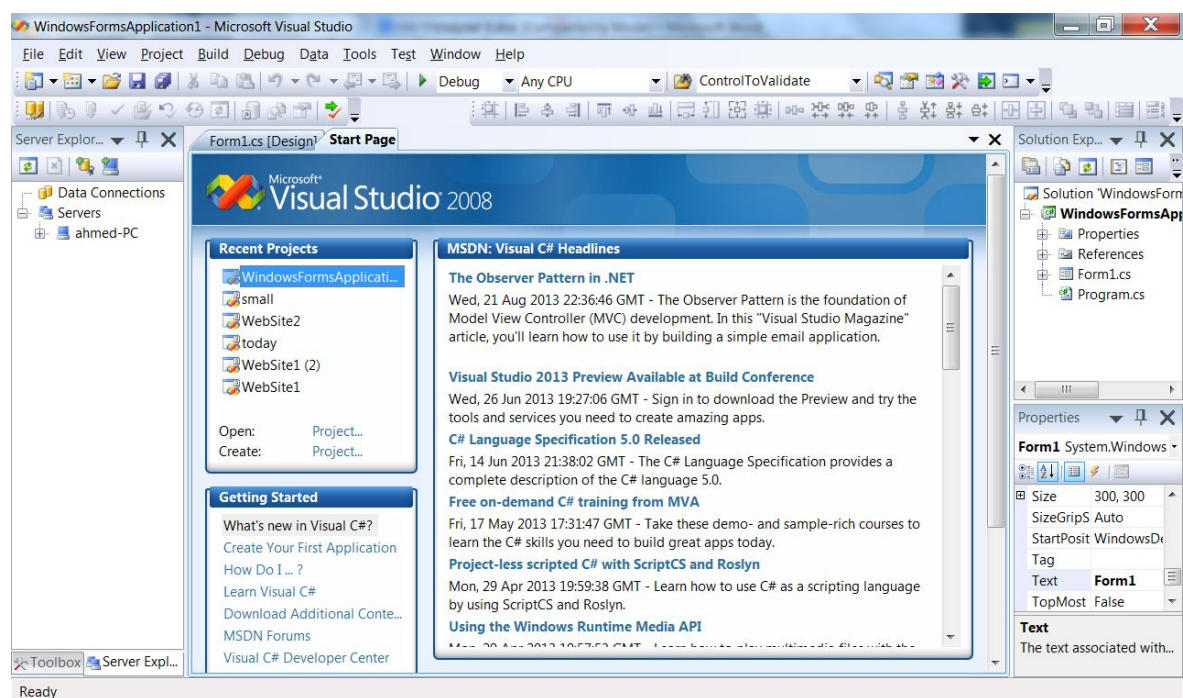


Fig. 3.3: Main window of ASP.NET of visual studio 2008

3.1.2 Operating System:

- The legacy software works on windows environment WIN 9X\NT\ME operating system.
- The re-engineered software works on windows environment WIN 9X\NT\ME and Linux operating system.

3.1.3 Database Tool:

- In the legacy supermarket management software, Microsoft SQL server was used to develop the database. This developed database is consisting of seven tables.

- In the re-engineered software, Microsoft SQL server 2008 was selected to develop the database. The main window of Microsoft SQL server 2008 is shown in Fig.3.4.

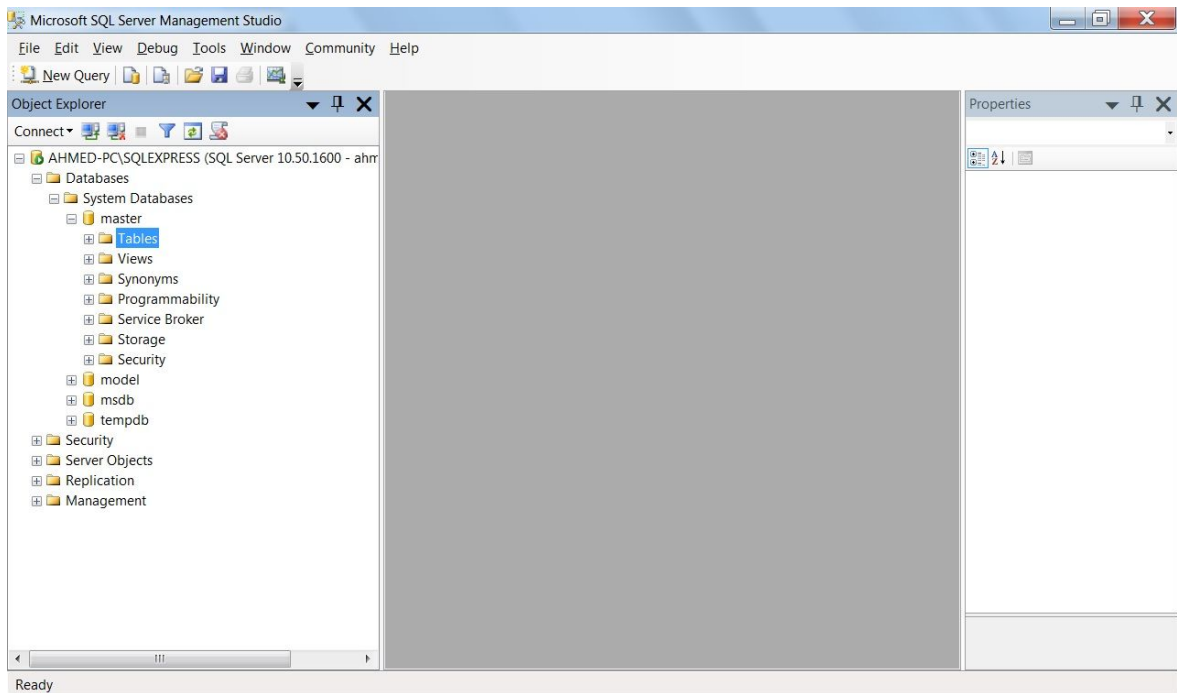


Fig.3.4: The main window of Microsoft SQL server 2008

3.1.4 Front-end tool:

- Visual Basic was used in the coding of legacy supermarket management system. It used to make front end for the application. The main window of Visual Basic is shown in Fig.3.5.
- ASP.NET was used in the coding of the re-engineered supermarket management system. It was used to make front end for the application.

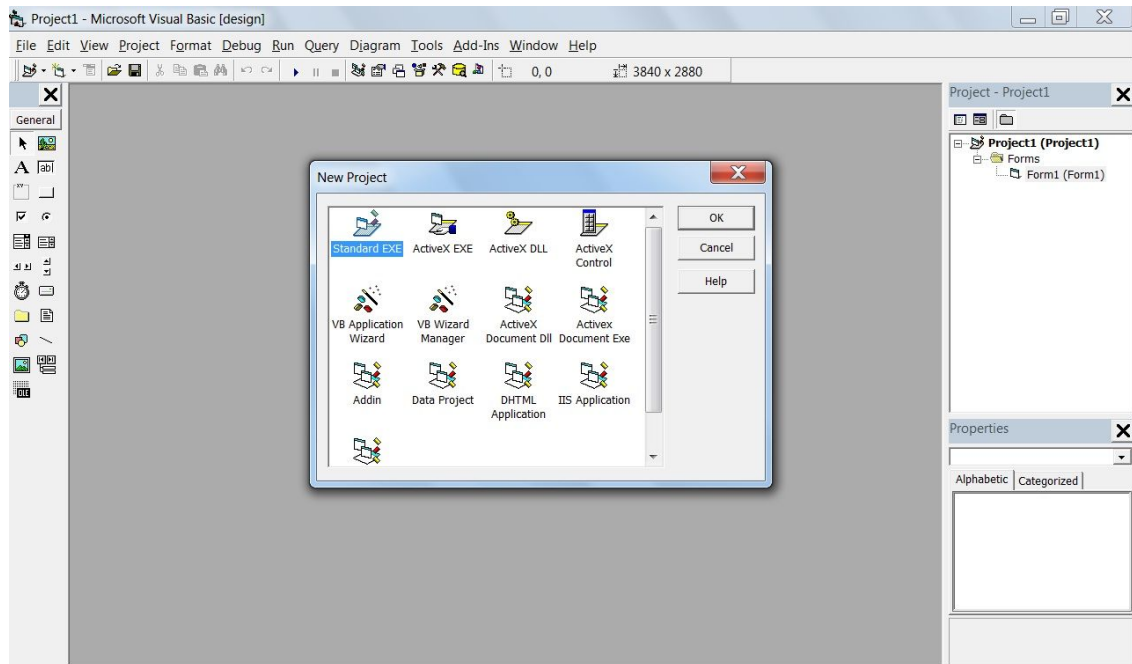


Fig.3.5: The main window of Visual Basic

3.1.5 Hardware and software configuration:

3.1.5.1 Hardware used:-

- Model: Acer Aspire 4930.
- Intel(R) Core(TM)2 Duo CPU T5800.
- With 300 GB Hard Disk.
- RAM: 3.00 GB
- 104 Key Standard Keyboard
- Canon LBP2900 Printer
- Internet connection.

3.1.5.2 Software configuration:-

The new re-engineered system will run on windows and web based environment with WIN 9X\NT\ME\Vista\7\8 or Linux. SQL server is used for database.

3.2 FUSION re-engineering model

The FUSION model is the re-engineering model that consists of set of activities to convert a legacy system to the target system with high quality and more functionality. The FUSION model performs not just the six activities of traditional re-engineering process,

but a combination of these activities and a set of framework activities of the spiral model to perform the transition of an existing system (legacy system) to a target system (new system). As shown in the Fig. 3.6, there are eight task regions, which were applied by the software engineering team on legacy software to get the target re-engineered software. These activities are: identification of problems and suggested optimization of legacy software, planning, reverse engineering, document re-structures, code re-structures, data re-structure, forward engineering, and deployment. The eight task regions are shown in the Fig. 3.6 and they are very necessary to re-engineer the legacy software. The FUSION model integrates the iteration nature of spiral model with traditional software re-engineering process, **Abbas et al. (2013)**.

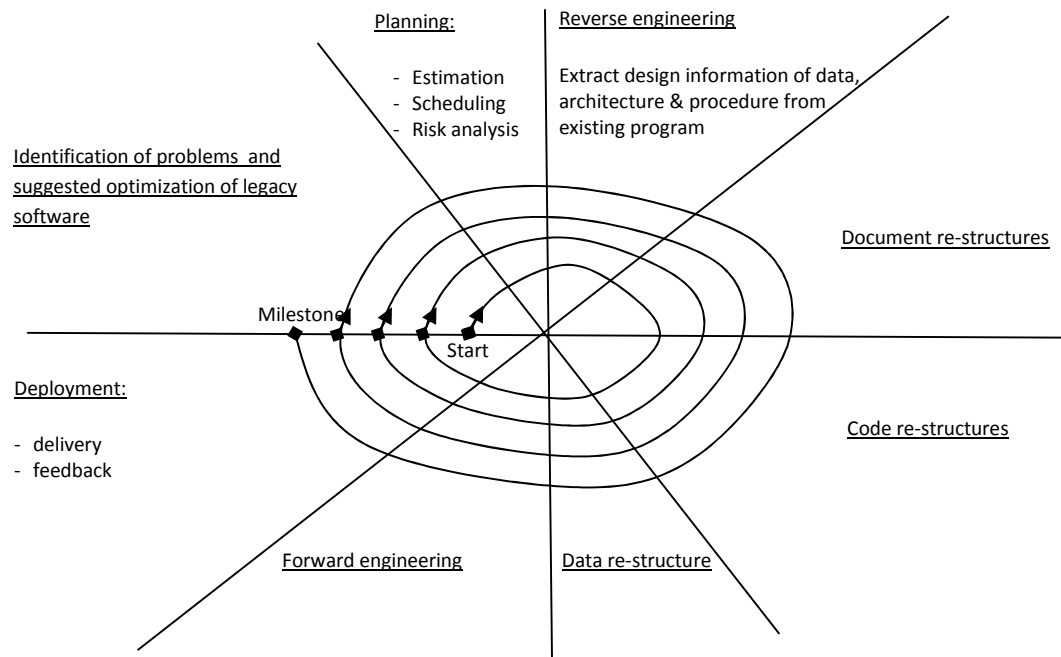


Fig. 3.6: FUSION model

3.3 The main activities of the FUSION model

As shown in the Fig.3.6, there are eight activities, which were applied by the software engineering team on legacy software to get the target re-engineered software, **Abbas et al. (2013)**. These activities are the following:

1. Identification of problems and suggested optimization of legacy software: This is the first step in the proposed re-engineering model. Within this step all errors and bugs that

causes problems in legacy software will be determined, new requirement will be specified according to customer opinion, new required facilities will be added, the feasibility study will be conducted, the Mean Time To Maintenance Request for the i-th component (MTTMR_i) will be calculated by the following formula:

$$MTTMR_i = \sum_{j=1}^n \frac{\Delta t_{j,i}}{n_i} \dots\dots\dots \text{Bianchi et al. (2001)}$$

, where:

N_i: is the total number of requests for maintenance made for the i-th component.

Δt_{j,i} = t_{j,i} – t_{j-1,i}: is the time interval between two successive requests for maintenance of the i-th component.

The values for **n_i** and **Δt_{j,i}** are obtained from the historical archives of the system. When the expected time for re-engineering the i-th component, RT_i, is less than MTTMR_i, then it is reasonable to suppose that it is unlikely to get requests for maintenance of this component during the time it is being re-engineered. On vice versa, if RT_i, is higher than MTTMR_i, then the component must be subdivided into several subcomponents, to prevent the requests for maintenance that are likely to be made during this time from being held up for very long time, and according to this analysis the decision to continue in re-engineering process on the legacy software will be made.

2. Planning: It is the activity of defining resources, timelines, cost, and other project related information, to carry out the re-engineering process. The cost benefit of re-engineering was calculated according to **Sneed (1995)**, formula, followed by risk analysis which was accomplished to assess both technical and management risks. According to the cost/benefit analysis model for re-engineering proposed by **Sneed (1995)**, the cost benefit for the proposed model can be calculated with respect to nine parameters as defined below:

P1= current annual maintenance cost for a software application

P2= current annual operation cost for a software application

P3= current annual business value of a software application

P4= predicted annual maintenance cost after re-engineering

P5= predicted annual operations cost after re-engineering

P6= predicted annual business value after re-engineering

P7= estimated re-engineering costs

P8= estimated re-engineering calendar time

P9= re-engineering risk factor (P9 = 1.0 is nominal)

L = expected life of the system

From the above, the following cost values can be obtained:

i. (Cmaint) the cost associated with the continuous maintenance of a candidate application (i.e. if re-engineering is not performed) is defined as

$$\{C_{\text{maint}} = [P3 - (P1 + P2)] * L\} \text{ - - - - Sneed (1995),}$$

ii. (Creeng) that is the cost associated with re-engineering a candidate application is defined as

$$\{C_{\text{reeng}} = [P6 - (P4 + P5)] * (L - P8) - (P7 * P9)\} \text{ - - - - Sneed (1995),}$$

iii. Using equations 1 and 2 above, the overall benefit of re-engineering can be computed thus:

$$\{ \text{Cost-benefit} = C_{\text{reeng}} - C_{\text{maint}} \} \text{ - - - - Sneed (1995),}$$

For a software organization, the cost-benefit presented above can be performed for all high priority applications identified during inventory analysis and those that show high cost-benefit can be targeted for re-engineering, while others can be postponed until resources are available. The model does not specify how to compute P4 to P9 hence relies on costs estimation models like Constructive Cost Model II (COCOMO II), Halstead, Expert Judgment, Function Point, and Lines-of-Code.

Those applications that show the highest cost/benefit can be targeted for re-engineering, while work on others can be postponed until resources are available.

There are a number of risks that can be recognized and the re-engineering teams have to take care of them, some of these risks are:

- Development Environment: Risks may be associated with the availability and quality of COTS (Common-Off-The-Shelf) tools.
- Product Size: Impact of risks is directly proportional to the overall size of the product that has to be re-engineered/re-build/re-modulated.
- Experience: Risks associated with the overall technical and project experience of the software engineers who are involved in the process of re-engineering.
- Customer characteristics: Risks linked with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.

- Technology to be built: Risks associated with the complexity of the system to be re-engineered and the newness of the technology that is packaged by the system. The complexity of the system can be determined through the Cyclomatic Complexity, which is a measurement of complexity of a program code and can be calculated by using the control flow graph of the program. All these values are calculated according to the following formulas:

- Code Cyclomatic Complexity = Number of graph edges – Number of Graph Nodes + Number of connected components
- Mean Cyclomatic complexity= Cyclomatic Complexity / Number of modules
- Decision density= Cyclomatic complexity / Number of lines

The following Table 3.1, displays the results published by SEI (Software Engineering Institute) and they are being used widely to determine the health of the code base and to evaluate the risk associated with the application.

Table 3.1: Risk Evaluation according to Mean Cyclomatic Complexity

Mean Cyclomatic Complexity	Risk Evaluation
1-10	A simple program, without much risk
11-20	More complex, moderate risk
21-50	Complex, high risk program
Greater than 50	Un testable program (very high risk)

- Process definition: Risks associated with respect to the process followed by the organization to perform re-engineering process.
 - Business impact: The constraints imposed by management or the market place and also the domain of the business are likely to cause risks in performing re-engineering process. In this case such risks are minimized through fine planning at the planning phase.
3. Reverse engineering: it is the process of analyzing software in an effort to create a representation of the software at a higher level of abstraction than source code. Reverse engineering is a process to recover the design of the legacy software. Reverse engineering process starts from the implementation phase and moving towards the

coding, design and requirement phase, as a result of which the data, architectural and procedural design information is extracted from existing legacy software.

4. Document re-structures: In this phase the mapping of re-structured SRS (software requirement specification) to the design document is being done, in other word;. As SRS changes the design structure consisting of DFD/ ER diagrams/UML diagrams need to be changed as per the extent of changed requirements, the algorithms have to be reviewed and modified and program logic should be verified.
5. Code re-structures: After getting all design diagrams of the legacy software and collecting the additional customer's requirements, the design structure will be changed according to these requirements. The code must to be re-structured according to the new design diagrams, the code has to be written by appropriate programming language, then the resultant re-structured code is reviewed and tested to ensure that no distortions have been introduced and the internal code documentation is updated.
6. Data re-structure: In most cases, data re-structuring begins with a reverse engineering activity. Data objects and attributes are identified, existing data structures are reviewed for quality and the data are re-engineered when data structure is weak, because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will always result in either architectural or code-level changes. From a conceptual point of view, the legacy data are partitioned into two classes:
 - Essential data, needed to carry out the application's business functions. The essential data contains:
 - Conceptual data, that is specific to the application domain, and for which they have a precise conceptual meaning;
 - Necessary structural data, which do not belong to the above class but constitute the primary keys to the tables making up the database.
 - Residual data, that are not necessary for carrying out the business functions, but are used by the legacy system and must therefore remain in the database until the procedures that use them have been re-engineered. Instead, the residual data class contains all the data that existed in the legacy database, and which should ultimately be eliminated to improve the software quality. They are classified as:
 - Control data, which communicate to a procedure that an event occurred during execution of other procedures, thus controlling the behavior of the former one.

- Redundant structural data, used to organize and support the data structures of the legacy system, but which are not strictly required, and can be removed with a better design of the database.
 - Semantically redundant data, whose definition domain is the same as, or is contained in, the definition domain of other data, while each equal value in the two definition domains is interpreted in the same way.
 - Computationally redundant data which can be computed starting from a different set of data included in the same database.
7. Forward engineering: It is the process which starts from requirement to the implementation phase. Forward engineering, also called renovation or reclamation, not only re-covers design information from existing software, but uses this information to alter or re-constitute the existing system in an effort to improve its overall quality. In most cases, re-engineered software re-implements the functions of the existing system and also adds new functions to improve overall performance.
8. Deployment: It is the final step of FUSION model that start by evaluating and testing the resulted re-engineered software, then deliver this software to the customer. The recommended testing techniques are Alpha testing and then Beta testing which is conducted when the project is delivered to the customer. The feedback process is necessary to obtain customer evaluation of that software, according to which the re-engineering process iterates to meet the customer demand.

3.4 Legacy Software System

Any legacy software system consists of various components, each one of these components will go through different states during the proposed re-engineering process. Each component may be in one of the following states during the various steps of the re-engineering process:

- Legacy, i.e. components of the legacy system that have not yet been re-engineered.
- Restored, i.e. functions with the same structure as in the legacy system, but access data through the new data banker, that alone recognizes the physical structure of the database.
- Re-engineered, i.e. components of the legacy system that have already been modified, and whose quality has reached the desired levels.

- New, i.e. components that did not exist in the legacy system, but have been added to introduce new functions in the same application domain.

3.4.1 The legacy supermarket management system

To implement the FUSION model, the legacy software has to be selected, so the selected legacy software is the supermarket management system. The FUSION model will be applied on that legacy software, but at first there is need to take a view on this legacy software and the nature of its work and objectives.

3.4.2 The objective of the legacy supermarket management system

This legacy software has designed and developed to a stage near to real life implementation. Although still it is not as proficient as is required to cater to the needs of a Supermarket Management. This legacy software objective is to remove some manual pitfalls that can be removed easily by computerization. The following are some common pitfalls:

1. Removal of error in calculation: - Some time there is an error in calculation of total amount. This is caused generally by a human being. A human can make mistakes while a machine can't. So, the chances of error are less. Computerization is very useful in removing the error.
2. Removal of Bargaining: - It is human nature that he wants to earn more money in less time. So, he will sell the product with large margin and if one knows the correct price only one will get that product in right price. So, this bargaining is abolished in computer system because the rate of item is prescribed having fewer margins. So, the shop owner will not bargain. This feature makes computer system very popular among honest shop owners.
3. Clean and record of item: - The bill of the item is neat and clear so everyone can understand it. This can give a good practice of business and a customer can trust on this system. In manual system the bills of the items is not so clear that everyone can understand it easily. Sometimes the shopkeeper's laziness restricts time from writing the bill, so the person (customer) will not be able to know about the status of each item .
4. Easy returning process: - In manual system sometimes a shop owner will not accept that this item is purchased from my shop. So, the customer face problem in returning the

defected goods. Using the computer system the returning of defected goods is very easy as the bill given by the store owner at purchasing time tells the quality and price of that good. The unwanted problem faced by the customer is abolished.

The following are some other objectives of the legacy supermarket system:

1. To give the bill receipt to the customer at the time of each purchase.
2. To provide the ability to the user of the system to add, delete, update different department.
3. To provide the ability to the user of the system to add, delete, update different employees to this system.
4. To provide the ability to the user of the system to add, delete, update different Vendors (suppliers) to this system.
5. To provide the ability to the user of the system to add, delete, update different Items (products) to this system.
6. To provide Interactive environment.

3.4.3 The Inputs to the legacy supermarket software are the following:

The legacy supermarket software has number of input items. They are:

1. Entry of all information related to vendors such as: vendor identification number, name, department, address, telephone, and fax.
2. Entry of salary and personnel information for each employee as: employee identification number, name, department, address (street, city, state, and zip), telephone, date of birth, salary, designation, qualification, and fax.
3. Entry of all information related to customer such as: name (first name, initial, last name), code, address (street, city, state, and zip), telephone, and email.
4. Entry of department information as: department name, department Identification number.
5. Entry of item's details as: department Id, item code, Item name, Item product, item rate, item discount, and item quantity for each item present in the Supermarket Management system.

6. Entry of bill information as: bill number, department, customer code, item code, and quantity from this item to be purchased.

3.4.4 Output of the legacy supermarket software

The legacy supermarket software has number of output items. They are:

1. The bill of the items purchased by the customer.
2. The stock status of each item.
3. Report contains the employee information.
4. Report contains the customer information.

3.4.5 Process Logic of the legacy supermarket software

The database of the legacy supermarket software consists of seven tables which keep the information about items, employees, vendors, customers, customer's bill and departments currently present in the system. All these information can be inserted, updated, deleted, and edited by the user of the system. When the user want to generate bill he has to select department and customer code, then he has to select the item code and enter the quantity, after that he has to click on purchase button to calculate the total price and update the quantity of the selected item that is available in the store, then click on "ready to buy" button to generate the bill. If the user wants to change the price of specific item, he can do that by going to Item page and select that item, then update the price of that item. This is the main process logic of the legacy supermarket system.

3.5 Implementation of FUSION model on legacy supermarket management system.

For the purpose of applying FUSION model, a legacy software known as supermarket management system was chosen, where that legacy software was developed in visual basic before many years ago and all the steps required to convert it to the target re-engineered system will be discussed, **Abbas et al. (2013)**. These steps are as follows:

1) Identification of problems and suggested optimization of legacy software:-

The software re-engineering team should meet with customer in order to re-specify the requirements as per the current expectations of the user. In the supermarket system the new requirements are identified such as convert the system from windows based to web based,

create two levels of authentication as administrator level and employee level, add the facility to generate many types of reports about the product, stock master, supplier, sale and purchase information, and feedback report, followed by mapping these requirements with the SRS. The reasons of problems in the selected legacy software were detected within this first step, some of which are that the legacy software did not have any type of security, there is weak error handling procedure that does not cover all the possible error cases, and that the old system can work on only one computer which results in poor service with only one counter in the supermarket to generate the bill for the customer. The legacy system is developed by using visual basic programming language that has number of disadvantages as mentioned in the MSDN of Microsoft. They are the following:

- It is not suited to complex modern programming techniques. On account of its age, little is being done to further the VB environment and it has been largely superseded by VB.NET and other languages (even by Microsoft!).
- Programs that are written in VB tend not to be the quickest, this is largely due to the additional code that is often included that is not really necessary for your program to run.
- VB is an interpreted language that again slows the execution of your program down.
- VB programs require large libraries to be present on your PC to enable them to work. If these libraries not present the programmer either has to supply them or you have to download them.
- Because of its over-simplified method VB can produce programmers that are sloppy in their work leading to workarounds having to be employed.
- Because of its age the VB does not allow many modern techniques such as Object Orientated Programming.
- As programmer can control the checking and warning systems in VB it often enables the programmer to write code that is very difficult to troubleshoot when a bug arises.
- VB consists of syntax and features borrowed from other languages (often ones that are now no longer used). Watch a programmers face when talking to them about "GoSub" and "On Error" commands!
- OOP - Object Orientated Programming - is missing from VB, this is one of the most popular techniques in all modern languages allowing code to be easily reused (although this is available in VB.NET)
- There is no threading support, although this is available in VB.NET.

- The programs a programmer developed in VB are not portable and cannot be used on non-Windows systems..
- Mathematical performance is poor which decrease the speed of your program.
- Service Packs! Most of programmers know about Microsoft's love of service packs to repair the many bugs that have accumulated over time.

After that, the feasibility study was conducted, and according to this analysis the decision to continue in re-engineering process for the supermarket management system was made.

2) Planning:-

It is the activity of defining resources, timelines, cost, and other project related information, to carry out the re-engineering process. It was decided to use ASP.NET with C# programming language to design and implement all web forms of the system, and to use Microsoft SQL server 2008 to re-design the database and implement it. The duration of the whole process was estimated to be three months; the values of the parameters that used in **Sneed (1995)** formula to calculate the cost benefit of re-engineering are supposed according to Iraqi markets, **Abbas et al. (2013)**, these values as following:

$P_1 = 400\$$, current annual maintenance cost for an application.

$P_2 = 150\$$, current annual operation cost for an application.

$P_3 = 600\$$, current annual business value of an application.

$P_4 = 100\$$, predicted annual maintenance cost after re-engineering.

$P_5 = 100\$$, predicted annual operations cost after re-engineering.

$P_6 = 1500\$$, predicted annual business value after re-engineering.

$P_7 = 150\$$, estimated re-engineering costs.

$P_8 = 3 \text{ months} = 0.25 \text{ years}$, estimated re-engineering calendar time.

$P_9 = 1$, re-engineering risk factor ($P_9 = 1.0$ is nominal).

$L = 4 \text{ years}$, expected life of the system.

According to Sneed's (1995) formula, the cost associated with the continuous maintenance of the legacy supermarket system before re-engineering is equal to 200\$ and the cost associated with re-engineering of the legacy supermarket system is equal to 600\$, and the

overall benefit of re-engineering can be computed from these two values and it is equal to 400\$, **Abbas *et al.* (2013)**.

Within the planning step, re-engineering team has to take care of a number of risks that can be recognized through the re-engineering process, some of which are:

- **Development Environment:** Risks may be associated with the availability and quality of COTS (Common-Off-The-Shelf) tools. In this project such risks are minimized by using ASP.NET tools as RAD (Rapid Application Development) from Microsoft, given that the Microsoft tools are most popular and trusted tools.
- **Product Size:** Impact of risks is directly proportional to the overall size of the product that has to be re-engineered/re-build/re-modulated. In this project the size of the legacy software is 2411 SLOC (Source Line of Code) and the number of modules is 212.
- **Experience:** There are so many risks associated with the overall technical and project experience of the software engineers who are involved in the process of re-engineering. In this project this risk was eliminated by advance technical course in ASP.NET and another in SQL server.
- **Customer characteristics:** Risks coming with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- **Technology to be built:** Risks associated with the complexity of the system to be re-engineered and the newness of the technology that is packaged by the system. In this project the total Cyclomatic complexity of the legacy supermarket management system is 578, the Mean Cyclomatic complexity is 2.726, and the decision density is 0.2397. All these values are calculated according to the formulas that discussed in previous section. The calculated values show that, the legacy supermarket software is a simple program, with little risk.
- **Process definition:** The used model for re-engineering is the FUSION model that has the cyclic features; also it depends on the feedback that comes from the customer. The feedback is coming at the final step of the iteration to evaluate and regenerate next versions of the re-engineered system, so the risks will reduce.
- **Business impact:** Such risks are minimized through well planning at the planning phase.

3) Reverse engineering:-

In this project the “Visustin v7” analysis tool was used to visualize the code of the legacy supermarket management system with flow charts! “Visustin” converts source code to flow charts and UML activity diagrams to understand existing code. The analysis process on the legacy software is made to see all the windows and operations of the legacy supermarket system, and to discover the output reports from this legacy system. Legacy supermarket management system has main window that contains a number of sub windows. All of these windows are analyzed and the codes analyzing was done for all the forms in the legacy software to produce the UML activity diagrams for each module and the details of samples of them are shown in **Appendix A1**. From the reverse engineering activity the data flow diagram, Execution Flow Chart, Use Case Diagrams, Entity Relationship Diagram and the Database design of the legacy supermarket system was extracted and are shown in **Appendix A2**.

4) Document re-structuring:

In this phase the mapping of re-structured SRS to the Design document is being done. The required software from applying proposed re-engineering process on legacy supermarket system must have good security and two levels of authentication (user, admin), robust error handling procedure that does cover most of the possible error cases, and that the new system can work on web based environment which results in good service to the customers of the supermarket and to generate the bills for many of customers at the same time. The design structure that has DFD, ERD, and UML diagrams will be changed according to the re-structured SRS. All of these modified diagrams are shown in **Appendix A3**.

5) Code re-structuring:

After getting all design diagrams of the legacy software and collecting the additional customer’s requirements, the design structure will be changed according to these requirements. The code has to be re-structured according to the new design diagrams that are shown in **Appendix A3**. The code was written by ASP.NET with C# programming language, and then the resultant re-structured code is reviewed and tested to ensure that no distortions have been introduced and the internal code documentation is updated. As

example, the code for bill generation in legacy software and re-engineered software are analyzed, and all the functionality of the old system available in the new system but in deferent programming languages. The simple comparison will be made between the code of the bill generation of legacy supermarket management system and the re-structured code of the re-engineered system, the following notes will be produced as shown in Table 3.2. The conclusion of this table is that the restructured code is better than the previous code.

Table 3.2: Comparison between the code of bill generation that belong to legacy software and the re-structured code of re-engineered software

	Legacy supermarket management system Bill generation form	Re-engineered supermarket management system Bill generation form
Functionality	same	same
Line of Code	307	162
Number of modules	18	9
Cyclomatic complexity	51	23
Mean Cyclomatic complexity	2.833	2.555
Decision density	0.1661	0.1419
Error Handling	Not available	Implemented

6) Data re-structuring:

In this project there are seven tables in the database of the legacy supermarket management system. They are customer bill, customer detail, department, employee, items, transaction, and vendor and three tables were added to the re-engineered database as per the customer need which are user login, product master, and feedback. The design of the re-engineered database is illustrated in **Appendix A3**.

7) Forward engineering:

In this project, the Rapid Application Development tool, Microsoft visual studio 2008 with C# used to renovate the selected legacy software. The resulted re-engineered software is web based software and has many pages that produce a lot of services to the user and

manager of the system. The samples of UML activity diagrams of the resulted re-engineered software are shown in **Appendix A4**.

8) Deployment:

It is the final step of FUSION software re-engineering model that start by evaluating and testing the resulted re-engineered software are then deliver this software to the customer. The recommended testing techniques are Alpha testing and then Beta testing which is conducted when the project is delivered to the customer. The result of testing is shown in Table 3.3. The feedback process is necessary to obtain customer evaluation of that software, according to which the re-engineering process iterates to meet the customer demand.

Table 3.3: Testing and result of testing

Process	Authorization	Testing	Result
Login	Administrator and Employee	1. input User name and Password for system user (Employees or administration) 2. click on login button	If wrong values are entered then the system view error message, else will log in to the system correctly.
View detail	Administrator	1. Select the detail information button of customer detail, sales detail, purchase detail, supplier detail, stock detail, or product detail.	All the information displayed correctly.
Product entry	Administrator	1. Click on add new to add new product. 2. Fill all blanket text 3. Click on save button 4. To update data click on	Product saved and updated, the quantity of the stock master updated successfully without any error. The ability for looking about any

		update button. 5. To Find product just enter the product Id, then Find button.	product work properly.
Purchase Item	Administrator and Employee	1. Click on add new to add new serial number of the purchased items. 2. Enter the supplier, Item, and quantity. 3. Click on calculate button. 3. Click on save button 4. To update data click on update button. 5. To find information about any purchase process just enter the serial number, then click on Find button.	Purchase information saved and updated, the quantity of the stock master updated successfully without any error. The ability for looking about any purchase process work properly.
Stock master	Administrator and Employee	1. Fill all blanket text 2. Click on update button. 3. To find the quantity of any product, just select the product and click on find button.	Stock master updated successfully without any error. The ability for looking about the quantity of any product work properly.
Sale (customer bill)	Administrator and Employee	1. Click on Add new button 2. Input all the bill information 2. Click a save button	The bill can be saved in the system and updated without any error. The printed bill contains

		3. Click on Add to Bill button to add item to bill. 4. To remove an Item just click on Remove from Bill button. 5. To print the bill click on Print button. 6. To clear the bill click on Clear button.	correct information without any mistake.
New customer	Administrator and Employee	1. Click Add new to generate new customer Id. 2. Fill all blanket text 3. Click save button 4. To update data click on update button. 5. To find any customer, just enter customer Id then click on Find button, 5. To delete any customer, just enter customer Id then click on Find button, then click on Delete button.	The customer information saved in the system, and can be updated successfully without errors. The find and delete process applied correctly.
New operator	Administrator and Employee	1. Click Add new to generate new operator Id. 2. Fill all blanket text 3. Click save button 4. To update data click on	The operator information saved in the system, and can be updated successfully without errors. The find and delete process applied

		update button. 5. To find any operator, just enter operator Id, then click on Find button, 5. To delete any operator, just enter operator Id then click on Find button, then click on Delete button,	correctly.
New supplier	Administrator	1. Click Add new to generate new supplier Id. 2. Fill all blanket text 3. Click save button 4. To update data click on update button. 5. To find any supplier, just enter supplier Id, then click on Find button, 5. To delete any supplier, just enter supplier Id, then click on Find button, then click on Delete button,	The supplier information saved in the system, and can be updated successfully without errors. The find and delete process applied correctly.
Create new user	Administrator	1. Enter user name. 2. Enter user password. 3. Confirm password. 4. Select user type (Admin or Employee).	The new user of the system was registered successfully and the user has all his authorization to login to the system according to his type; admin or employee.

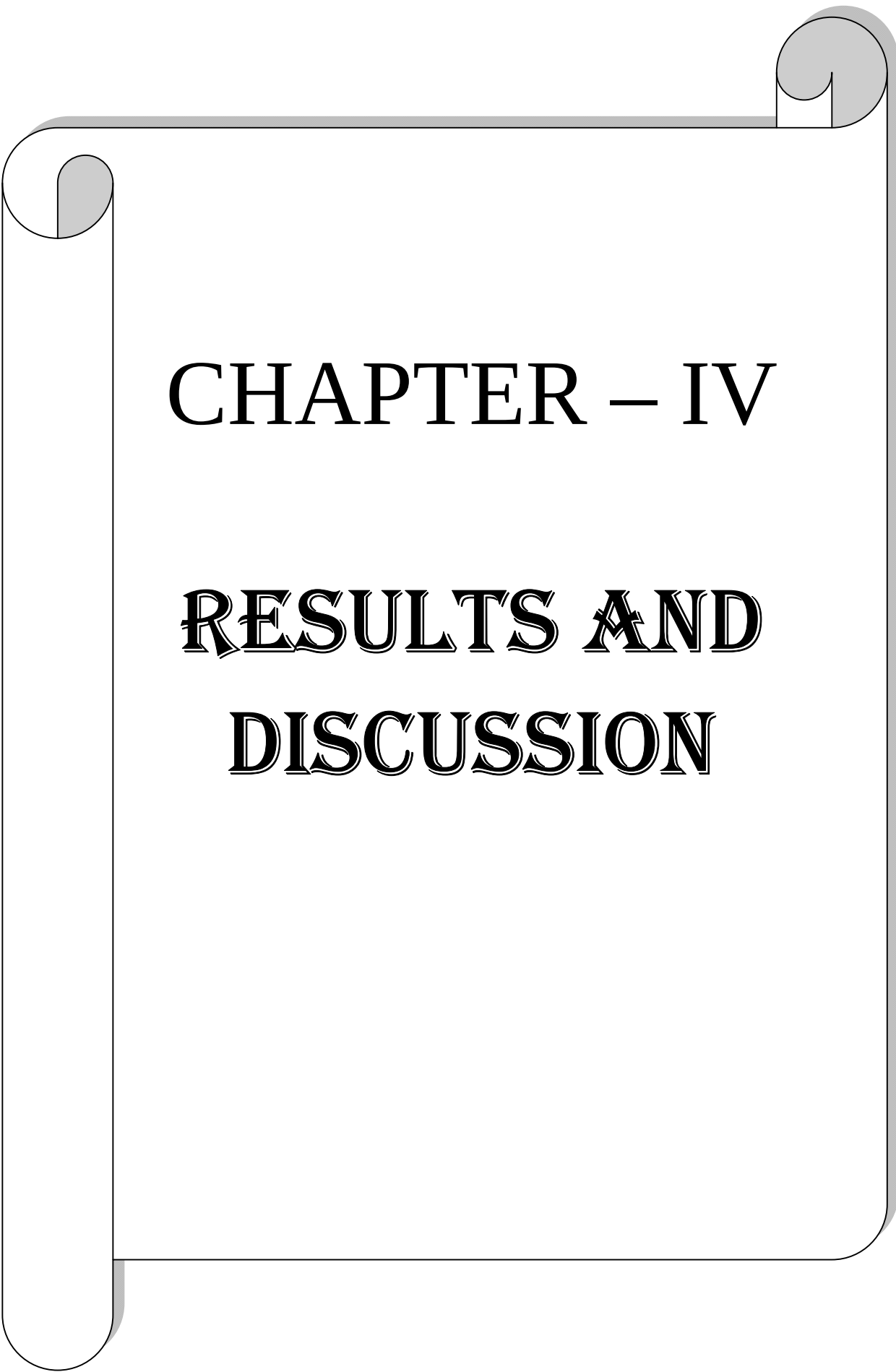
		5. Click on Register button. 6. To reset the entered values just click on Reset button.	
Change user password	Administrator	1. Select user and his type. 2. Input the new password. 3. Confirm the new password. 4. Click on Update button. 5. To reset this information just Click on Reset button.	The password changed successfully. Without any mistake.
Delete user	Administrator	1. Enter the name of user. 2. Click on Delete button.	The delete process done successfully.
View the list of system's users	Administrator	1. Click on List of User button.	All system's user information displayed correctly.
View the feedback details	Administrator	1. Click on Feedback Detail button. 2. To delete any record from the feedback table just click on delete button.	The Feedback information displayed to the admin correctly.
View reports	Administrator	1. Click on Reports button. 2. Report page displayed and clicks on the required report to be displayed.	All nine reports displayed correctly (User, Supplier, Customer, Employee, Sale, Stock master, Product, Purchase, and Feedback reports).

Add new department	Administrator	1. Click on Add new button. 2. Enter department name. 2. Click the save button. 3. To delete department, just enter department's Id, then find and delete. 5. To update department information just click on update button. 6. To clear the page click on Reset button.	The new department added correctly. Updated process done successfully without any error. The ability for looking about any product work properly.
Log out process	Administrator and Employee	1. To log out from the system just click on log out button.	Log out process done successfully.
Feedback process	Public	1. Fill all blanket text. 2. Click on Submit button. 3. To clear the page just click on Reset button.	The feedback saved successfully and the administrator can check it.

3.6 Applying FUSION model:

In this chapter newly proposed FUSION re-engineering model is applied on legacy supermarket management system and is analyzed with other similar existing models is explained. The first step is the identification of problems and suggested optimization of legacy software, and the second step is planning to define resources, timelines, cost, and other project related information, to carry out the re-engineering process. The third step is to do reverse engineering process to analyze the legacy system, the samples of UML activity diagrams, and all DFD, ERD, use case diagrams, execution flowchart and database of legacy supermarket system is explained in **Appendix A1**, and **Appendix A2**, respectively.

The Document re-structures, Code re-structures, and Data re-structures steps were applied and the resulted DFD, ERD, use case diagrams, execution flowchart and database of re-engineered supermarket system is illustrated in **Appendix A3**. The seventh step is the Forward engineering, where it is the process which starts from requirement to the implementation phase. Then the deployment step is applied, finally, the samples of UML activity diagrams of re-engineered system is shown in **Appendix A4**.

A decorative border resembling a scroll, with rounded corners and a grey shadow effect. The top-left and top-right corners feature a scroll-like fold, and the bottom-left corner has a vertical extension.

CHAPTER – IV

RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Comparison between legacy supermarket management system and the re-engineered system by using FUSION model

The comparison between the legacy system and the re-engineered system is shown in Table 4.1. Metrics such as Size (LOC), Number of modules, Cyclomatic Complexity, Mean Cyclomatic Complexity, and Decision Density are calculated by using “Visustin v7” analysis tool. The metrics are evaluated as Low (L), Medium (M), and High (H). Result of technical comparison between legacy supermarket management system and the re-engineered system is shown in Table 4.2 and this comparison includes the difference in Platform, Application, Programming Language, Front end, Backend, Database, Number of tables, Number of generated reports, Execution time, Coding, Making dll (dynamic link library), Authentication levels, Stock management and Customer Feedback. Also, the size (LOC) comparison is illustrated in Fig.4.1; the number of modules comparison is illustrated in Fig.4.2; the Cyclomatic complexity comparison is illustrated in Fig.4.3; the Mean Cyclomatic complexity comparison is illustrated in Fig.4.4, and the decision density comparison is illustrated in Fig.4.5.

From the data in Table 4.1 and Table 4.2, it is proved that the re-engineered system is better than the legacy system and it meets the customer need, so the FUSION re-engineering model is useful model for re-engineering processes.

Table 4.1: Metrics Comparison between legacy supermarket management system and the re-engineered system using FUSION model

		Legacy Supermarket management System	Re-engineered Supermarket management System
Complexity	Size (LOC)	2411	2323
	Number of source code modules	212	139
	Cyclomatic Complexity	578	354

	Mean Cyclomatic complexity	2.726					2.546				
	Decision density	0.239					0.152				
Reusability	Integrability		L	M	H			L	M	H	
	Testability		L	M	H			L	M	H	
	Portability		L	M	H			L	M	H	
	Modifiability		L	M	H			L	M	H	
Quality	Functionality		L	M	H			L	M	H	
	Reliability		L	M	H			L	M	H	
	Performance		L	M	H			L	M	H	
	Security		L	M	H			L	M	H	

Table 4.2: Technical comparison between legacy supermarket management system and the re-engineered system using FUSION model

	Legacy Supermarket Management System	Re-engineered Supermarket Management System
Platform	Windows	Windows & linux
Application	Windows based (stand alone based)	Web Based (client-server)
Language	Visual Basic 6.0	C#
Front end	Visual Basic 6.0	ASP.NET
Backend	Sqlserver 2005	Sqlexpress 2008 R2
Database	All tables are dependent upon the reference constraint which results in changing the condition difficult as it can lead to the removal of old data that do not match the condition.	The constraints are implemented through C# application codes which results in data independency and thereby eases alteration of condition by just changing the code with no effect

		on the tables and database.
Execution time	The legacy system with Visual Basic as its front end takes more execution time, because it includes the execution of both the control and its backend code.	The ASP.NET with active-x controls takes less time to execute as it requires the execution of only the control and not the back end code.
Coding	Visual Basic programming language which supports the structured programming paradigm is used and hence leads to rewriting the codes since Visual Basic does not support grouping control.	C# which supports Object Oriented Programming is used as the programming language and thus enables the programmer to reuse the predefined objects and classes in an easy manner without having any need to write them again.
Making dll (dynamic link library) library	Not supported in Visual Basic	The ASP.NET supports the development of of dll files that can be used easily when needed, reducing time and effort.
Code Reusability	Not available	Available
Number of tables	7	10
Authentication levels	None	Two levels, Administrator and Employee
Number of reports	3	9
Stock management	Not available	Available
Customer Feedback	Not available	Available

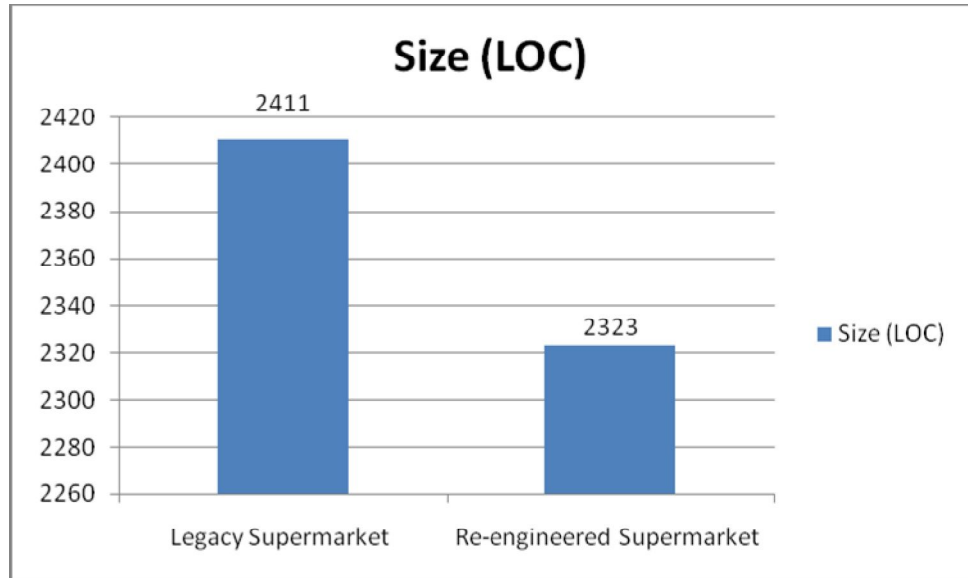


Fig.4.1: The comparison between the size (LOC) of the legacy supermarket system and the re-engineered system using FUSION model.

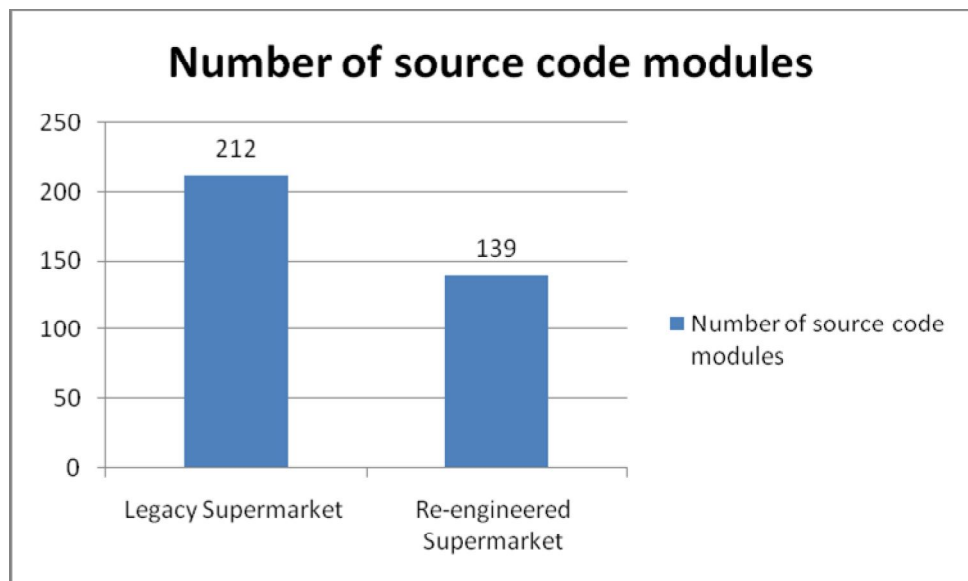


Fig.4.2: The comparison between the number of source code modules of the legacy supermarket system and the re-engineered system using FUSION model.

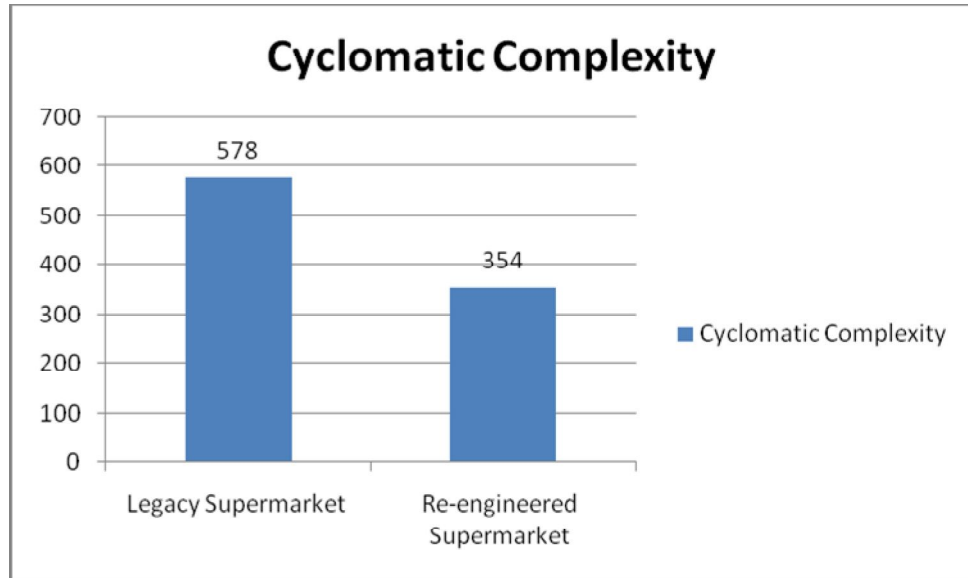


Fig.4.3: The comparison between the Cyclomatic complexity of the legacy supermarket system and the re-engineered system using FUSION model.

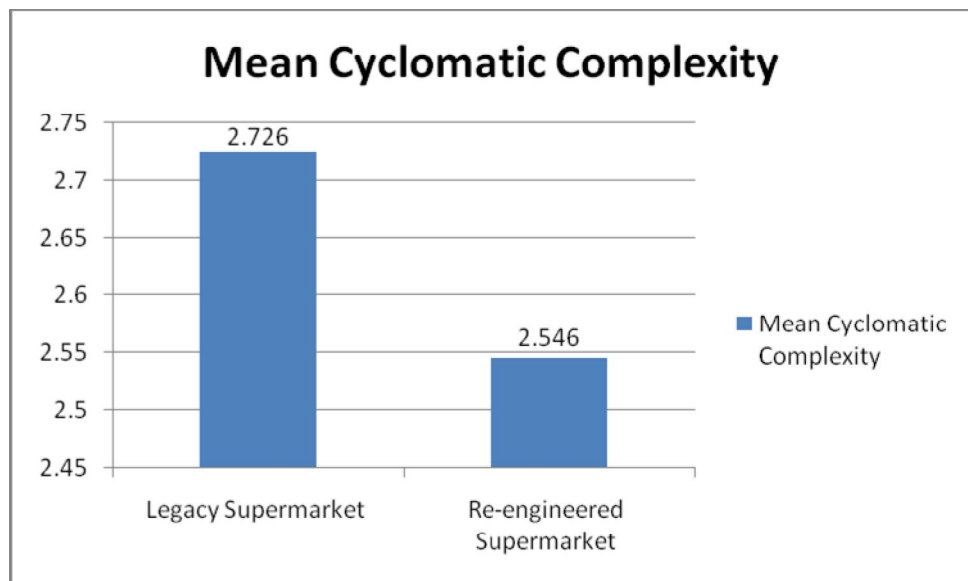


Fig.4.4: The comparison between the Mean Cyclomatic complexity of the legacy supermarket system and the re-engineered system using FUSION model.

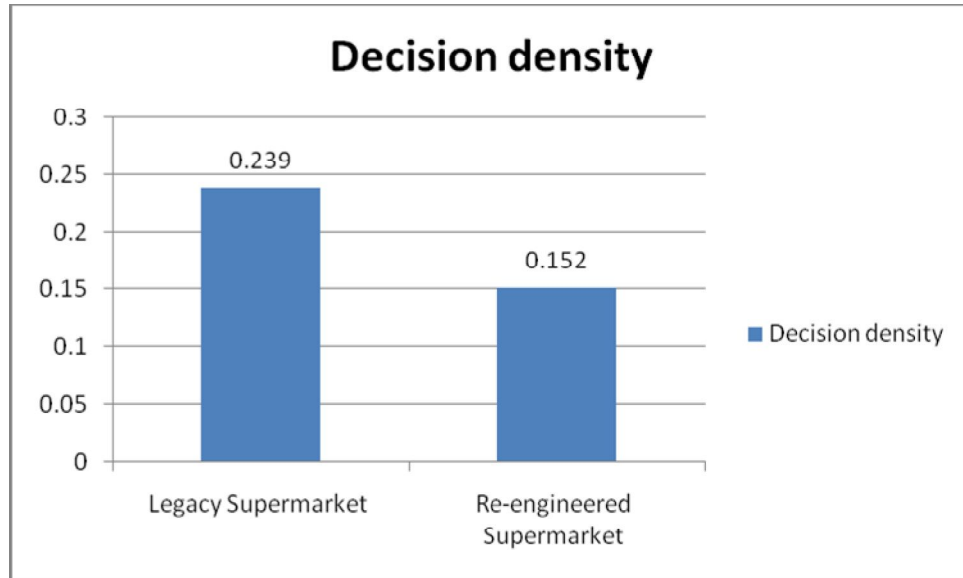


Fig.4.5: The comparison between the decision density of the legacy supermarket system and the re-engineered system using FUSION model.

From the tabulated values in Table 4.1 and Table 4.2, and from depicted charts in Fig.4.1, Fig.4.2, Fig.4.3, Fig.4.4, and Fig.4.5, the produced conclusion is that the FUSION Software re-engineering process model reduces risk level, cost, size (LOC), Cyclomatic complexity, Mean Cyclomatic complexity, decision density, and effort. The use of this re-engineering approach provides a way to increase testability, portability, modifiability, functionality, reliability, performance, and security of legacy software. These results prove that the FUSION model is an effective process to perform the re-engineering of legacy software.

In this thesis the FUSION re-engineering model is applied on legacy supermarket management system to convert it to a new system with more functionality, high security, and good quality. The FUSION re-engineering model is applied to get all the benefits of re-engineering mentioned in chapter one. The software metrics which are used to make comparison between the legacy system and new system, and the result of this comparison for some metrics are the following:

1. Complexity: that include these metrics:

- Size (LOC) = 2411 in legacy system, Size (LOC) = 2323 in Re-engineered system
- Number of source code modules = 212 in legacy system, Number of source code modules = 139 in Re-engineered system

- Cyclomatic Complexity = 578 in legacy system, Cyclomatic Complexity = 354 in Re-engineered system
- Mean Cyclomatic complexity = 2.726 in legacy system, Mean Cyclomatic complexity = 2.546 in Re-engineered system
- Decision density = 0.239 in legacy system, Decision density = 0.152 in Re-engineered system

2. Reusability: that include these metrics:

- Integrability is low in legacy system, Integrability is medium in Re-engineered system
- Testability is medium in legacy system, Testability high is in Re-engineered system
- Portability is low in legacy system, Portability is high in Re-engineered system
- Modifiability is low in legacy system, Modifiability is high in Re-engineered system

3. Quality: that include these metrics:

- Functionality is medium in legacy system, Functionality is high in Re-engineered system
- Reliability is medium in legacy system, Reliability is high in Re-engineered system
- Performance is low in legacy system, Performance is high in Re-engineered system
- Security is low in legacy system, Security is high in Re-engineered system

Result of technical comparison between legacy supermarket management system and the re-engineered system is shown bellow and this comparison includes the difference in Platform, Application, Programming Language, Front end, Backend, Database, Number of tables, Number of generated reports, Execution time, Coding, Making dll (dynamic link library), Authentication levels, Stock management and Customer Feedback.

Technical comparison between legacy supermarket management system and the re-engineered system are the following:

- 1- The legacy software works in windows platform but the re-engineered system can work on windows & linux platform.
- 2- The legacy software is a standalone based application but the re-engineered system is a web based application.
- 3- The programming Language is Visual Basic in legacy software but it is C# in re-engineered software.

- 4- The frontend is Visual Basic in legacy software but it is ASP.NET in re-engineered software.
- 5- The backend is Sqlserver 2005 in legacy software but it is sqlexpress 2008 R2 in the re-engineered software.
- 6- All the tables in the database of legacy software are dependent upon the reference constraint which results in changing the condition difficult as it can lead to the removal of old data that do not match the condition. But in the re-engineered software the constraints are implemented through C# application codes which results in data independency and thereby eases alteration of condition by just changing the code with no effect on the tables and database.
- 7- The Execution time of the legacy system with VB as its front end takes more execution time, because it includes the execution of both the control and its backend code. But in the re-engineered software the ASP.NET with active-x controls is used and it takes less time to execute because it requires the execution of only the control and not the back end code.
- 8- The Number of tables in the database of legacy software is seven but in the re-engineered software is ten.
- 9- There was no authentication level in the legacy software, but there are two levels in the re-engineered software. They are Administrator and Employee.
- 10- The Number of reports in the legacy software is three but in the re-engineered software are nine.
- 11- Stock management and customer Feedback are not available in the legacy software, but they are available in the re-engineered software.

4.2 Differences between FUSION Re-engineering Model and other Models

The comparison between FUSION re-engineering model and the General Re-Engineering Model, Iterative Re-Engineering Model, Dual-Spiral Re-Engineering Model and Hybrid Re-Engineering Model is depicted in the next section.

4.2.1 Comparison between FUSION Re-engineering Model and General Model for Software Re-engineering

A comparison between FUSION model as suggested by **Abbas *et al.* (2013)** and general model of re-engineering discussed by **Linda (2010)**, gave results depicted in Table 4.3.

Table 4.3: Comparison between FUSION and General Software Re-engineering Model

FUSION Software Re-engineering Model	General Software Re-engineering Model
Starts with the source code of an existing legacy system and concludes with the source code of a target system.	Starts with the source code of present legacy system and concludes with the source code of a target system.
This process has systematic procedure that can be applied for simple and complex system.	This process may be as simple as using a code translation tool to translate the code from one language to another or from one operating system to another (UNIX to DOS). Furthermore, the re-engineering task may be very complex, using the present source code to recreate the design, specify the requirements in the existing system then compare them to current requirements, removing those no longer applicable, re-structure and re-design the system (using object-oriented design), and finally code the modern target system.
This model in Fig.3.6 combines between traditional re-engineering process and the spiral software engineering process, so it has eight task regions.	This model in Fig.1.2 applies three principles of re-engineering: abstraction, alteration, and refinement.
There are eight task regions, they are: - Identification of problems and suggested optimization of legacy software. - Planning it is the activity of defining	The applied principles are: - Abstraction is a gradual increase in the abstract level of system. Abstraction produces a representation that emphasizes certain system

(resources, timelines, cost, and other project related information, to carry out the re-engineering process), - Reverse engineering is a process to recover the design of the legacy software, - Document re-structures. - Code re-structures. - Data re-structure. - Forward engineering. - Deployment: It is the final step of proposed software re-engineering model that start by evaluating and testing the resulted re-engineered software, then deliver this software to the customer	characteristics by suppressing information about others. This upward movement is termed reverse engineering and has associated sub-processes, tools and techniques. - Alteration is the making of one or more changes to a system representation without changing the degree of abstraction, including addition, deletion and modification of information, but not functionality. - Refinement is the gradual decrease in the abstraction level of system representation and is caused by the successive replacement of existing system information with more detailed information. This is termed forward engineering and resembles software development of new code, but with some process refinements
There are milestones for management purpose	There are no milestones
Cyclic process has number of iterations	There is no cyclic process
According to customer feedback, the decision is made if the re-engineering process is completed or not	Does not consider customer feedback, when the last task is done, the process will finish.

4.2.2 Comparison between FUSION Re-engineering Model and Iterative re-engineering process

A comparison between FUSION model as suggested by **Abbas *et al.* (2013)** and Iterative re-engineering model suggested by **Bianchi *et al.* (2001)**, gave results depicted in Table 4.5.

Table 4.4: Comparison between FUSION and Iterative Software Re-engineering Model

FUSION Software Re-engineering Model	Iterative Software Re-engineering Model
Starts with the source code of an existing legacy system and concludes with the source code of a target system.	Starts with gradual re-engineering of the procedural components of a legacy system, and at the end of the process, a single system will be in existence: the re-engineered one.
This process has systematic procedure that can be applied for simple and complex system.	This process has systematic procedure that can be applied for simple and complex system.
This model in Fig.3.6 combines between traditional re-engineering process and the spiral software engineering process, so it has eight task regions.	This model in Fig.2.16 made up of evolving the legacy system components toward firstly a restored system and then toward the re-engineered system, it has nine phases.
There are eight task regions, they are: - Identification of problems and suggested optimization of legacy software. - Planning: It is the activity of defining (resources, timelines, cost, and other project related information, to carry out the re-engineering process), - Reverse engineering is a process to recover the design of the legacy software, - Document re-structures,	There are nine phases, they are: - Analyses legacy system: It produce set of components - Classify data: It produce data classified as classes. - Re-design data phase: It produce re-engineered data. - Adapt legacy components: It result in restored functions - Migrate data phase: It transfers the data to a new database. - Equivalence test, which produce

- Code re-structures, - Data re-structure, - Forward engineering, - Deployment: It is the final step of proposed software re-engineering model that start by evaluating and testing the resulted re-engineered software then, deliver this software to the customer.	decision if there is need to go back to the Adapt legacy components phase. - Re-design functions phase: It produce the re-engineered functions. - Equivalence test, produce decision if there is need to go back to the Re-design functions phase. - Empty residual data phase: go back to classify data phase.
There are milestones for management purpose	There are no milestones
Cyclic process has number of iterations	It is semi-cyclic process that iterates in some phases.
According to customer feedback the decision is made if the re-engineering process is completed or not	Does not consider customer feedback, when the last task is done, the process will finish.

4.2.3 Comparison between FUSION Re-engineering Model and Dual-Spiral Re-engineering Model

A comparison between FUSION model as suggested by **Abbas *et al.* (2013)** and Dual-Spiral re-engineering model suggested by **Yang *et al.* (2005)**, gave results depicted in Table 4.6.

Table 4.5: Comparison between FUSION and Dual-Spiral Software Re-engineering Model

FUSION Software Re-engineering Model	Dual-Spiral Software Re-Engineering Model
Starts with the source code of an existing legacy system and concludes with the source code of a target system.	Starts with the two systems legacy and target system work together, at beginning legacy one is full system and target one is empty system, then it end with target system with all functionalities.
This process has systematic procedure that	This process has systematic procedure that can

can be applied for simple and complex system.	be applied for simple and complex system.
This model in Fig.3.6 combines between traditional re-engineering process and the spiral software engineering process, so it has eight task regions.	This model in Fig.2.17 depends on decremented legacy system pattern at legacy system and incremental target system Pattern at target system.
There are eight task regions, they are: - Identification of problems and suggested optimization of legacy software. - Planning, - Reverse engineering. - Document re-structures. - Code re-structures. - Data re-structure. - Forward engineering, - Deployment.	There are four phases at legacy system side and four phases at target system side, they work together; they are: - Remove function, Build proxy, Regression test, and stable system, these phases at legacy system side. - Add function, Build proxy, Regression test, and stable system, these phases at target system side.
There are milestones for management purpose	There are no milestones
Cyclic process has number of iterations	Cyclic process has number of iterations
According to customer feedback the decision is made whether the re-engineering process is completed or not	Does not consider customer feedback, when the last function removed from the legacy system and added to the target system, the dual-spiral process finished.

4.2.4 Comparison between FUSION Re-engineering Model and Hybrid Re-Engineering Model

A comparison between FUSION model as suggested by **Abbas *et al.* (2013)** and Hybrid re-engineering model suggested by **Tarar and Kumar (2012)**,), gave results depicted in Table 4.6.

Table 4.6: Comparison between FUSION and Hybrid Software Re-engineering Model

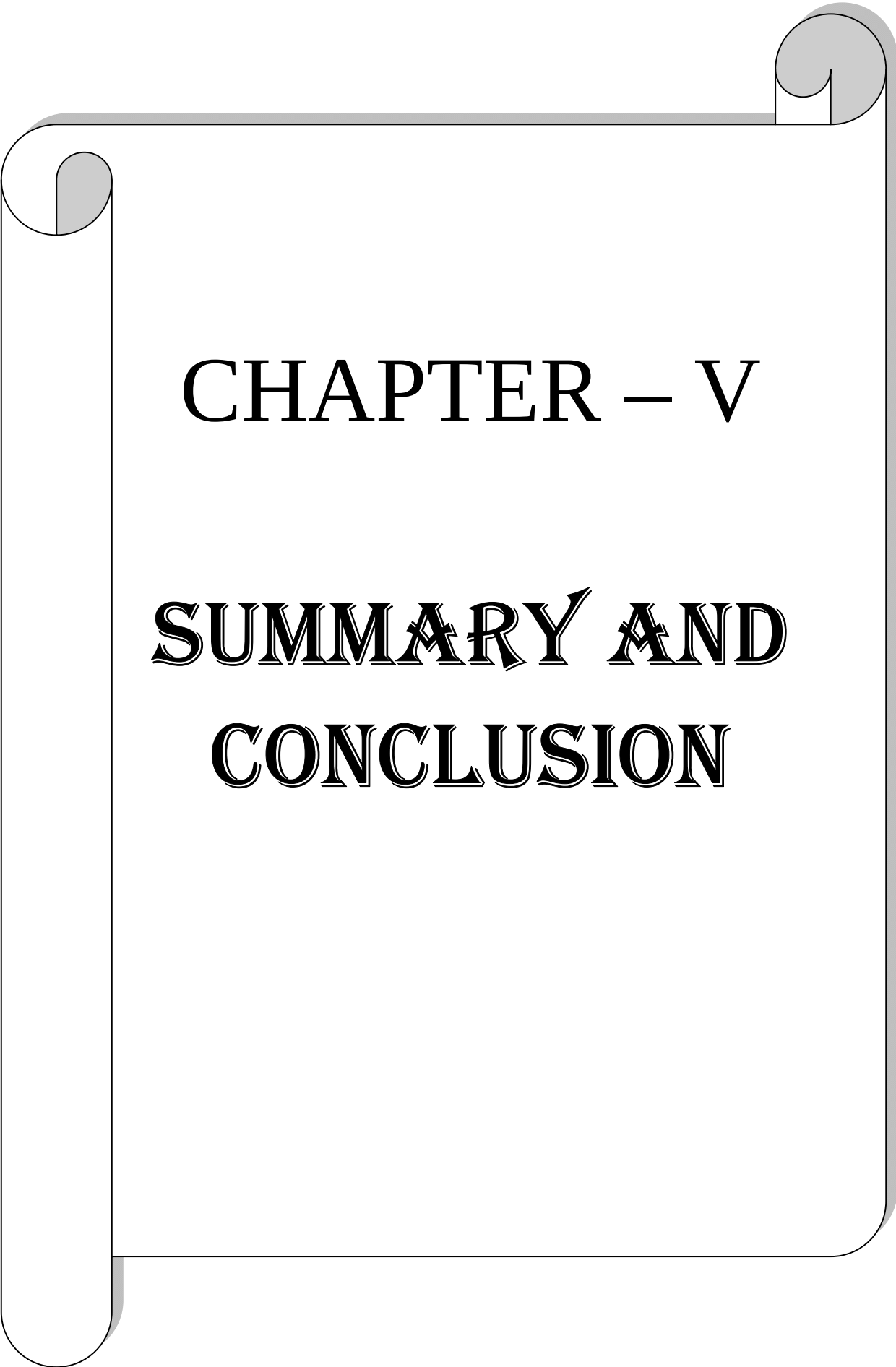
FUSION Software Re-engineering Model	Hybrid Software Re-Engineering Model
Starts with the source code of an existing legacy system and concludes with the source code of a target system.	Starts with the source code of a present legacy system and end with the source code of a target system.
This process has systematic procedure that can be applied for simple and complex system.	This process has systematic procedure that can be applied for simple and complex system.
This model in Fig.3.6 combines between traditional re-engineering process and the spiral software engineering process, so it has eight task regions.	This model in Fig. 2.18 is a re-engineering process that uses a collection of abstraction levels and alteration methods to transition an existing system to a target system.
There are eight task regions, they are: - Identification of problems and suggested optimization of legacy software - Planning. - Reverse engineering. - Document re-structures. - Code re-structures. - Data re-structure. - Forward engineering. - Deployment.	There are four phases, they are: - Re-specify requirement and feasibility study, produced restructured SRS. - Re-design, mapping restructured SRS to design document. - Re-code the design document. - Re-testing and implementation.
This model can use the Commercial off the Shelf (COTS) components, but it does not depend on it.	This model depend on Commercial off the Shelf (COTS), COTS have some risk associated with it as: a package will not perform an anticipated or advertised, or that will be unreliable, incomplete or immature. Also COTS product may limit further enhancements to the system because changes in COTS provided functions may not be possible due to legal or contractual

	issues.
There are milestones for management purpose	There are no milestones
Cyclic process has number of iterations	Cyclic process has number of iterations
According to customer feedback the decision is made if the re-engineering process is completed or not	Does not consider customer feedback, when the last task is done, the process will finish.

4.3 The diagnostic features of FUSION Re-engineering Model

There are many basic tasks that have to be accomplished by any software re-engineering process but there are diagnostic features of the proposed process, **Abbas *et al.* (2013)**, they are as the following:

- The milestones are inserted after each one of iterations to check if the process is performed within scheduled time or not.
- The re-engineering process can stop after any iteration.
- The reasons that cause the problems of the legacy software are determined at first step to overcome those problems reasons.
- In the FUSION re-engineering process, there is planning task that includes feasibility study, scheduling, estimation and risk analysis to determine the expected benefit and estimation of the cost and effort, so that the developer can make his decision on whether to re-engineer that legacy software or re-build it from scratch.
- The third, fourth, fifth, sixth and seventh activities are the same as in traditional software re-engineering process except that in the FUSION model there is a non deterministic number of iterations that lead to the final re-engineered product with new features, capabilities and high quality.
- Deployment and waiting for customer feedback is also a diagnostic activity of the FUSION re-engineering process. According to this feedback the decision is made whether the re-engineering process is completed or not. If the customer is not satisfied, the next iteration will begin.

A decorative border resembling a scroll, with a grey shaded area at the top right corner and a grey shaded area at the bottom left corner.

CHAPTER – V

SUMMARY AND CONCLUSION

CHAPTER 5

SUMMARY AND CONCLUSION

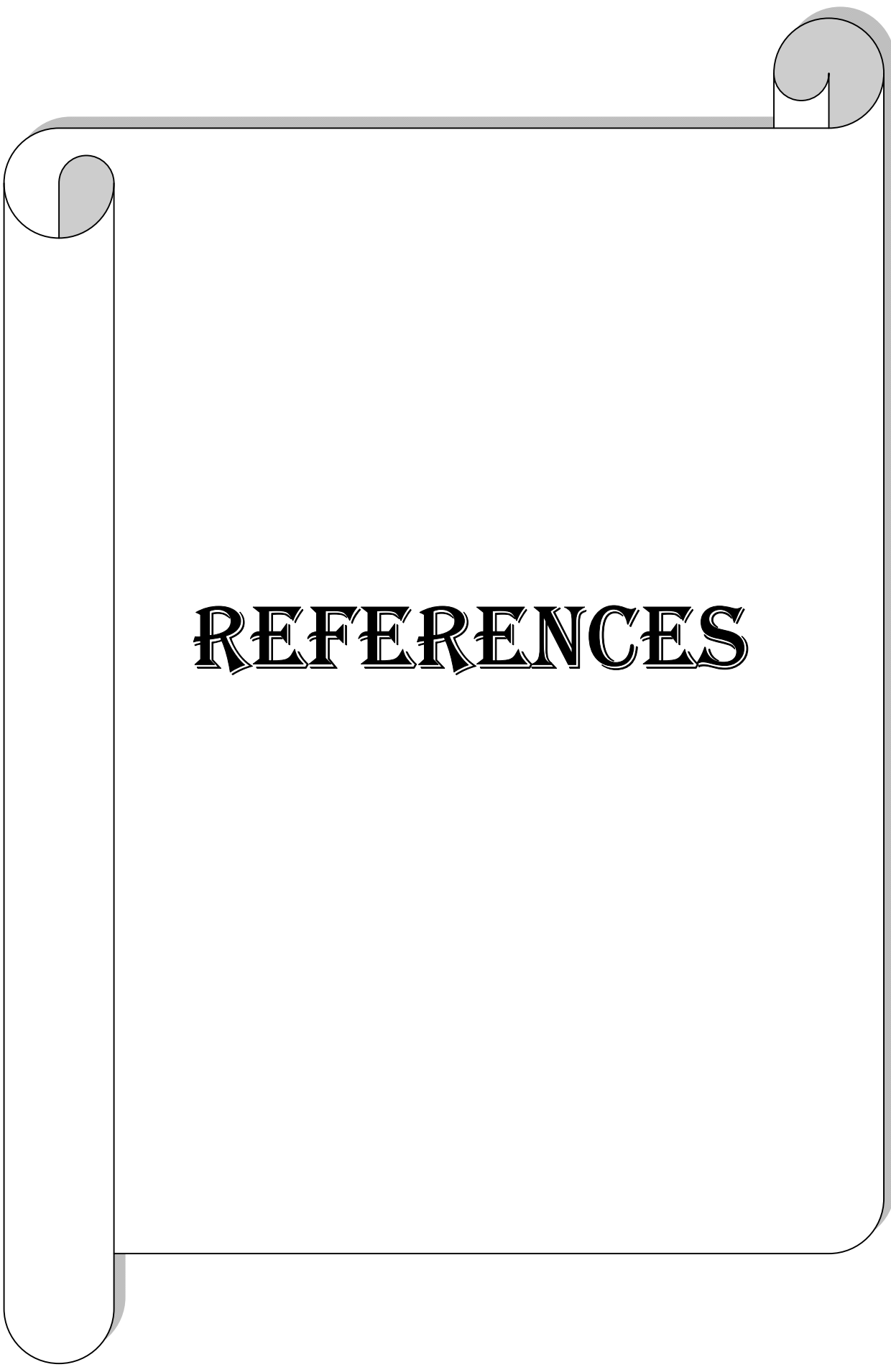
5.1 Conclusion:

The new era is marked by the rapid changes in the computer industry that introduces new hardware and software, making older system suffers from market competition and difficulty in maintenance. In this thesis, a new re-engineering model has been developed and tested. It is based on the integration between spiral software engineering model and the traditional re-engineering model. With this model, all the functionalities of the legacy system can be re-engineered to develop a new system and at the same time by incorporating new features and functional enhancements. The proposed model of re-engineering is successfully applied on a legacy supermarket management system. Through the analysis of FUSION model, it was found that the FUSION Software re-engineering process model reduces risk level, cost, Cyclomatic complexity, and effort. The use of this re-engineering approach provides a way to increase testability, portability, modifiability, functionality, reliability, performance, and security of legacy software. So it is concluded that the FUSION model is an effective software engineering process to perform the re-engineering of legacy software.

5.2 Future works

The future opportunities of the researchers are the following:

- 1- Automate this proposed re-engineering process by CASE tool, to make it more effective and less time consuming.
- 2- Apply the FUSION model on more complicated legacy software, then re-evaluate the proposed re-engineering process.



REFERENCES

References:

Addicott Rachael, McGivern Gerry, Ferlie Ewan (2006), "Networks, Organizational Learning and Knowledge Management: NHS Cancer Networks". Public Money & Management 26 (2): 87–94. doi:10.1111/j.1467-9302.2006.00506.x.

Ahmed Saleem Abbas, W. Jeberson, and V. V. Klinsega (2012), "The Need of Re-engineering in Software Engineering", ISSN: 2049-3444 © 2012 – IJET Publications UK. Volume 2 No. 2, February, 2012.

Ahmed Saleem Abbas, W. Jeberson, and V. V. Klinsega (2012), "A Literature Review and Classification of Selected Software Engineering Researches", The International Journal of Engineering & Technology (IJET), ISSN: 2049-3444 © 2012 – IJET Publications UK. Volume 2 No. 7, July 2012.

Ahmed Saleem Abbas, W. Jeberson, and V. V. Klinsega (2013), "Implementation of FUSION model to Re-engineer Legacy Software", International Journal of Computer Science and Engineering (IJCSE), ISSN(Print): 2278-9960, ISSN(Online): 2278-9979, Impact Factor (JCC): 2.3176, Vol. 2, Issue 2, May 2013, 53-64 © IASET

Ahmed Saleem Abbas, W. Jeberson, and V. V. Klinsega (2013), "Proposed Software Re-engineering Process That Combine Traditional Software Reengineering Process with Spiral Model", International Journal of Advanced Research in Computer Science (IJARCS), ICV (Index Copernicus Value) = 5.47, ISSN 0976 – 5697, Volume 4, No. 2, Jan-Feb 2013.

Alessandro Bianchi, Danilo Caivano, Vittorio Marengo, Giuseppe Visaggio (2001), "Iterative Reengineering of Legacy Systems ", IEEE International Conference on ISBN: 07695118 Year: 2001 Pages: 632-641 Provider: IEEE Publisher: IEEE.

Alexandre Bergel, St'éphane Ducasse, Jannik Laval and Romain Peirs (2008), "Enhanced Dependency Structure Matrix for Moose", Accepted to FamoosR, 2008

André Cristiano Kalsing, Gleison Samuel do Nascimento, Cirano Iochpe and Lucinéia Heloisa Thom (2010), "An Incremental Process Mining Approach to Extract Knowledge from, Legacy Systems", 2010 14th IEEE International Enterprise Distributed Object Computing Conference.

Anthony I. Wasserman (2010), "Software Engineering Issues for Mobile Application Development", Moffett Field, CA 94035 USA, tonyw@sv.cmu.edu.

Ashok Kumar Behera and Rajesh Tiwari (2011), "Analysis and Design of Software Visualization Tool for the Behavior of Object Oriented Programming", International Journal of Computer Applications (0975 – 8887), 2011.

Asit Kumar Gahalaut and Padmavati Khandnor (2010), "Reverse Engineering: An Essence for Software Re-engineering and Program Analysis", International Journal of Engineering Science and Technology Vol. 2(06), 2010, 2296-2303.

Beck, Kent *et al.* (2001), "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010.

Bell and Michael (2008), "Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons. pp. 3. ISBN 978-0-470-14111-3.

Bredemeyer, Dana and Ruth Malan (2003), "Architecture Strategy Choreographs the Dance of Change", slides from the presentation to be given at the Enterprise Architecture Conference in Phoenix, AZ in November, 2003.

Brian Randell (2001), "The 1968/69 NATO Software Engineering Reports". Brian Randell's University Homepage. The School of the Computer Sciences, Newcastle University.<http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOREports/index.html>. Retrieved 2008-10-11 .

Bruce Schneier and Beyond Fear (2003), "Thinking Sensibly about Security in an Uncertain World", Copernicus Books, ISBN 0-387-02620-7 SPIN 10926468,2003, pages 26-27.

C. R. Kothari (2004), "Research Methodology: Methods and Techniques", book, second edition, 2004.

Cadet Pamela A. Sanders, and John A. Hamilton (2002), "A Process for Interoperability", Auburn University, 2002.

Canning R. (1972), "The Maintenance 'Iceberg'," EDP Analyzer, vol. 10, no. 10, October 1972.

Ceruzzi and Paul E. (2000), "A History of Modern Computing", Cambridge, Mass.: MIT Press. ISBN 0-262-03255-4.

Cheng *et al.* (2009), "Software Engineering for Self-Adaptive Systems", book, ISBN 978-3-642-02160-2, 2009.

- Chikofsky E.J. and J.H. Cross (1990)**, "Reverse Engineering and Design Recovery: A Taxonomy" IEEE Software, January 1990, pp. 13–17.
- Choquet Christophe and Alain Corbière (2006)**, "Re-engineering Framework for Systems in Education", University of Maine, France.
- Christopher Poile, Andrew Begel, Nachiappan Nagappan, and Lucas Layman (2009)**, "Coordination in Large-Scale Software Development: Helpful and Unhelpful Behaviors", Dept. of Computer Science, North Carolina State University, September 28, 2009.
- Chu C. William and Chih-Wei Lu (2002)**, "Pattern-based Software Re-engineering", Department of Computer Science and Information Engineering, TungHai University, Taichung, Taiwan, ROC.
- Craig Niiyama, Sam Chung, Donald Chinn, and Sergio Davolos (2008)**, "Service-Oriented Software Reengineering Methodology for Composite Services", Computing & Software Systems, Institute of Technology, University of Washington, Tacoma, {cniiyama, chungsa, dchinn}@u.washington.edu, Milgard School of Business, University of Washington, Tacoma, sergioid@u.washington.edu, 2008
- Csaba Nagy, László Vidács, Rudolf Ferenc, Tibor Gyimóthy, Ferenc Kocsis, and István Kovács (2011)**, "Solutions for Reverse Engineering 4GL Applications, Recovering the Design of a Logistical Wholesale System", 2011 15th European Conference on Software Maintenance and Reengineering, IEEE computer Society.
- Daniel Ratiu (2009)**, "Challenges for Domain Knowledge Driven Program Analysis", FAMOOSr 2009 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.
- Davenport, T.H. and J.E. Young (1990)**, "The New Industrial Engineering: Information Technology and Business Process Redesign," Sloan Management Review, Summer 1990, pp. 11–27.
- David Rothlisberger (2009)**, "Why and How to Substantiate the Good of our Reverse Engineering Tools?", FAMOOSr 2009 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.
- DeMarco T. (1995)**, "Lean and Mean," IEEE Software, November 1995, pp. 101–102.
- Eilam Eldad and Chikofsky Elliot J. (2007)**, "Reversing: secrets of reverse engineering", p. 3. ISBN 9780764574818. <http://books.google.com/books>, (2007).

Fabrizio Perin (2009), "Driving the refactoring of Java Enterprise Applications by evaluating the distance between application elements", FAMOOSr 2009 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.

Fowler M. (1999), Refactoring—improving the design of existing code, Addison-Wesley, Reading, MA., 1999, p. 53

Francesca Arcelli Fontana and Marco Zanoni (2011), "A tool for design pattern detection and software architecture reconstruction", Elsevier, 2011.

Hammer M. (1990), "Reengineer Work: Don't Automate, Obliterate, "Harvard Business Review, July–August 1990, pp. 104–112.

Harry M. Sneed, Anecon GmbH and Wien (2006), "Re-engineering for Testability", Universities of Regensburg and Passau, May, 2006

IEEE Standard Glossary (1990), IEEE, IEEE Standard Glossary of Software Engineering Terminology, report IEEE Std 610.12- 1990, IEEE, 1990.

J. Laval et al. (2009), "Identifying cycle causes with CycleTable", FAMOOSr, paper, 11-Oct-2009

J. W. Wesner et al. (1995), "Winning with Quality, Addison-Wesley", Reading, MA, 1995

James Wilcox (2011), "Paying Too Much for Custom Application Development", March 2011.

Jannik Laval, Simon Denier and St´ephane Ducasse (2009), " Identifying cycle causes with CycleTable", FAMOOSr 2009, 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.

Jannik Laval, Simon Denier, St´ephane Ducasse, and Alexandre Bergel (2009), "Identifying cycle causes with Enriched Dependency Structural Matrix", Accepted to WCRE, 2009.

Jannik Laval, Simon Denier, St´ephane Ducasse, and Jean-R´emy Falleri (2010), "Supporting Simultaneous Versions for Software Evolution Assessment", Elsevier, 2010.

Jasmine K.S and R. Vasantha (2009), “Re-engineering Test Management for Increasing Testing Effectiveness in Component Based Enterprise Applications”, International Journal of Computer Theory and Engineering.

Jiantao Pan (1999), “Software Testing” by Jiantao Pan, Carnegie Mellon University, 1999, jpan@cmu.edu.

Jörg Bartholdt and Detlef Becker (2011), "Re-Engineering of a Hierarchical Product Line", 2011 15th International Software Product Line Conference, IEEE computer Society.

Jurgen Ebert Volker Riediger and Andreas Winter (2008), "Graph Technology in Reverse Engineering, The TGraph Approach", University of Koblenz-Landau, Institute for Software Technology, Universitätsstr. 1. 56070 Koblenz, Germany ebert,riediger@uni-koblenz.de, and Johannes-Gutenberg-University Mainz, Institute for Computer Science, Staudingerweg 9. 55128 Mainz, Germany, winter@uni-mainz.de, 2008.

Klaus Krogmann (2007), “Re-engineering of Software Component Models to Enable Architectural Quality of Service Predictions”, Institute for Program Structures and Data Organisation Faculty of Informatics University of Karlsruhe (TH), Germany, 2007.

Laplante Phillip (2007), “What Every Engineer Should Know about Software Engineering”, ISBN 9780849372285, <http://books.google.com>, 2007 .

Lina Tankeleviciene (2008), "Using Re-engineering Techniques for Distance Study Course Improvement", International Book Series "Information Science and Computing, The paper is selected from Third International Conference "Modern (e-) Learning" MeL 2008, Varna, Bulgaria, June-July 2008.

Lloyd G. Williams and Connie U. Smith (2002), "Five Steps to Solving Software Performance Problems", June, 2002.

Ma Y. Chen and Thimm G. (2008), "Paradigm Shift: Unified and Associative Feature-based Concurrent Engineering and Collaborative Engineering", Journal of Intelligent Manufacturing, DOI 10.1007/s10845-008-0128-y, 2008.

Mamta Garg, and Manoj Kumar Jindal (2009), "Reverse Engineering – Roadmap to Effective software Design", International Journal of Recent Trends in Engineering, Vol. 1, No. 2, May 2009.

Manna M. (1993), “Maintenance Burden Begging for a Remedy,” Datamation, April 1993, pp. 53–63.

Martin Wirsing, Matthias Hölzl, Lucia Acciai, Federico Banti, Allan Clark, Alessandro Fantechi, Stephen Gilmore, Stefania Gnesi, László Gónczy, Nora Koch, Alessandro Lapadula, Philip Mayer, Franco Mazzanti, Rosario Pugliese, Andreas Schroeder, Francesco Tiezzi, Mirco Tribastone, and Dániel Varró (2010), "SENSORIA Patterns: Augmenting Service Engineering with Formal Analysis, Transformation and Dynamicity", This work has been partially sponsored by the project SENSORIA, IST-2 005-016004, 2010.

Mathworks (2007), "About Aerospace Coordinate Systems".(<http://www.mathworks.com/access/helpdesk/help/toolbox/aeroblks/index.html?access/helpdesk/help/toolbox/aeroblks/f3-22568.html>). Retrieved 2007-11-23.

McCarthy J., Brayton, R., Edwards D., Fox P., Hodes L., Luckham D., Maling K., Park D. et al. (1960), LISP I Programmers Manual. Boston, Massachusetts: Artificial Intelligence Group, M.I.T. Computation Center and Research Laboratory. p. 88f. "In the local M.I.T. patois, association lists [of atomic symbols] are also referred to as "property lists", and atomic symbols are sometimes called "objects", March 1960.

Mehrdad Nojoumian and Timothy C. Lethbridge (2008), "Re-engineering PDF-Based Documents Targeting Complex Software Specifications", University of Waterloo, Canada., University of Ottawa, Canada, 2008.

Mohammed Abdul-Bari and Shahanawaj Ahamad (2011), "Process of Reverse Engineering of Enterprise Information System Architecture", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 3, September 2011 ISSN (Online): 1694-0814 www.IJCSI.org, 2011.

MSDN, Microsoft, <http://msdn.microsoft.com/vbasic/>

Nicholas T. Pilkington, Juncao Li, and Fei Xie (2010), "ESIDE: An Integrated Development Environment for Component-Based Embedded Systems", Department of Computer Science, Portland State University Portland, OR 97207, USA, 2010.

Nicolas Anquetil, Jannik Laval, and RMoD Team (2010), "Legacy Software Restructuring: Analyzing a Concrete Case", INRIA Lille-Nord-Europe, France, 2010.

NIST (2011), "The NIST Definition of Cloud Computing". National Institute of Science and Technology.<http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Retrieved 24 July 2011.

Osborne W. M. and E. J. Chikofsky (1990), "Fitting Pieces to the Maintenance Puzzle," IEEE Software, January 1990, pp. 10–11.

Peter Naur and Brian Randell (1968), "Software Engineering: Report of a conference sponsored by the NATO Science Committee" (PDF). Garmisch, Germany: Scientific Affairs Division, NATO. <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Retrieved 2008-12-26 .

Philippe Dugerdil and Julien Repcnd (2010), "Automatic Generation of Abstract Views for Legacy Software Comprehension", ISEC '10. February 25-27. 2010. Mysore. India.

Phuc V. Nguyen (2011), "The Study and Approach of Software Re-engineering", Department of Cipher & Information Technology, Ho Chi Minh City's Vietnamese Communist Party Committee, Ho Chi Minh City, Vietnam, 2011.

Pierre Bourque and Robert Dupuis (2004), "Guide to the Software Engineering Body of Knowledge", IEEE Computer Society. pp. 1–1. ISBN 0-7695-2330-7. <http://www.swebok.org>, 2004.

Rajeev Misra (2009), "Software Product Management", InfoPro Worldwide, Inc. 12 Perrine Road, Suite # 104 Monmouth Jn., NJ 08852, USA, Tel: 1 (609) 606 9020, Fax: 1 (609) 452 0335, Web site : www.infoproworldwide.com

Rikard Land (2002), "Measurements of Software Maintainability", Department of Computer Engineering, Mälardalen University, 2002.

Roger S. Pressman (2005), Software Engineering: A Practitioner's Approach (Sixth, International ed.), McGraw-Hill Education.

Rosenberg H. Linda (2010), "Software Re-engineering" Software Assurance Technology Center, Linda.Rosenberg@gsfc.nasa.gov.

S.K.Mishra, D.S.Kushwaha and A.K.Misra (2009), "Creating Reusable Software Component from Object-Oriented Legacy System through Reverse Engineering", journal of object technology, Online at <http://www.jot.fm>., 2009.

Sandhya Tarar and Ela Kumar (2012), "Design Paradigm and Risk Assessment of Hybrid Re-engineering with an approach for development of Re-engineering Metrics", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.

- Satwinder Singh, K.S.Kahlon, and Parvinder S. Sandhu (2010)**, "Re-engineering To Analyze and Measure Object Oriented Paradigms", IEEE, 2010.
- SEI.cmu**, SEI (Software Engineering Institute), <http://www.sei.cmu.edu/>
- Shoham Y. (1990)**, "Agent-Oriented Programming", Technical Report STAN-CS-90-1335, Stanford University, Computer Science Department, 1990.
- Simon Denier, Foutse Khomh, and Yann-Gaël Gueheneuc (2008)**, "Reverse Engineering the Literature on Design Patterns and Reverse-Engineering", PTIDEJ Team DIRO, Universite de Montreal and DGIGL, Ecole Polytechnique de Montreal, 2008
- Sneed H. (1995)**, "Planning the Reengineering of Legacy Systems," IEEE Software, January 1995, pp. 24–25.
- Sneed H. (1998)**, "Measuring the Reusability of Legacy Software Systems“, Software Process – Improvement and Practice, Wiley Pub., No. 4, March, 1998, p. 43
- Sneed H. and Nyary E. (1994)**, "Downsizing large Application Programs“, Journal of Software Maintenance, Vol. 6, No. 5, Oct. 1994, p. 235-248
- Steve McConnell (2006)**, "instantly indispensable Software Estimation": book, 2006.
- Stewart T. A. (1993)**, "Reengineering: The Hot New Managing Tool," Fortune, August 23, 1993, pp. 41–48.
- Swanson E. B. (1976)**, "The Dimensions of Maintenance,"Proc. Second Intl. Conf. Software Engineering, IEEE, October 1976, pp. 492–497.
- T. Hillman Willis and Ann Hillary Willis-Brown (2002)**, "Extending the value of ERP", Industrial Management & Data Systems, Vol. 102 Iss: 1, pp.35 - 38
- Thomas Erl (2005)**, "Service-Oriented Architecture: Concepts, Technology, and Design", book, Kindle edition, 2005.
- Umesh Banodha and Kanak Saxena (2011)**, "A Software Architecture Style for Medical Process Re-engineering", Proceedings of the World Congress on Engineering and Computer Science 2011 Vol I, WCECS 2011, October 19-21, 2011, San Francisco, USA
- V. S. Coerne et al. (2010)**, "Automotive Software: Characteristics and Re-engineering Challenges", Corporate Sector Research and Advance Engineering Software, P.O. Box 30 02 40, 70442 Stuttgart, Germany, paper, 2010.
- V. Carchiolo et al. (2010)**, "Object-oriented re-engineering of manufacturing models: A case study", Springer Science, Business Media, LLC, paper, 2010

Verts, William T. (2008), "Open source software". World Book Online Reference Center, 2008-01-13.

Vincent Schulte Coerne, Andreas Thums, Jochen Quante and Robert Bosch GmbH (2010), "Automotive Software: Characteristics and Re-engineering Challenges", Corporate Sector Research and Advance Engineering Software P.O. Box 30 02 40, 70442 Stuttgart, Germany, 2010.

Vincenza Carchiolo, Sebastiano D'Ambra, Alessandro Longheu and Michele Malgeri(2010), "Object-oriented re-engineering of manufacturing models: A case study", Springer Science + Business Media, LLC, 2010.

Wallace Bruce (2010), "A hole for every component, and every component in its hole, Existential Programming", There is no such thing as a Component, (May 19, 2010.(

Wiegers and Karl E. (2003), "Software Requirements", Second Edition. Microsoft Press. ISBN 0-7356-1879-8 .

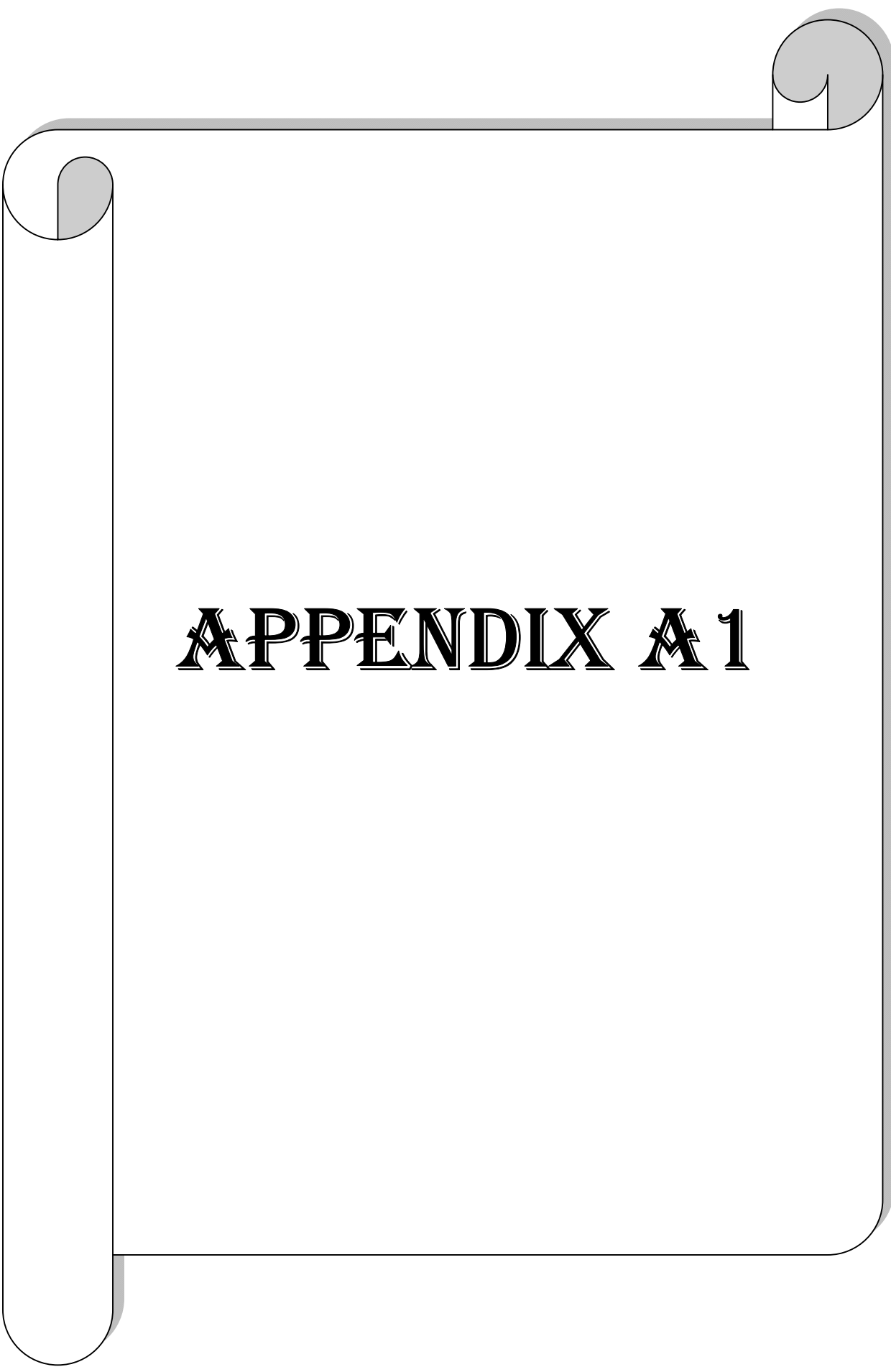
William M. Mongan, Maxim Shevertalov and Spiros Mancoridis (2008), "Re-engineering a Reverse Engineering Portal to a Distributed SOA", Drexel University, Department of Computer Science, 3141 Chestnut Street, Philadelphia, PA 19104, fwmm24, ms333, spirosg@drexel.edu, 2008.

Xiaohu Yang et al. (2005), "A Dual-Spiral Reengineering Model for Legacy System", TENCON 2005 - 2005 IEEE Region 10 Conference ISBN: 0780393112 Year: 2005 Pages: 1-5 Provider: IEEE Publisher: IEEE.

Yang LI, Hongji Yang, and William Chu (2001), "Information Elicitation From Software Code", World Scientific Publishing Company, 2001.

Zhang Zhuopeng, Ruimin Liu and Hongji Yang (2005), "Service Identification and Packaging in Service Oriented Re-engineering", De Montfort University, Leicester, LE1 9BH, England.

Zou Ying (2003), "Techniques and Methodologies for the Migration of Legacy Systems to Object Oriented Platforms", University of Waterloo, thesis.



APPENDIX A1

APPENDIX A1

THE UML ACTIVITY DIAGRAMS OF LEGACY SUPERMARKET SYSTEM

The “Visustin v7” analysis tool was used to visualize the code of the legacy supermarket management system with flow charts; Visustin converts source code to flow charts to understand existing code. Now the analysis process start by analyzing the codes of the main window to convert the code of “fmEmployee.frm” to UML activity diagrams as shown in Fig.A1.1.a, Fig.A1.1.b, Fig.A1.1.c, and Fig.A1.1.d.

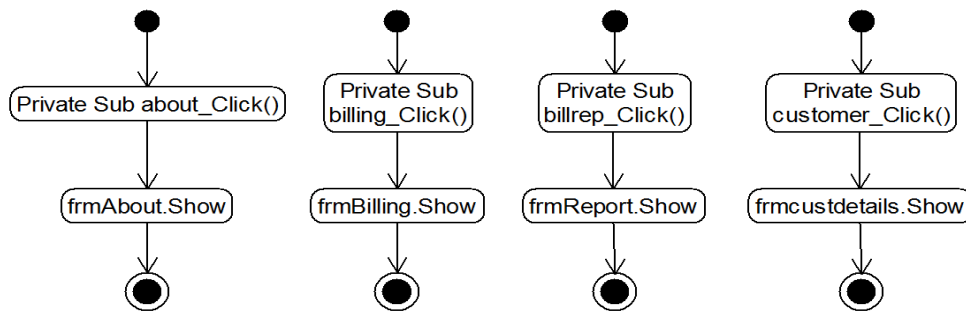


Fig.A1.1.a: UML activity diagram of the structures [about_Click(), billing_Click(), billrep_Click(), customer_Click()] in “fmEmployee.frm”

As shown in Fig.A1.1.a, when the user click on about command the about window will be displayed and that window contains the information of the developer of that legacy software, when the user click on billing command the billing window will be displayed and through this window the bill will be generated according to the Items that purchased by the customer, when the user click on billrep command, the report about the bill will be displayed and that report contain the information of purchased items, amount and total amount. When the user click on customer command, the customer detail window will be displayed, through this window the new customer can be added and his information can be updated,

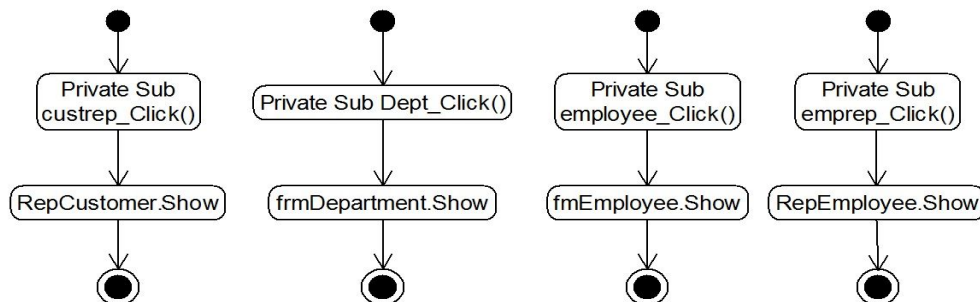


Fig.A1.1.b: UML activity diagram of the structures [custrep_Click(), dept_Click(), employee_Click(), emprep_Click()] in “fmEmployee.frm”

As shown in Fig.A1.1.b, when the user click on customer report command the report about the customer will be displayed and that report contain all information that is related to the customer. When the user click on department command the department detail window will be displayed, through this window the new department can be added and it's information can be updated. When the user click on employee command the employee details window will be displayed through this window the new employee can be added and it's information can be updated. When the user click on employee report command, the report about the employee will be displayed and that report contain all information that related to employee,

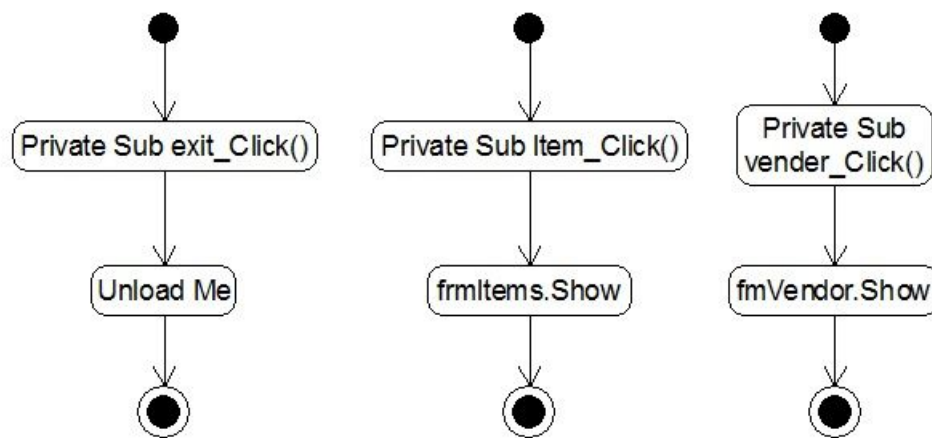


Fig.A1.1.c: UML activity diagram of the structures [exit_Click(), itm_Click(), vender_Click()] in “fmEmployee.frm”

As shown in Fig.A1.1.c, when the user click on exit command this window will be unloaded, when the user click on item command the items detail window will be displayed, through this window the new item can be added and it's information can be updated. When the user click on vendor command, the vendor detail window will be displayed, through this window the new vendor can be added and his information can be updated.

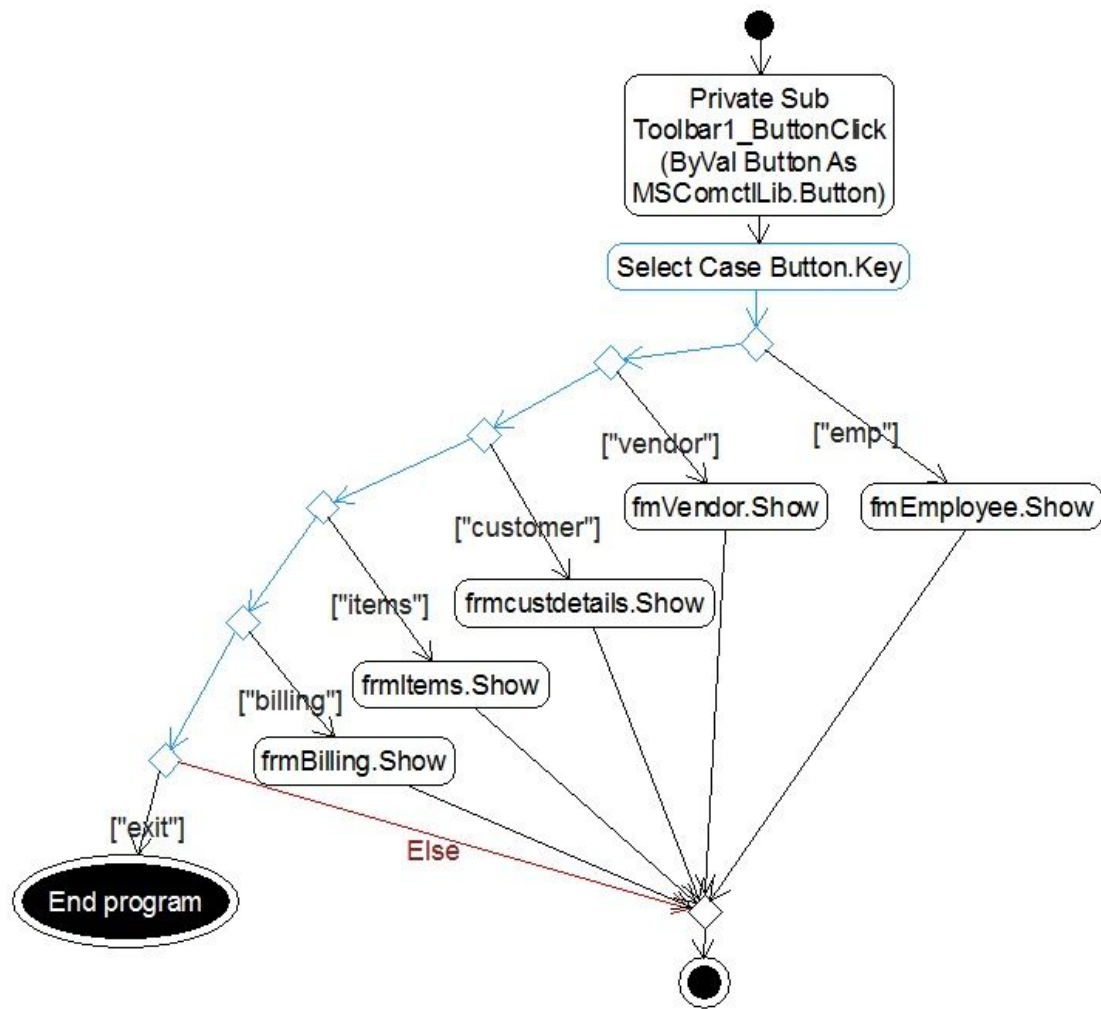


Fig.A1.1.d: UML activity diagram of the structure of Toolbar1_ButtonClick() in “fmEmployee.frm”

As shown in Fig.A1.1.d, when the user click on Toolbar1 button, the user can select one option and these options are; employee, vendor, customer, items, billing, and exit option. The next step is to analyze the codes of the employee window to convert this code to UML activity diagrams as shown in Fig.A1.2.a, Fig.A1.2.b, Fig.A1.2.c, Fig.A1.2.d, Fig.A1.2.e, Fig.A1.2.f, Fig.A1.2.g, Fig.A1.2.h, Fig.A1.2.i, Fig.A1.2.j, Fig.A1.2.k, Fig.A1.2.l, Fig.A1.2.m, Fig.A1.2.n, Fig.A1.2.o, Fig.A1.2.p, Fig.A1.2.q and Fig.A1.2.d:

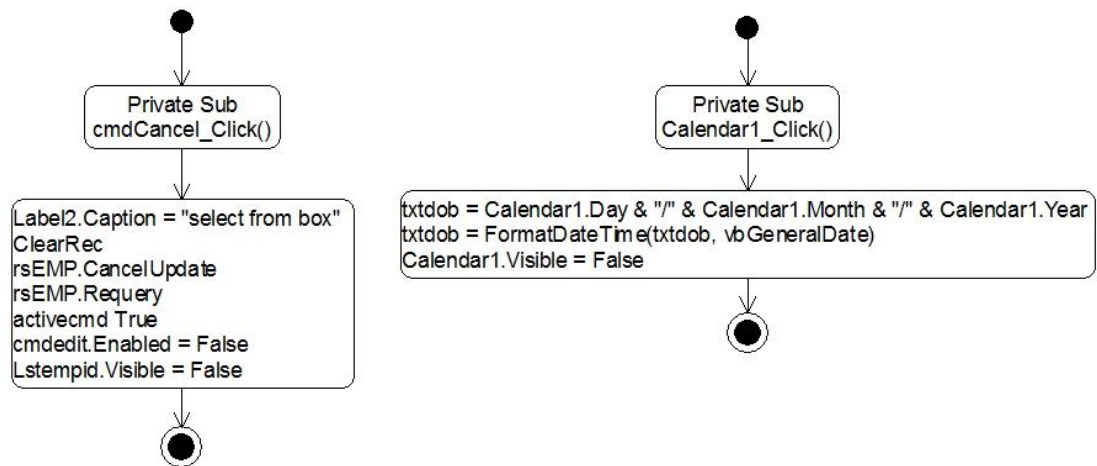


Fig.A1.2.a: UML activity diagram of the structures [cmdCancel_Click(), Calendar1_Click()]in “fmEmployee.frm”

As shown in Fig.A1.2.a, when the user click on cancel button, the message “select from box” will be displayed, record cleared, update process canceled, the edit command disabled, and the employee information will be not visible. When the user click on calendar button, the date will be determined by the user, then the calendar will be invisible.

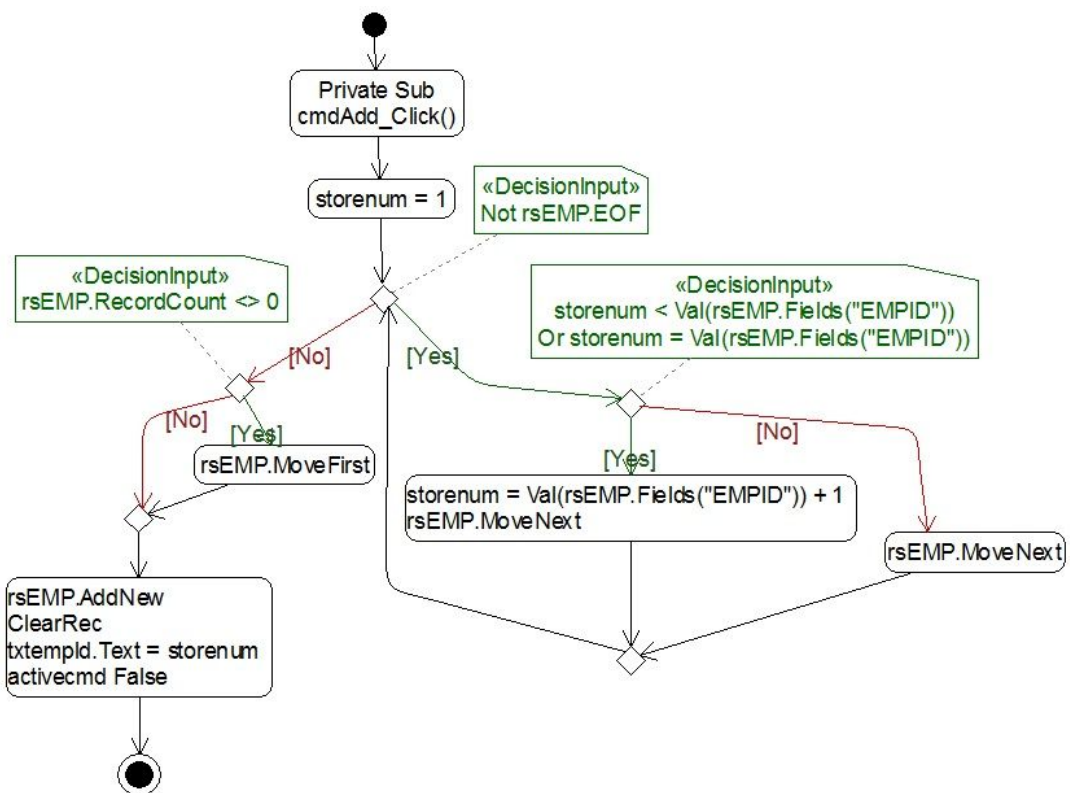


Fig.A1.2.b: UML activity diagram of the structures of cmdAdd_Click() in “fmEmployee.frm”

As shown in Fig.A1.2.b, when the user click on Add button, the employee Identification number will be generated automatically according to the above structure and the record will be cleared and will be ready to enter employee information.

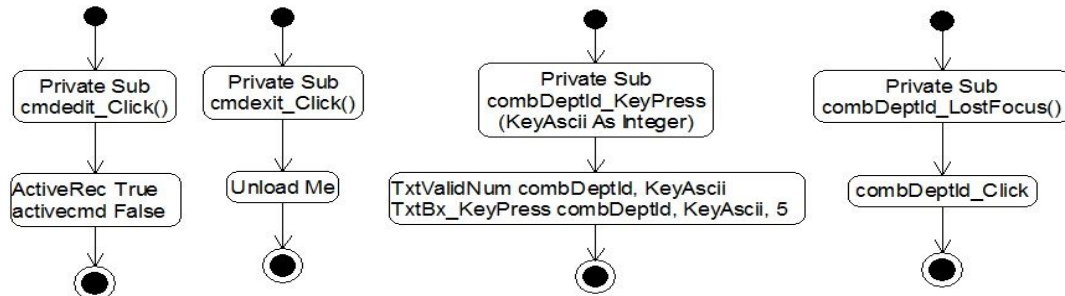


Fig.A1.2. c: UML activity diagram of the structures [cmdedit_Click(), cmdexit_Click(), combDeptId_Click(), combDeptId_Click()]in “fmEmployee.frm”

As shown in Fig.A1.2.c, when the user click on edit button, the record will be activated and commands will be deactivated until make modification and click on submit button. When the user click on exit button the employee window will be unloaded, when the user clicks on (combDeptid) combo box, list of department id will be displayed, so the user can select with department that the specified employee belong to it, after that ‘combDeptid’ will lost focus.

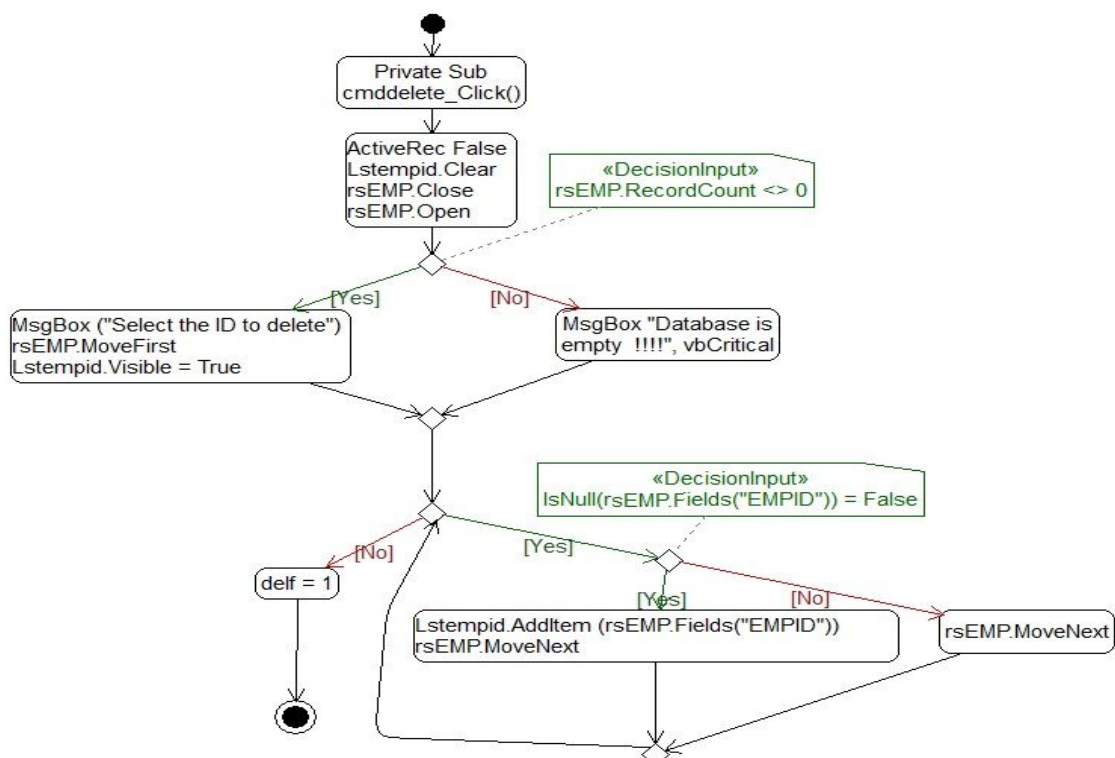


Fig.A1.2.d: UML activity diagram of the structures of cmddelete_Click() in “fmEmployee.frm”

As shown in Fig.A1.2.d, when the user click on delete button, first of all the system check if the database is empty or not, then check if the determined employee exists in the database or not, if it exists then he will be deleted and next record will be displayed.

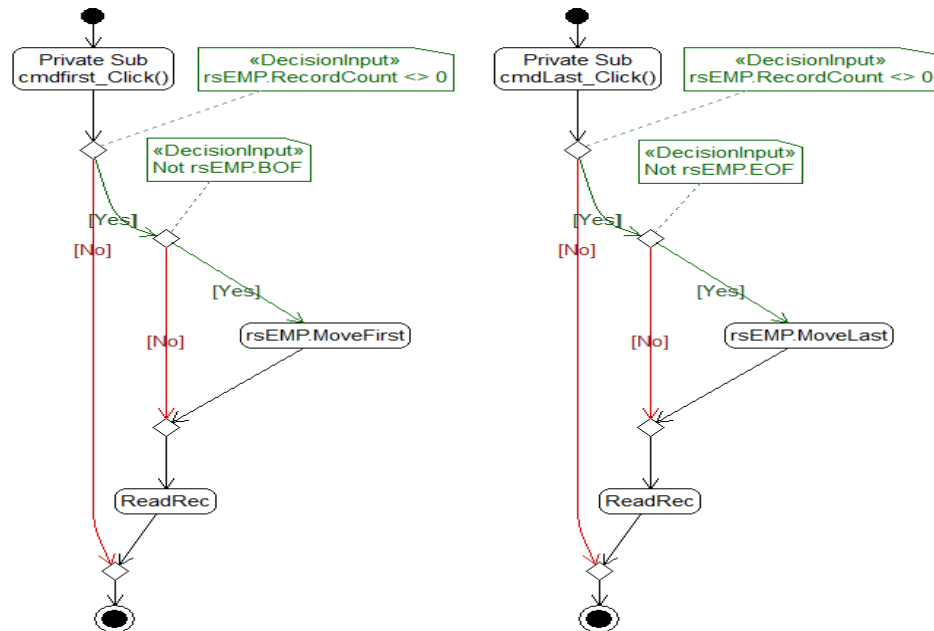


Fig.A1.2.e: UML activity diagram of the structures [cmdfirst_Click(), cmdLast_Click()] in “fmEmployee.frm”

As shown in Fig.A1.2.e, when the user clicks on first button, first of all the system checks if the database is empty or not, then moves to the first record and will read this record and display it. When the user clicks on last button, the system will check if the database is empty or not, then move to the last record and read this record and display it.

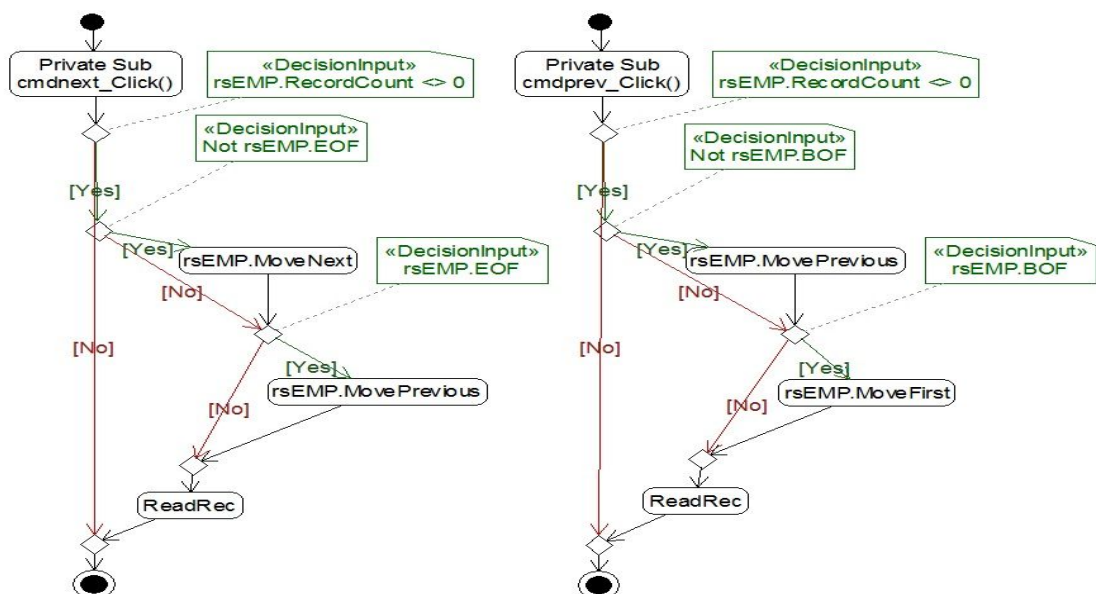


Fig.A1.2.f: UML activity diagram of the structures [cmdnext_Click(), cmdprev_Click()] in “fmEmployee.frm”

As shown in Fig.A1.2.f, when the user clicks on next button, first of all the system check if the record count is equal to zero or not, then check if database is empty or not, then move to the next record and check if it is end of file, then move to previous otherwise read this record and display it. When the user clicks on previous button, first of all the system check if the record count is equal to zero or not, then check if the database is empty or not, then move to the previous record and check if it is end of file then move to next, otherwise read this record and display it.

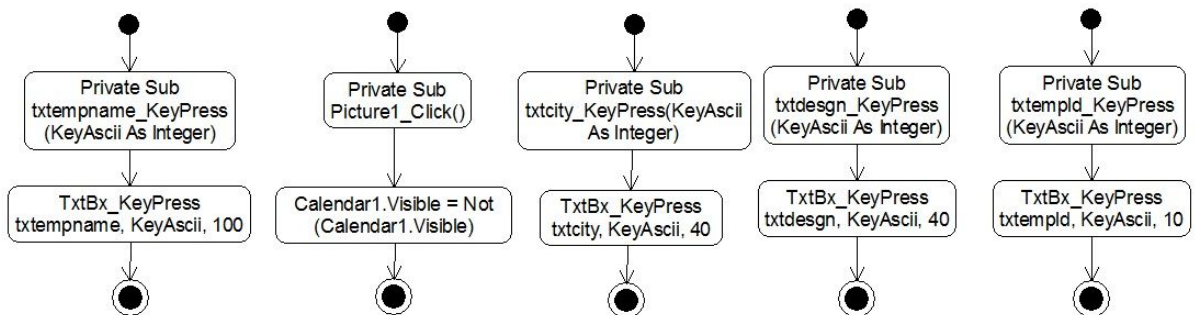


Fig.A1.2.g: UML activity diagram of the structures [txttempname_KeyPress(KeyAscii As Integer), Picture1_Click(), txtcity_KeyPress(KeyAscii As Integer), txtdesign_KeyPress(KeyAscii As Integer), txtempld_KeyPress(KeyAscii As Integer)] in “fmEmployee.frm”

As shown in Fig.A1.2.g, when the user perform KeyPress on employee name textbox, the keyAscii returned integer value equal to 100, when the user click on picture1 the calendar will be displayed to determine the date. When the user perform KeyPress on city textbox, the keyAscii returned integer value equal to 40, when the user perform KeyPress on designation textbox the keyAscii returned integer value equal to 40, when the user perform KeyPress on employee identification number textbox the keyAscii returned integer value equal to 10.

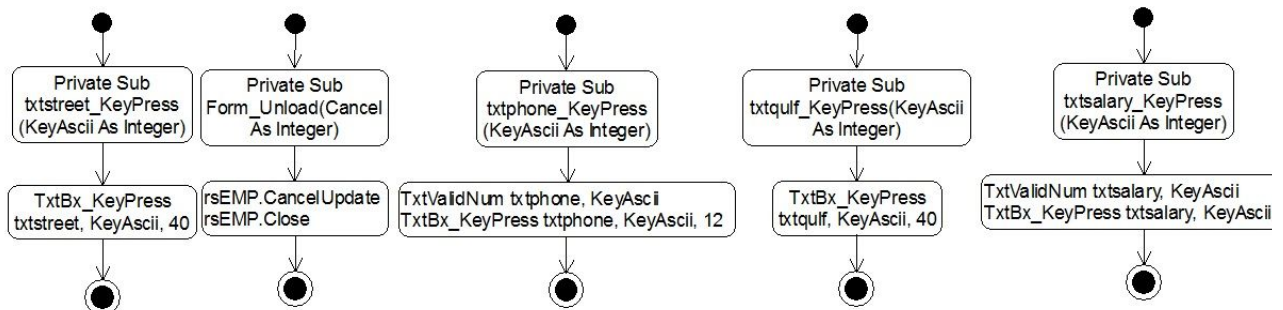


Fig.A1.2.h: UML activity diagram of the structures [txtstreet_KeyPress(KeyAscii As Integer), Form_Unload(Cancel As Integer), txtphone_KeyPress(KeyAscii As Integer), txtqulf_KeyPress(KeyAscii As Integer), txtsalary_KeyPress(KeyAscii As Integer)] in “fmEmployee.frm”

As shown in Fig.A1.2.h, when the user perform KeyPress on street textbox the keyAscii returned integer value equal to 40, when the user click on close the update will be cancelled. when the user perform KeyPress on phone textbox the keyAscii returned integer value equal to 12, when the user perform KeyPress on qualification textbox the keyAscii returned integer value equal to 40, when the user perform KeyPress on salary textbox the keyAscii returned integer value equal to 8.

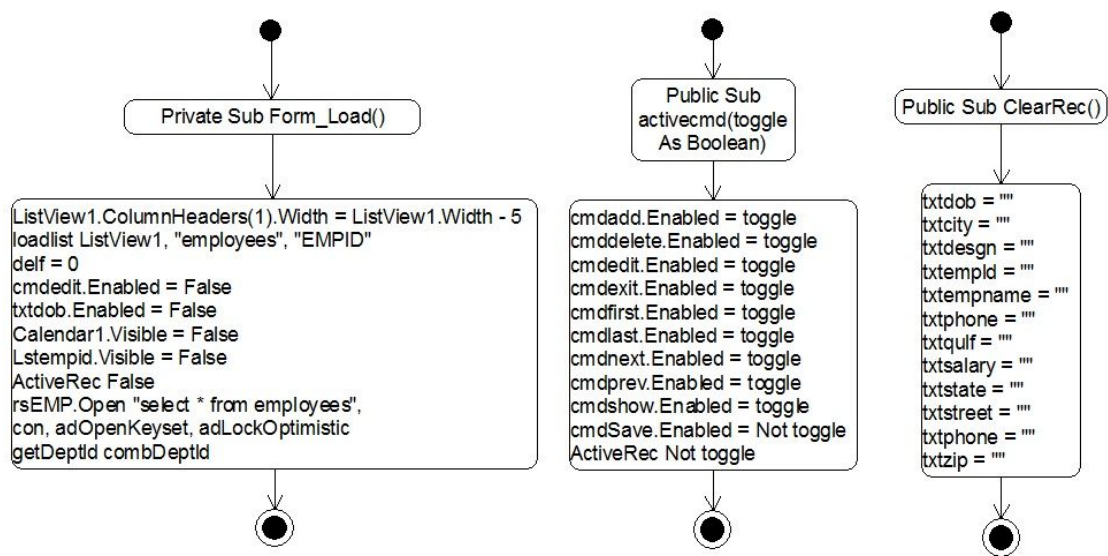


Fig.A1.2.i: UML activity diagram of the structures [Form_Load(), activecmd(toggle As Boolean), ClearRec()]in “fmEmployee.frm”

As shown in Fig.A1.2.i, when the employee form is loaded, the width of column header of list view1 will be determined, then load list items as ‘employee, EMPID’, the buttons disabled. These buttons are: cmdedit, txtdob, calender1, and lstempid, then read data from employee table. When the user click on activecmd all the button will be activated and these buttons are: add, delete, edit, first, last, next, previous, and show, but save button and record are inactive . clearRec is a function to clear all the text box and make them empty to be ready.

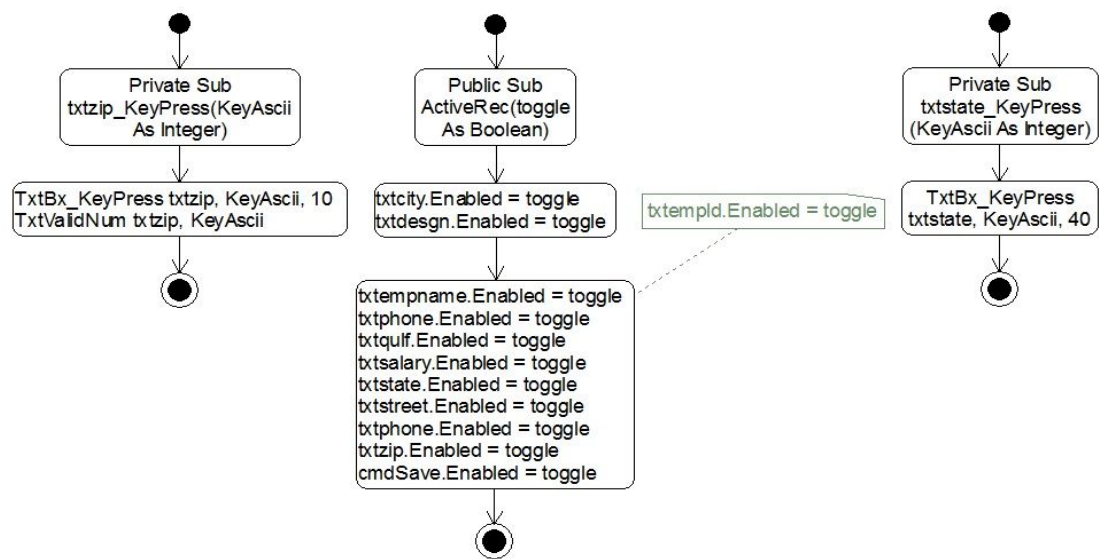


Fig.A1.2.j: UML activity diagram of the structures [txtzip_KeyPress(KeyAscii As Integer), ActiveRec (toggle As Boolean), txtstate_KeyPress(KeyAscii As Integer)] in “fmEmployee.frm”

As shown in Fig.A1.2.j, when the user perform KeyPress on zip textbox the keyAscii returned integer value equal to 10, when the user perform KeyPress on state textbox the keyAscii returned integer value equal to 40. When activeRec function is done all textboxes are activated. These textboxes are for: city, employee name, phone, qualification, salary, state, street, zip, and designation, save button will also be activated.

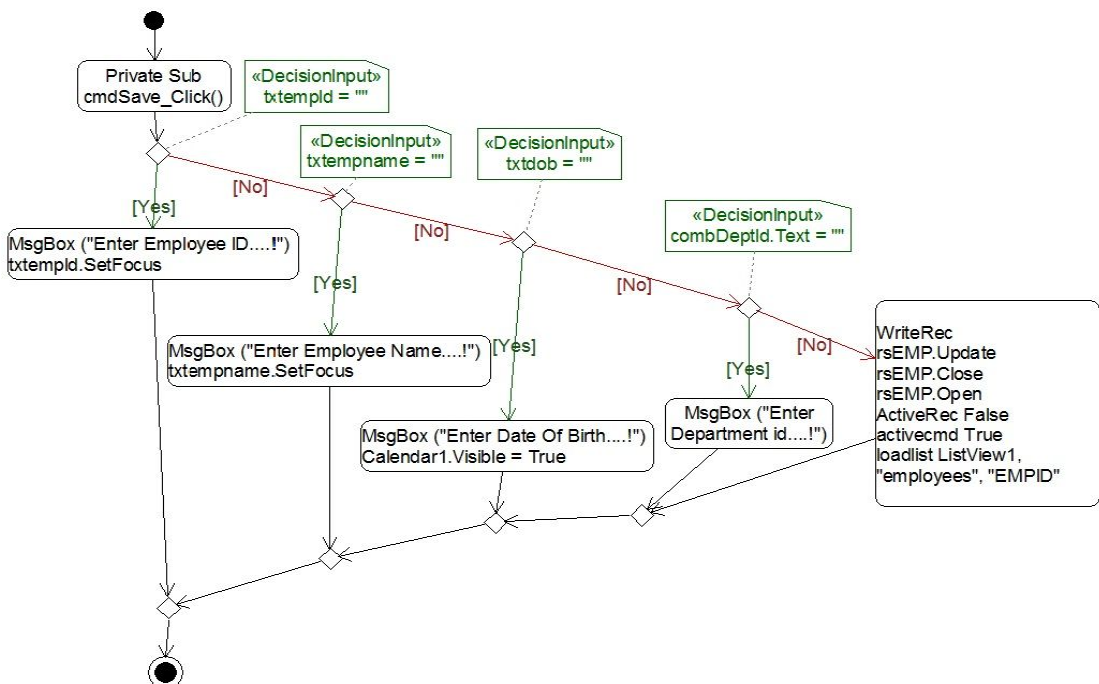


Fig.A1.2.k: UML activity diagram of the structures of cmdSave_Click()in “fmEmployee.frm”

As shown in Fig.A1.2.k, when the user clicks on submit button (i.e to save), first of all the system will check all the textboxes one by one that are not empty, then the record will be saved in the employee table within the database, otherwise the text message will be appeared to the user to inform him about the missing information that he must enter them.

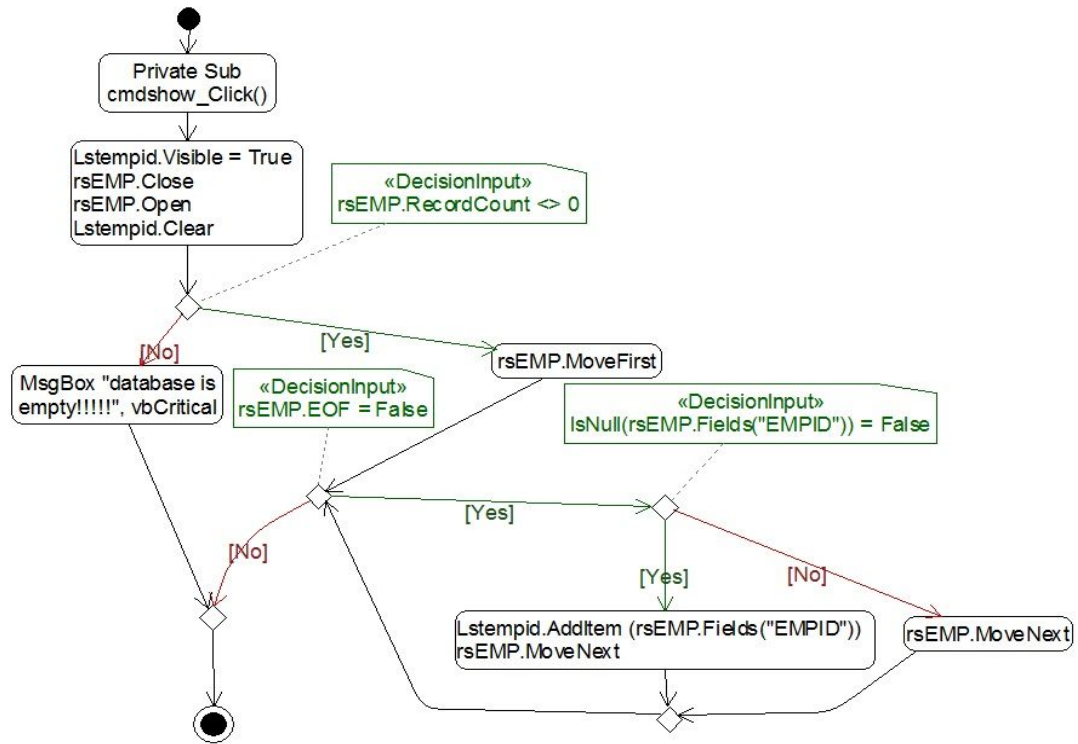


Fig.A1.2.l: UML activity diagram of the structures of cmdshow_Click()in
“fmEmployee.frm”

As shown in Fig.A1.2.l, when the user clicks on show button, first of all the list of employee identification will be visible and cleared, then the system check the database is empty or not, if it is empty then message to user will appear to inform him that the database is empty, otherwise move to first record and move step by step to find the required employee according to selected id to be displayed.

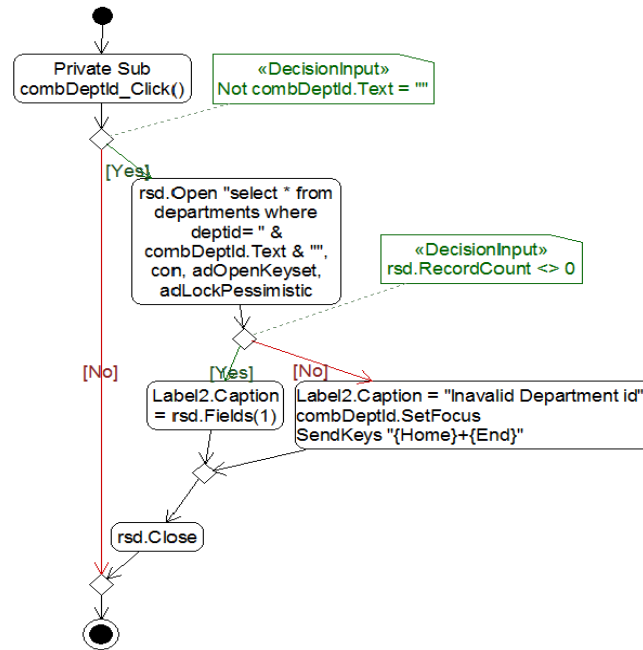


Fig.A1.2.m: UML activity diagram of the structures of combDeptid_Click() in “fmEmployee.frm”

As shown in Fig.A1.2.m, when the user clicks on ‘combDeptid’ combo box, first of all check the combo box of department identification is not empty, then the system fetch the department name from the database and display in ‘label2’ that if it exist, otherwise wrong message will be displayed to inform the user that it is invalid department id, after that the connection is closed.

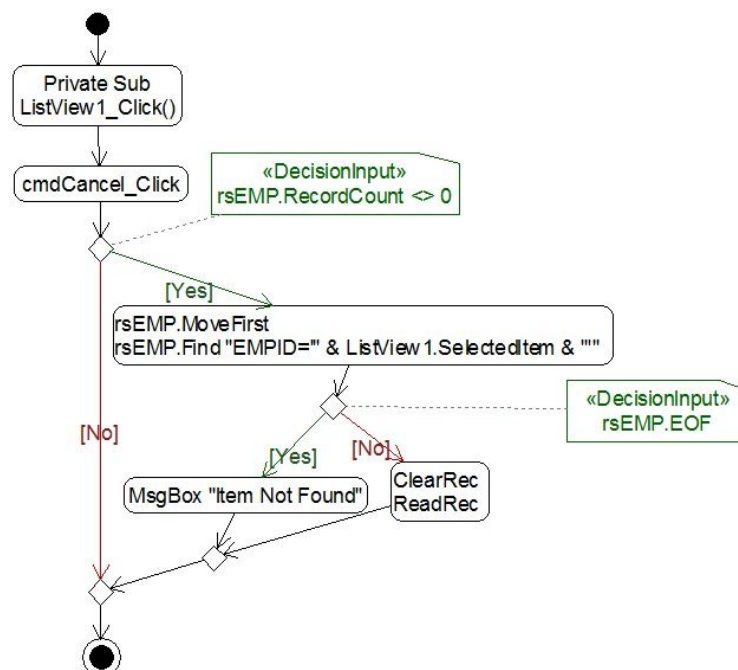


Fig.A1.2.n: UML activity diagram of the structures of ListView1_Click() in “fmEmployee.frm”

As shown in Fig.A1.2.n, when the user clicks on 'listView1', first of all clear all the textboxes and check the record count not equal to zero, then move to first record and find the employer that his id is equal to selected item, if it exist then display his information, else display message that says 'item not found'.

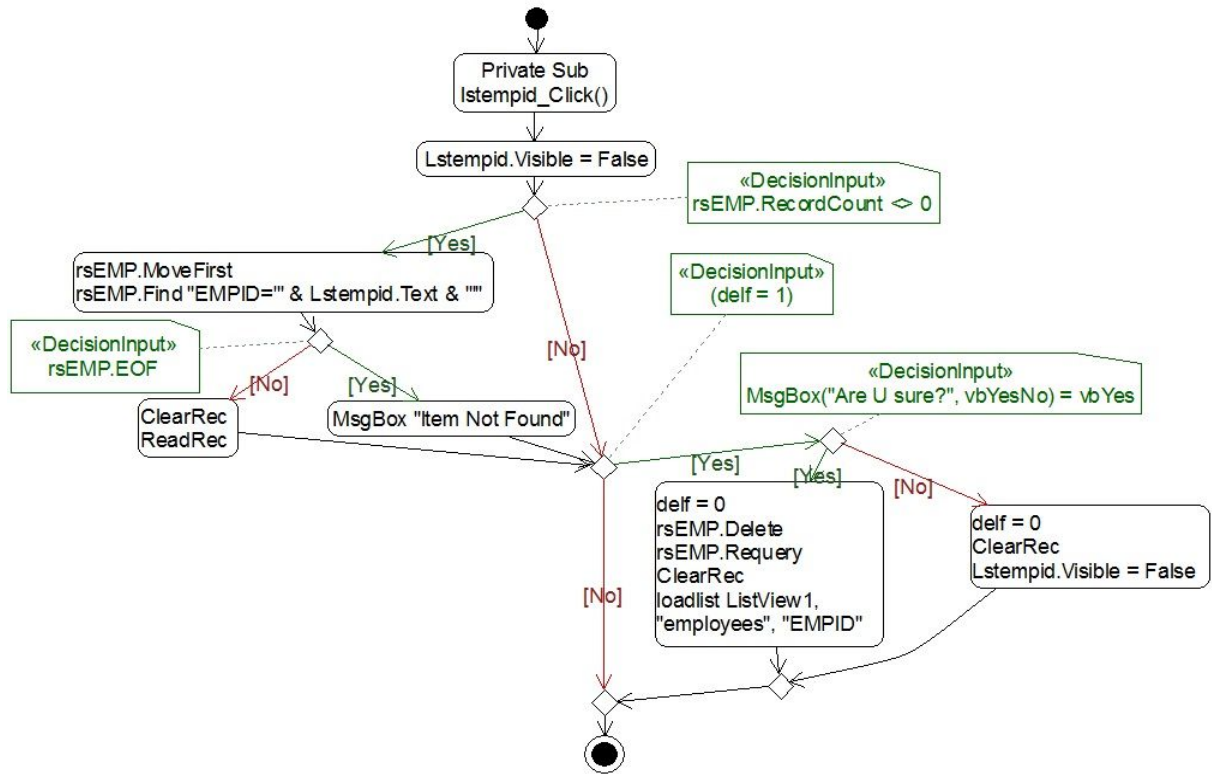


Fig.A1.2.o: UML activity diagram of the structures of Istempid_Click() in
"fmEmployee.frm"

As shown in Fig.A1.2.o, when the user clicks on 'listempId' list, first of all 'listempId' will be invisible then check record count not equal to zero then move to first record and find the employer that his id equal to selected item, if it exist then display his information, else display message that says 'item not found'. Else if record count equal to zero then remove this record according to user choice (yes or no).

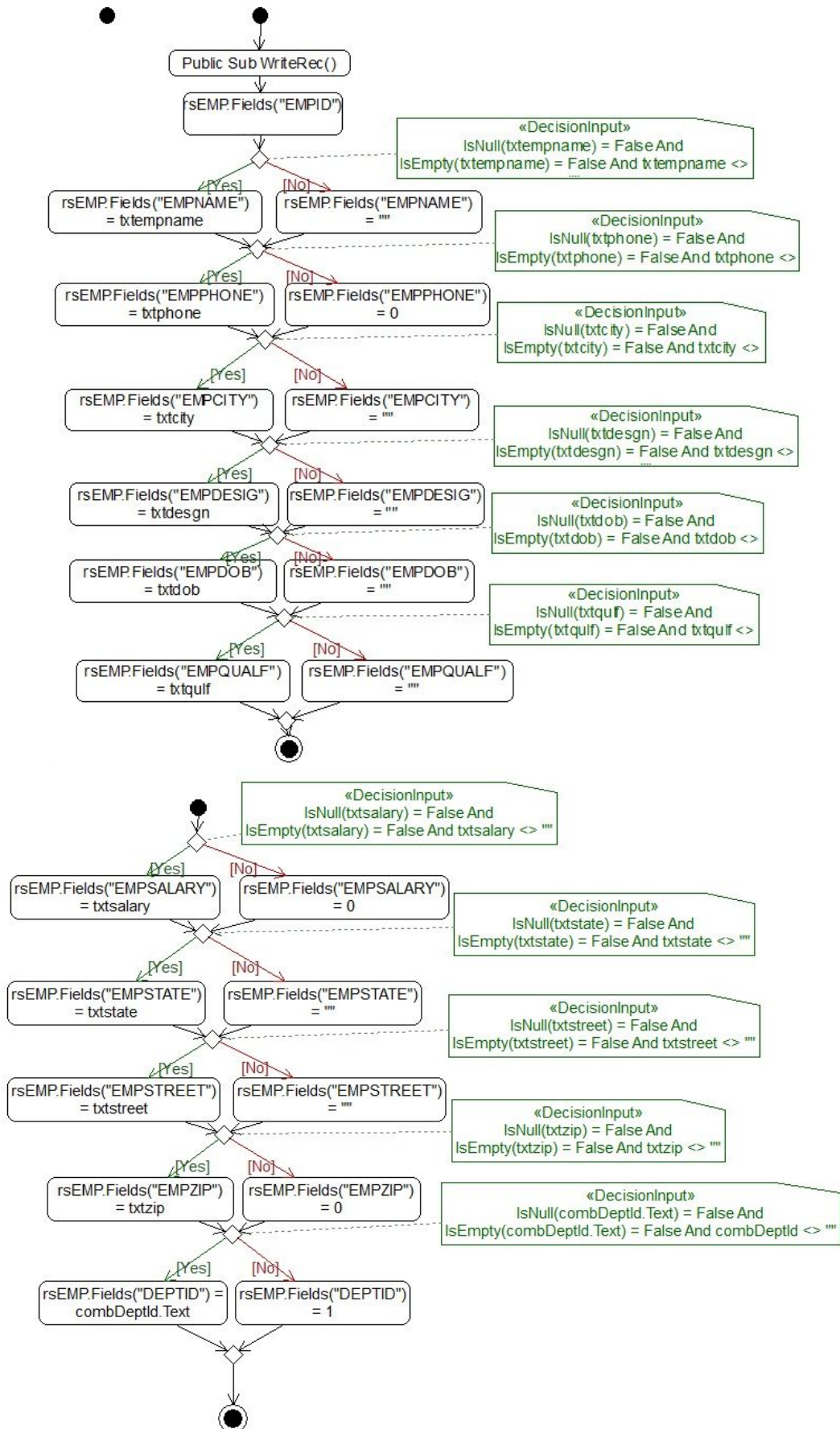


Fig.A1.2.p: UML activity diagram of the structures of WriteRec() in “fmEmployee.frm”

As shown in Fig.A1.2.p, when the WriteRec() function was executed, all field about employee id, name, city, phone, zip, designation, salary, qualification, and department Id will be checked and saved in the employee table.

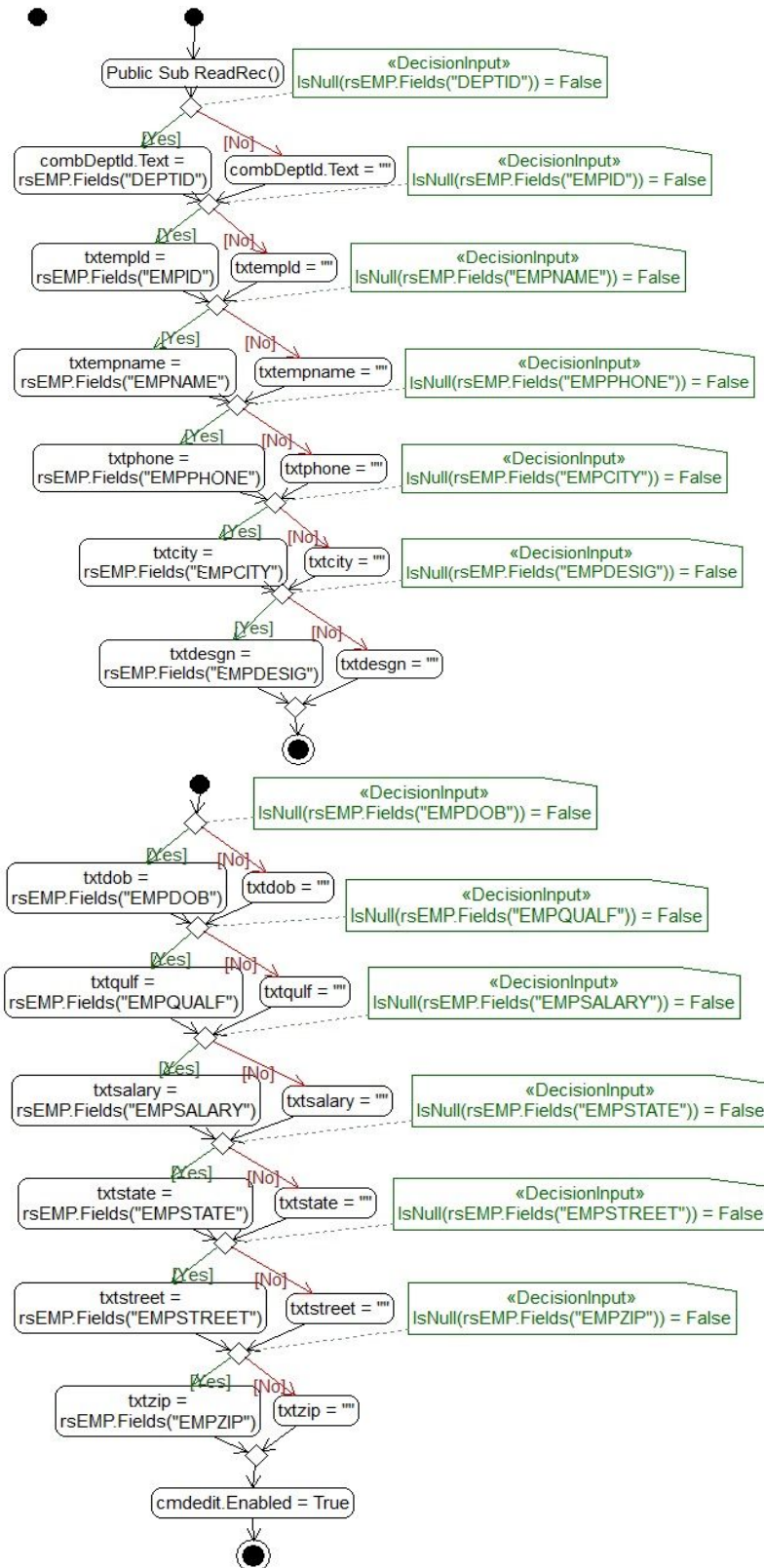


Fig.A1.2.q: UML activity diagram of the structures of ReadRec() in “fmEmployee.frm”,

As shown in Fig.A1.2.q, when the ReadRec() function was executed, all field about employee id, name, city, phone, designation, salary, qualification, and zip code will be checked and displayed on the corresponding textboxes in the employee form.

The next step is to analyze the codes of the billing window to convert this code to UML activity diagrams as shown in Fig.A1.3.a, Fig.A1.3.b, Fig.A1.3.c, Fig.A1.3.d, Fig.A1.3.e, Fig.A1.3.f, Fig.A1.3.g, Fig.A1.3.h, Fig.A1.3.i, Fig.A1.3.j-1 and Fig.A1.3.j-2:

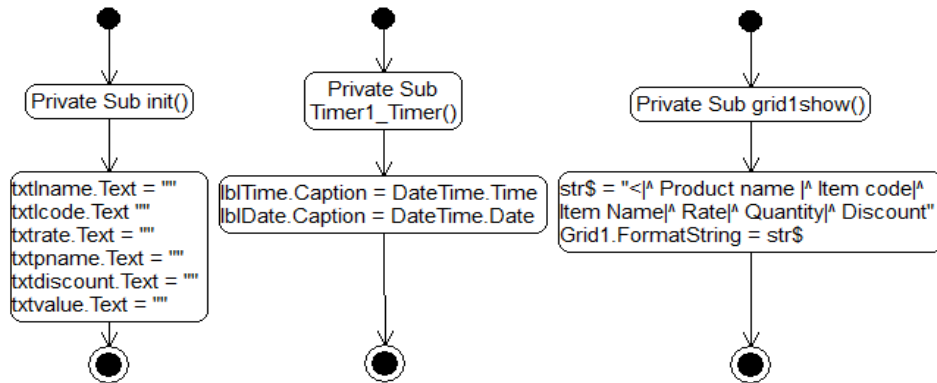


Fig.A1.3.a: UML activity diagram of the structures of [init (), Timer1_Timer (), grid1show()] in “frmBilling”

As shown in Fig.A1.3.a, When the init() function is executed, all the textboxes of billing form will be cleared. When the timer() function of timer1 is executed, the caption of lblTime label will be equal to time and the caption of lblDate label will be equal to date, When the grid1show() function is executed, all the data of billing table will be displayed on the grid.

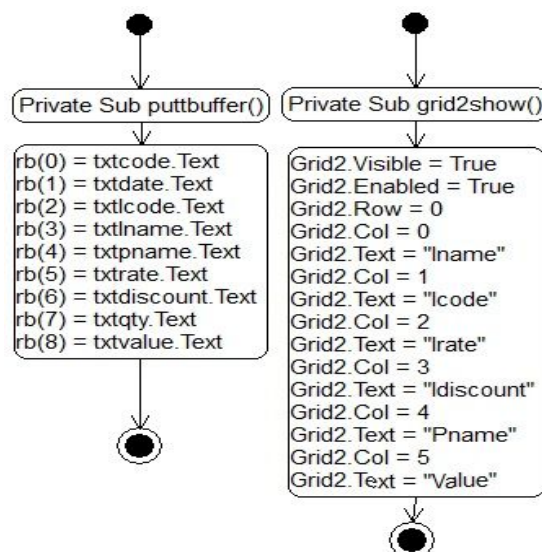


Fig.A1.3.b: UML activity diagram of the structures of [puttbuffer (), grid2show()] in “frmBilling”

As shown in Fig.A1.3.b, When the putbuffer () function executed, all the textboxes of billing form will be saved temporarily on list as a buffer. When the grid2show () function executed, all the data of billing table will be displayed on the grid as a table.

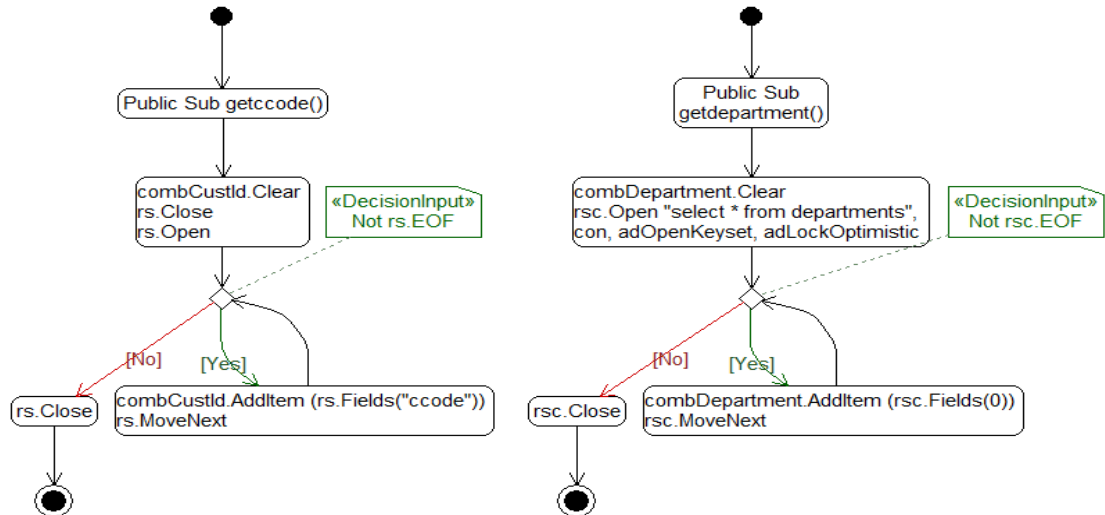


FIG.A1.3.c: UML activity diagram of the structures of [getccode (), getdepartment()] in “frmBilling”

As shown in Fig.A1.3.c, When the getccode () function is executed, the customer identification code will be fetched for the customer id combo box. When the getdepartment () function is executed, the department identification code will be fetched for the department id combo box.

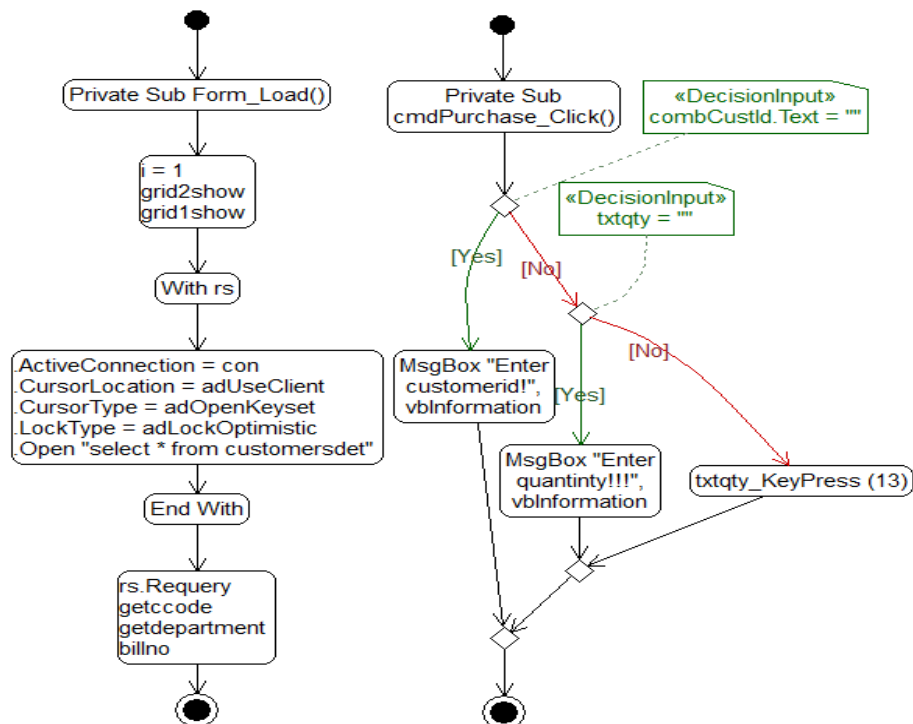


Fig.A1.3.d: UML activity diagram of the structures of [Form_Load (), cmdPurchase_Click()] in “frmBilling”

As shown in Fig.A1.3.d, When the billing form is loaded, first of all it set I equal to 1, then invoke grid1show () and grid2show (), then activate connection, set the cursor type and location, then determine lock type, then select customer detail, after that invoke getccode () and getdepartment () and billno () functions. When the user clicks on the purchase button, first of all chec the combo box of customer id is not empty, then check the quantity is not empty, then invoke the txtty_keyPress(13), otherwise ask the user to enter the missing information.

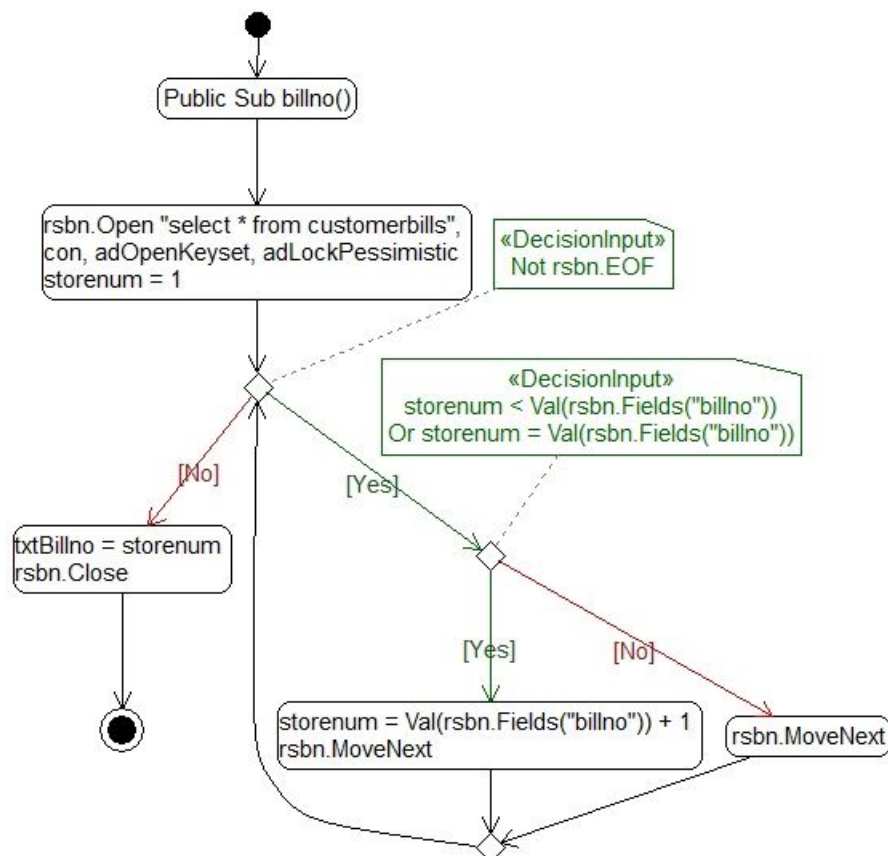


Fig.A1.3.e: UML activity diagram of the structures of billno () in “frmBilling”

As shown in Fig.A1.3.e, When the billno () function is executed, first of all it is connected to the customerbills table, then select record from the table, and set storenum equal to 1, then check it is not end of file, then check storenum less than or equal to billno, if yes then storenum equal to billno plus one, otherwise move next. Now if it is end of file then bill number textbox equal to storenum.

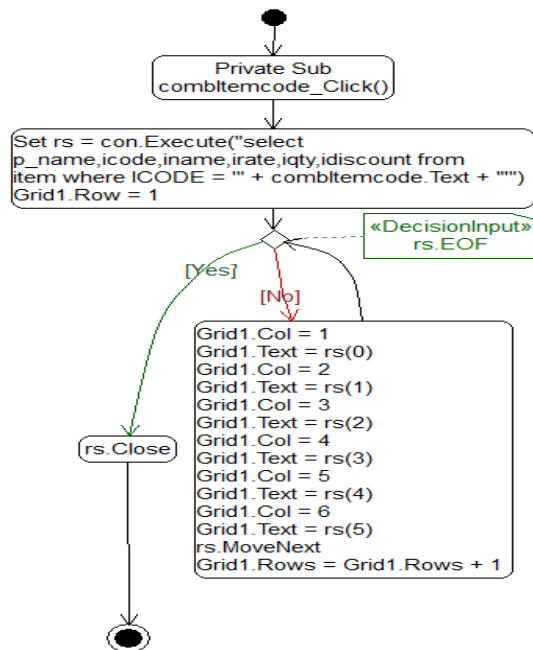


Fig.A1.3.f: UML activity diagram of the structures of combitemcode_Click () in “frmBilling”

As shown in Fig.A1.3.f, When the user clicks on item code combo box, first of all it is connected to the item table, then select record from the table according to item code, then check it is end of file, then close connection, otherwise, fill the grid field by the corresponding fetched values as product name, item code, item name, item rate, item quantity and item discount.

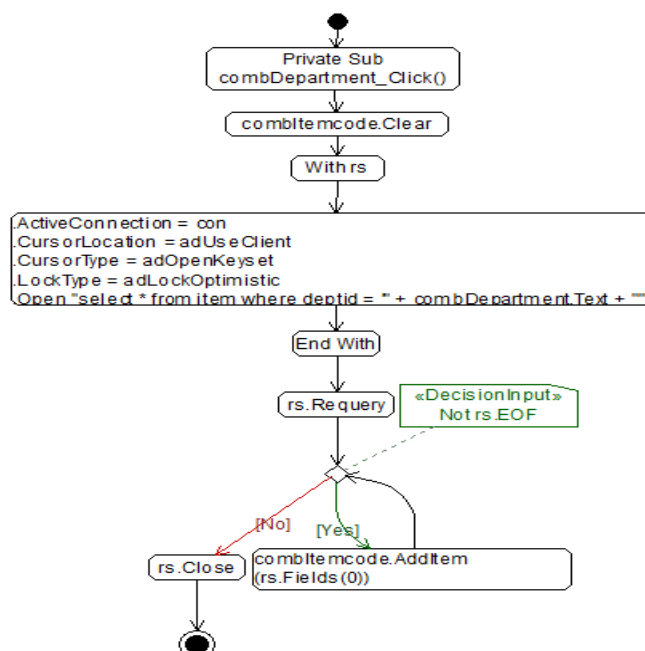


Fig.A1.3.g: UML activity diagram of the structures of combDepartment_Click () in “frmBilling”

As shown in Fig.A1.3.g, When the user clicks on department combo box, first of all it is clear item code combo box, then activate connection, set cursor location and type, and set lock type, the select record from the item table according to department id, then check it is not end of file, then add item otherwise, close connection.

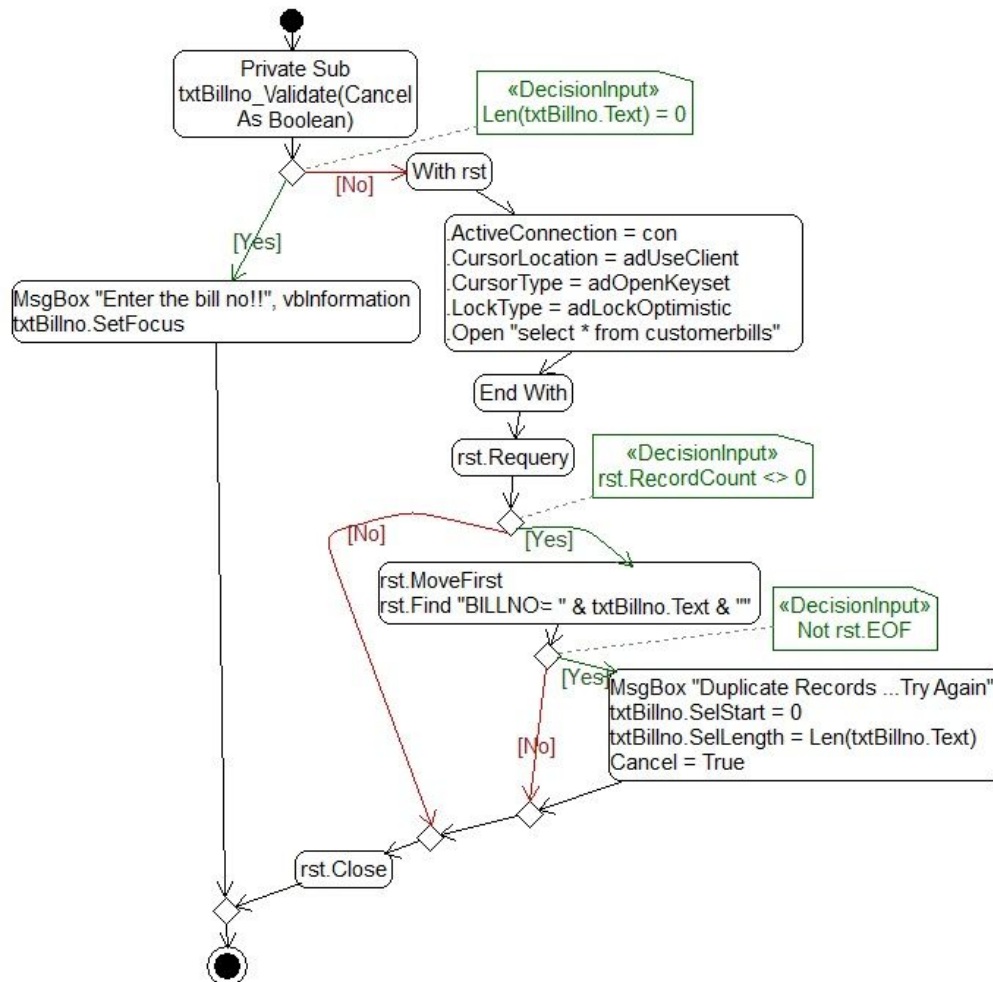


Fig.A1.3.h: UML activity diagram of the structures of txtBillno_Validate (Cancel as Boolean) in “frmBilling”

As shown in Fig.A1.3.h, When the txtBillno_Validate (Cancel as Boolean) function is executed, first of all it is check the length of the text of ‘billno’ textbox if it is equal to zero or not, if yes then a message will appear to the user inform him to enter the bill number, else if length not equal to zero, then activate connection, set cursor location and type, and set lock type, the select record from the ‘customerbill’ tables, then check record count is not equal to zero, then move to first record and find ‘billno’ according to the value that entered in the textbox, then add item otherwise, then check it is not end of file, if yes that mean there is a duplicate and make cancel true, finally close connection.

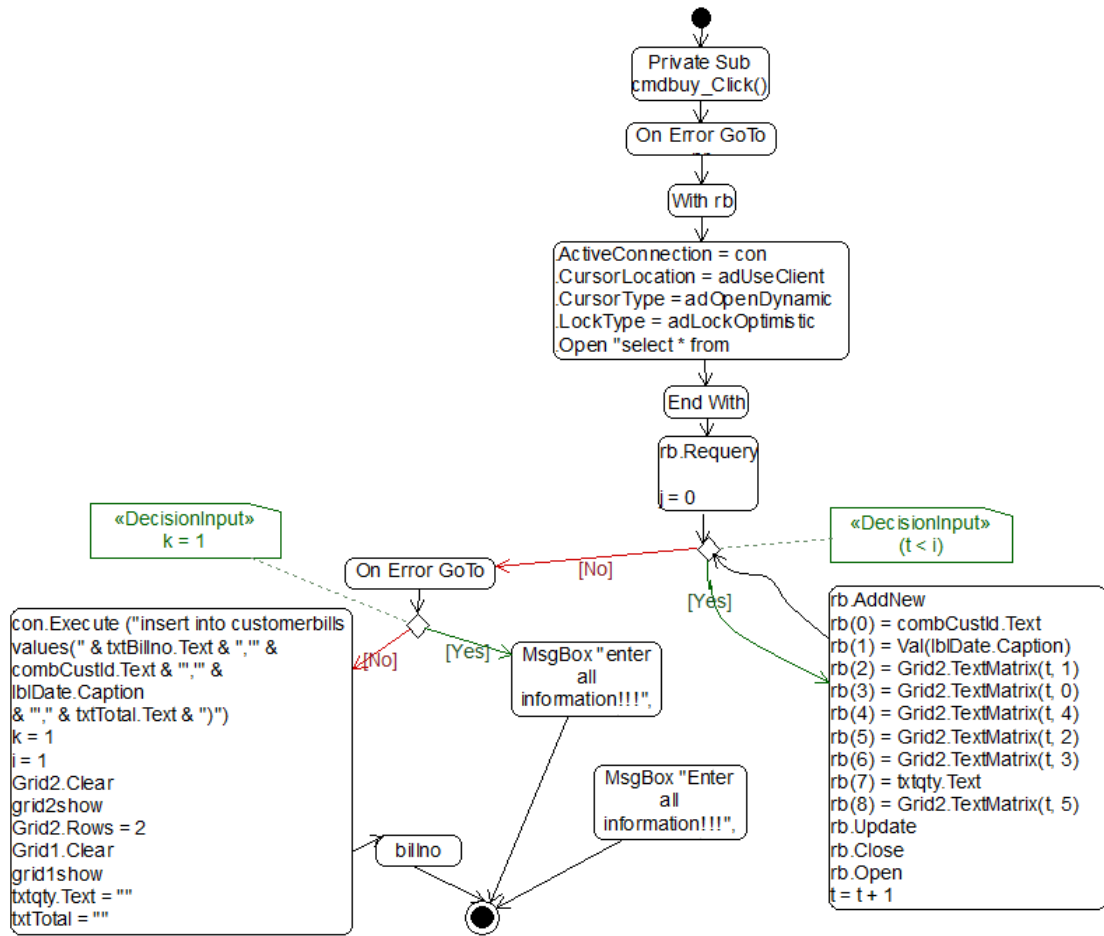
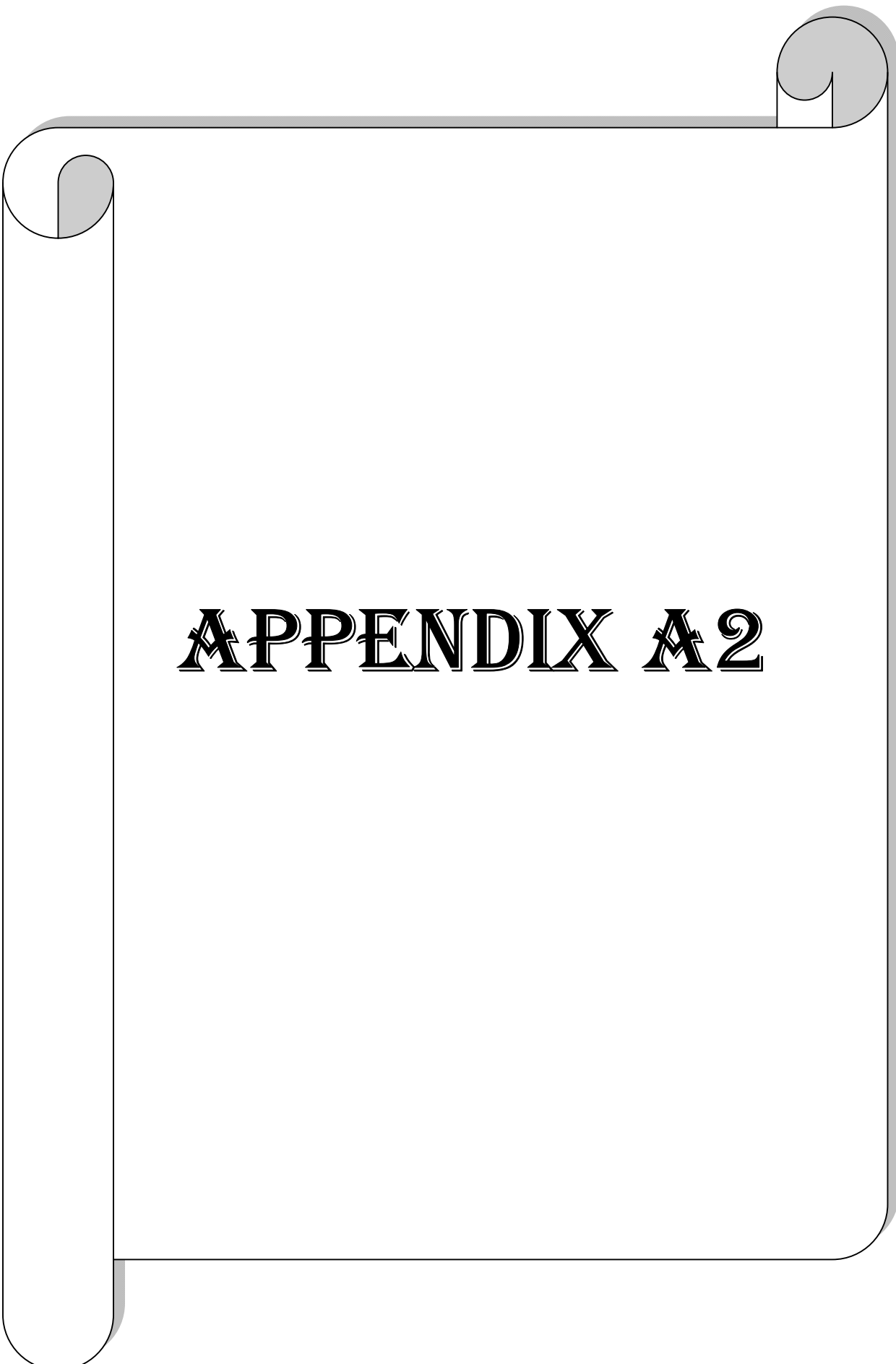


Fig.A1.3.i: UML activity diagram of the structures of cmdbuy_Click () in “frmBilling”

As shown in Fig.A1.3.i, When the user clicks on buy button, first of all it is check if there is any error go to pr to display error message, then activate connection, set cursor location and type, and set lock type, the select record from the ‘customerbill’ table, then set items to the ‘customerbill’ table, if there is some information not exist then a message will appear to user tell him that he must enter all the information, finally the purchased item will be added to the ‘customerbills’ table.

As shown in Fig.A1.3.j, When the user perform KeyPress on quantity textbox, when the quantity specified the system check if this quantity is available in the store or not, if it is available then update the item table with new quantity and update the transaction table. If it is not available a message will appear to the user tell him that this quantity is more than the available.

Then perform the same analysis process to analyze the codes of vendor details window, department window, customer details window, and item details window to convert this code to UML activity diagrams, there is no need to show the details about the analysis of these windows because the analysis process was explained and some sample was shown.



APPENDIX A2

APPENDIX A2

THE DFD, ERD, USE CASE DIAGRAMS, EXECUTION FLOWCHART AND DATABASE OF LEGACY SUPERMARKET SYSTEM

The DFD, ERD, use case diagrams, execution flowchart and database of legacy supermarket system will be illustrated in this appendix; all these diagrams and flowcharts were extracted from the legacy supermarket management system within the reverse engineering activity.

Data Flow Diagram (DFD)

The data flow diagram of the legacy supermarket system is shown below in Fig.A2.1.

Zero level DFD



Fig.A2.1: Zero level DFD of legacy supermarket system

Zero Level DFD gives a brief overview of the entire system. The **legacy Supermarket Management System** is enable the user to fill/view/print the bill that contain the items that selected by customer and compute the total amount of the bill then bill generated and stock master updated by the number of items that sold, in this legacy system there is no security, any user can use the system. After performing the required transactions on the system and generating the bill, the user can close the system or start to provide service to another customer.

First level DFD

First level DFD of a system gives a broad picture of all the processes of the system and the data transformation within the system. 1st Level Data Flow Diagram was shown in Fig.A2.2, The major processes involved in this system are as follows:

- 1) Order process.
- 2) Bill reparation.
- 3) Manage employee's details.
- 4) Manage vendor's details.
- 5) Manage customer's details.
- 6) Manage department's details.
- 7) Manage item's details

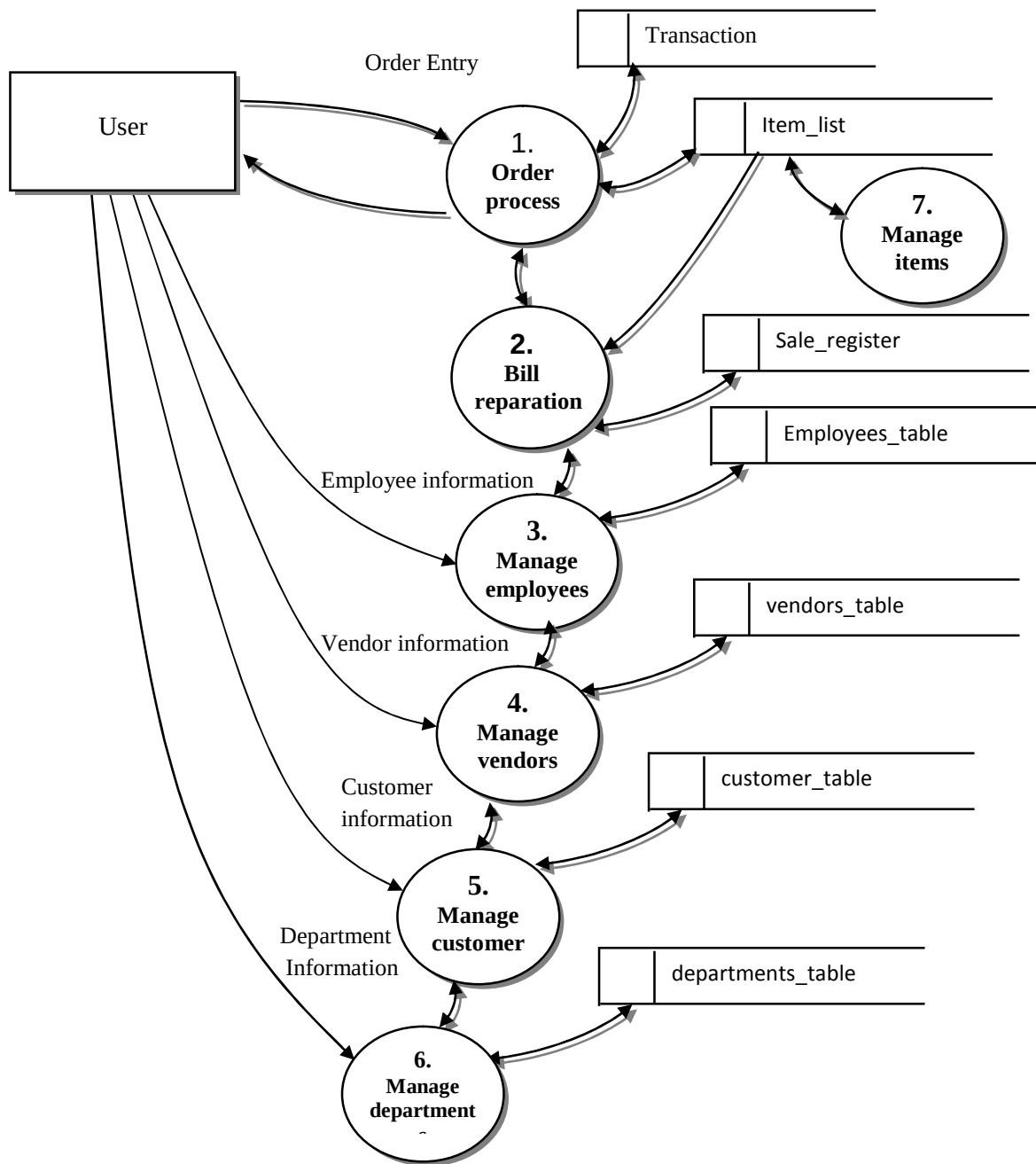
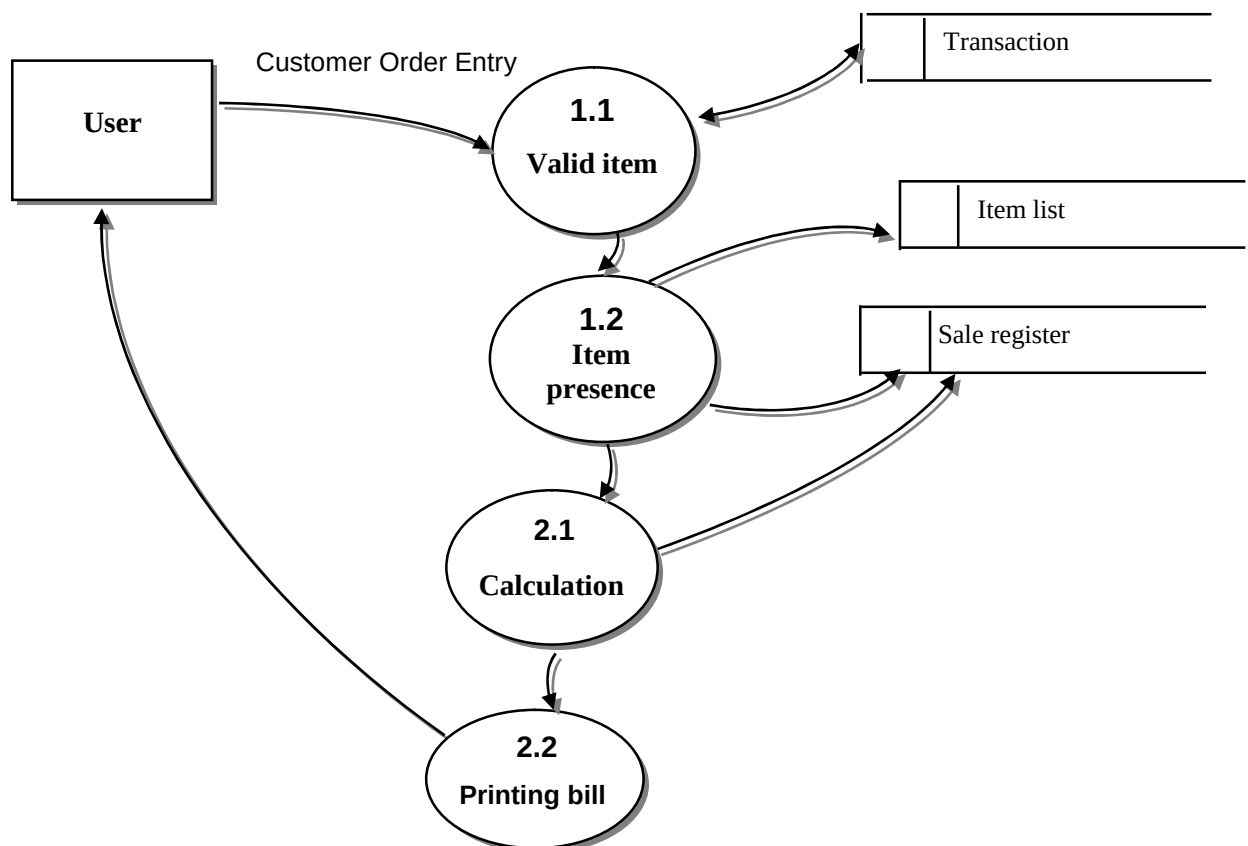
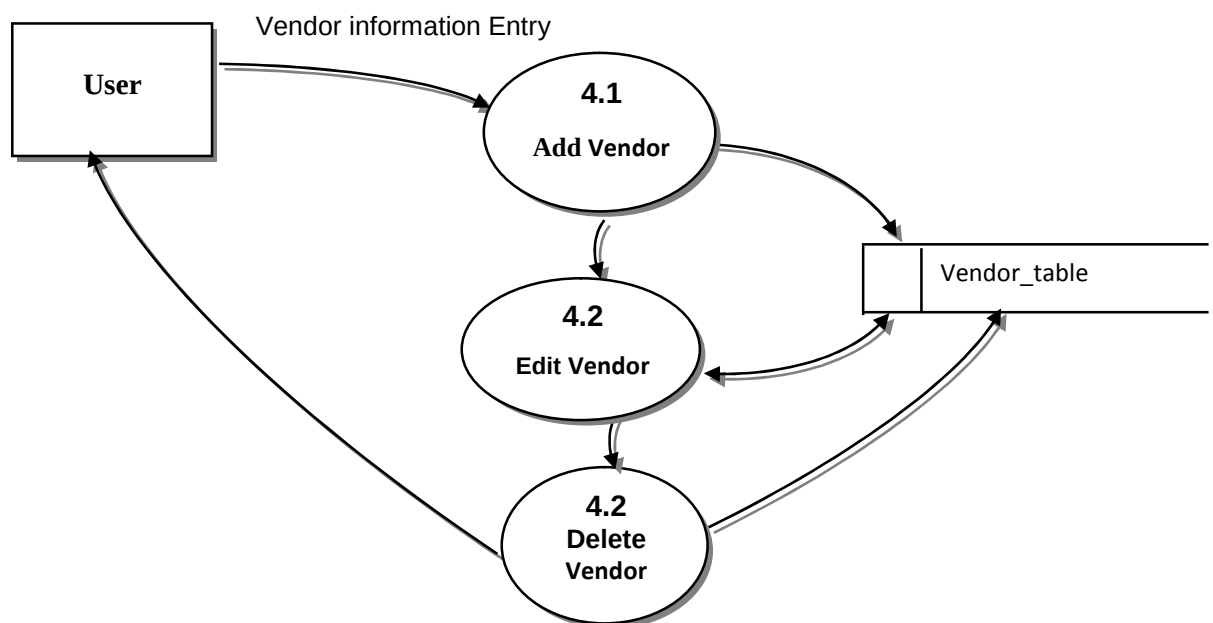
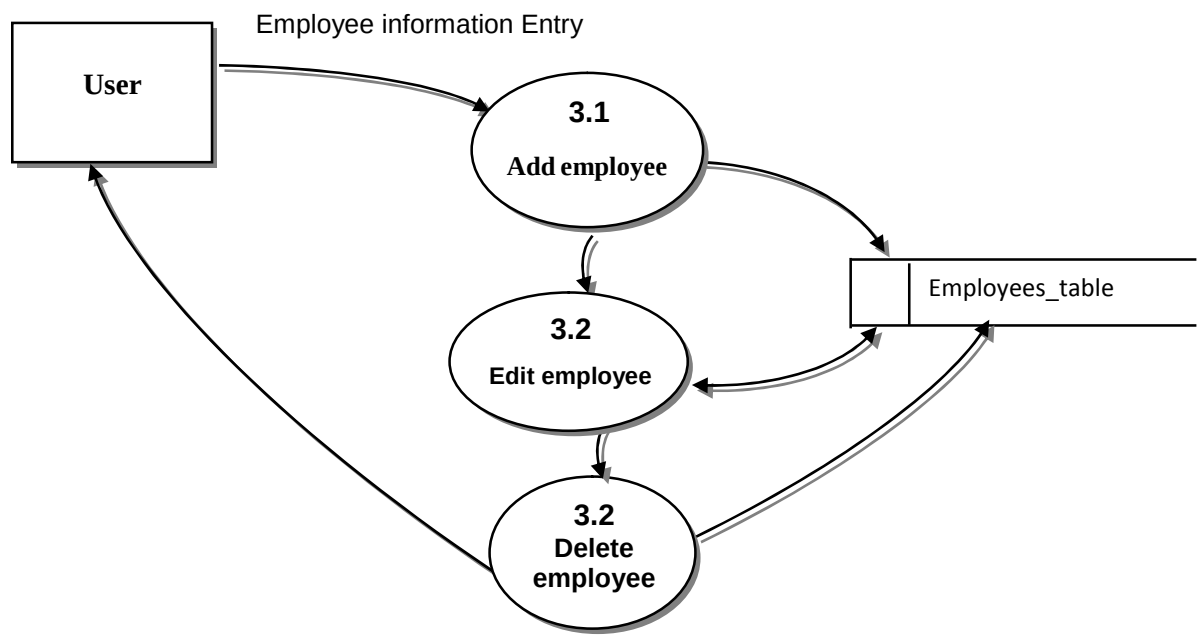


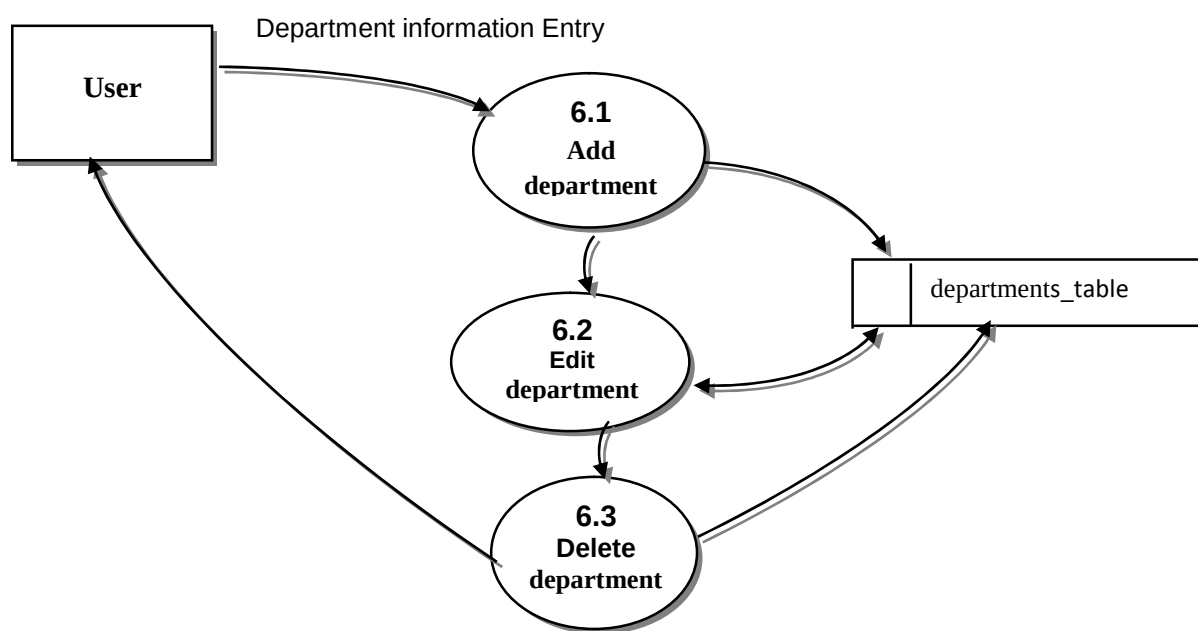
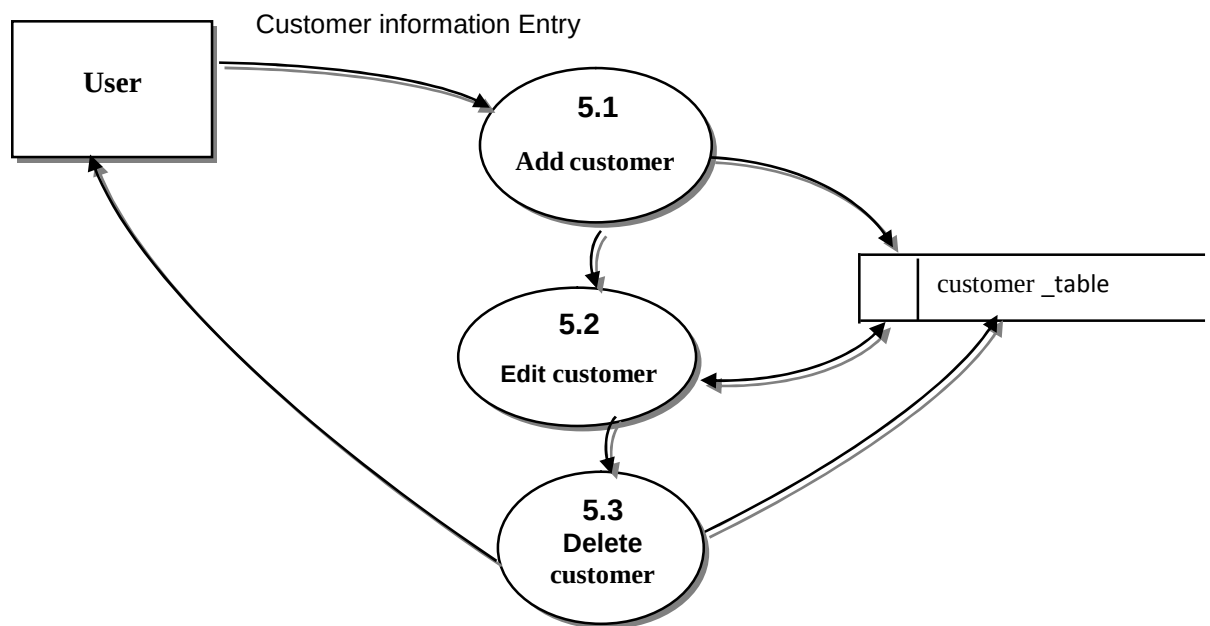
Fig. A2.2: the 1st Level Data Flow Diagram of legacy supermarket system

Second Level Data flow Diagram

In this level all the transaction of this legacy Supermarket Management will be expanded as shown in Fig.A2.3. The different subsection is indicated by decimal notation which can easily understand by the reader. First when a customer gives its requirement details the system check whether this item is sold in the supermarket or not, this validation check is first provided. Then if that item is present then the availability of the item is processed. Whether this item is currently present in the stock or not .The presence of item is assured then the bill is processed by the computerization according to the requirement of the customer. This process is repeated for all items whose request is given by the customer. When all the item is finished then the printed bill including all the details of the item is given to customer and the customer pay the bill provided by the store. This whole process is the summary of the legacy Supermarket Management automation system as the result from the analysis of that legacy software.







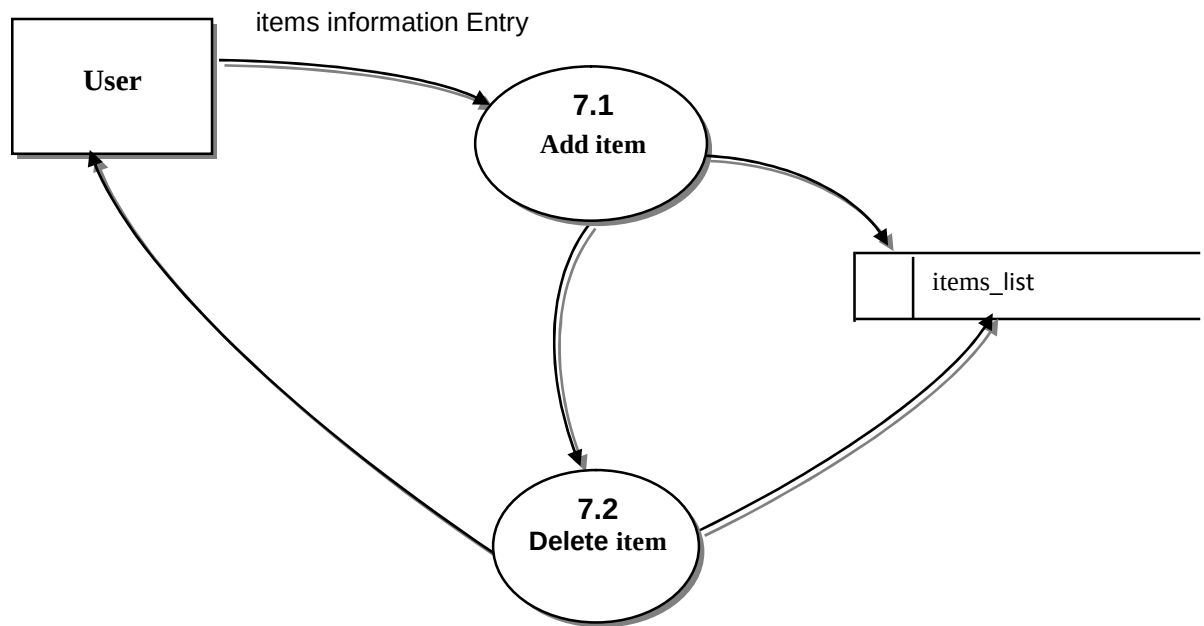


Fig. A2.3: 2nd Level Data Flow Diagram of legacy supermarket system

Execution Flow Chart of The legacy supermarket system

The flowcharts are the easiest way to understand the execution of this project. The Flow Chart to Add New employee is shown in Fig.A2.4.a. The Flow Chart to Edit employee information is shown in Fig. A2.4.b. The Flow Chart to delete employee is shown in Fig.A2.4.c. The Flow Chart to Add New vendor is shown in Fig.A2.5.a. The Flow Chart to edit vendor information is shown in Fig.A2.5.b. The Flow Chart to delete vendor is shown in Fig.A2.5.c. The Flow Chart to Add New customer is shown in Fig.A2.6.a. The Flow Chart to Edit customer information is shown in Fig.A2.6.b. The Flow Chart to delete customer is shown in Fig.A2.6.c. The Flow Chart to Add New item is shown in Fig.A2.7.a. The Flow Chart to delete item is shown in Fig.A2.7.b. The Flow Chart for selling an item is shown in Fig.A2.8.

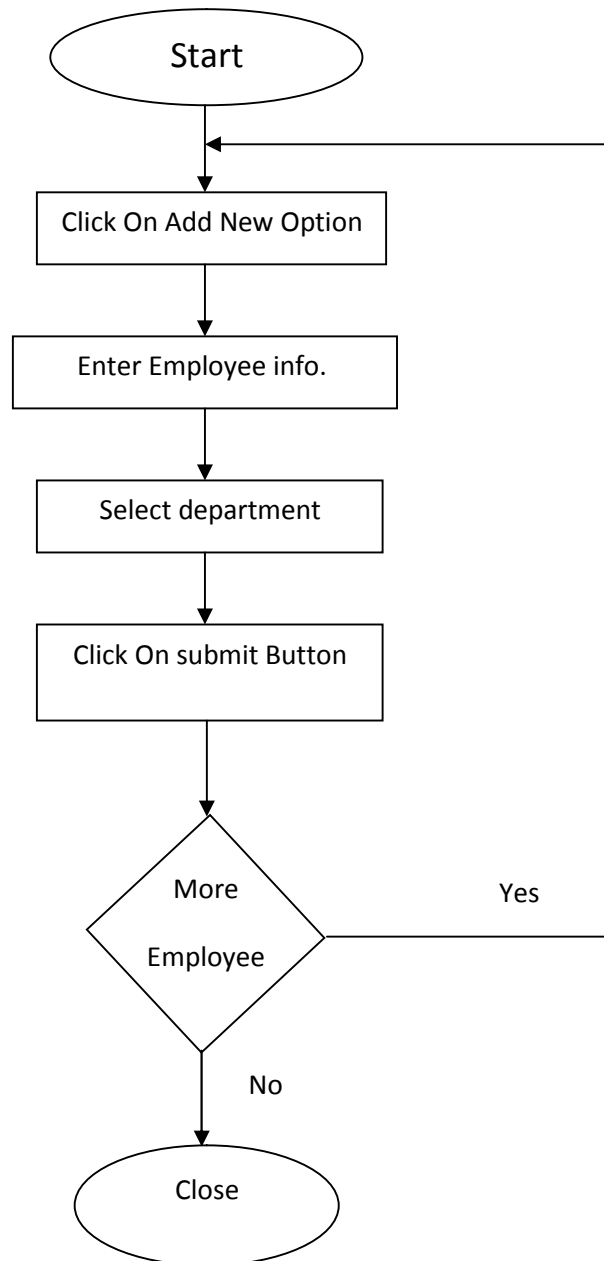


Fig.A2.4.a: The Flow Chart to Add New employee

As shown in Fig.A2.4.a, when the user want to add new employee he has to clicks on add new button then enter employee information as name, street, city, state, zip, and phone, then select the department and clicks on submit button to save these information in the database in the employees table. Finally check if there is another employee to be added if yes then do the procedure again, otherwise close this window.

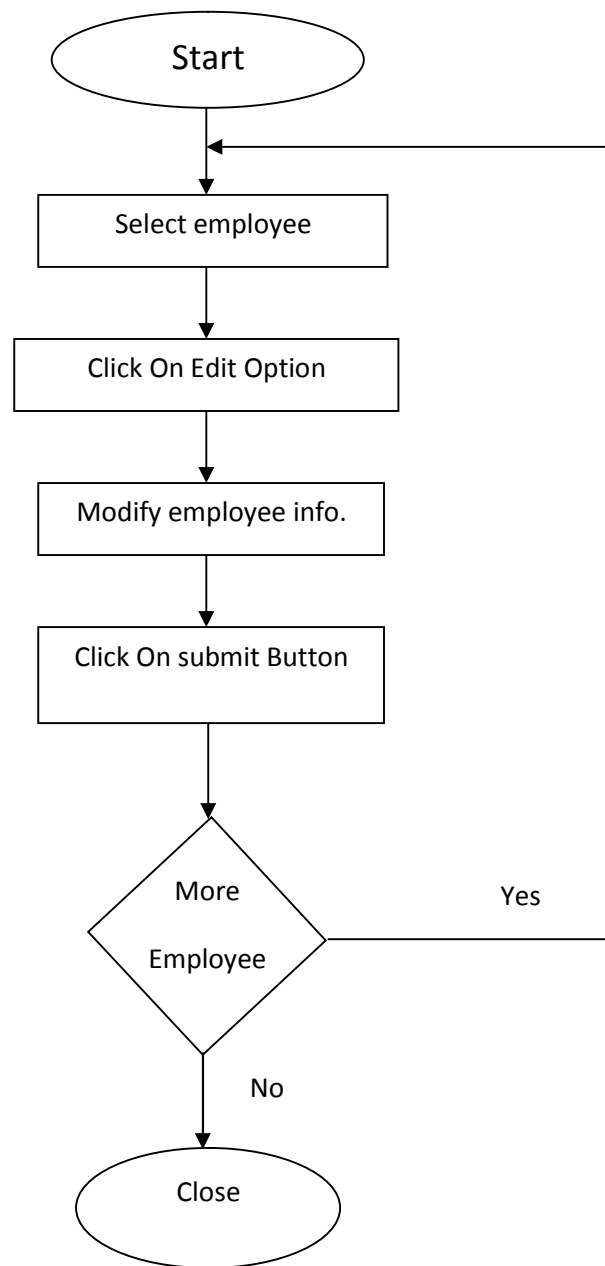


Fig.A2.4.b: The Flow Chart to edit employee information

As shown in Fig.A2.4.b, when the user want to edit employee information he has to select the employee and clicks on edit button then modify employee information and clicks on submit button to update these information in the employees table. Finally check if there is another employee need update his information, if yes then do the procedure again, otherwise close this window.

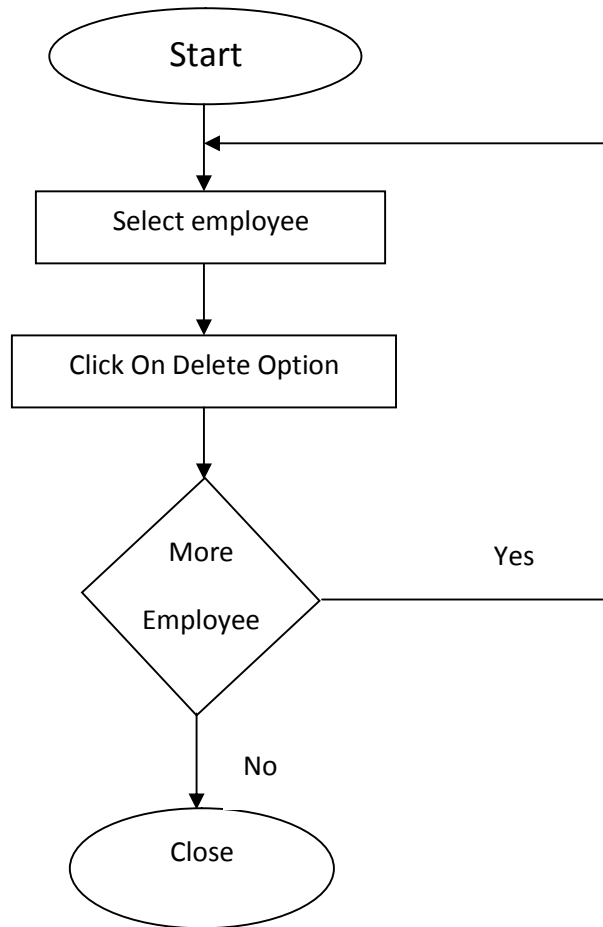


Fig.A2.4.c: The Flow Chart to delete employee

As shown in Fig.A2.4.c, when the user want to delete employee from the database he has to select the employee and clicks on delete button then employee information will be deleted from employees table. Finally check if there is another employee there is need to delete his information, if yes then do the procedure again, otherwise close this window.

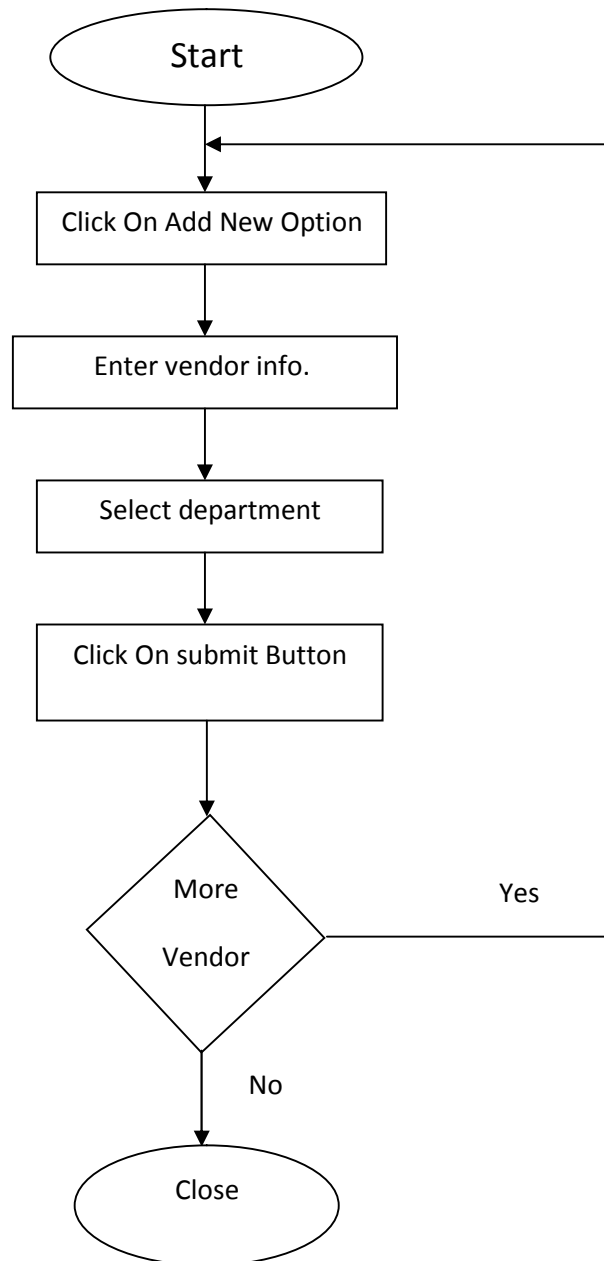


Fig.A2.5.a: The Flow Chart to Add New vendor

As shown in Fig.A2.5.a, when the user want to add new vendor he has to clicks on add new button then enter vendor information as name, address, phone, and fax, then select the department and clicks on submit button to save these information in the database in the vendor table. Finally check if there is another vendor to be added, if yes then do the procedure again, otherwise close this window.

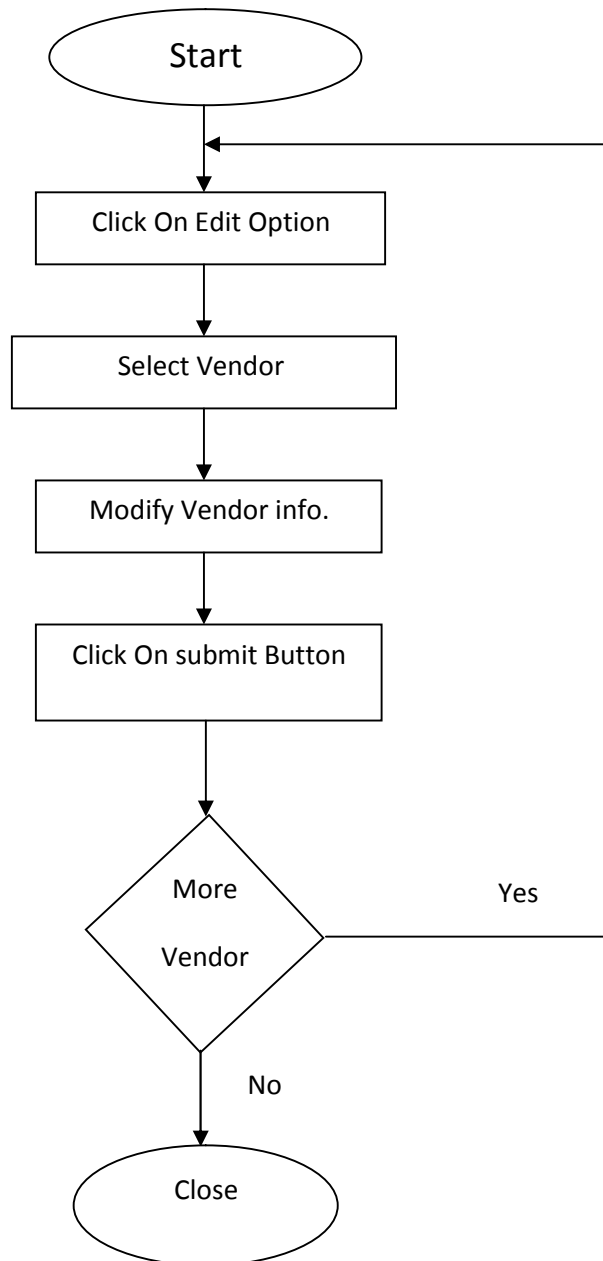


Fig.A2.5.b: The Flow Chart to edit vendor information

As shown in Fig.A2.5.b, when the user want to edit vendor information he has to select the vendor and clicks on edit button then modify vendor information and clicks on submit button to update these information in the vendors table. Finally check if there is another vendor need to update his information, if yes then do the procedure again, otherwise close this window.

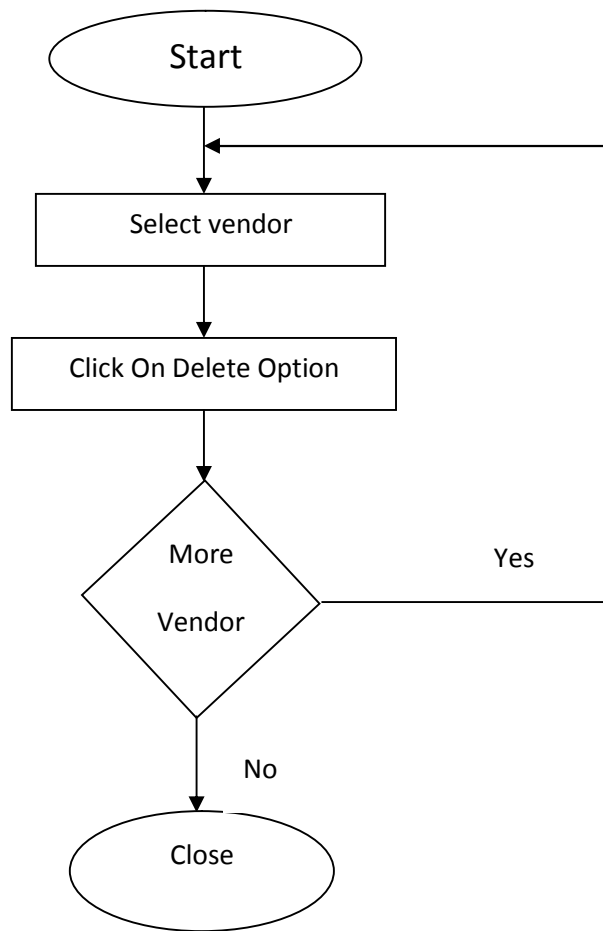


Fig.A2.5.c: The Flow Chart to delete vendor

As shown in Fig.A2.5.c, when the user want to delete vendor from the database he has to select the vendor and clicks on delete button then vendor information will be deleted from vendors table. Finally check if there is another vendor there is need to delete his information, if yes then do the procedure again, otherwise close this window.

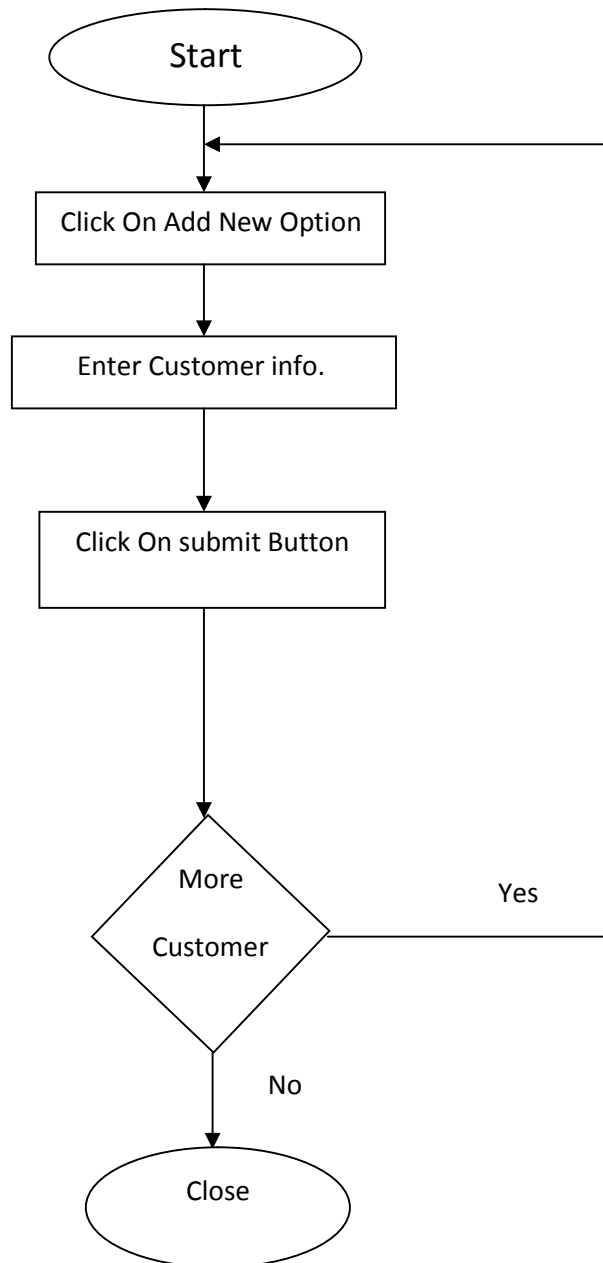


Fig.A2.6.a: The Flow Chart to Add New customer

As shown in Fig.A2.6.a, when the user want to add new customer he has to clicks on add new button then enter customer information as first name, last name, customer code, street, city, state, zip, phone, and email, then clicks on submit button to save these information in the database in the customer table. Finally check if there is another customer to be added, if yes then do the procedure again, otherwise close this window.

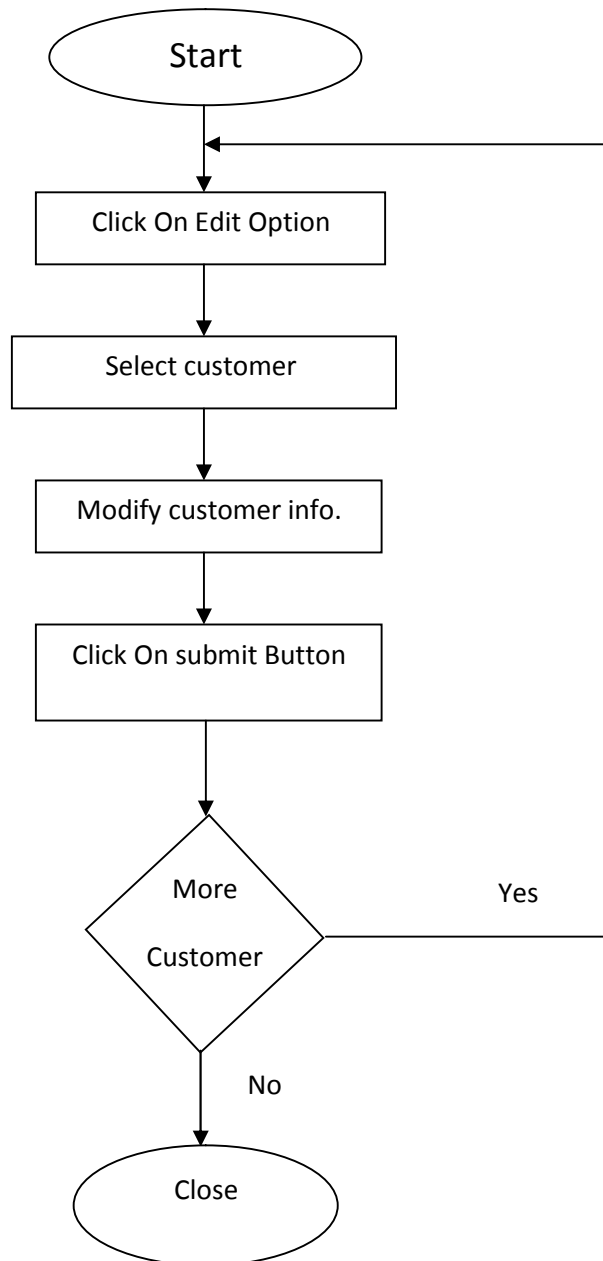


Fig.A2.6.b: The Flow Chart to edit customer information

As shown in Fig.A2.6.b, when the user want to edit customer information he has to clicks on edit button and select the customer then modify customer information and clicks on submit button to update these information in the customer table. Finally check if there is another customer there is need to update his information, if yes then do the procedure again, otherwise close this window.

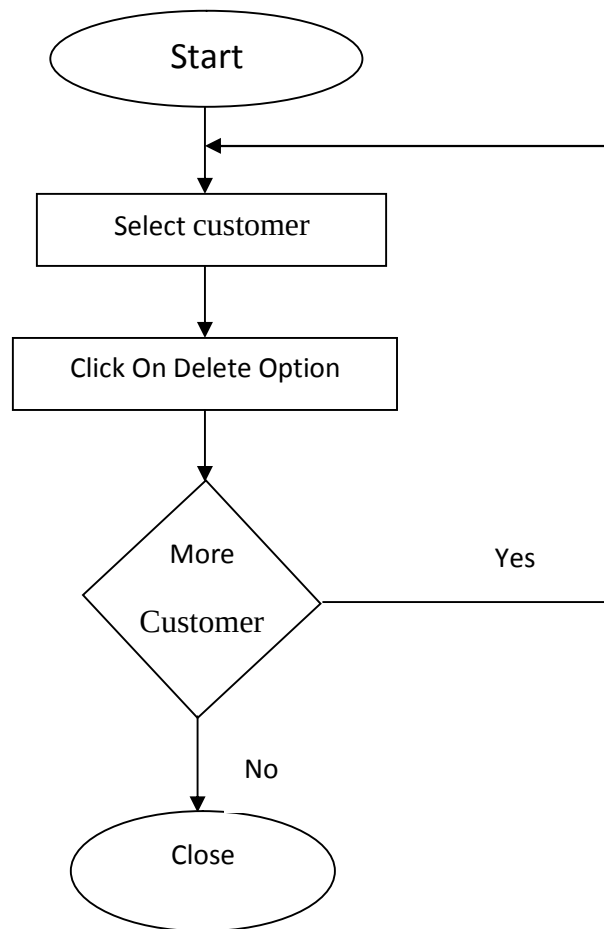


Fig.A2.6.c: The Flow Chart to delete customer

As shown in Fig.A2.6.c, when the user want to delete customer from the database he has to select the customer and clicks on delete button then customer information will be deleted from customer table. Finally check if there is another customer there is need to delete his information, if yes then do the procedure again, otherwise close this window.

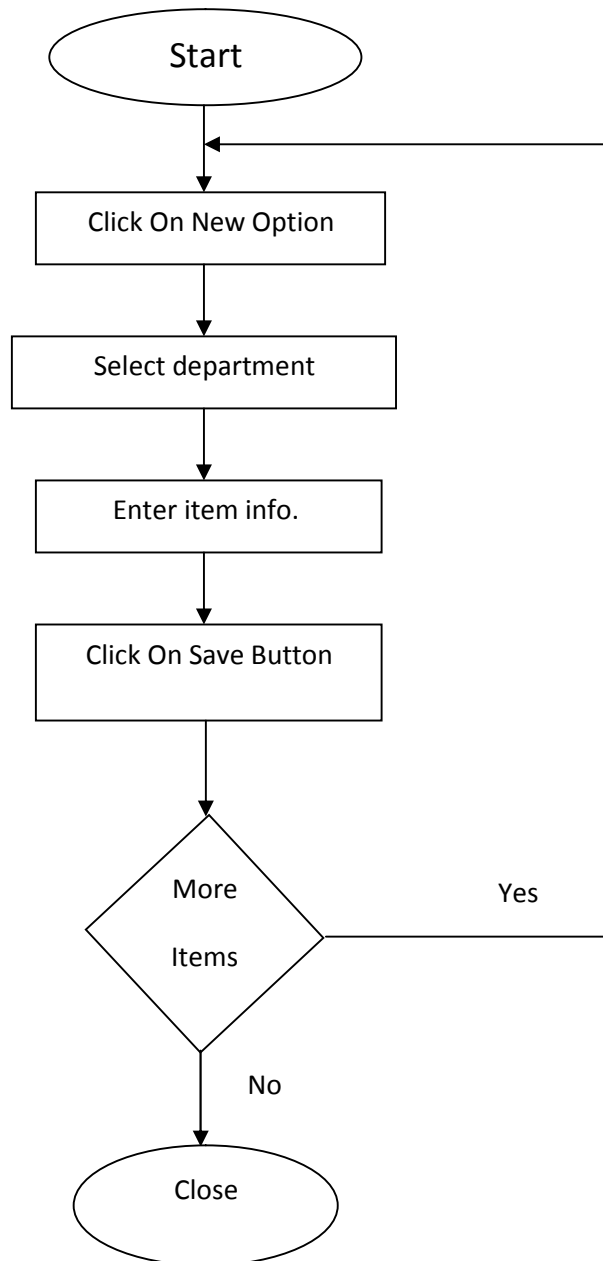


Fig.A2.7.a: The Flow Chart to Add New item

As shown in Fig.A2.7.a, when the user want to add new item he has to clicks on add new button then select the department and enter item information as product name, rate, discount, and quantity, then clicks on save button to save these information in the database in the items table. Finally check if there is another item to be added, if yes then do the procedure again, otherwise close this window.

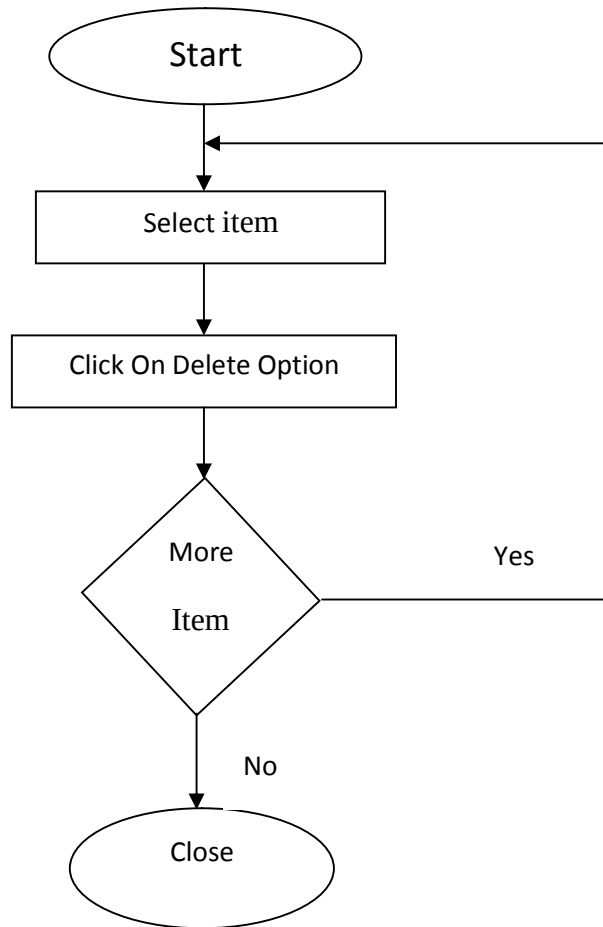


Fig.A2.7.b: The Flow Chart to delete item

As shown in Fig.A2.7.b, when the user want to delete item from the database he has to select the item and clicks on delete button then item information will be deleted from items table. Finally check if there is another item there is need to delete his information, if yes then do the procedure again, otherwise close this window.

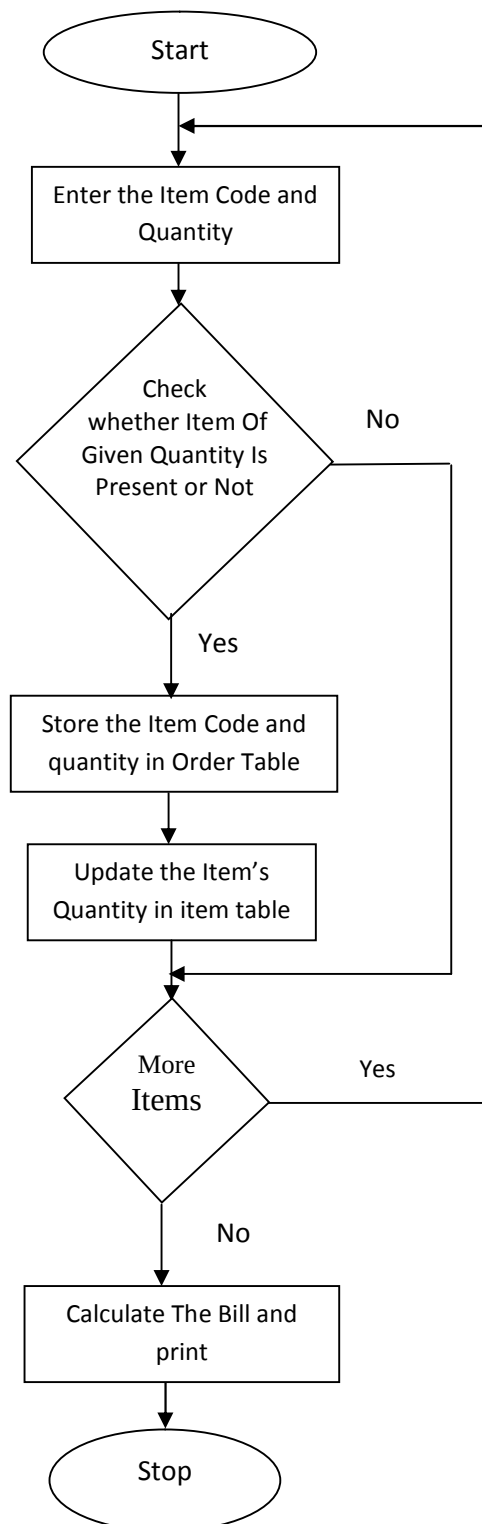


Fig.A2.8: The Flow Chart for selling an item

As shown in Fig.A2.8, when the user want to sell an item to the customer he has to select the item according to item code, then determine the required quantity from that item, then

the system check if the required quantity available in the store or not, if yes then save the item code and quantity in the order table and update the item's quantity in the items table, if there is insufficient quantity in the store ask the user to select another item, finally clicks on calculate the bill button then print the bill.

Use Case Diagrams:

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well. The use case diagram of the legacy supermarket management system are shown in Fig.A2.9:

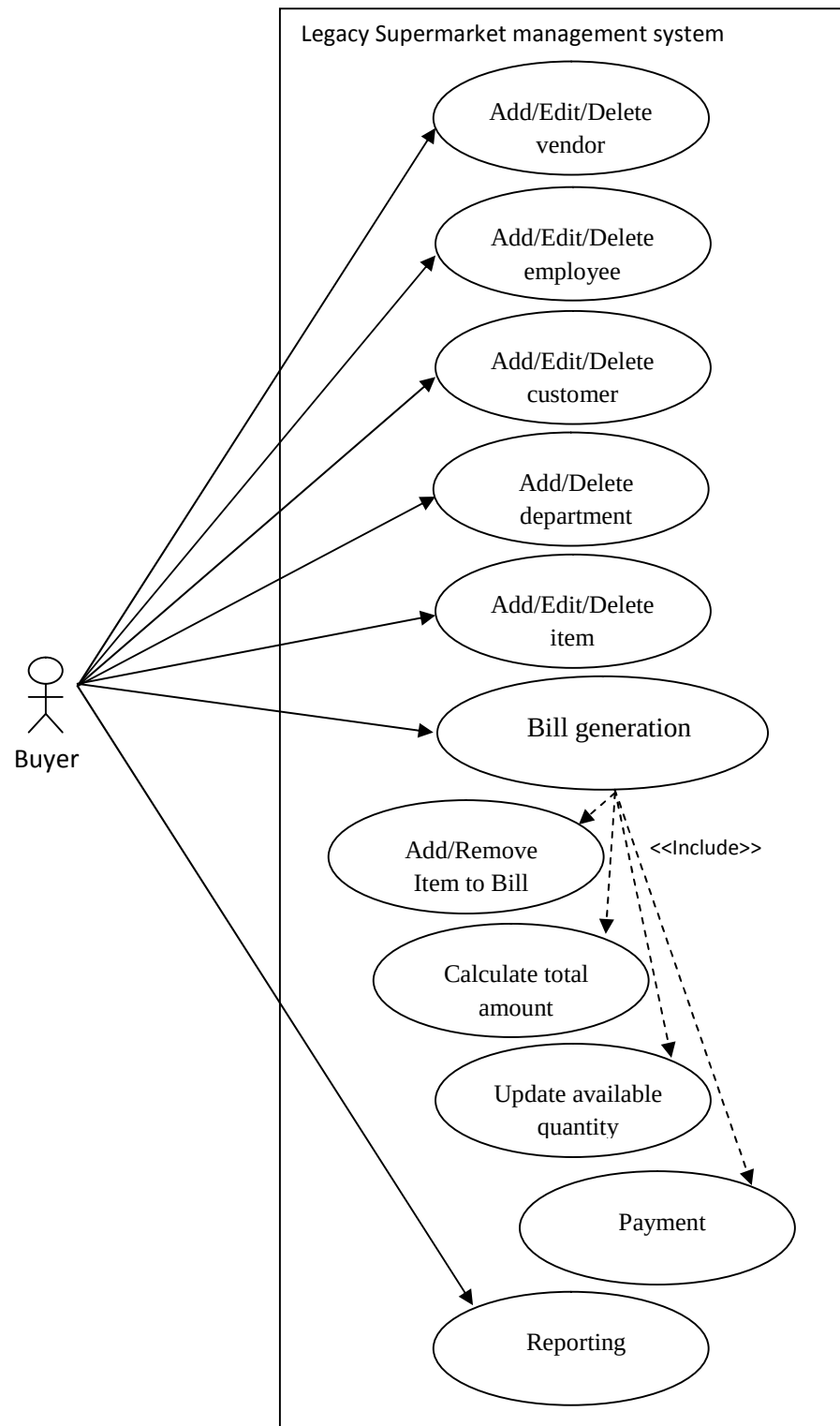


Fig.A2.9: The use case diagram of the legacy supermarket management system

Entity Relationship Diagrams:

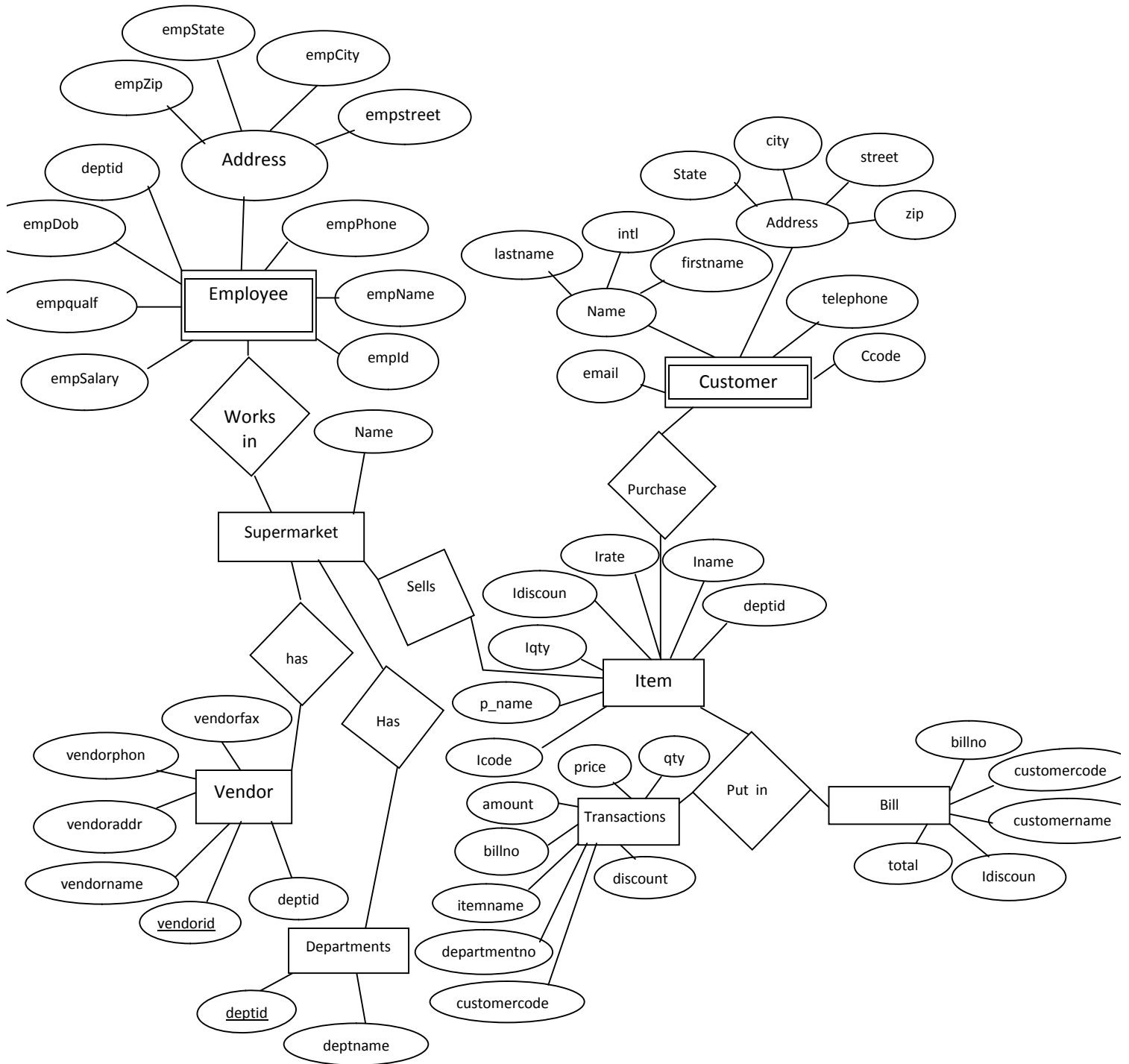


Fig. A2.10: The ERD diagram of the legacy supermarket management system

Database Design:

The database code to design the employee table is shown below, and the table content shown in Table A2.1:

```
create table employees(  
empId char(10),  
empName varchar(30),  
empStreet varchar(20),  
empCity varchar(20),  
empState varchar(20),  
empZip varchar(20),  
empPhone char(12),  
empDob datetime,  
empSalary integer,  
empDesig varchar(20),  
empqualf varchar(40),  
deptid integer);
```

Table A2.1: Employees information

Data field	Type	Constraint	Description
empId	char(10)	-----	Employee identification number
empName	varchar(30)	-----	Employee name
empStreet	varchar(20)	-----	Name of the street
empCity)	varchar(20	-----	Name of the city
empState,	varchar(20)	-----	Name of the state
empZip	varchar(20)	-----	Zip code for the employee
empPhone	char(12)	-----	Phone number of the employee
empDob	datetime	-----	Date of employee birth
empSalary	integer	-----	Employee salary
empDesig	varchar(20)	-----	Employee designation
empqualf	varchar(40)	-----	Employee Qualification
deptid	integer	-----	Department identification number

The database code to design the Vendor table is shown below, and the table content shown in Table A2.2:

```
create table vendors(  
deptid integer,  
vendorid integer PRIMARY KEY,  
vendorname varchar(30),  
vendoraddr varchar(30),  
vendorphone decimal(12),
```

vendorfax decimal(15));

Table A2.2: Vendors information

Data field	Type	Constraint	Description
deptid	integer	-----	Department identification number
vendorid	integer	PRIMARY KEY	Vendor identification number
vendorname	varchar(30)	-----	Vendor name
vendoraddr	varchar(30)	-----	Vendor address
vendorphone	decimal(12)	-----	Vendor phone number
vendorfax	decimal(15))	-----	Fax of vendor

The database code to design the customer table is shown below, and the table content shown in Table A2.3:

```
create table customersdet(
  firstname varchar(20),
  lastname varchar(20),
  Ccode varchar(20),
  street varchar(20),
  city varchar(20),
  State varchar(20),
  zip varchar(20),
  telephone varchar(20),
  email varchar(20),
  intl varchar(2)
)
```

Table A2.3: Customers information

Data field	Type	Constraint	Description
firstname	varchar(20)	-----	Customer first name
lastname	varchar(20)	-----	Customer last name
Ccode	varchar(20)	-----	Customer code
street	varchar(20)	-----	Name of the street
city	varchar(20)	-----	Name of the city
State	varchar(20)	-----	Name of the state
zip	varchar(20)	-----	Zip code
telephone	varchar(20)	-----	Customer telephone number
email	varchar(20)	-----	Customer email address
intl	varchar(2)	-----	Initialization

The database code to design the Item table is shown below, and the table content shown in Table A2.4:

```
create table item (
Icode varchar(6),
deptid integer,
Iname varchar(50),
Irate integer,
Idiscount integer,
Iqty integer,
p_name varchar(50),
primary key (Icode),
FOREIGN KEY (deptid) REFERENCES departments(deptid));
```

Table A2.4: Items information

Data field	Type	Constraint	Description
Icode	varchar(6)	Primary key	Item code number
deptid	integer	FOREIGN KEY REFERENCES departments(deptid)	Department identification number
Iname	varchar(50)	-----	Item name
Irate	integer	-----	Item rate
Idiscount	integer	-----	Item discount
Iqty	integer	-----	Item quantity
p_name	varchar(50)	-----	Product name

The code to create the customer bill table is shown below, and the table content shown in Table A2.5;

```
create table customerbills
(
billno integer primary key,
customercode integer,
customername varchar(100),
total integer )
```

Table A2.5: Customer bill information

Data field	Type	Constraint	Description
billno	integer	Primary key	Bill number
customercode	integer	-----	Customer code
customername	varchar(100)	-----	Customer name
total	integer	-----	Total amount of the items price

The code of transaction table is shown below, and the table content shown in Table A2.6.

create table transactions

```
(
billno integer,
departmentno integer,
customercode integer,
itemname varchar(50),
qty integer,
discount integer,
price integer,
amount integer);
```

Table A2.6: Transactions information

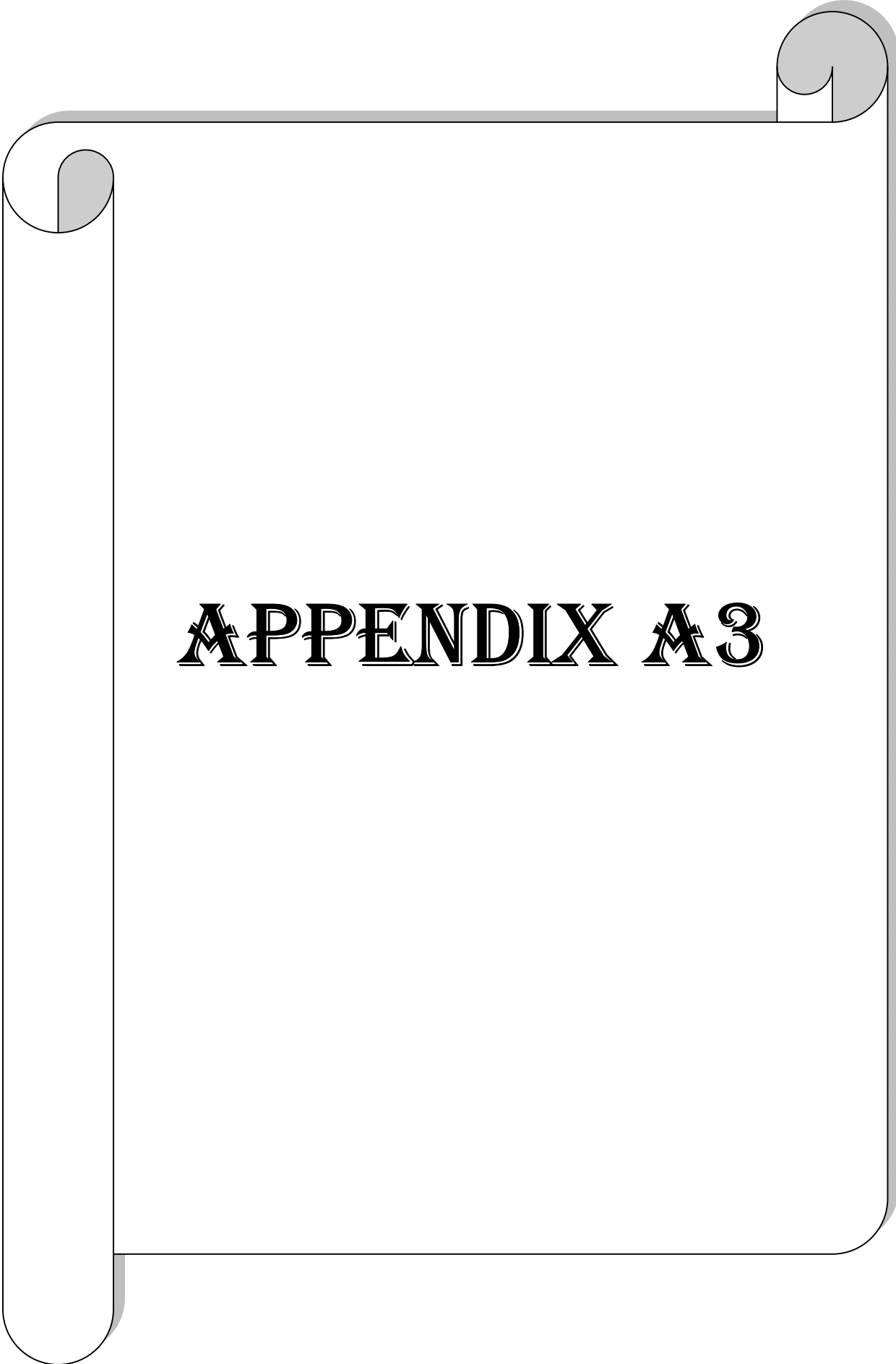
Data field	Type	Constraint	Description
Billno	integer	-----	Bill number
Departmentno	integer	-----	Department identification number
Customercode	integer	-----	Customer code
itemname	varchar(50)	-----	Item name
Qty	integer	-----	Quantity of the item
Discount	integer	-----	Discount amount
Price	integer	-----	Price of the item
amount	integer	-----	Amount of total price

The code to create the department table is shown below; the table content shown in Table A2.7.

```
create table departments(  
deptid integer PRIMARY KEY,  
deptname varchar(100));
```

Table A2.7: Department information

Data field	Type	Constraint	Description
deptid	integer	PRIMARY KEY	Department identification number
deptname	varchar(100)	-----	Department name



APPENDIX A3

APPENDIX A3

THE DFD, ERD, USE CASE DIAGRAMS, EXECUTION FLOWCHART AND DATABASE OF RE-ENGINEERED SUPERMARKET SYSTEM

The DFD, ERD, use case diagrams, execution flowchart and database of re-engineered supermarket system will be illustrated in this appendix; all these diagrams and flowcharts were produced according to the new requirements of the system's users, where some modifications will be applied on the legacy supermarket management system within the Document restructure activity.

Data Flow Diagram (DFD)

The data flow diagram of the re-engineered supermarket system is shown below in Fig.A3.1,

Zero level DFD

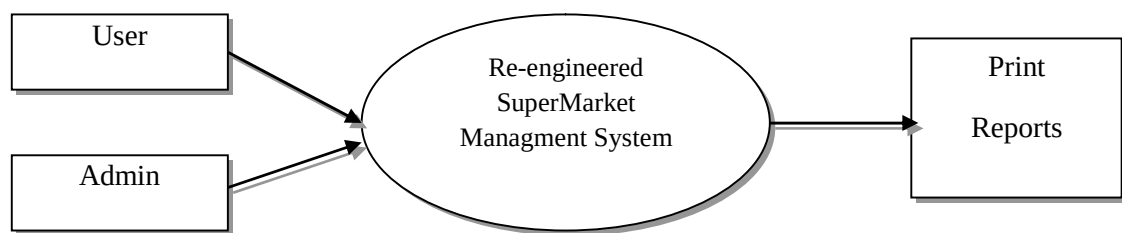


Fig. A3.1: Zero level DFD of the re-engineered Supermarket system

Zero Level DFD gives a brief overview of the entire system. The **Automated re-engineered Supermarket Management System** is enable the admin and user to fill/view/print the bill that contain the items that selected by customer and compute the total amount of the bill then bill generated and stock master updated by the number of items that sold, and enable the admin to add/ delete/ modified users of the system, also the admin can manage the price and all related information of the items that available in his supermarket. In this re-engineered system there is two level of authentication: user level and admin level, this authentication was protected by security password; After performing the required transactions on the system and generating the bill, the user can close the system or start to provide service to another customer.

First level DFD

First level DFD of a system gives a broad picture of all the processes of the system and the data transformation within the system. 1st Level Data Flow Diagram was shown in Fig.A3,2, The major processes involved in this system are as follows:

- 1) View/Edit/ Delete operator's details, by admin and user.
- 2) View/ Edit/ Delete customer's details, by admin and user.
- 3) View/ Edit/ Delete product's details, by admin and user.
- 4) View/ Edit/ Delete purchase's details, by admin and user.
- 5) View/ Edit/ Delete stock master's details, by admin and user.
- 6) Sale's an Generation of bill, by admin and user.
- 7) View/ Edit/ Delete department's details, by admin only.
- 8) View/ Edit/ Delete user's details, by admin only.
- 9) View/ Edit/ Delete supplier's details, by admin only.
- 10) View/ Edit/ Delete feedback's details, by admin only.
- 11) Generation of users, suppliers, customers, employees, sales, stocks, products, purchases, and feedbacks reports for the administrator for management purposes.

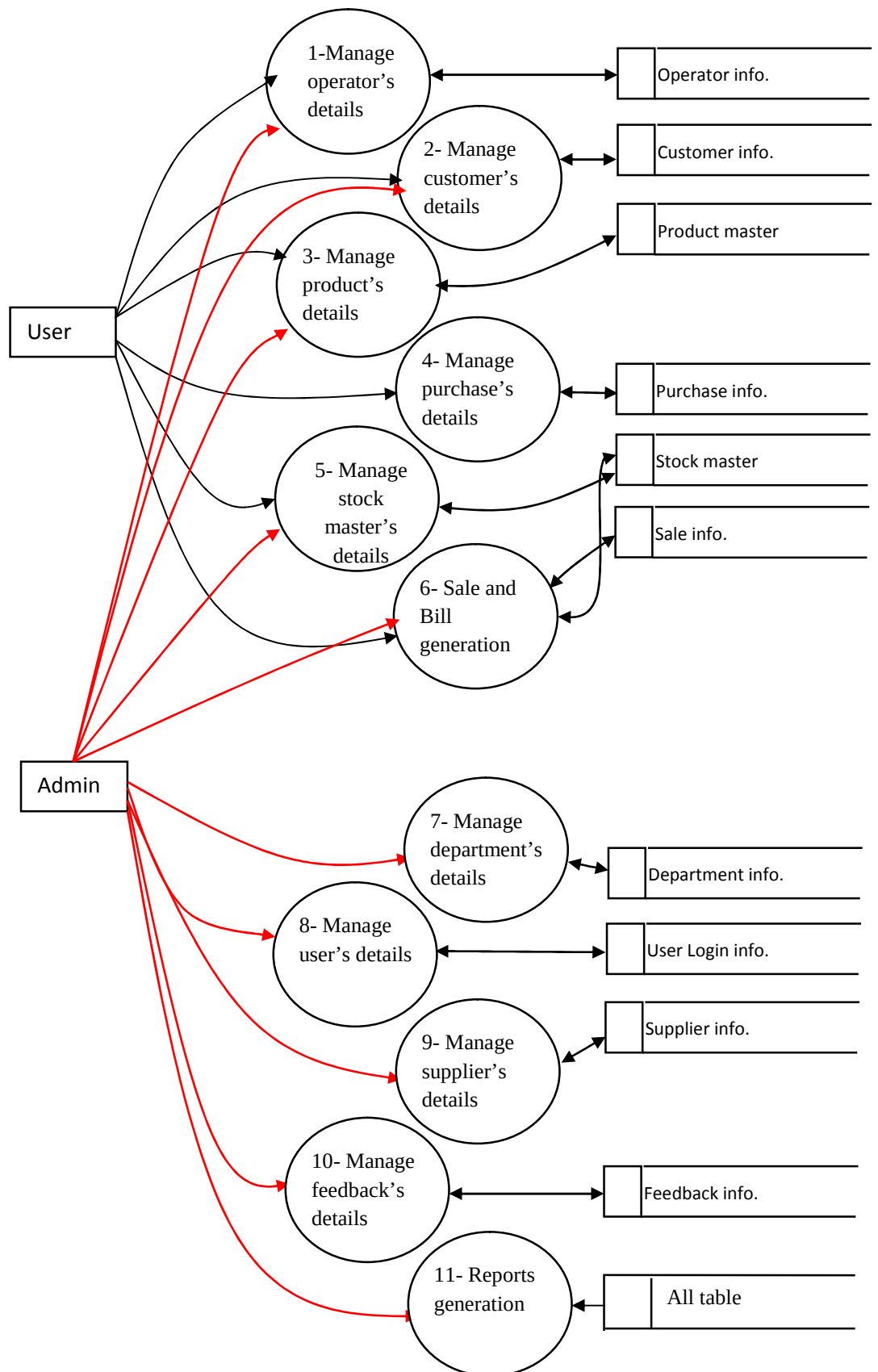
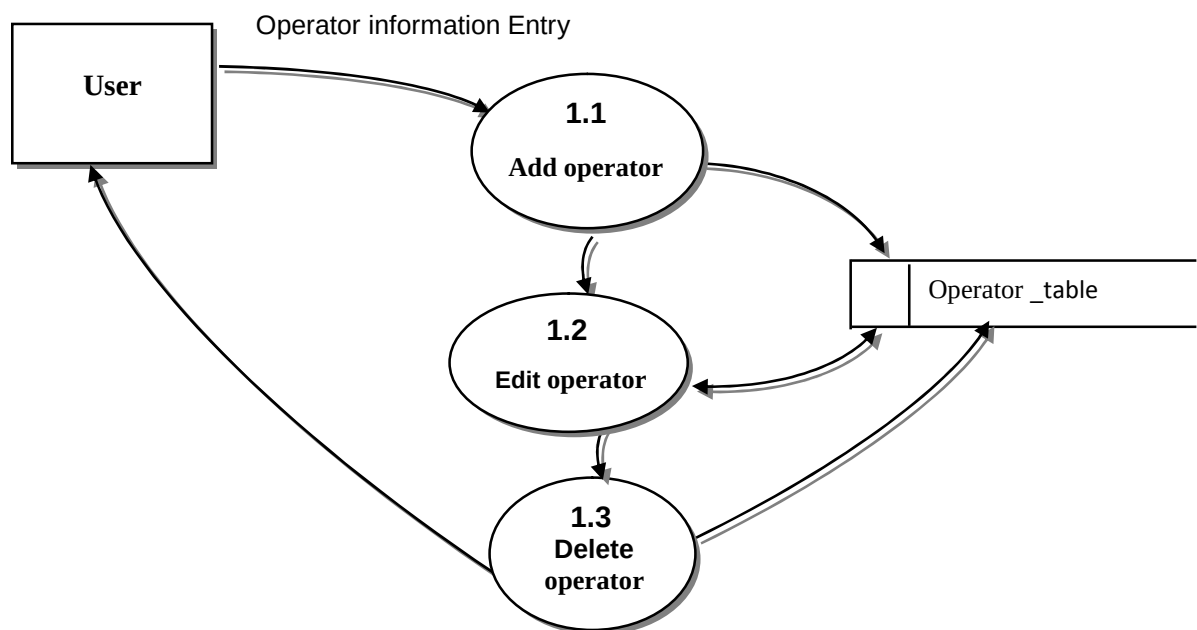
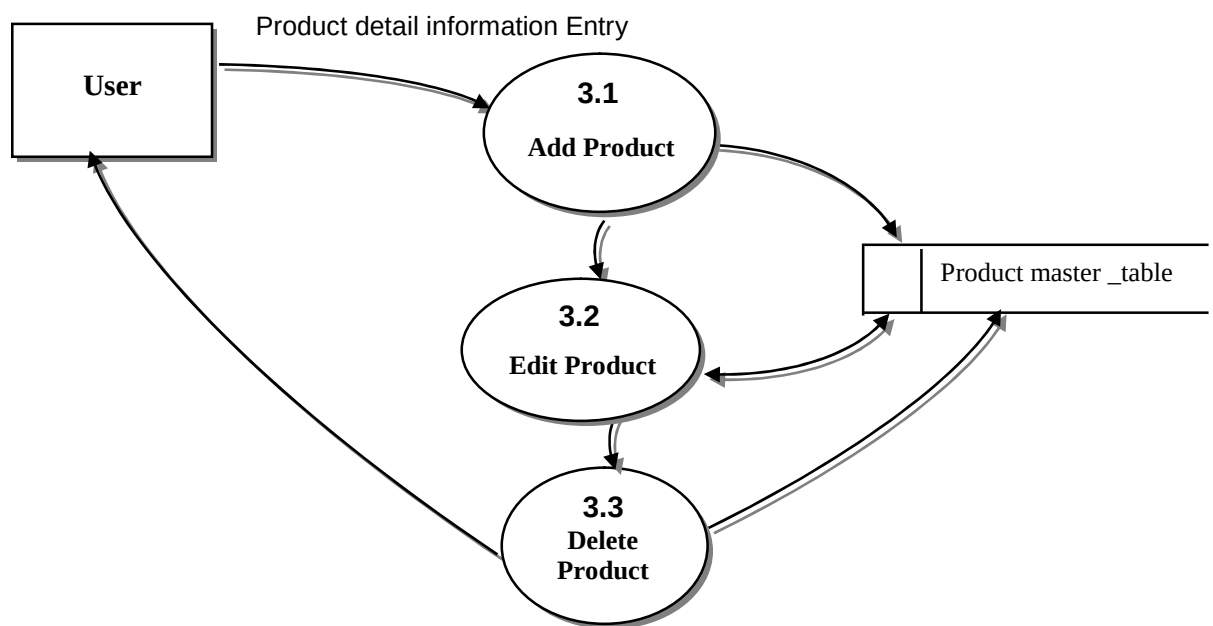
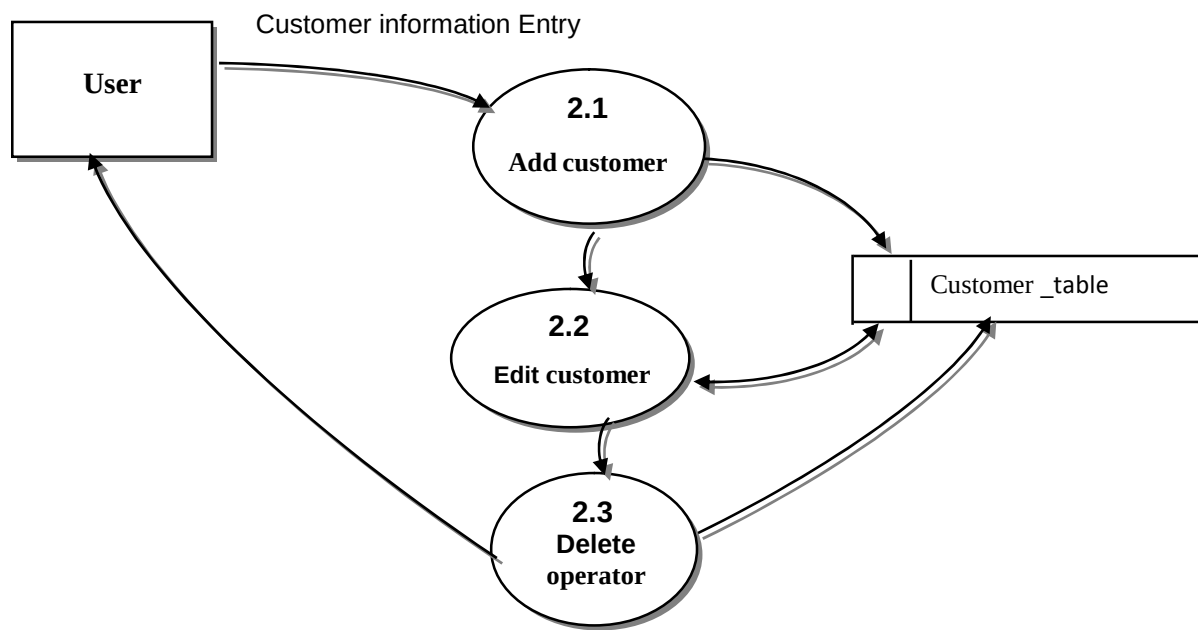


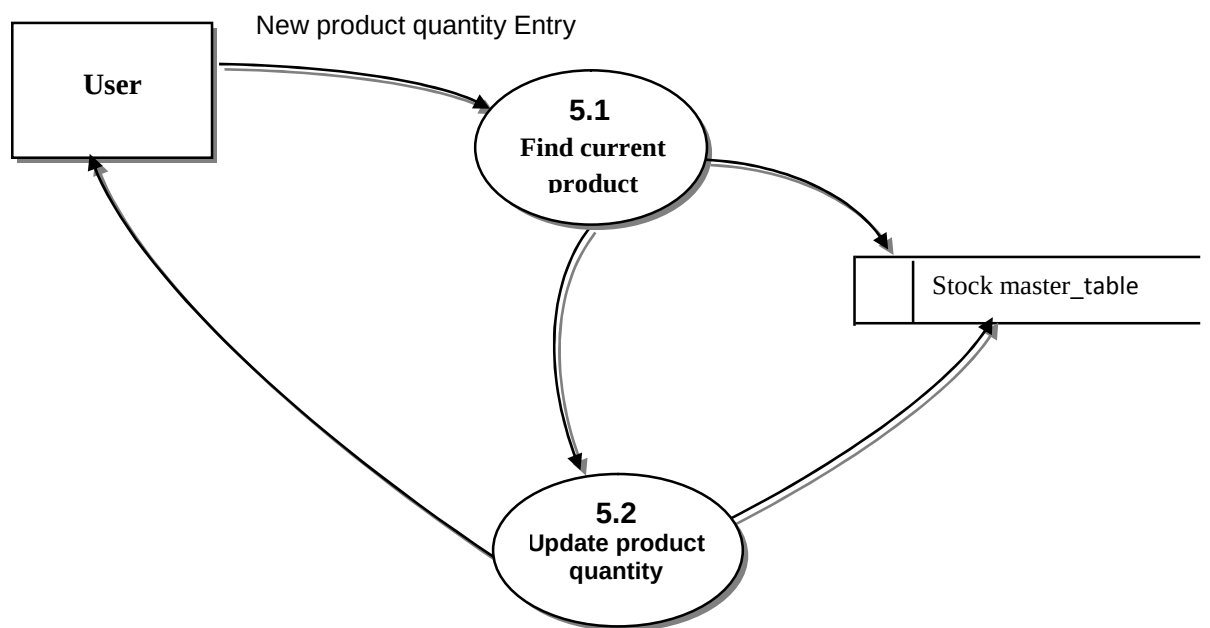
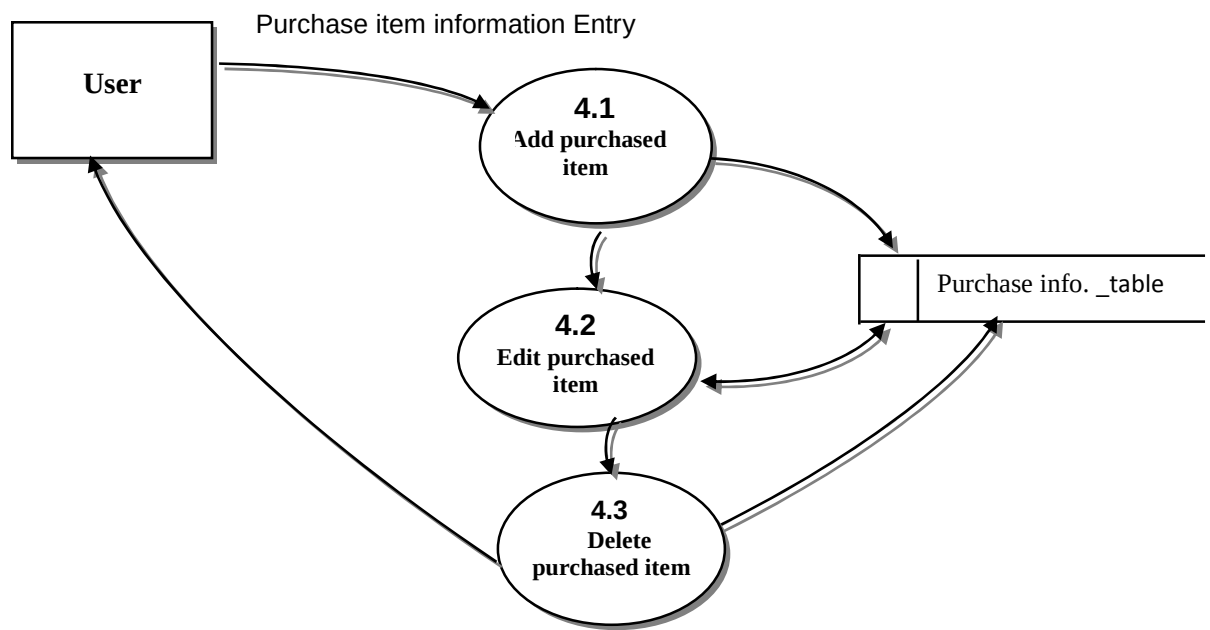
Fig.A3.2: 1st Level Data Flow Diagram of the re-engineered system

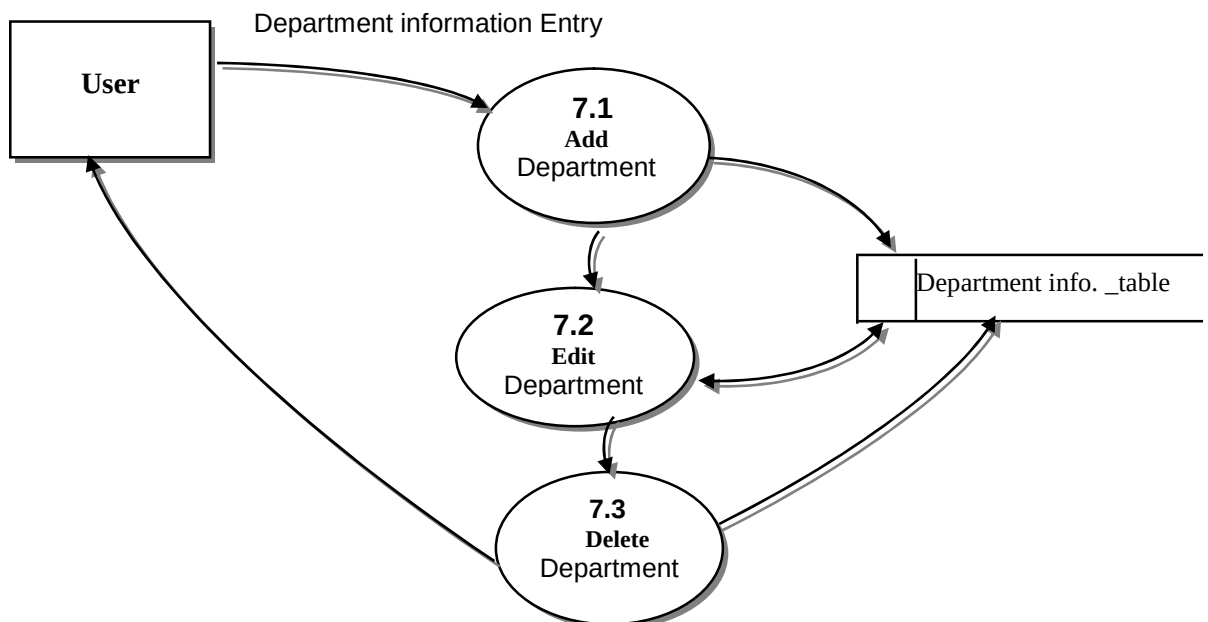
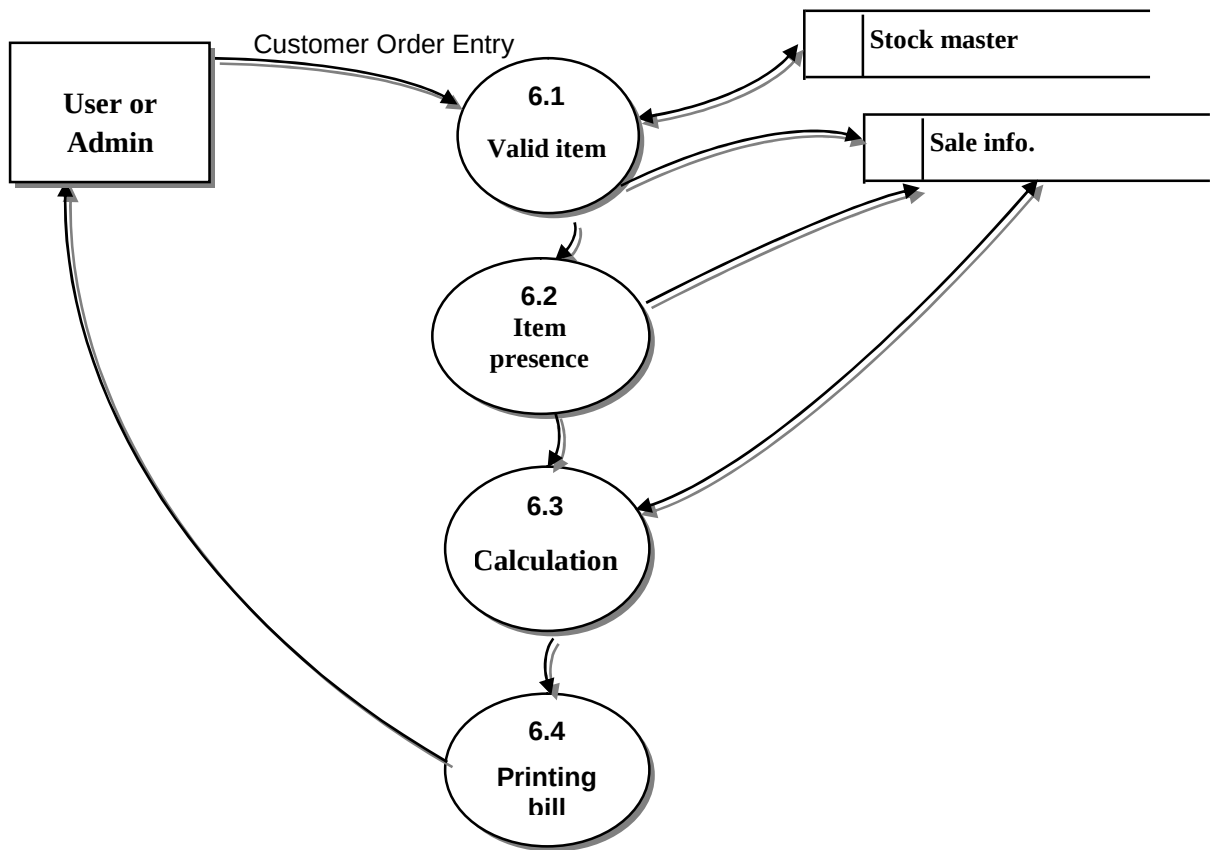
Second Level Data flow Daigram

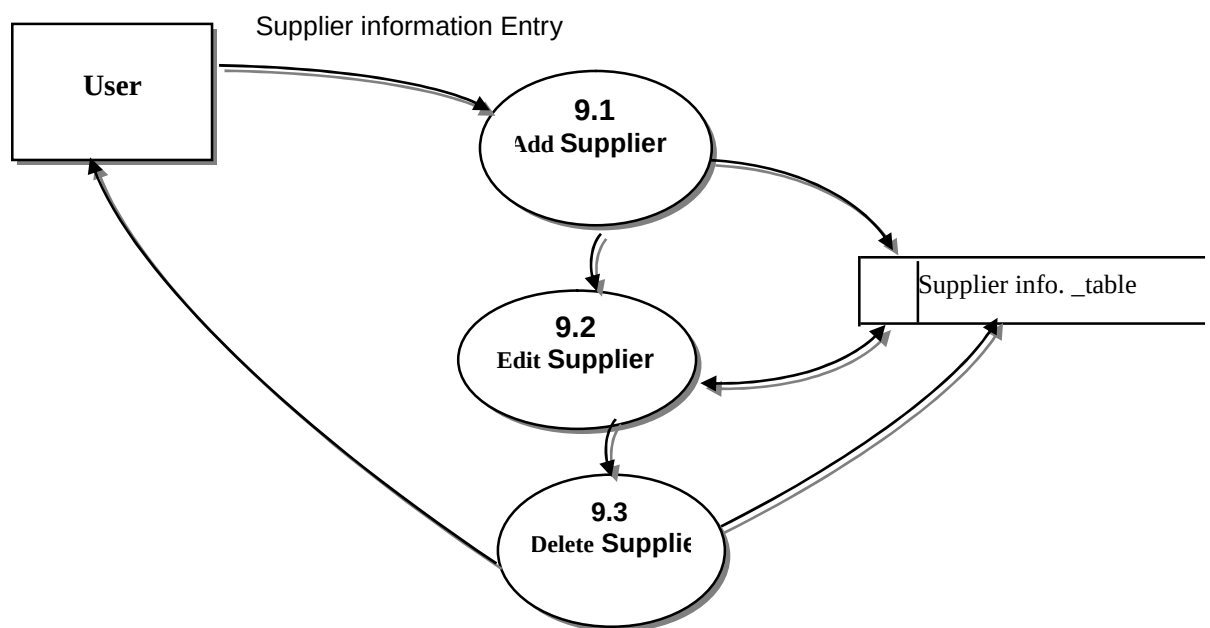
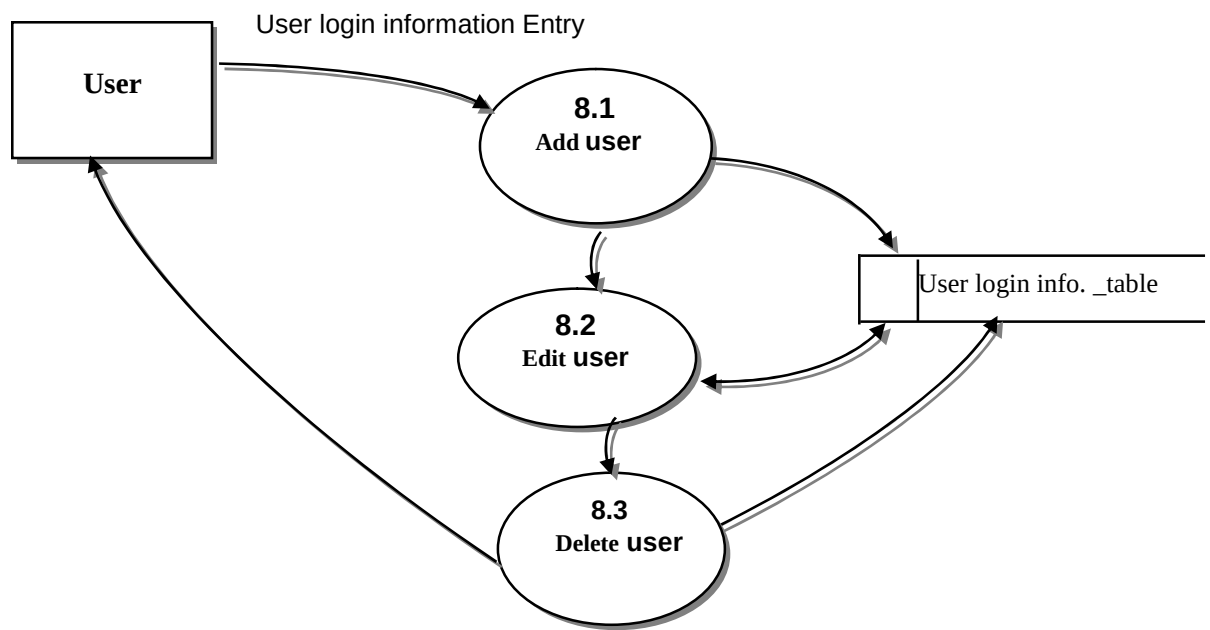
In this level all the processes of this re-engineered SuperMarket Management system will be expanded, as shown in Fig.A3.3. The different subsection is indicated by decimal notation which can easily understand by the reader. First when a customer gives its requirement details the system check whether this item is sold in the supermarket or not, this validation check is first provided. Then if that item is present then the availability of the item is processed. Whether this item is currently present in the stock or not .The presence of item is assured then the bill is processed by the computerization according to the requirement of the customer. This process is repeated for all items whose request is given by the customer. When all the item is finished then the printed bill including all the details of the item is given to customer and the customer pay the bill provided by the store.











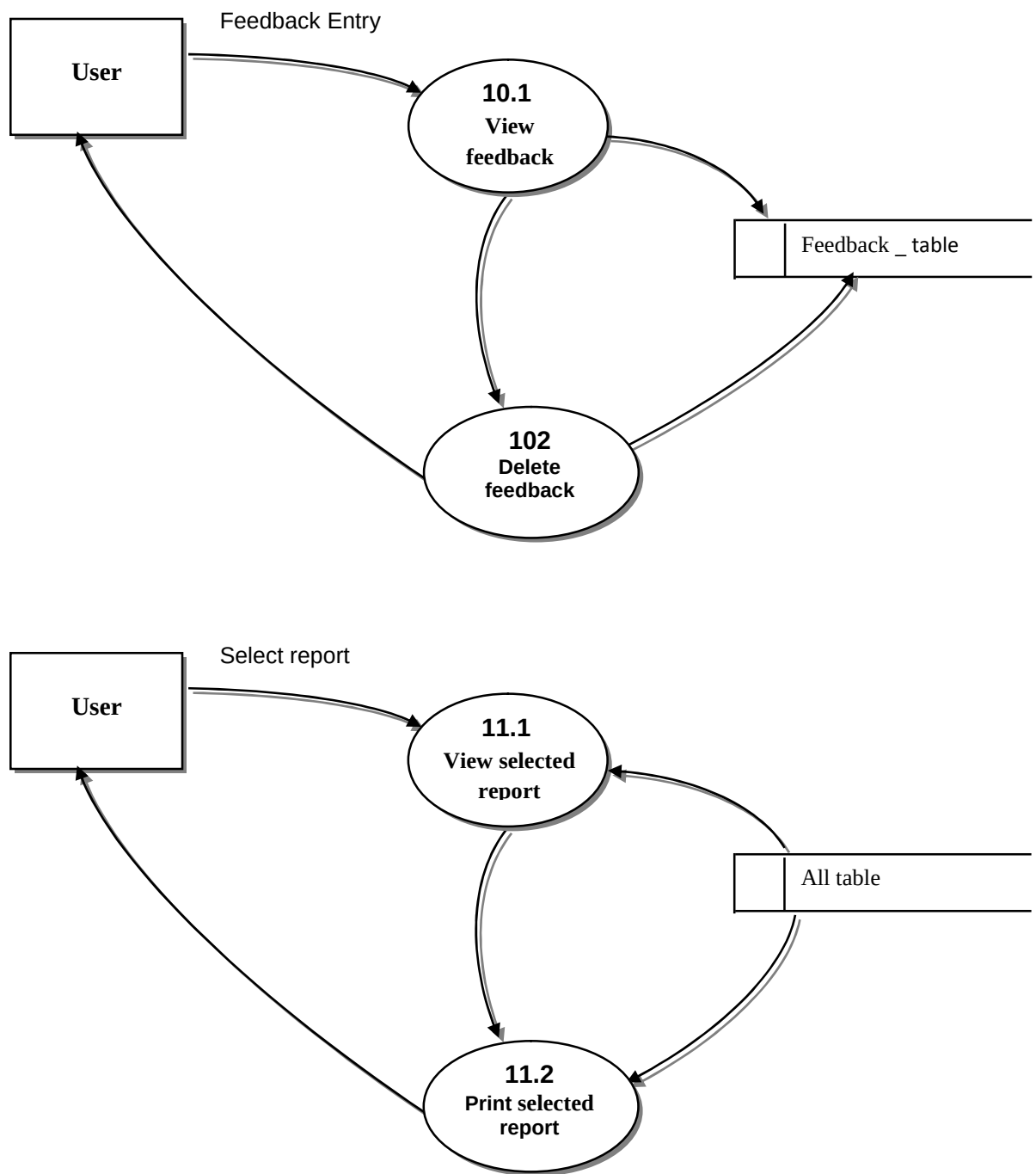


Fig.A3.3: 2'nd Level Data Flow Diagram of supermarket re-engineered system

Execution Flow Chart of The re-engineered supermarket system

The execution Flow chart of the re-engineered supermarket system was made by doing some modification on that of legacy software, where the flowcharts are the easiest way to understand the execution of the project. The Flow Chart to Add New Operator is shown in Fig.A3.4.a. The Flow Chart to Edit operator information is shown in Fig. A3.4.b. The Flow Chart to delete operator is shown in Fig. A3.4.c. the Flow Chart to Add New supplier is shown in Fig. A3.5.a. The Flow Chart to edit supplier information is shown in Fig. A3.5.b. The Flow Chart to delete supplier is shown in Fig. A3.5.c. The Flow Chart to Add New customer is shown in Fig. A3.6.a. the Flow Chart to Edit customer information is shown in Fig.A3.6.b. The Flow Chart to delete customer is shown in Fig.A3.6.c. The Flow Chart to Add New product is shown in Fig.A3.7.a. The Flow Chart to Edit product information is shown in Fig.A3.7.b. The Flow Chart to delete product is shown in Fig.A3.7.c. The Flow Chart to Add New user is shown in Fig. A3.8.a. The Flow Chart to Edit user information is shown in Fig.A3.8.b. The Flow Chart to delete user is shown in Fig.A3.8.c. The Flow Chart to update the stock master is shown in Fig.A3.9. The Flow Chart for selling an item is shown in Fig.A3.10. The Flow Chart for Purchase an item is shown in Fig.A3.11. The Flow Chart to Add New department is shown in Fig.A3.12.a. The Flow Chart to Edit department is shown in Fig.A3.12.b. The Flow Chart to delete department is shown in Fig.A3.12.c. The flow chart to login process at main page is shown in Fig.A3.13.

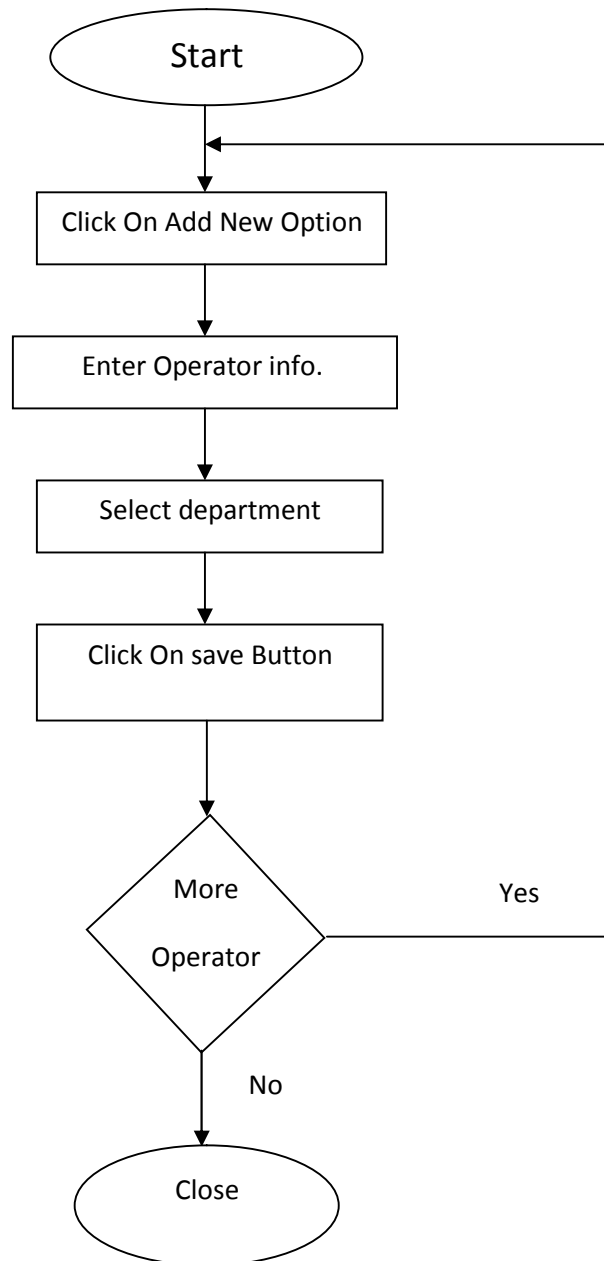


Fig.A3.4.a: The Flow Chart to Add New operator

As shown in Fig.A3.4.a, when the user want to add new operator he has to clicks on add new button then enter operator information as name, address, city, pin code, phone, and email, then select the department and clicks on save button to save these information in the database in the operator table. Finally check if there is another operator to be added if yes then do the procedure again, otherwise close this page.

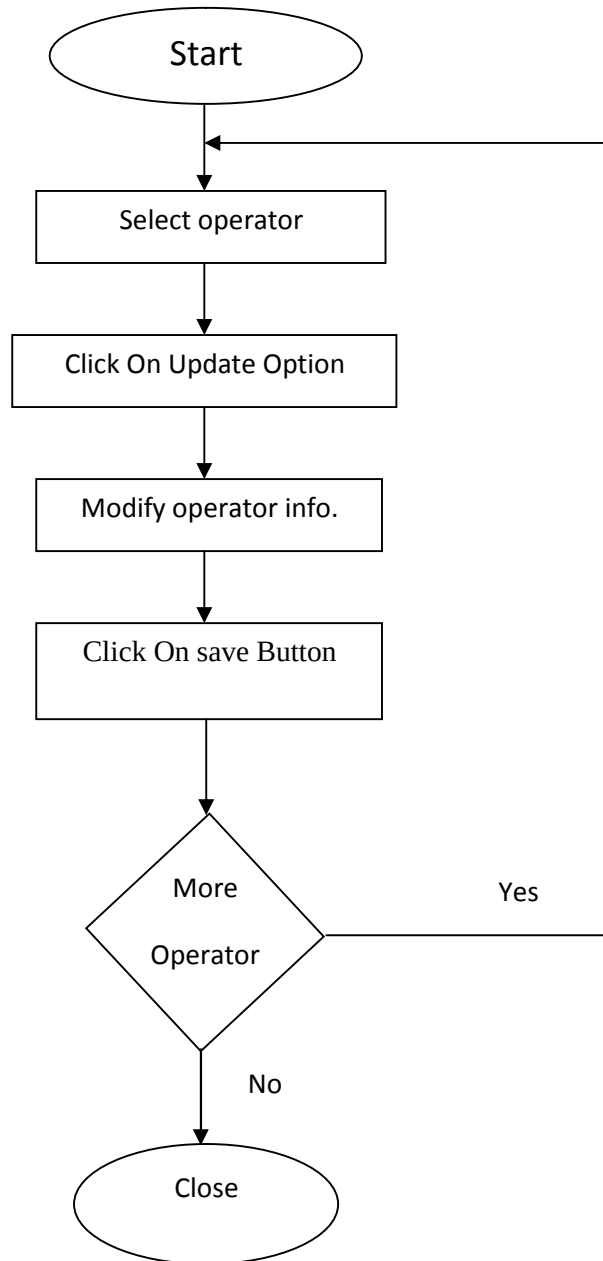


Fig.A3.4.b: The Flow Chart to edit operator information

As shown in Fig.A3.4.b, when the user want to edit operator information he has to select the operator then click on update button and modify operator information then clicks on save button to update these information in the operator table. Finally check if there is another operator need to update his information, if yes then do the procedure again, otherwise close this page.

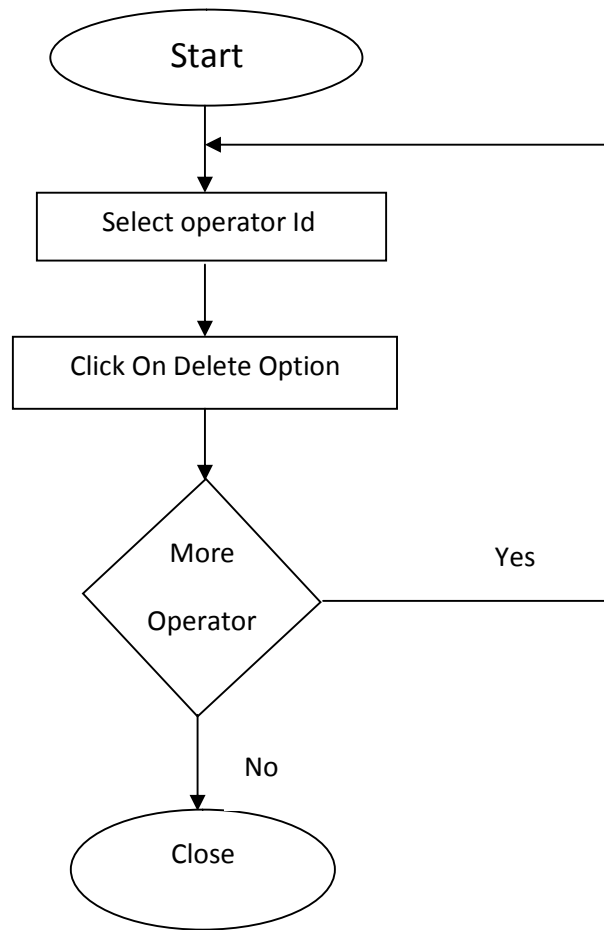


Fig.A3.4.c: The Flow Chart to delete operator

As shown in Fig.A3.4.c, when the user want to delete operator from the database he has to select the operator and clicks on delete button then operator information will be deleted from operator table. Finally check if there is another operator, there is need to delete his information, if yes then do the procedure again, otherwise close this page.

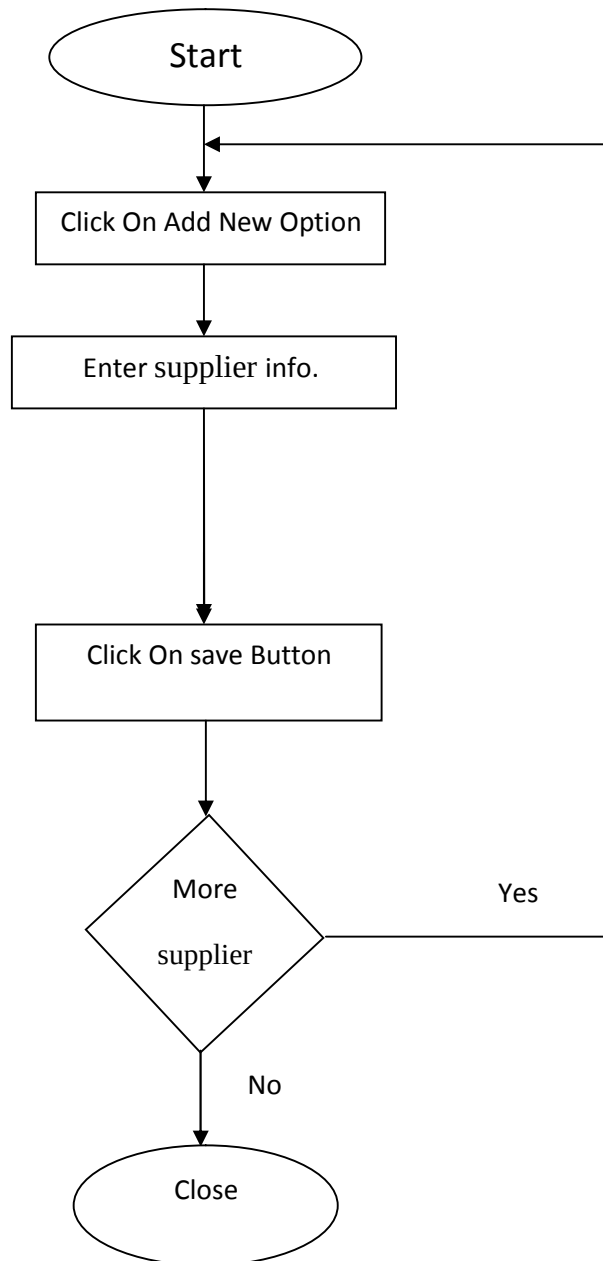


Fig.A3.5.a: The Flow Chart to Add New supplier

As shown in Fig.A3.5.a, when the administrator want to add new supplier he has to clicks on add new button then enter supplier information as name, address, city, pin code, phone, and email, then clicks on save button to save these information in the database in the supplier table. Finally check if there is another supplier to be added if yes then do the procedure again, otherwise close this page.

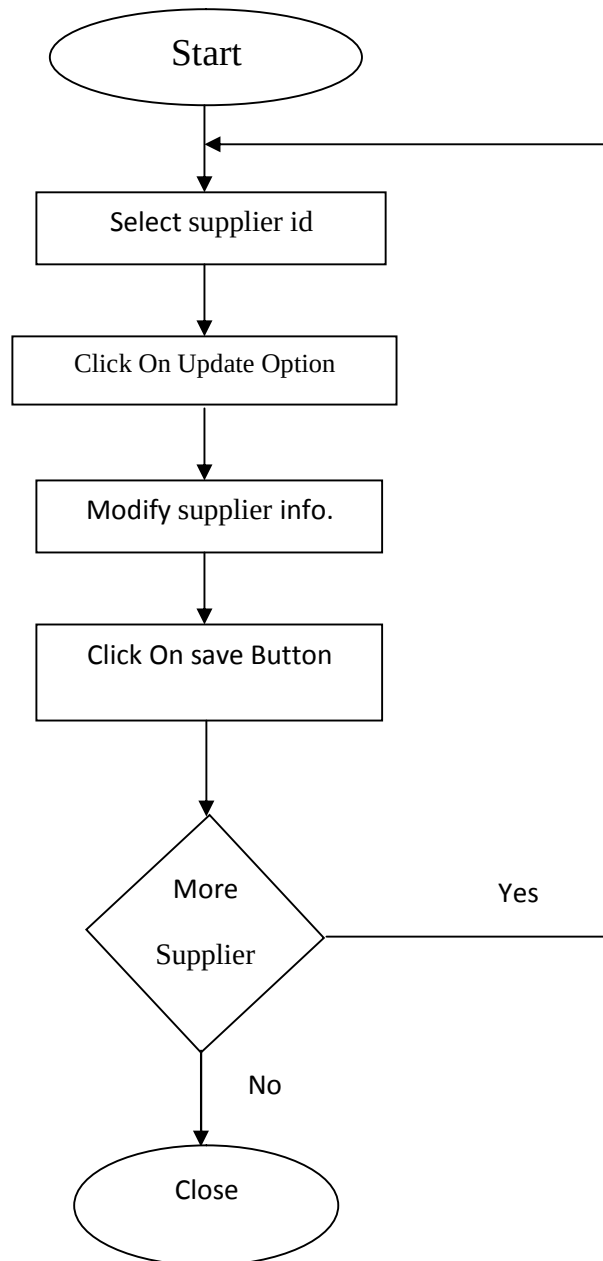


Fig.A3.5.b: The Flow Chart to edit supplier information

As shown in Fig.A3.4.b, when the administrator want to edit supplier information he has to select the supplier then click on update button and modify supplier information then clicks on save button to update these information in the supplier table. Finally check if there is another supplier there is a need to update his information, if yes then do the procedure again, otherwise close this page.

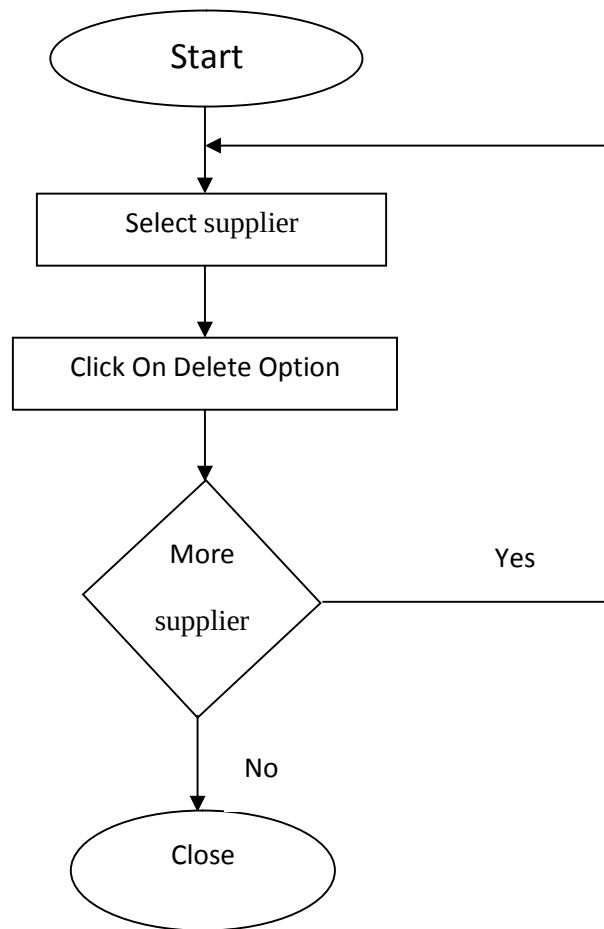


Fig.A3.5.c: The Flow Chart to delete supplier

As shown in Fig.A3.5.c, when the user want to delete supplier from the database he has to select the supplier and clicks on delete button then supplier information will be deleted from supplier table. Finally check if there is another supplier, there is need to delete his information, if yes then do the procedure again, otherwise close this page.

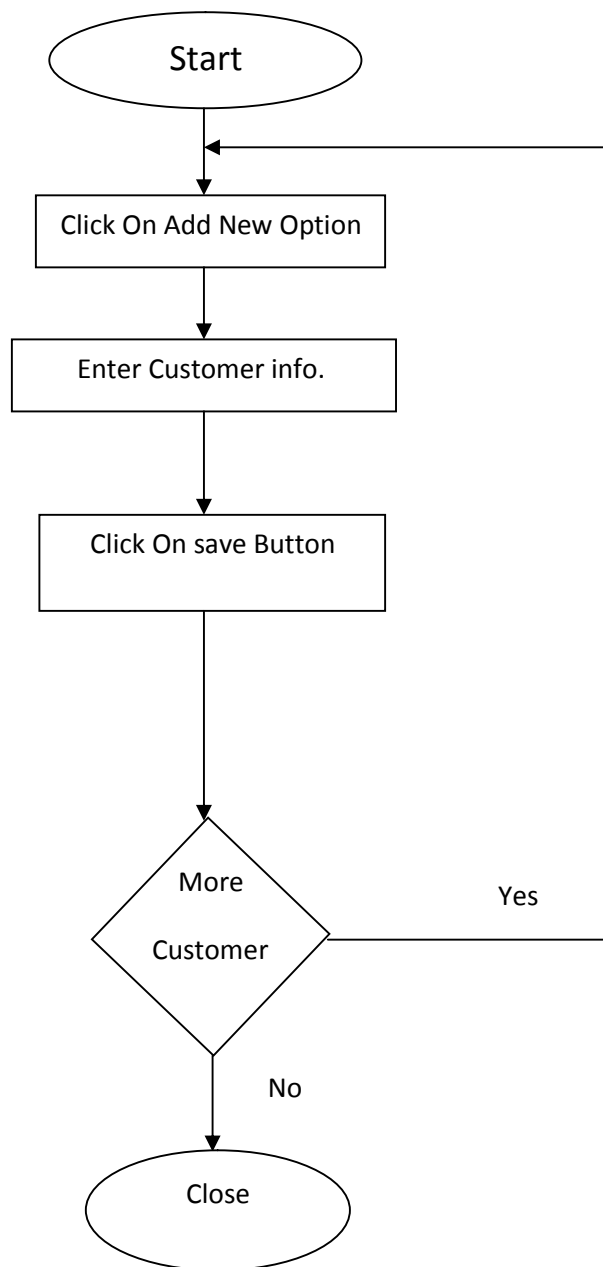


Fig.A3.6.a: The Flow Chart to Add New customer

As shown in Fig.A3.6.a, when the user want to add new customer he has to clicks on add new button then enter customer information as name, address, city, pin code, phone, and email, then clicks on save button to save these information in the database in the customer table. Finally check if there is another customer to be added if yes then do the procedure again, otherwise close this page.

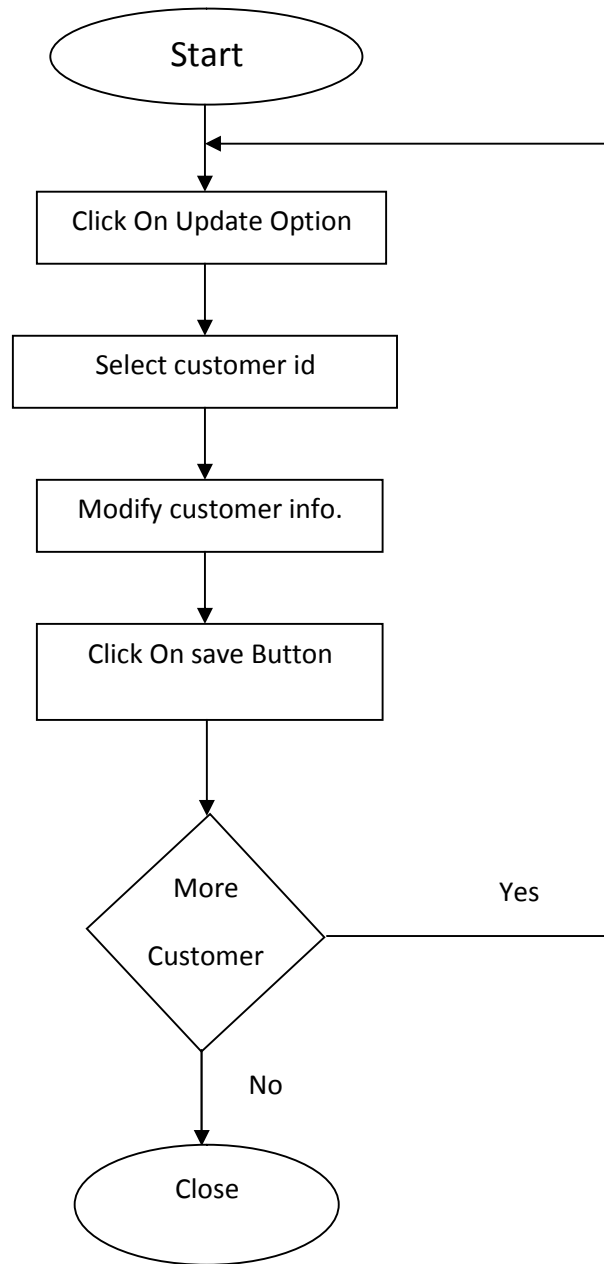


Fig.A3.6.b: The Flow Chart to edit customer information

As shown in Fig.A3.6.b, when the user want to edit customer information he has to select the customer then click on update button and modify customer information then clicks on save button to update these information in the customer table. Finally check if there is another customer there is a need to update his information, if yes then do the procedure again, otherwise close this page.

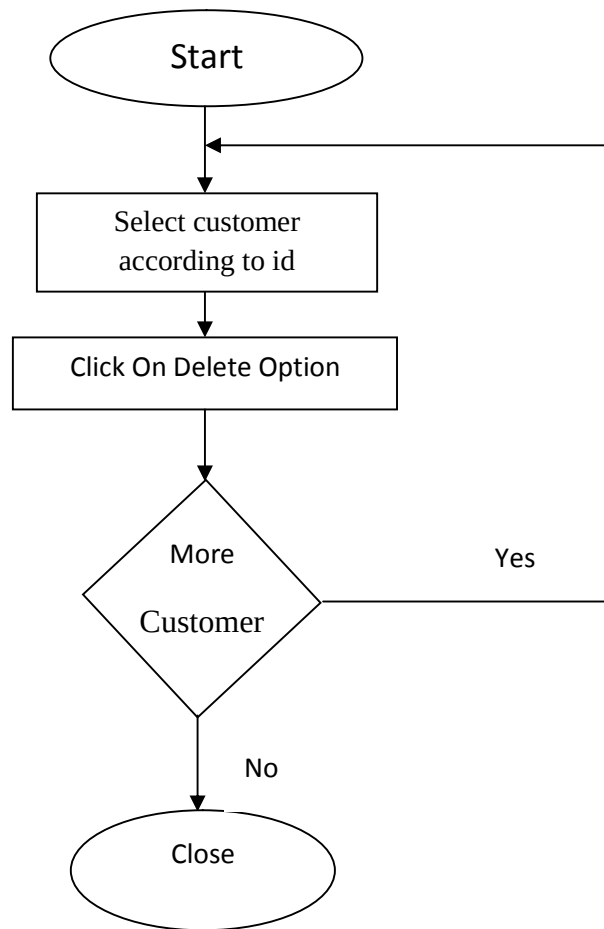


Fig.A3.6.c: The Flow Chart to delete customer

As shown in Fig.A3.6.c, when the user want to delete customer from the database he has to select the customer and clicks on delete button then customer information will be deleted from customer table. Finally check if there is another customer, there is need to delete his information, if yes then do the procedure again, otherwise close this page.

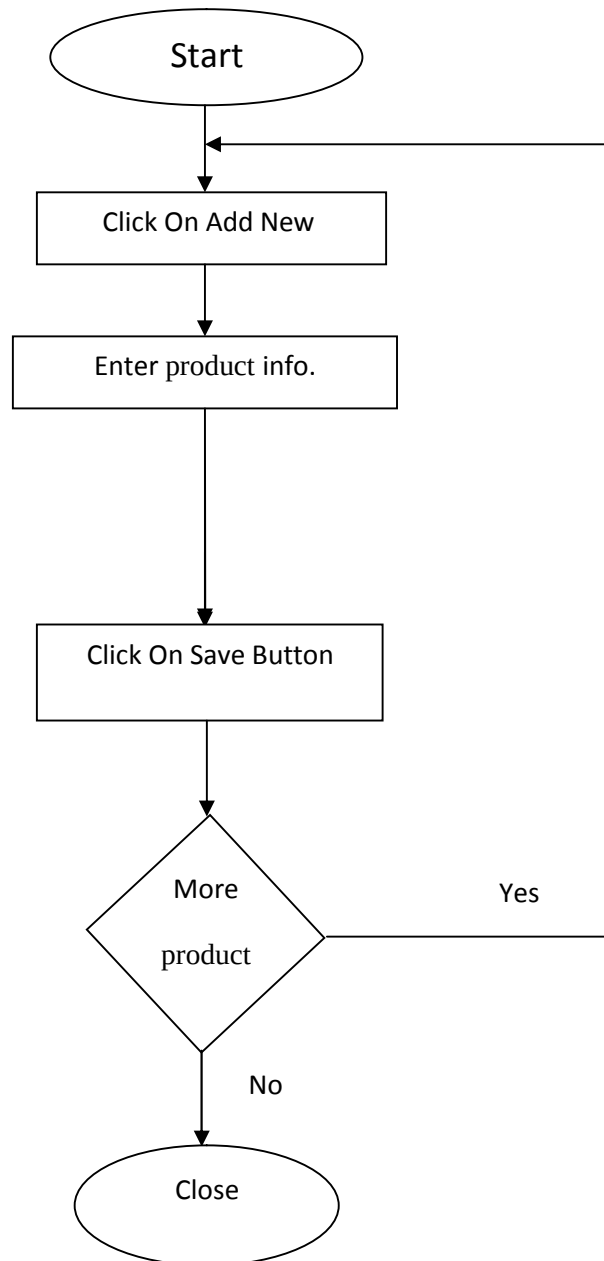


Fig.A3.7.a: The Flow Chart to Add New product

As shown in Fig.A3.7.a, when the user want to add new product he has to clicks on add new button then enter product information as name, maximum price, purchase price, product type, and sale price, then clicks on save button to save these information in the database in the product master table. Finally check if there is another product to be added if yes then do the procedure again, otherwise close this page.

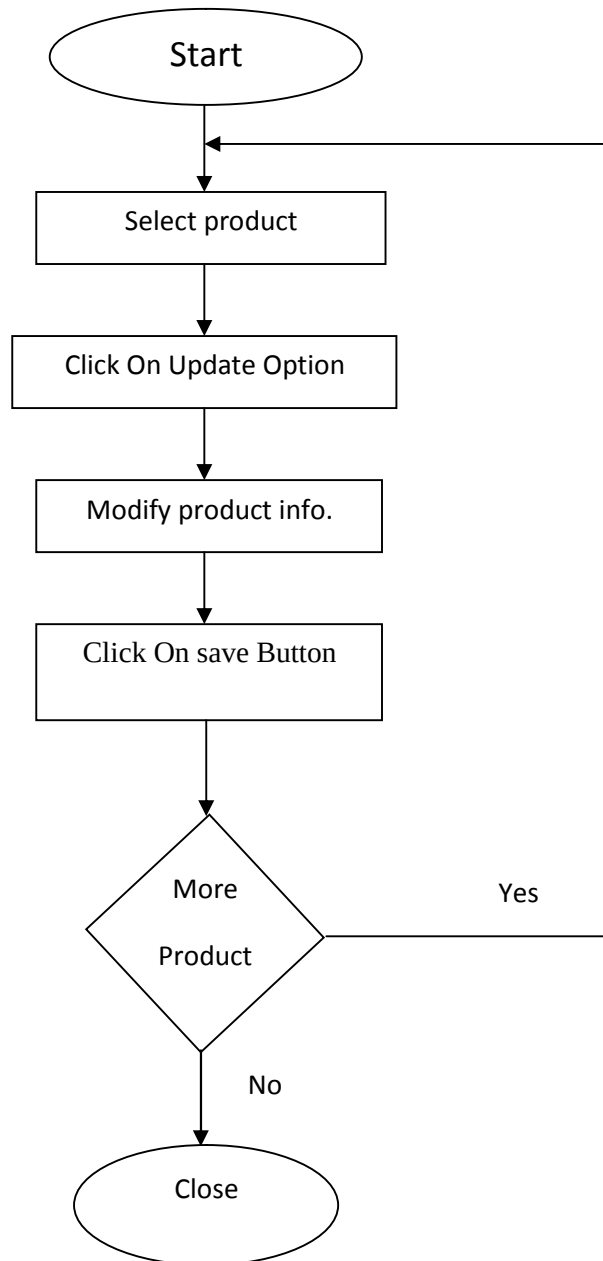


Fig.A3.7.b: The Flow Chart to edit product

As shown in Fig.A3.7.b, when the user want to edit product information he has to select the product then click on update button and modify product information then clicks on save button to update these information in the product table. Finally check if there is another product need to update it's information, if yes then do the procedure again, otherwise close this page.

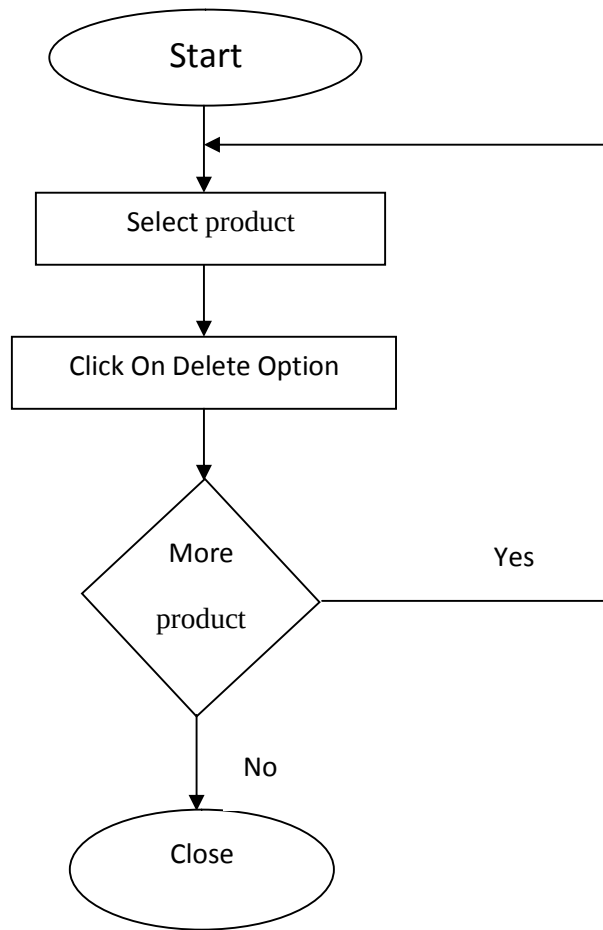


Fig.A3.7.c: The Flow Chart to delete product

As shown in Fig.A3.7.c, when the user want to delete product from the database he has to select the product and clicks on delete button then product information will be deleted from product table. Finally check if there is another product, there is need to delete it's information, if yes then do the procedure again, otherwise close this page.

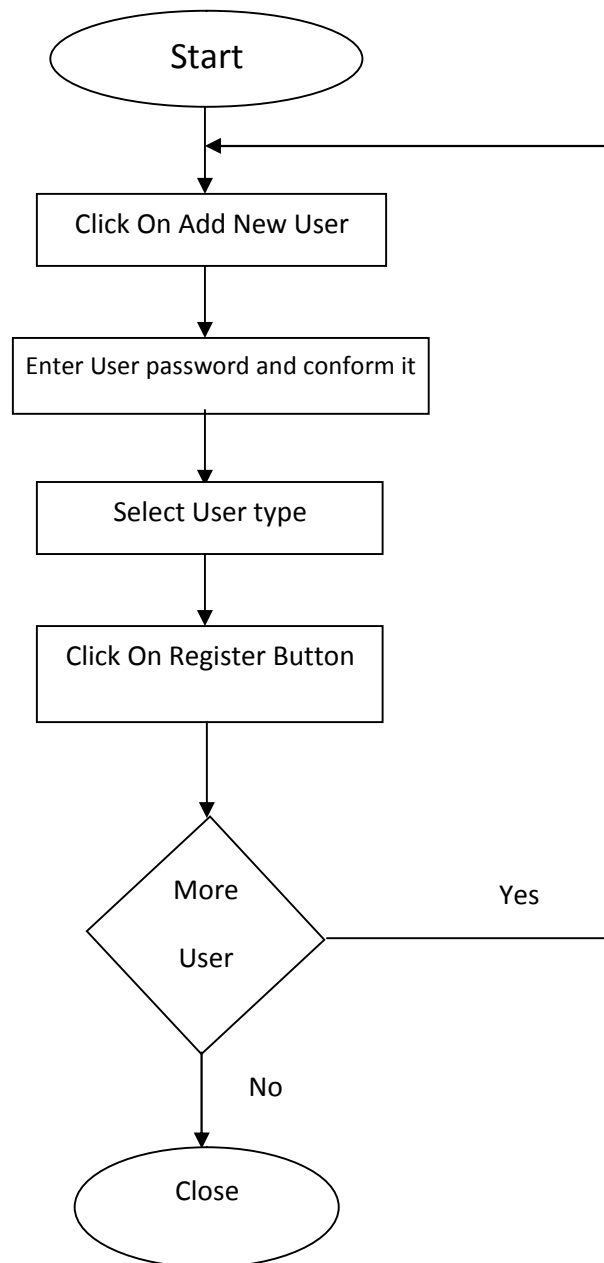


Fig.A3.8.a: The Flow Chart to Add New user

As shown in Fig.A3.8.a, when the administrator want to add new user he has to clicks on add new button then enter user name, and password then confirm this password, then select user type and clicks on register button to save these information in the database. Finally check if there is another user to be added if yes then do the procedure again, otherwise close this page.

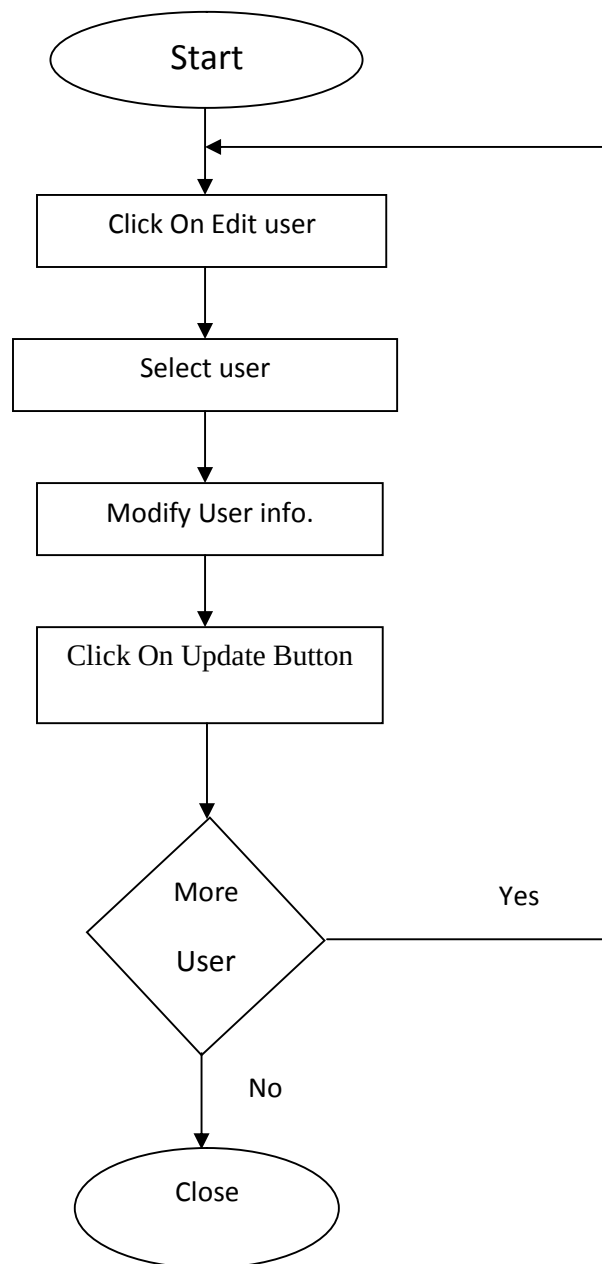


Fig.A3.8.b: The Flow Chart to edit user

As shown in Fig.A3.8.b, when the administrator want to edit user information he has to click on edit button then select the user and modify user information then clicks on update button to update these information in the user login table. Finally check if there is another user need to update his information, if yes then do the procedure again, otherwise close this page.

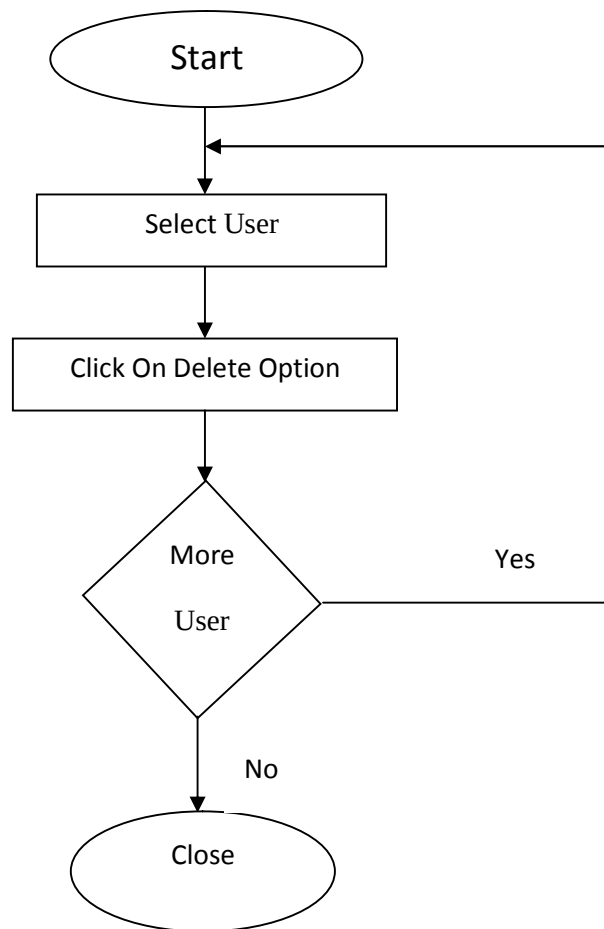


Fig.A3.8.c: The Flow Chart to delete user

As shown in Fig.A3.8.c, when the user want to delete user from the database he has to select the user and clicks on delete button then user information will be deleted from user login table. Finally check if there is another user, there is need to delete his information, if yes then do the procedure again, otherwise close this page.

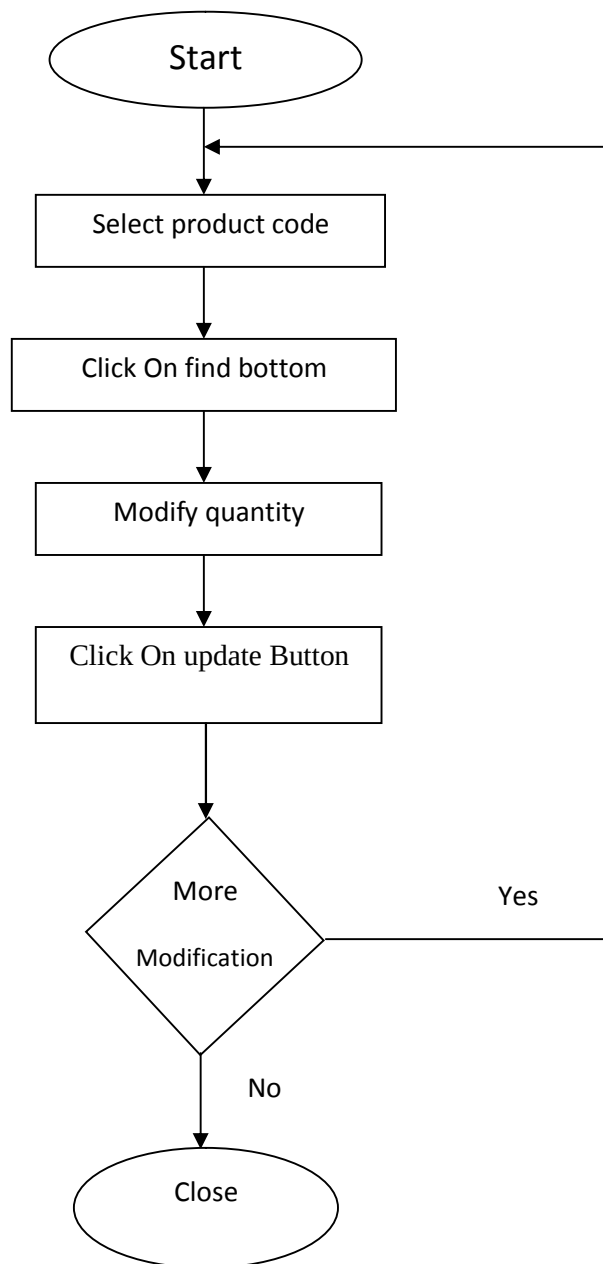


Fig.A3.9: The Flow Chart to update stock master

As shown in Fig.A3.9, when the user want to edit stock master information he has to select the product, then click on find button, then modify quantity then clicks on update button to update these information in the stock master table. Finally check if there is another modification required, if yes then do the procedure again, otherwise close this page.

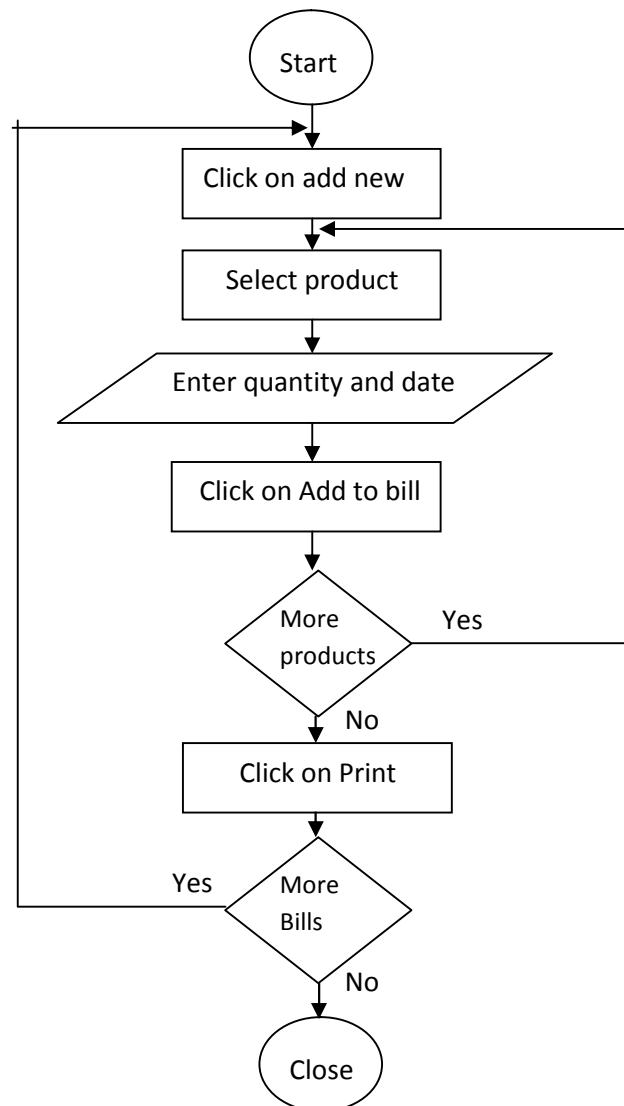


Fig.A3.10: The Flow Chart for selling an item

As shown in Fig.A3.10, when the user want to sale item to customer, first of all click on add new button to generate order number, then select product according to customer requirement, then enter the quantity and date, after that click on add to bill, then check if there is another product the customer want to be added to the bill, if yes then go to select product and repeat same procedure, if no need to add other product click on print bill. Finally check if there is another bill required, if yes then do the procedure again, otherwise close this page.

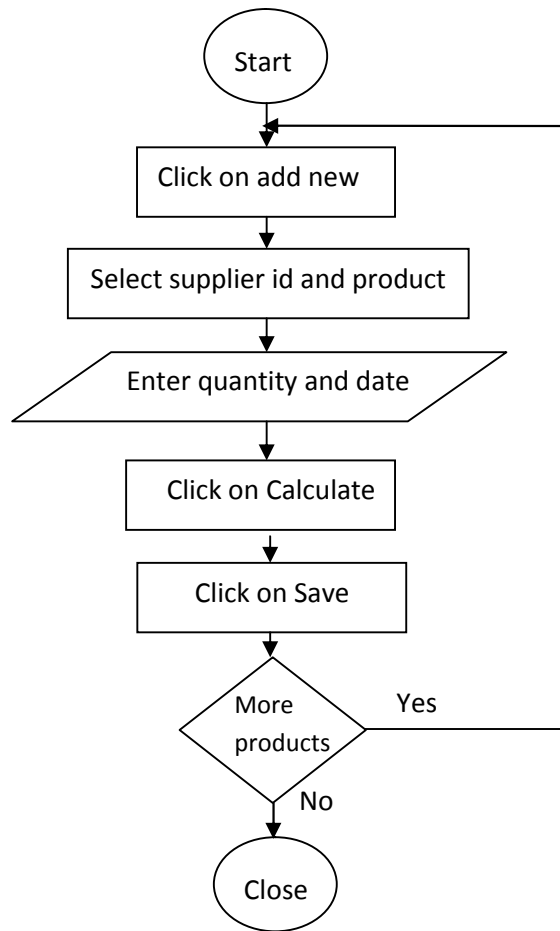


Fig.A3.11: The Flow Chart for Purchase an item

As shown in Fig.A3.11, when the user want to purchase product, first of all click on add new button to generate serial number, then select supplier id and product, then enter the quantity and date, after that click on calculate button, then click on save button to save information to purchase table and update product quantity in stock master table. Finally check if there is another product required, if yes then do the procedure again, otherwise close this page.

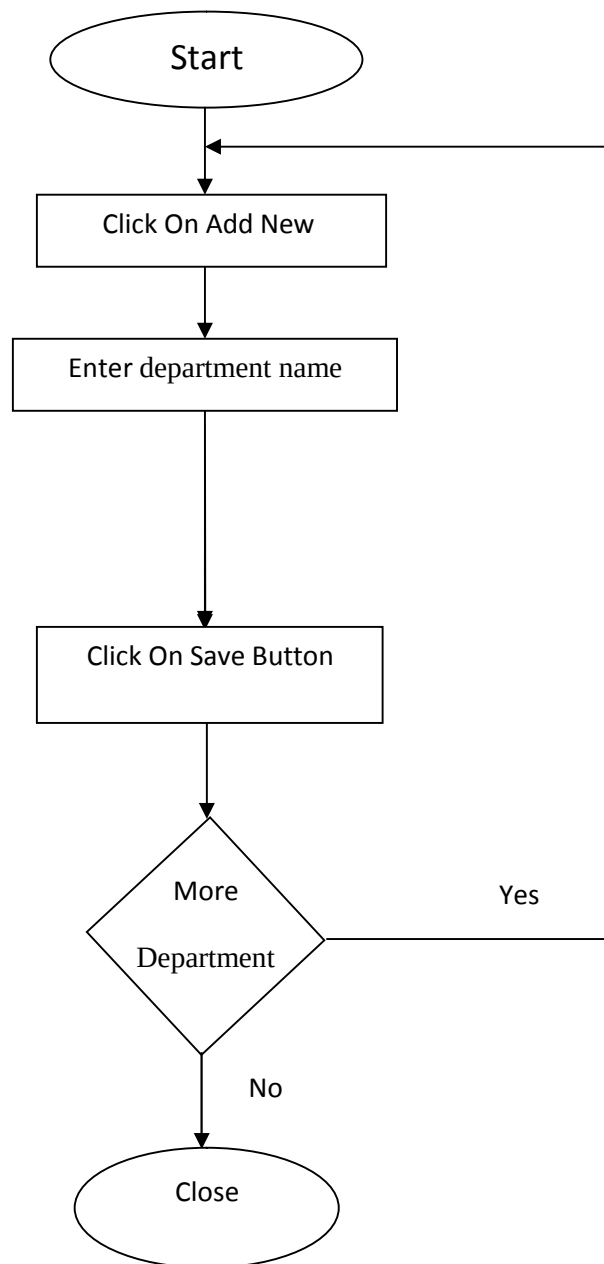


Fig.A3.12.a: The Flow Chart to Add New department

As shown in Fig.A3.12.a, when the administrator want to add new department he has to clicks on add new button then enter department name, then clicks on save button to save these information in the database in the department table. Finally check if there is another department to be added if yes then do the procedure again, otherwise close this page.

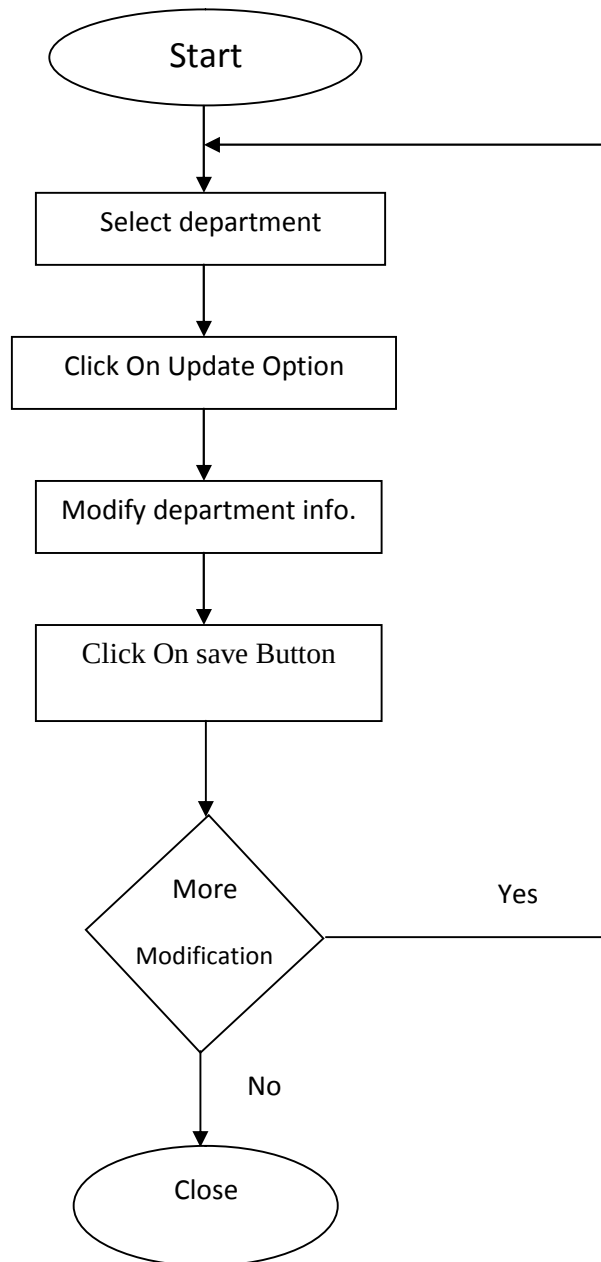


Fig.A3.12.b: the flowchart to edit department

As shown in Fig.A3.12.b, when the administrator want to edit department information he has to select the department then click on update button and modify department information then clicks on save button to update these information in the department table. Finally check if there is another department need to update his information, if yes then do the procedure again, otherwise close this page.

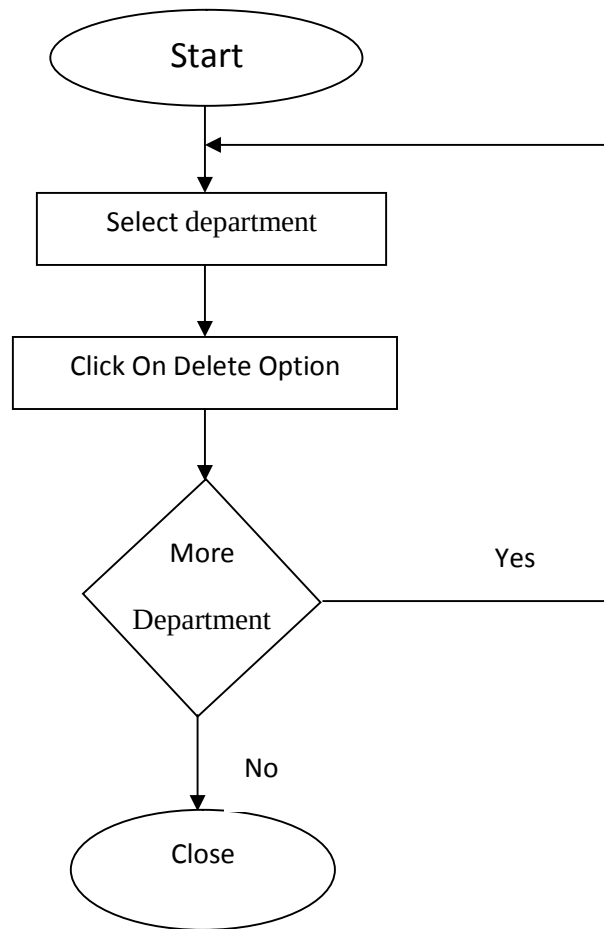


Fig.A3.12.a: The Flow Chart to delete department

As shown in Fig.A3.12.c, when the user want to delete department from the database he has to select the department and clicks on delete button then department information will be deleted from department table. Finally check if there is another department, there is need to delete it's information, if yes then do the procedure again, otherwise close this page.

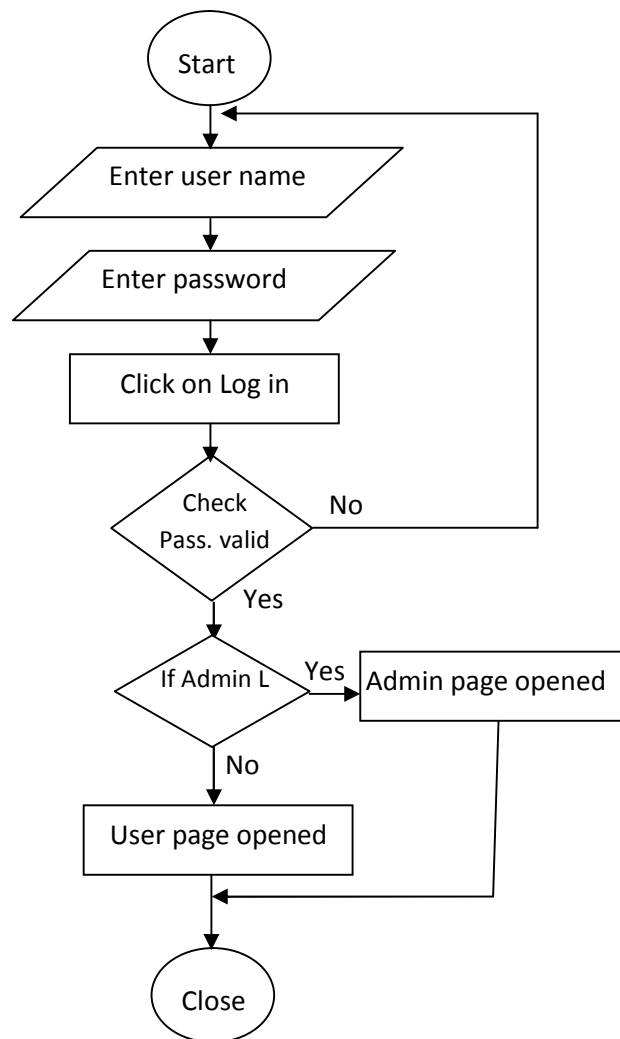


Fig.A3.13: The Flow Chart to Login process

As shown in Fig.A3.13, when the user want to log in to the system, he has to enter user name and password,. Then click on log in button, then th system will check the password if not valid re-enter password, otherwise the system check user type, if he is admin then administrator page will be opened, otherwise user page will be opened. Finally when the user completes his work he can logout by just click on log out button.

Use Case Diagrams:

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well. The use case diagram of the re-engineered supermarket management system are shown in Fig.A3.14:

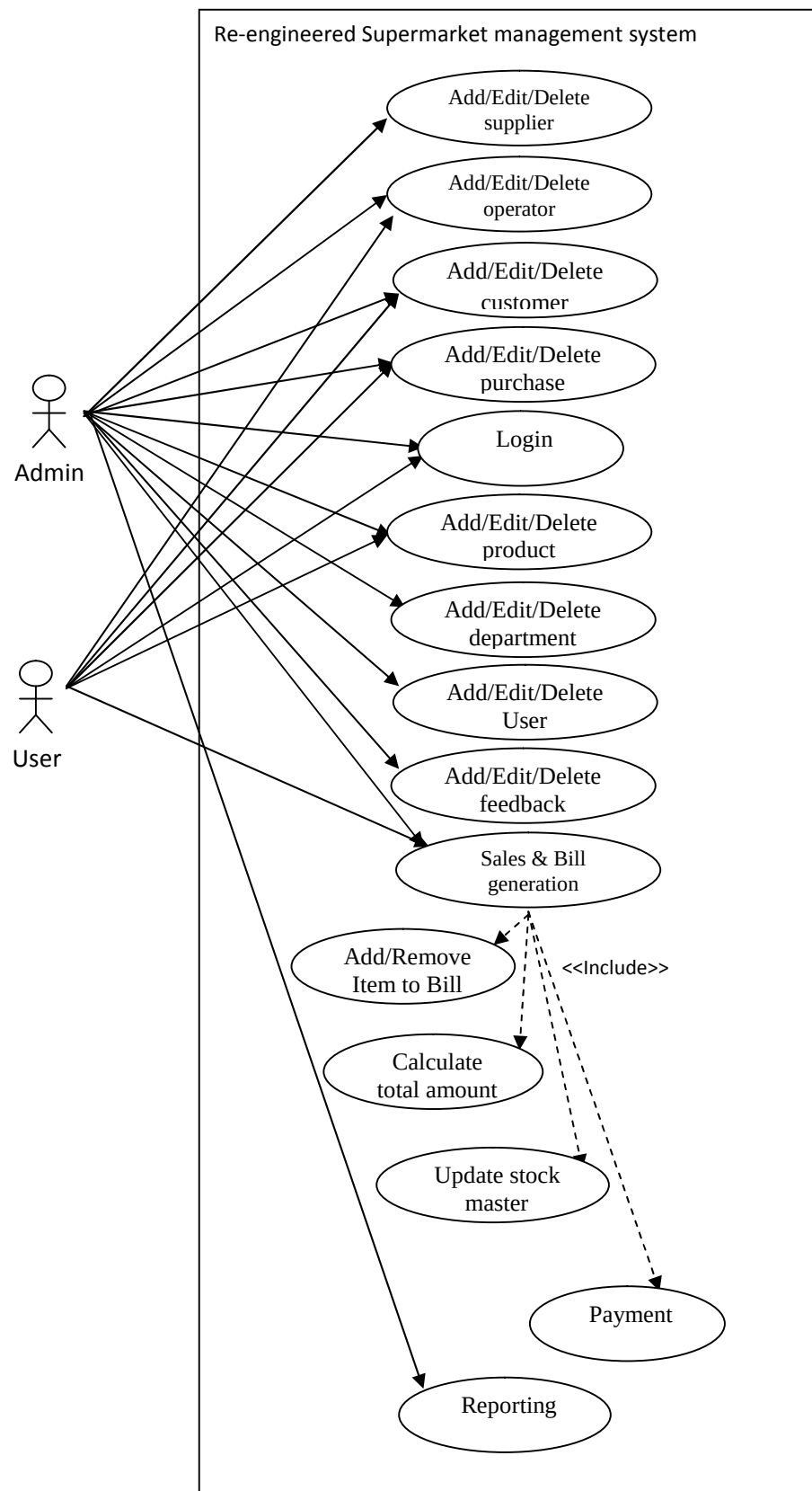


Fig.A3.14: The use case diagram of the re-engineered supermarket management system

Entity Relationship Diagrams:

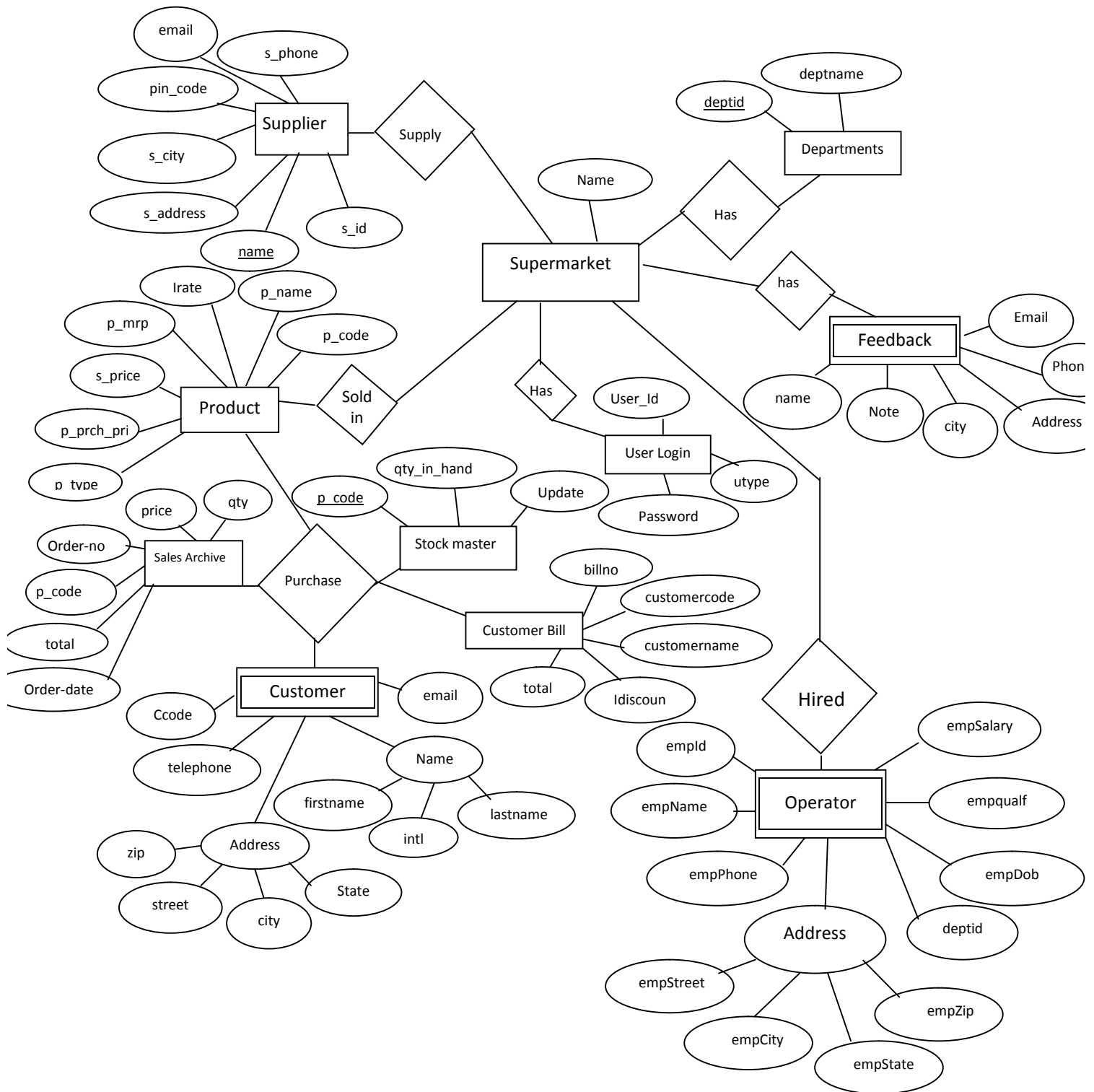


Fig.A3.15: The ERD diagram of the re-engineered supermarket management system

Database Design:

The database code to design the operator information table is shown below, and the table content shown in Table A3.1:

```
create table [operator information](  
o_id int,  
o_dept varchar(50),  
o_salary float,  
o_name varchar(50),  
join_date datetime(50),  
o_address varchar(50),  
o_phon varchar(50),  
o_email varchar(50),  
PRIMARY KEY (o_id));
```

Table A3.1: Operators information

Data field	Type	Constraint	Description
o_id	int	PRIMARY KEY,	Operator identification number
o_dept	varchar(50)	-----	Department, where operator work
o_salary	float	-----	The salary of the operator
o_name	varchar(50)	-----	Operator name
join_date	varchar(50)	-----	Date when operator hired in supermarket
o_address	varchar(50)	-----	Operator address
o_phon	varchar(50)	-----	Operator phone
o_email	varchar(50)	-----	Operator email

The database code to design the supplier information table is shown below, and the table content shown in Table A3.2:

```
create table [supplier information](  
s_id int,  
s_name varchar(50),  
s_address varchar(50),  
s_city varchar(50),  
pin_code varchar(50),  
s_phone varchar(50),  
e_mail varchar(50),  
PRIMARY KEY (s_id));
```

Table A3.2: Suppliers information

Data field	Type	Constraint	Description
s_id	int	PRIMARY KEY	Supplier identification number
s_name	varchar(50)	-----	Supplier name
s_address	varchar(50)	-----	Suppliers address
s_city	varchar(50)	-----	Supplier city
pin_code	varchar(50)	-----	Pin-code of the city
s_phone	varchar(50)	-----	Supplier phone
e_mail	varchar(50)	-----	Supplier email

The database code to design the customer information table is shown below, and the table content shown in Table A3.3:

```
create table [customer information](
c_id int,
c_name varchar(50),
c_address varchar(50),
c_city varchar(50),
pin_code varchar(50),
c_phone varchar(50),
c_e_mail varchar(50),
PRIMARY KEY (c_id));
```

Table A3.3: Customers information of re-engineered system

Data field	Type	Constraint	Description
c_id	Int	PRIMARY KEY	Customer identification number
c_name	varchar(50)	-----	Customer name
c_address	varchar(50)	-----	Customer address
c_city	varchar(50)	-----	Customer city
pin_code	varchar(50)	-----	Pin-code of the city
c_phone	varchar(50)	-----	Customer phone
c_e_mail	varchar(50)	-----	Customer email

The database code to design the product master table is shown below, and the table content shown in Table A3.4:

```
create table [product master] (
p_code int,
p_name varchar(50),
```

```

p_mrp float,
p_prch_price float,
p_type varchar(50),
s_price float,
PRIMARY KEY (p_code));

```

Table A3.4: Products information

Data field	Type	Constraint	Description
p_code	int	Primary key	Product code
p_name	varchar(50)	-----	Product name
p_mrp	float	-----	Product maximum price
p_prch_price	float	-----	Product purchase price
p_type	varchar(50)	-----	Product type
s_price	float	-----	Product sale price

The code to create the sale table is shown below, and the table content shown in Table A3.5;

```

create table sale
(
order_no int,
p_code varchar(50),
qty int,
s_price float,
total float,
order_date datetime,
);

```

Table A3.5: Sale information

Data field	Type	Constraint	Description
order_no	int	-----	The number of the order
p_code	varchar(50)	-----	Product code
qty	int	-----	quantity
s_price	Float	-----	Sale price
total	Float	-----	Total amount
order_date	datetime	-----	The date of this order

The code of purchase table is shown below, and the table content shown in Table A3.6.

```
create table purchase
(
sr_no int,
s_id int,
i_code varchar(50),
qty int,
p_price float,
total float,
p_date datetime,
PRIMARY KEY (sr_no));
```

Table A3.6: Purchase information

Data field	Type	Constraint	Description
sr_no	int	PRIMARY KEY	Serial number of the purchased information
s_id	int	-----	Supplier id
i_code	varchar(50)	-----	Item code
qty	int	-----	Quantity of purchased items
p_price	float	-----	Purchase price
total	float	-----	Total amount of price
p_date	datetime,	-----	Purchase date

The code to create the department table is shown below; the table content shown in Table A3.7.

```
create table department(
deptId varchar(50),
deptname varchar(50));
```

Table A3.7: Department information of re-engineered system

Data field	Type	Constraint	Description
deptid	varchar(50)	-----	Department identification number
deptname	varchar(50)	-----	Department name

The code to create the feedback table is shown below; the table content shown in Table A3.8:

```
create table feedback(
name varchar(50),
address varchar(50),
city varchar(50),
phone varchar(50),
email varchar(50),
note text,
);
```

Table A3.8: Feedback information

Data field	Type	Constraint	Description
name	varchar(50)	-----	Name of the person that send the feedback
Address	varchar(50)	-----	Address of that person
city	varchar(50)	-----	City of that person
phone	varchar(50)	-----	Phone number of that person
email	varchar(50)	-----	Email address of that person
note	text	-----	The feedback message that sent by that person

The code to create the stock master table is shown below; the table content shown in Table A3.9:

```
create table [stock master](
p_code int,
qty_in_hand int,
[update] int);
```

Table A3.9: Stock information

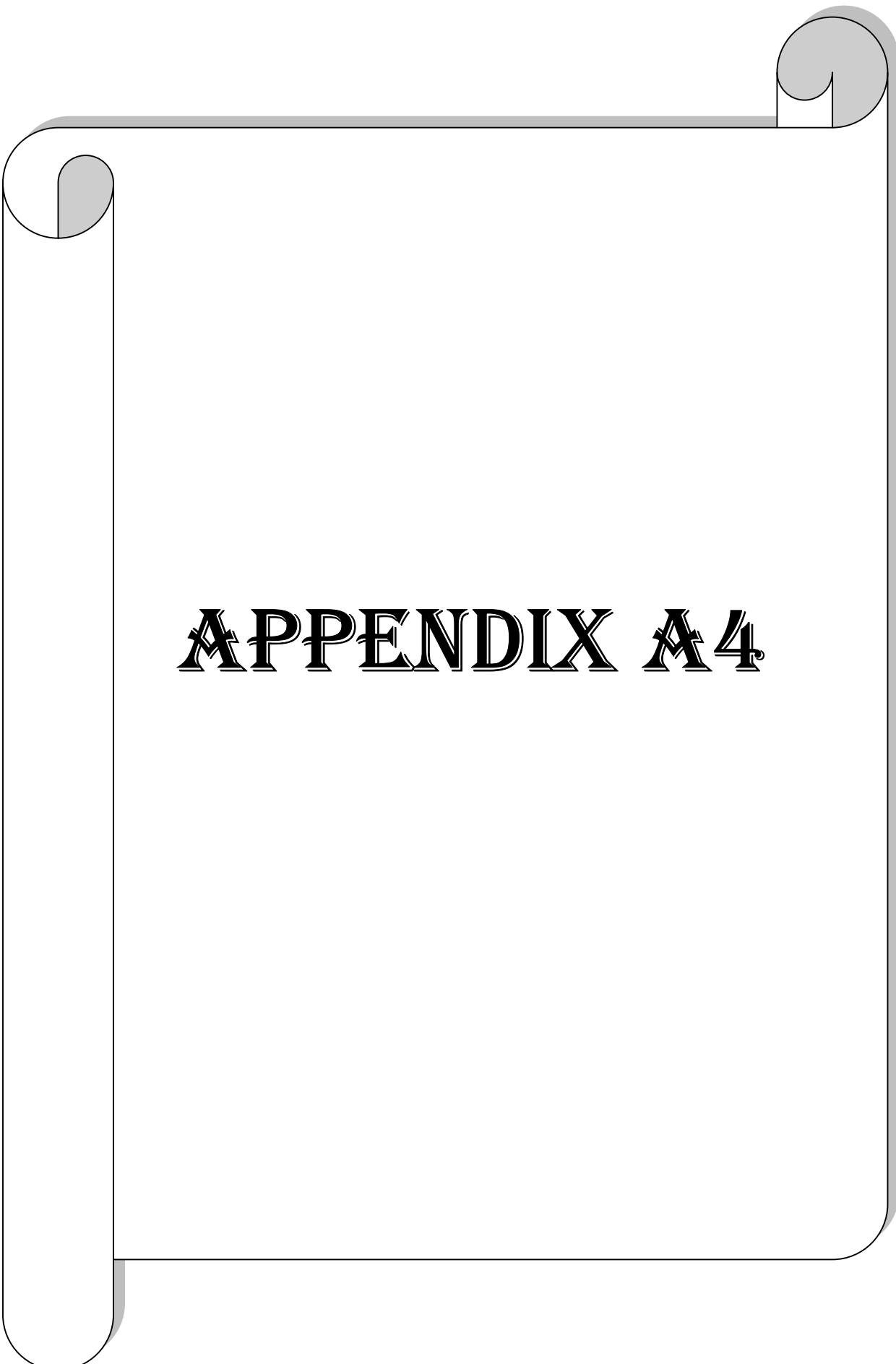
Data field	Type	Constraint	Description
p_code	int	-----	Product code
qty_in_hand	int	-----	Quantity available in stock
[update]	int	-----	New added quantity

The code to create the “User Login” table is shown below; the table content shown in Table A3.10.

```
create table [User Login](  
User_Id varchar(50) PRIMARY KEY,  
Password varchar(50),  
Utype varchar(50));
```

Table A3.10: User login information

Data field	Type	Constraint	Description
User_Id	varchar(50)	PRIMARY KEY	User identification
Password	varchar(50)	-----	Password required to log in to the system
Utype	varchar(50)	-----	User type which he either employee or administrator



APPENDIX A4

APPENDIX A4

THE UML ACTIVITY DIAGRAMS OF THE RE-ENGINEERED SUPERMARKET SYSTEM

The “Visustin v7” analysis tool was used to visualize the code of the re-engineered supermarket management system with flow charts; Visustin converts source code to flow charts to visualize the code of re-engineered system for the next iteration, if required, in FUSION model. Now the code of the “about.aspx.cs” was analyzed to convert it to UML activity diagrams as shown in Fig.A4.1:

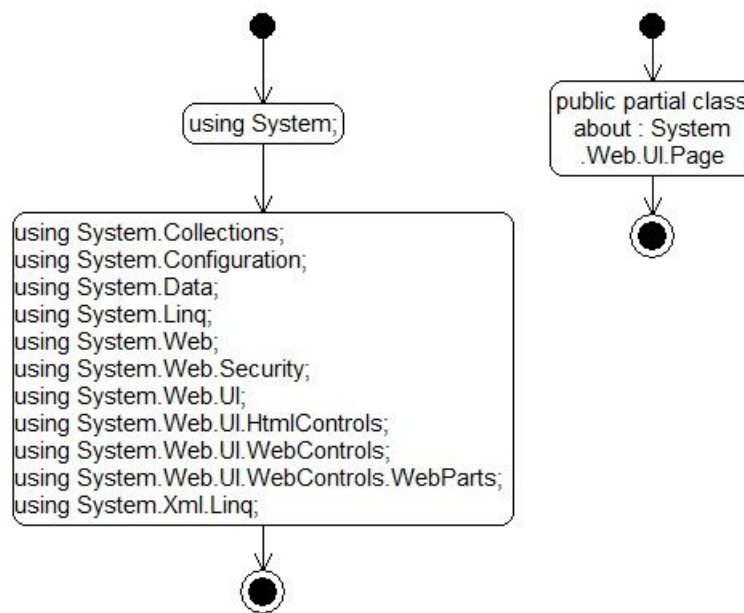


Fig.A4.1: UML activity diagram of the structures [using system, public partial class about: System.Web.UI.Page] in “about.aspx.cs”

As shown in the Fig.A4.1, at first, all the used name spaces were determined, where the ‘System.Collections’ namespace contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hash tables and dictionaries, The ‘System.Configuration’ namespace contains the types that provide the programming model for handling configuration data. The ‘System.Data’ namespace provides access to classes that represent the ‘ADO.NET’ architecture. ‘ADO.NET’ lets you build components that efficiently manage data from multiple data sources. The ‘System.Linq’ namespace provides classes and interfaces that support queries that use Language-Integrated Query (LINQ). The ‘System.Web’ namespace supplies classes and interfaces that enable browser-

server communication. This namespace includes the `HttpRequest` class, which provides extensive information about the current HTTP request; the `HttpResponse` class, which manages HTTP output to the client; and the `HttpServerUtility` class, which provides access to server-side utilities and processes. 'System.Web' also includes classes for cookie manipulation, file transfer, exception information, and output cache control. The 'System.Web.Security' namespace contains classes that are used to implement ASP.NET security in Web server applications. The 'System.Web.UI' namespace provides classes and interfaces that enable you to create ASP.NET server controls and ASP.NET Web pages for the user interface of your ASP.NET Web applications. This namespace includes the `Control` class, which provides a common set of functionality for all server controls, which includes HTML server controls, Web server controls, and user controls. It also includes the `Page` class. This class is generated automatically whenever a request is made for an .aspx file in an ASP.NET Web application. You can inherit from both of these classes. The namespace also includes classes that provide the server controls with data-binding functionality, the ability to save the view state of a given control or page, and parsing functionality. 'System.Web.UI.HtmlControls' class defines the methods, properties, and events common to all HTML server controls in the ASP.NET page framework. The 'System.Web.UI.WebControls' namespace contains classes that allow you to create Web server controls on a Web page. Web server controls run on the server and include form controls such as buttons and text boxes. They also include special-purpose controls such as a calendar. Because Web server controls run on the server, you can programmatically control these elements. Although Web server controls are rendered as HTML, their object model does not necessarily reflect HTML syntax. The 'System.Web.UI.WebControls.WebParts' namespace contains classes and interfaces known collectively as the Web Parts control set. You can use these classes to create Web pages whose appearance and behavior can be modified 'personalized' by end users. Each user's personalized settings for a page are saved for future browser sessions. This contains the classes for (LINQ to XML), it is an in-memory XML programming interface that enables you to modify XML documents efficiently and easily. Then define class named as 'about' and use 'System.Web.UI.Page' that represents an (.aspx file), also known as a Web Forms page, requested from a server that hosts an ASP.NET Web application.

The next step is to visualize the codes of the “admin.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.2:

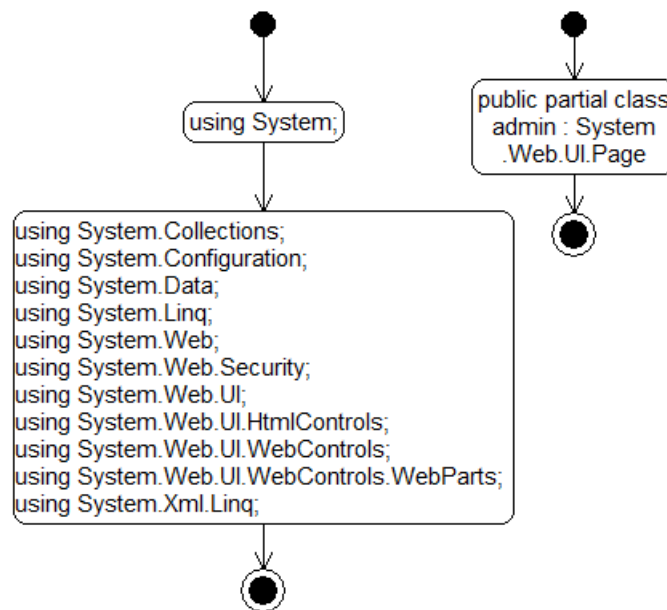


Fig.A4.2: UML activity diagram of the structures [using System, public partial class admin : System.Web.UI.Page] in “admin.aspx.cs”

As shown in the Fig.A4.2, at first, all the used name spaces were determined, they are (System.Collections), (System.Configuration), (System.Data), (System.Linq), (System.Web), (System.Web), (System.Web.Security), (System.Web.UI), (System.Web.UI.HtmlControls), (System.Web.UI.WebControls.WebParts) namespace and (XML.LINQ), all these name spaces were previously defined in this Appendix. Then define class named as (admin) and use (System.Web.UI.Page) that represents an (.aspx file), also known as a Web Form page, requested from a server that hosts an ASP.NET Web application.

The next step is to visualize the codes of the “admin master.cs” to convert this code to UML activity diagrams as shown in Fig.A4.3.a and Fig.A4.3.b:

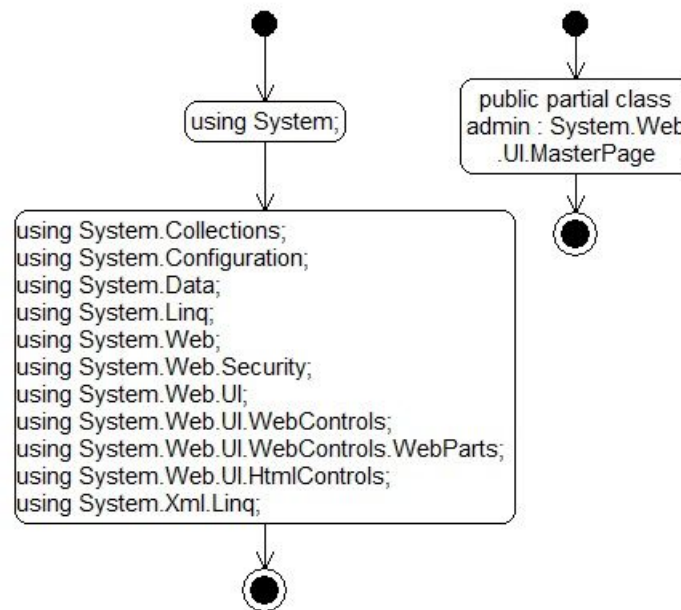


Fig.A4.3.a: UML activity diagram of the structures [using System, public partial class admin : System.Web.UI.MasterPage] in “admin master.cs”

As shown in the Fig.A4.3.a, at first, all the used name spaces were determined, they are (System.Collections), (System.Configuration), (System.Data), (System.Linq), (System.Web), (System.Web.Security), (System.Web.UI), (System.Web.UI.HtmlControls), and (System.Web.UI.WebControls.WebParts) namespace and (XML.LINQ), all these name spaces were previously defined in this Appendix. Then define class named as (admin) and use (System.Web.UI. MasterPage) that represents an (.aspx file) for master page, also known as a Web Forms page, which acts as a template and merging container for pages that are composed only of Content controls and their respective child controls.

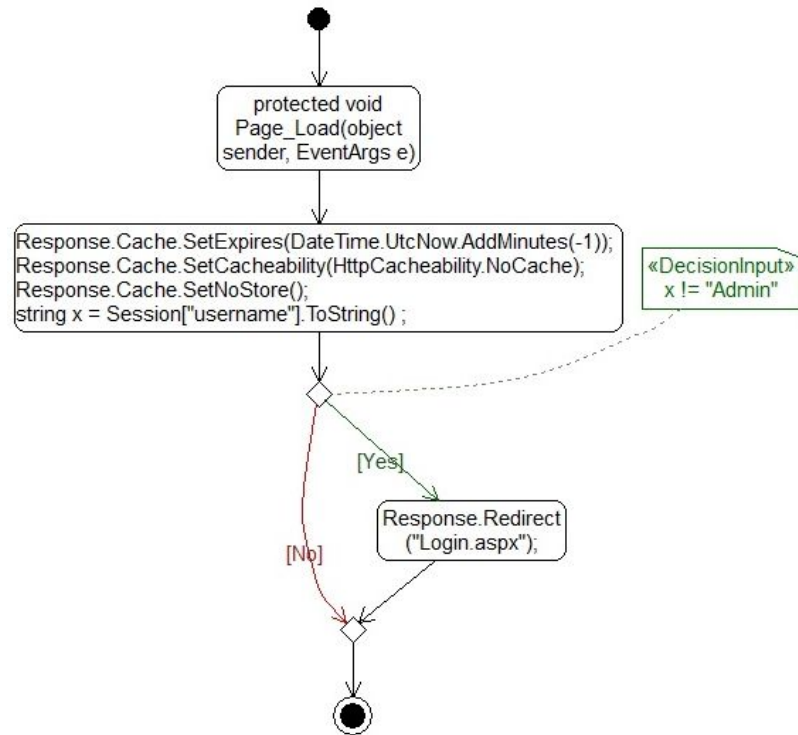


Fig.A4.3.b: UML activity diagram of the structures [protected void Page_Load(object sender, EventArgs e)] in “admin master.cs”

As shown in Fig.A4.3.b, when protected admin master page is loaded, the system will check the session if it is not equal to ‘admin’ that mean that person is not authorized to enter this page, so the system automatically redirect to (login page).

The next step is to visualize the codes of the “employee.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.4.a, Fig.A4.4.b, and Fig.A4.4.c:

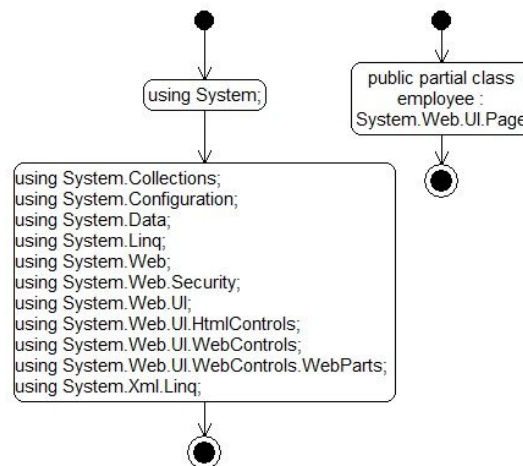


Fig.A4.4.a: UML activity diagram of the structures of [using System, public partial class employee : System.Web.UI.Page] in " employee.aspx.cs"

As shown in the Fig.A4.4.a, at first, all the used name spaces were determined. Then define class named as ‘employee’.

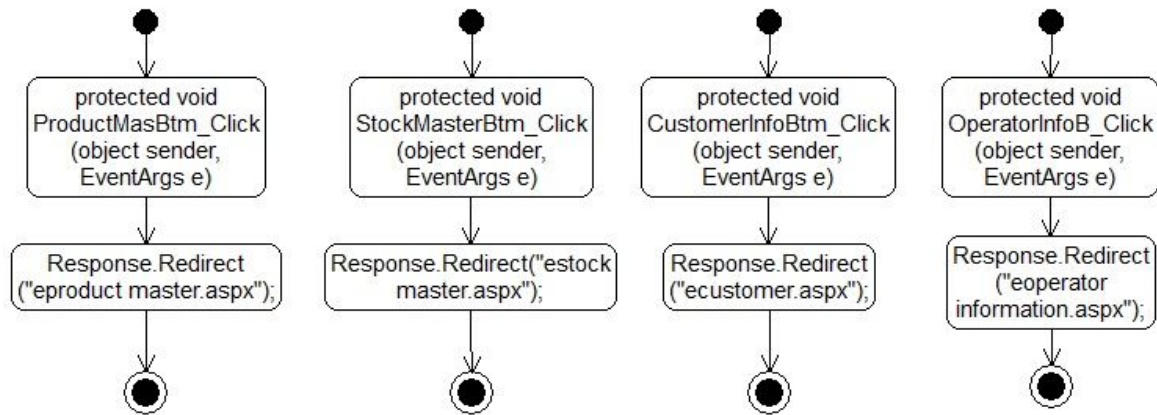


Fig.A4.4.b: UML activity diagram of the structures of [protected void ProductMasBtm_Click (object sender, EventArgs e), protected void StockMasterBtm_Click (object sender, EventArgs e), protected void CustomerInfoBtm_Click (object sender, EventArgs e), protected void OperatorInfoB_Click (object sender, EventArgs e)] in “employee.aspx.cs”

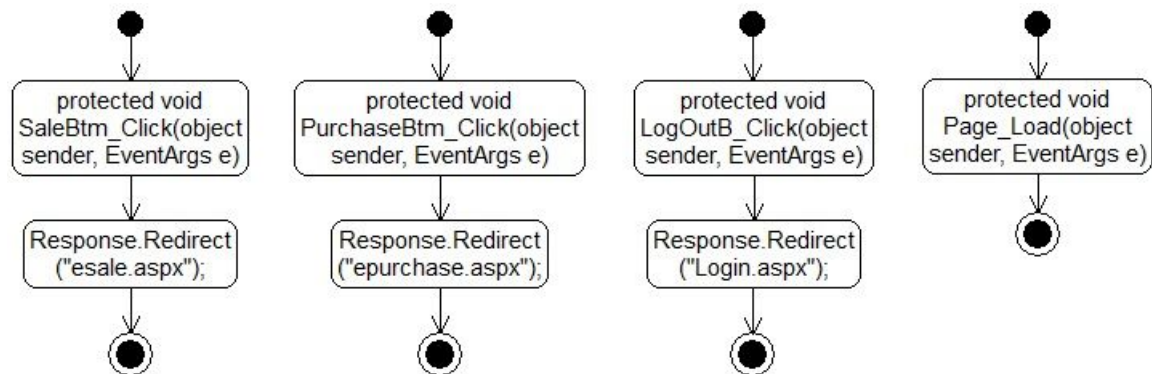


Fig.A4.4.c: UML activity diagram of the structures of [protected void SaleBtm_Click (object sender, EventArgs e), protected void PurchaseBtm_Click (object sender, EventArgs e), protected void LogOutB_Click (object sender, EventArgs e), protected void Page_Load (object sender, EventArgs e)] in “employee.aspx.cs”

As shown in Fig.A4.4.c, when the user wants to open sale page, he has to clicks on sale button, when the user want to open purchase page, he has to clicks on purchase button. When the user wants to log out, he has to clicks on log out button. When Page_Load function is executed the employee page will be loaded.

The next step is to visualize the codes of the “employee.master.cs” to convert this code to UML activity diagrams as shown in Fig.A4.5.a, and Fig.A4.5.b:

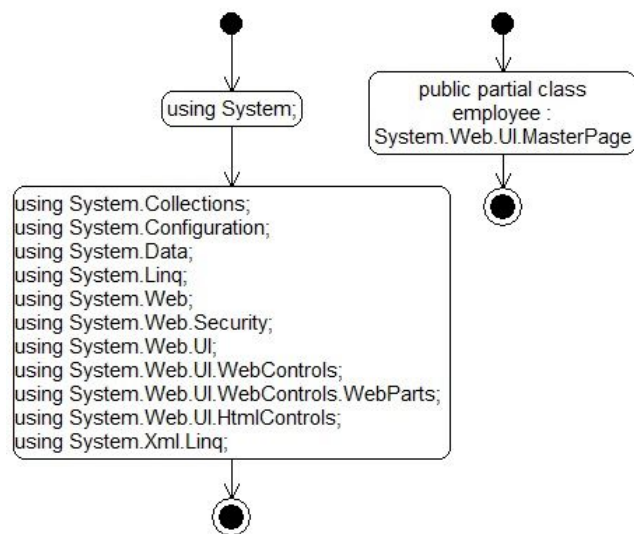


Fig.A4.5.a: UML activity diagram of the structures of [using System, public partial class employee: System.Web.UI.MasterPage] in “employee.master.cs”

As shown in the Fig.A4.5.a, at first, all the used name spaces were determined. Then define class named as ‘employee’ and use (System.Web.UI. MasterPage) that represents an (.aspx file) for master page, also known as a Web Forms page, Acts as a template and merging container for pages that are composed only of Content controls and their respective child controls.

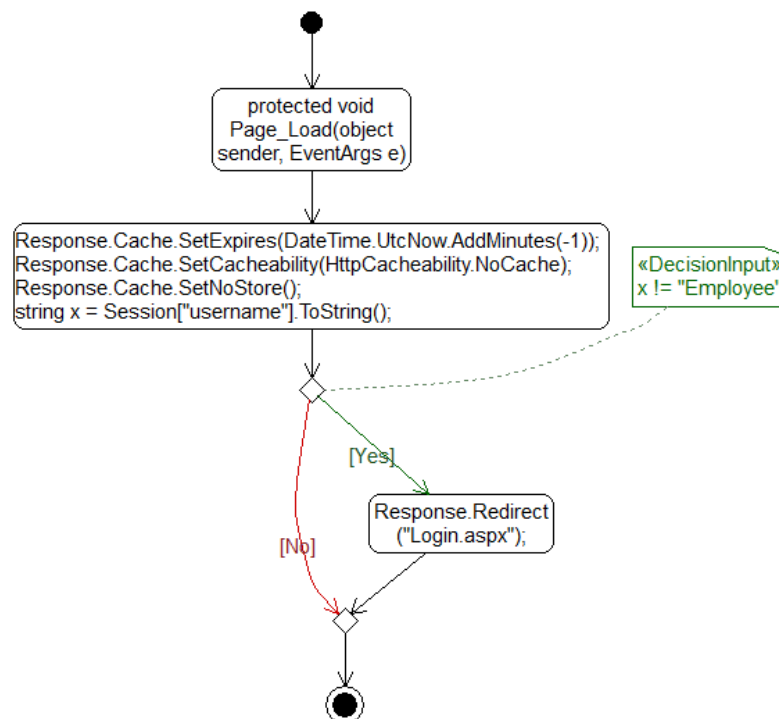


Fig.A4.5.b: UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e)] in “employee.master.cs”

As shown in Fig.A4.5.b, when protected employee master page loaded, the system will check the session if it is not equal to ‘employee’ that mean that person is not authorized to enter this page, so the system automatically redirect to ‘login page’.

The next step is to visualize the codes of the “EmployeeRe.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.6.a, and Fig.A4.6.b:

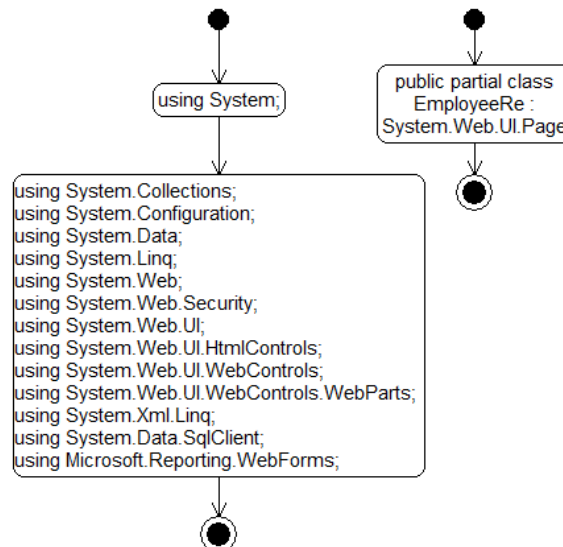


Fig.A4.6.a: UML activity diagram of the structures of [using System, public partial class EmployeeRe : System.Web.UI.Page] in “EmployeeRe.aspx.cs”

As shown in the Fig.A4.6.a, at first, all the used name spaces were determined. Then define class named as ‘EmployeeRe’.

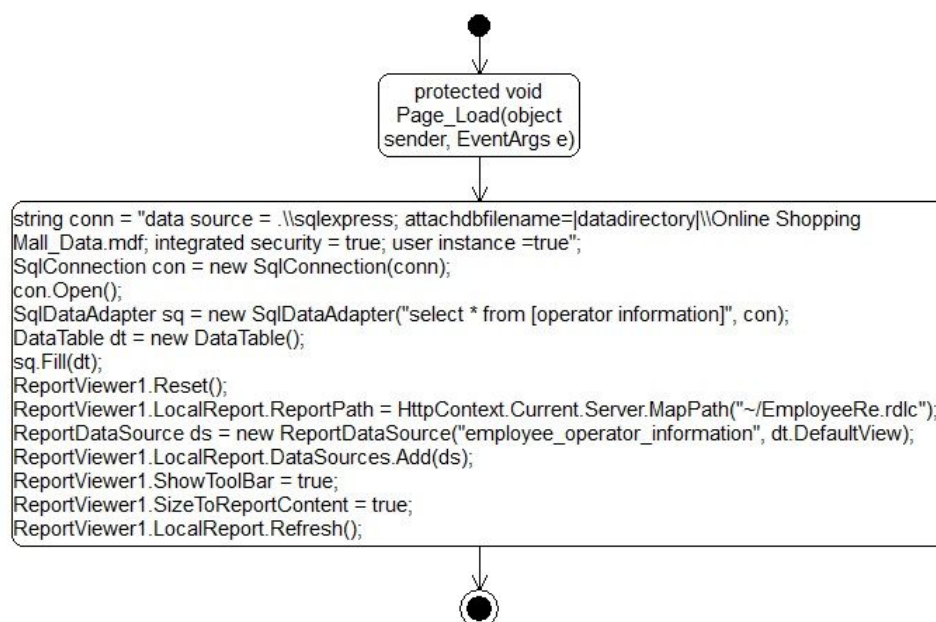


Fig.A4.6.b: UML activity diagram of the structure of [using System, protected void Page_Load (object sender, EventArgs e)] in “EmployeeRe.aspx.cs”

As shown in Fig.A4.6.b, when the ‘EmployeeRe’ page was loaded, the system connected to the database and selects the data from ‘operator information’ table, then justifies report and displays it.

The next step is to visualize the codes of the “feedback.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.7.a, Fig.A4.7.b, and Fig.A4.7.c:

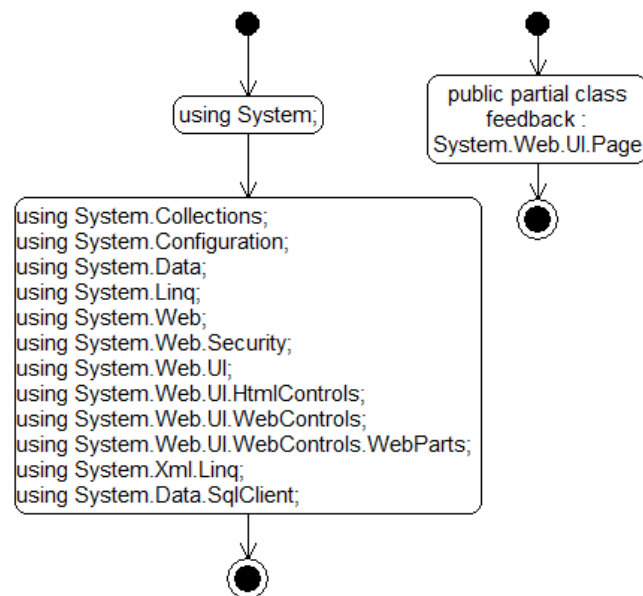


Fig.A4.7.a: UML activity diagram of the structures of [using System, public partial class feedback : System.Web.UI.Page] in “feedback.aspx.cs ”

As shown in the Fig.A4.7.a, at first, all the used name spaces were determined. Then define class named as ‘feedback’.

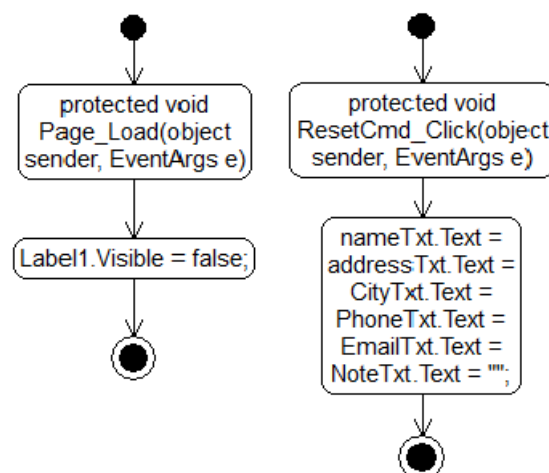


Fig.A4.7.b: UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void ResetCmd_Click(object sender, EventArgs e)] in “feedback.aspx.cs ”

As shown in Fig.A4.7.b, when the feedback page loaded, that makes 'label1' invisible. When the user wants to reset the feedback page, he has to click on Reset button to clear all the textboxes and label.

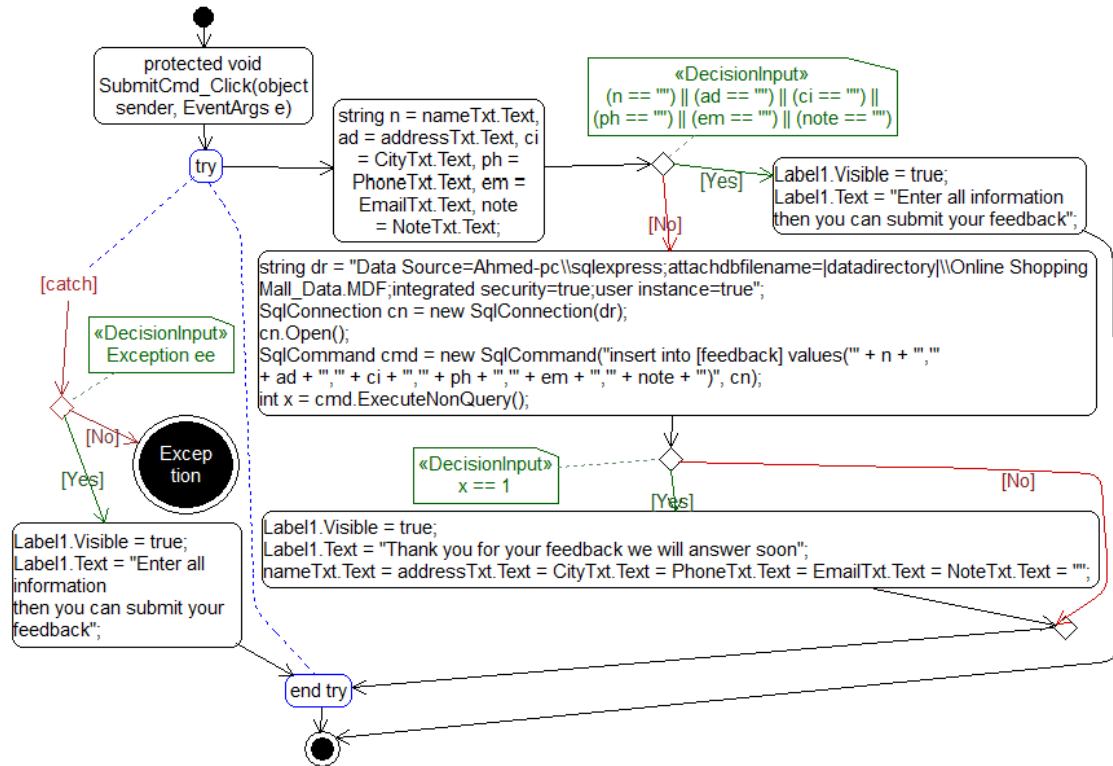


Fig.A4.7.c: UML activity diagram of the structures of [protected void SubmitCmd_Click(object sender, EventArgs e)] in "feedback.aspx.cs "

The next step is to visualize the codes of the "feedbackRe.aspx.cs" to convert this code to UML activity diagrams as shown in Fig.A4.8.a, and Fig.A4.8.b:

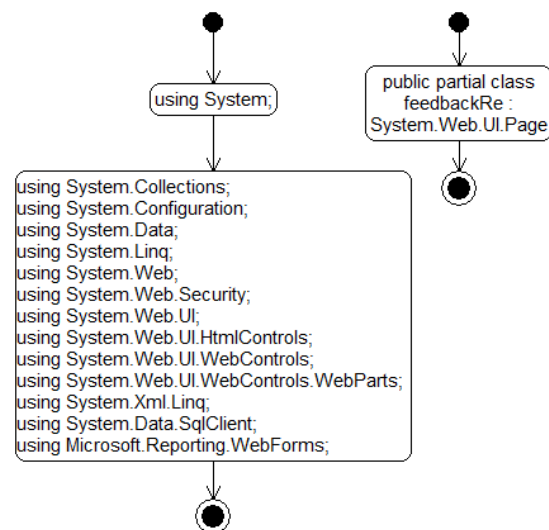


Fig.A4.8.a: UML activity diagram of the structures of [using System, public partial class feedbackRe : System.Web.UI.Page] in "feedbackRe.aspx.cs "

As shown in the Fig.A4.8.a, at first, all the used name spaces were determined. Then define class named as ‘feedbackRe’.

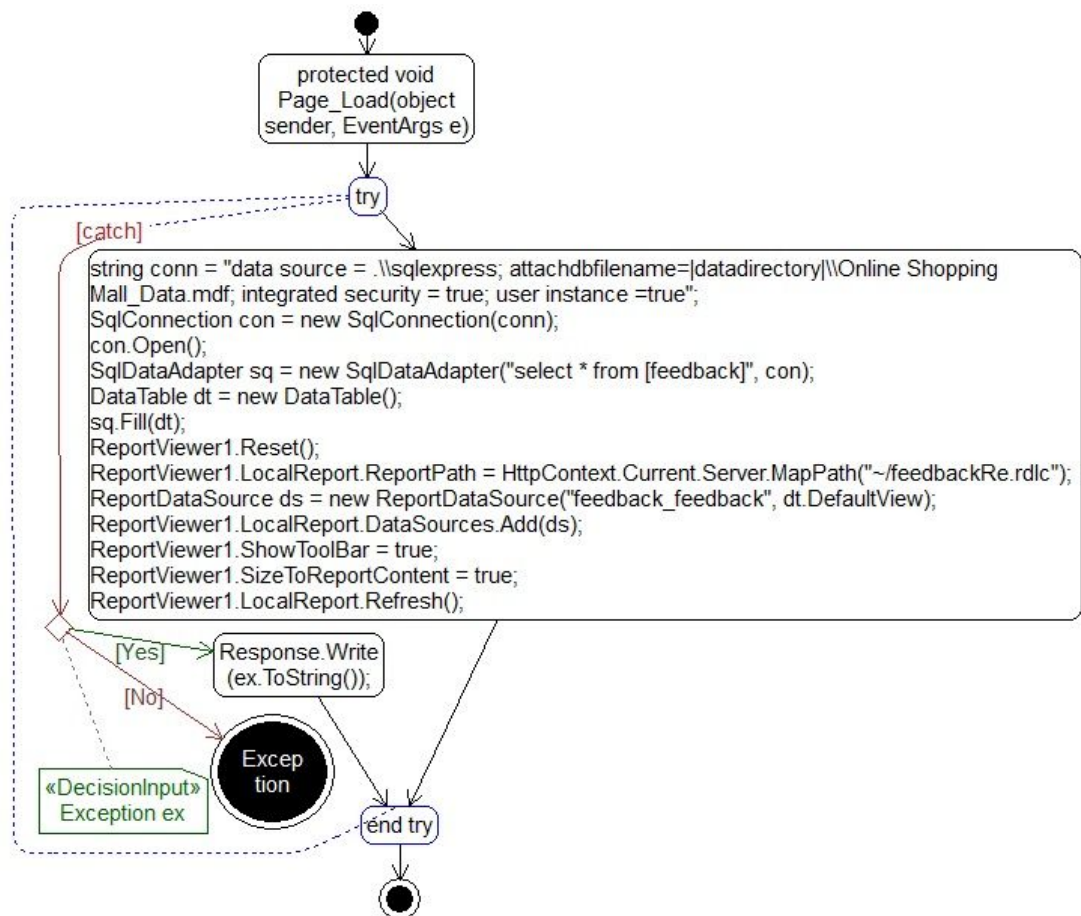


Fig.A4.8.b: UML activity diagram of the structures of [using System, protected void Page_Load(object sender, EventArgs e)] in “feedbackRe.aspx.cs ”

The next step is to visualize the codes of the “Login.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.9.a, Fig.A4.9.b, and Fig.A4.9.c:

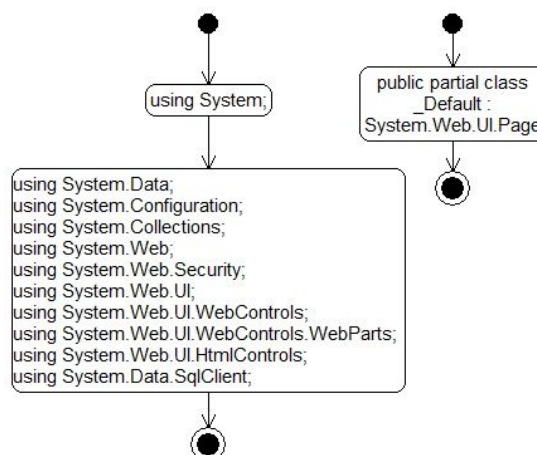


Fig.A4.9.a: UML activity diagram of the structures of [using System, public partial class _Default : System.Web.UI.Page] in “Login.aspx.cs ”

As shown in the Fig.A4.9.a, at first, all the used name spaces were determined. Then define class named as 'Default' for login page.

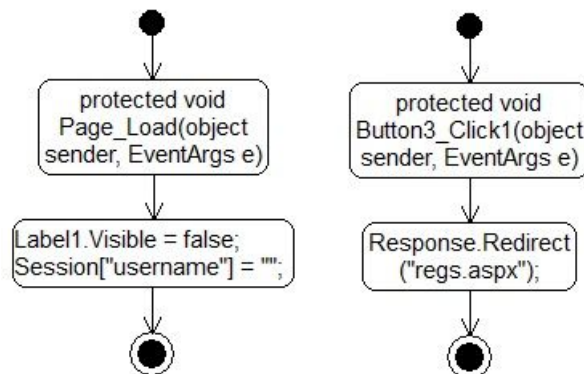


Fig.A4.9.b: UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void Button3_Click1(object sender, EventArgs e)] in "Login.aspx.cs "

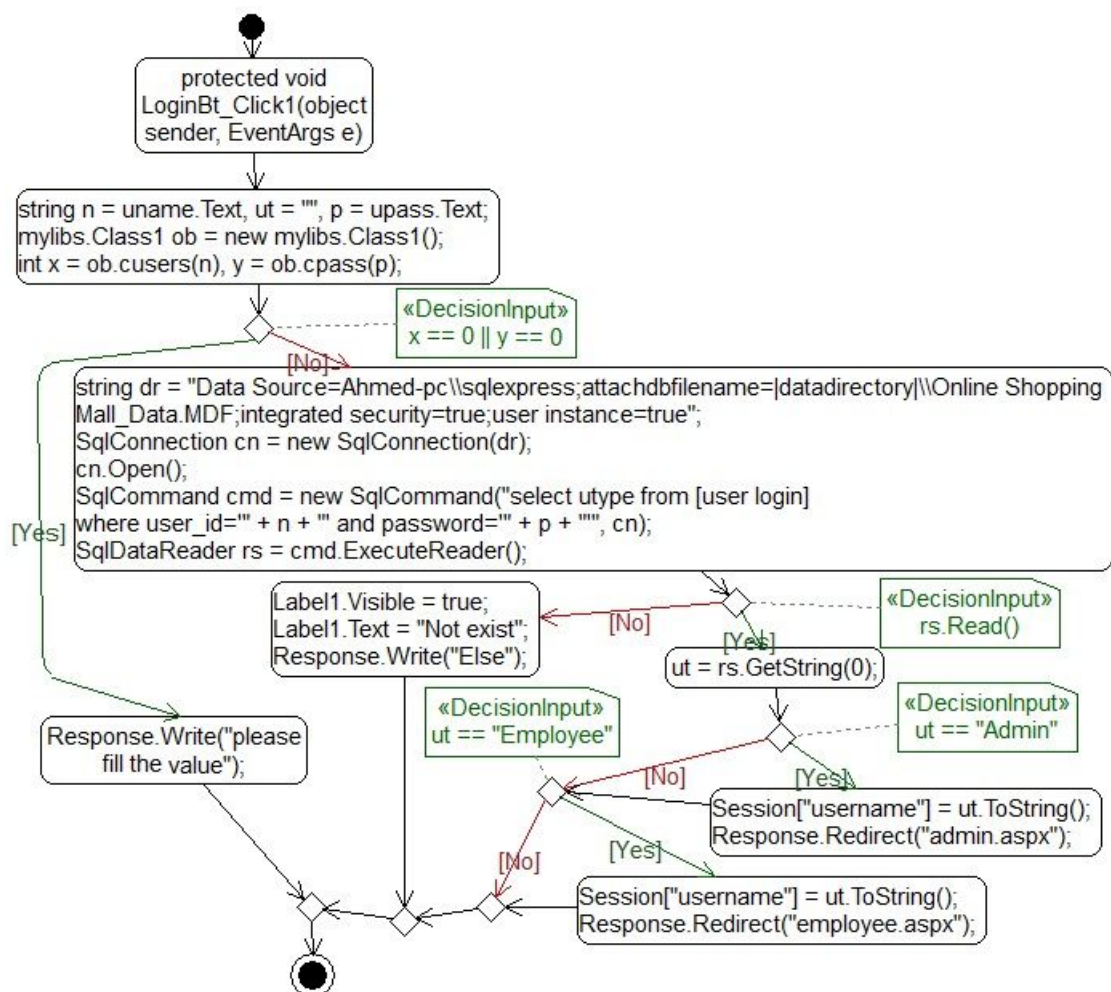


Fig.A4.9.c: UML activity diagram of the structures of [protected void LoginBt_Click1(object sender, EventArgs e)] in "Login.aspx.cs "

The next step is to visualize the codes of the “regs.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.10.a, Fig.A4.10.b, and Fig.A4.10.c:

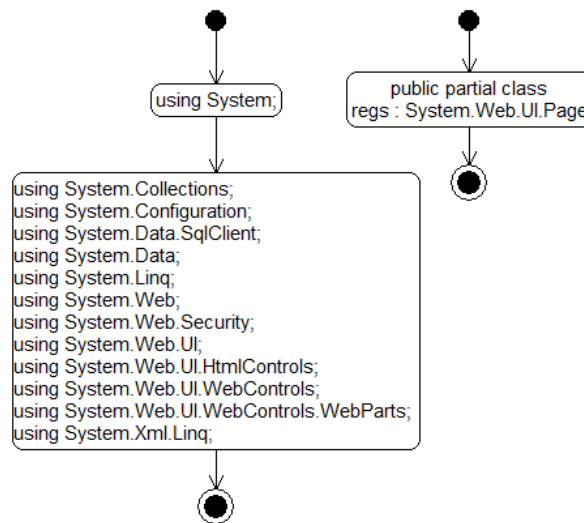


Fig.A4.10.a: UML activity diagram of the structures of [using System, public partial class regs : System.Web.UI.Page] in “regs.aspx.cs”

As shown in the Fig.A4.10.a, at first, all the used name spaces were determined. Then define class named as ‘regs’.

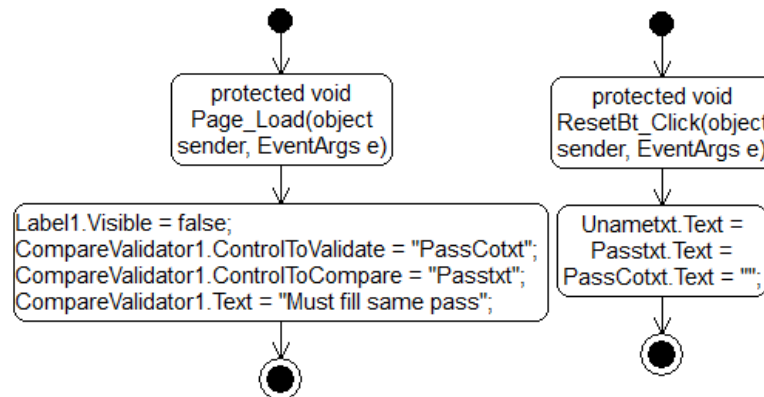


Fig.A4.10.b: UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void ResetBt_Click(object sender, EventArgs e)] in “regs.aspx.cs”

As shown in Fig.A4.10.b, when the ‘regs’ page was loaded, the ‘label1’ will be invisible, then set validators. When the user wants to reset the ‘regs’ page, he has to click on Reset button to clear all the textboxes and label.

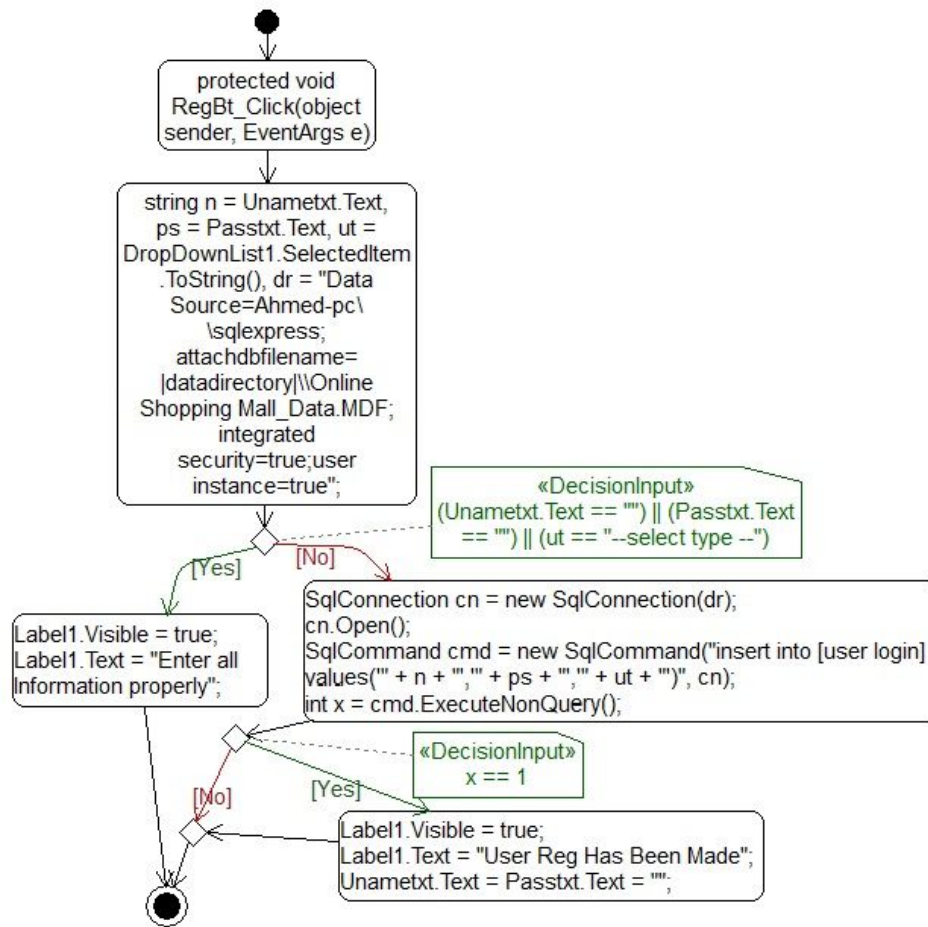


Fig.A4.10.c: UML activity diagram of the structures of [protected void
RegBt_Click(object sender, EventArgs e)] in “regs.aspx.cs”

As shown in Fig.A4.10.c, when the admin wants to register a new user for the supermarket system, he can do that by just clicks on register button, then the system will check the entered data then save it in the ‘user login’ table and message appear to the admin to inform him that the user registration has been made.

The next step is to visualize the codes of the “edituser.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.11.a, Fig.A4.11.b, and Fig.A4.11.c:

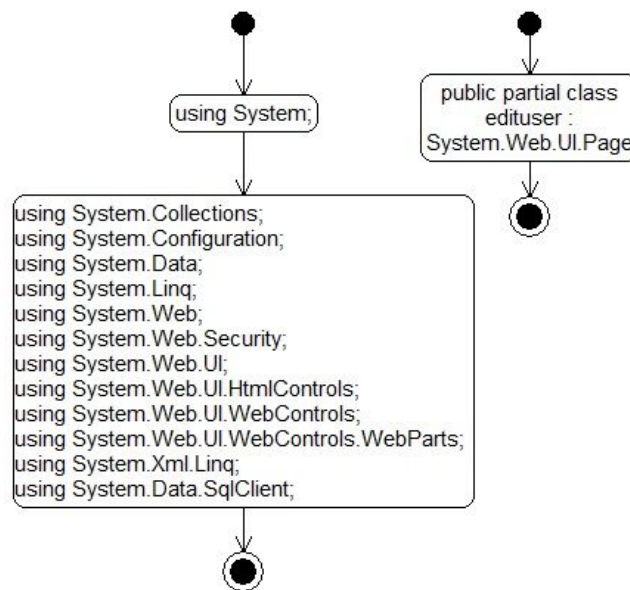


Fig.A4.11.a: UML activity diagram of the structures of [using System, public partial class edituser : System.Web.UI.Page] in "edituser.aspx.cs"

As shown in the Fig.A4.11.a, at first, all the used name spaces were determined. Then define class named as 'edituser'.

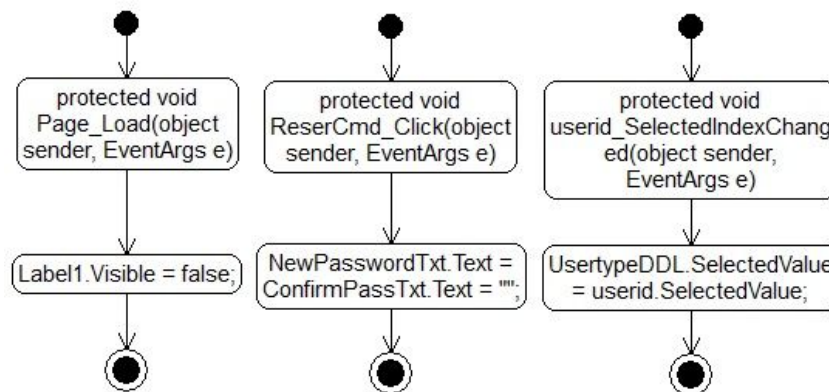


Fig.A4.11.b: UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e), protected void ReserCmd_Click (object sender, EventArgs e), protected void userid_SelectedIndexChang ed (object sender, EventArgs e)] in "edituser.aspx.cs "

As shown in Fig.A4.11.b, when Page_Load function was executed, the 'edituser' page will be loaded and set 'label1' invisible. When the user clicks on reset button all the text boxes will be cleared. When the user id selected then the system will get his type either administrator or employee according to the registered information.

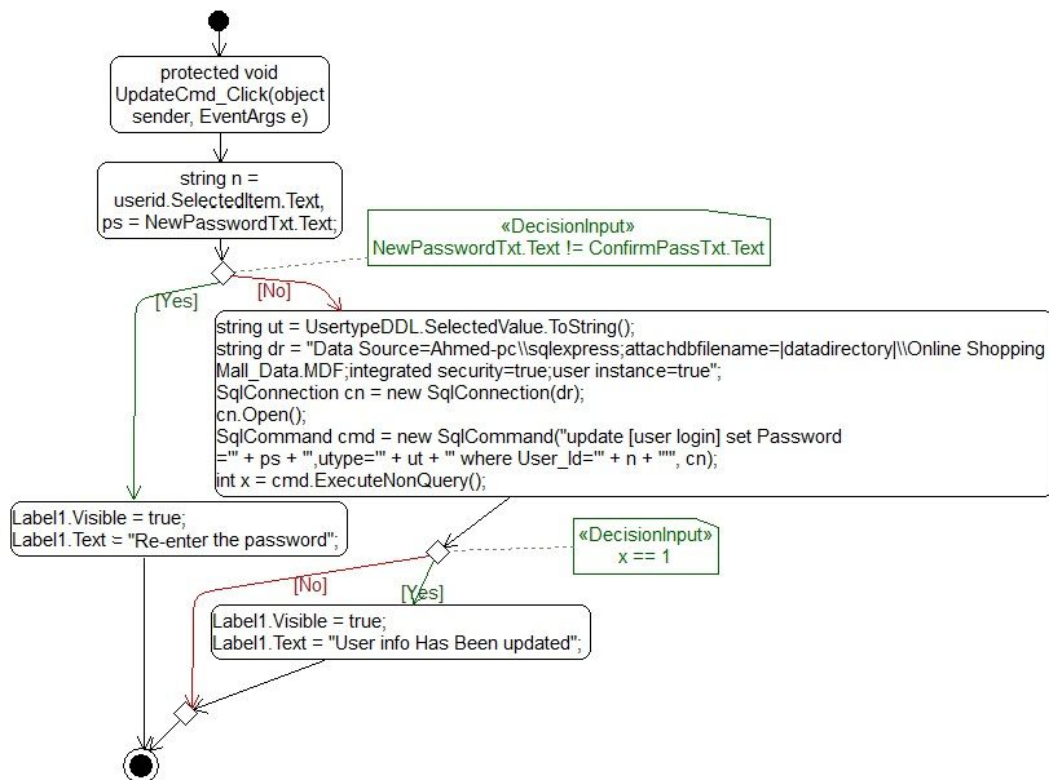


Fig.A4.11.c: UML activity diagram of the structures of [protected void UpdateCmd_Click (object sender, EventArgs e)] in "edituser.aspx.cs"

As shown in Fig.A4.11.c: when the administrator clicks on update button, first of all he must enter the user name, select authentication level 'employee or administrator', and enter new password, then system will update the 'user login' table, and check if the update is done correctly then send a message to the user to inform him that the user information has been updated, otherwise. Within this process if there is any error, the system will catch it and send a message to the user to inform him that he must re-enter the password.

The next step is to visualize the codes of the "deleteuser.aspx.cs" to converts this code to UML activity diagrams as shown in Fig.A4.12.a, and Fig.A4.12.b:

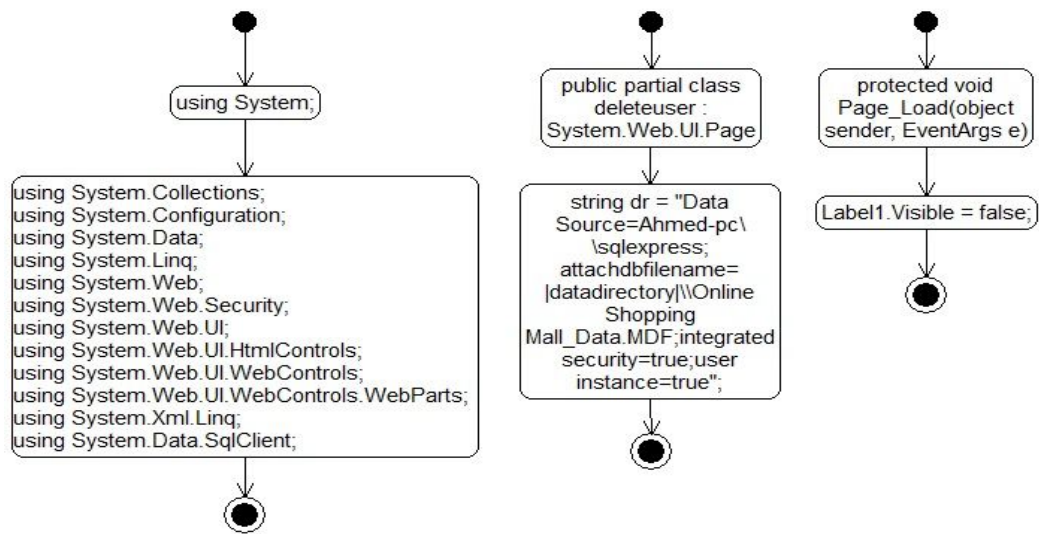


Fig.A4.12.a: UML activity diagram of the structures of [using System, public partial class deleteuser:System.Web.UI.Page, protected void Page_Load (object sender, EventArgs e)] in " deleteuser.aspx.cs "

As shown in the Fig.A4.12.a, at first, all the used name spaces were determined. Then define class named as 'deleteuser', and specify the connection to the database. Then 'Page_Load' function is executed to load the 'delete user' page and set 'label1' to invisible state.

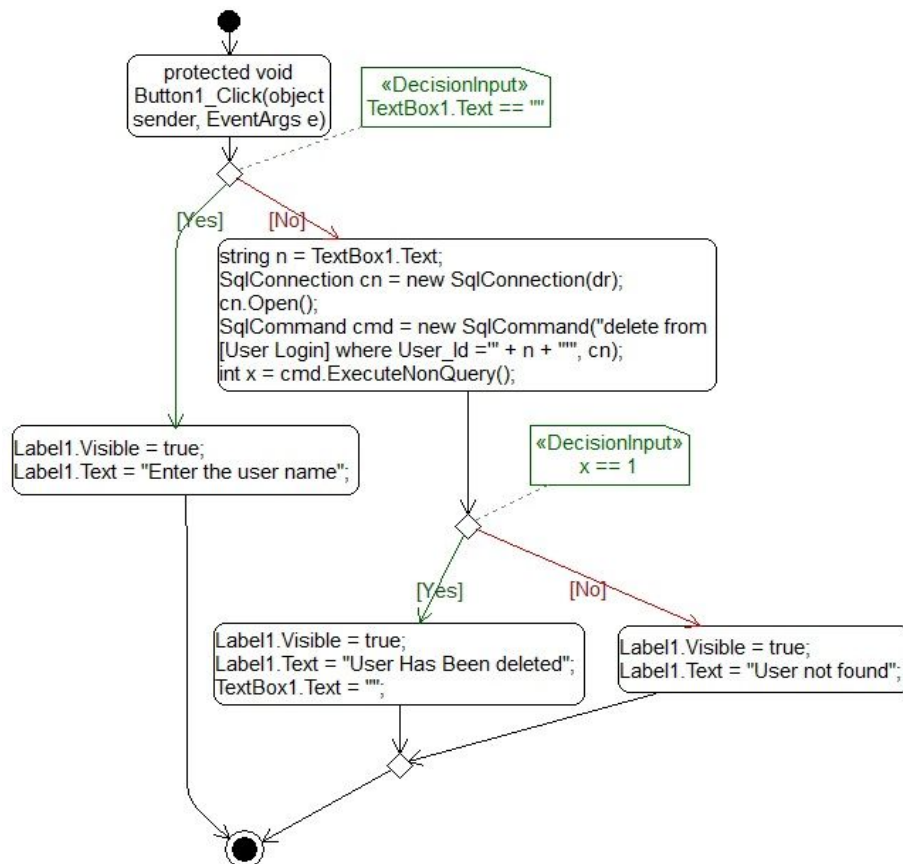


Fig.A4.12.b: UML activity diagram of the structures of [protected void Button1_Click(object sender, EventArgs e)] in " deleteuser.aspx.cs "

As shown in Fig.A4.12.b: when the administrator want to delete user, first of all he must enter the user name, and clicks on delete button, then it will be deleted from ‘user login’ table, then check if the delete is done correctly, if yes then the system display message to inform administrator that the user has been deleted, otherwise inform him that the user was not found. Within this process if the administrator did not enter the user name, then the system will display a message to inform him that he must enter the user name.

The next step is to visualize the codes of the “sale.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.13.a, Fig.A4.13.b, Fig.A4.13.c, Fig.A4.13.d, Fig.A4.13.e, Fig.A4.13.f, Fig.A4.13.g, Fig.A4.13.h, and Fig.A4.13.i:

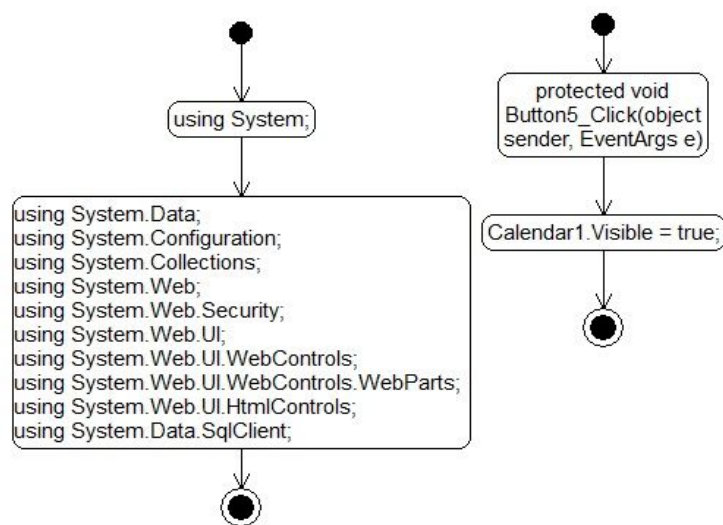


Fig.A4.13.a: UML activity diagram of the structures of [using System, protected void Button5_Click(object sender, EventArgs e)] in “sale.aspx.cs ”

As shown in the Fig.A4.13.a, at first, all the used name spaces were determined. When the user clicks on ‘Button5’ the colander will be visible to select the date.

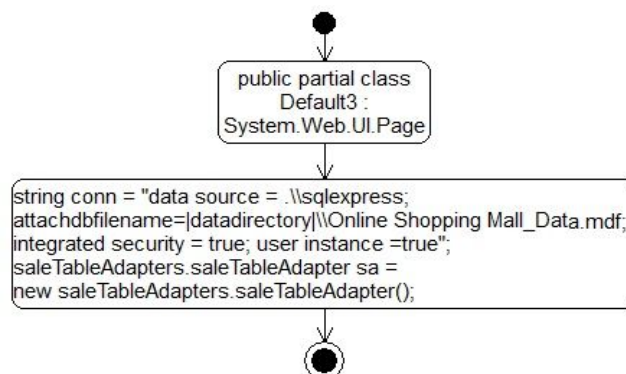


Fig.A4.13.b: UML activity diagram of the structures of [public partial class Default3: System.Web.UI.Page] in “sale.aspx.cs”

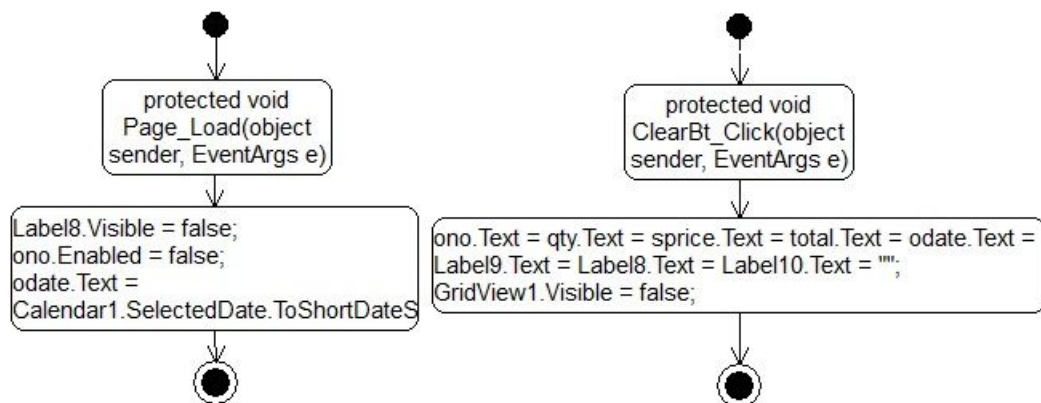


Fig.A4.13.c: UML activity diagram of the structures of [protected void Page_Load(object sender, EventArgs e), protected void ClearBt_Click(object sender, EventArgs e)] in “sale.aspx.cs ”

As shown in Fig.A4.13.c, when the sale page loaded, that makes ‘label8’ invisible, and ono ‘orde number’ disable. When the user wants to clear the sale page, he has to click on Clear button to clear all the textboxes and label.

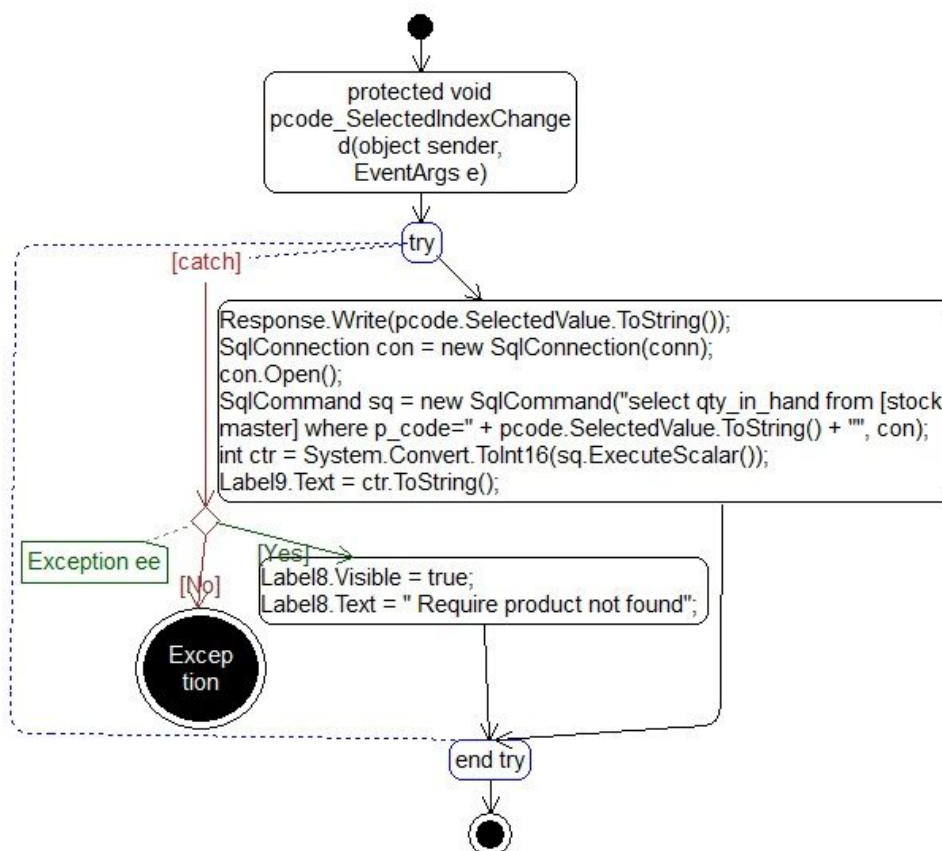


Fig.A4.13.d: UML activity diagram of the structures of [protected void pcode_SelectedIndexChanged(object sender, EventArgs e)] in “sale.aspx.cs ”

As shown in Fig.A4.13.d: when the user select the product according to the product code the system will check the quantity of selected item if available or not. If it is not available then message will appear to user to inform him about that.

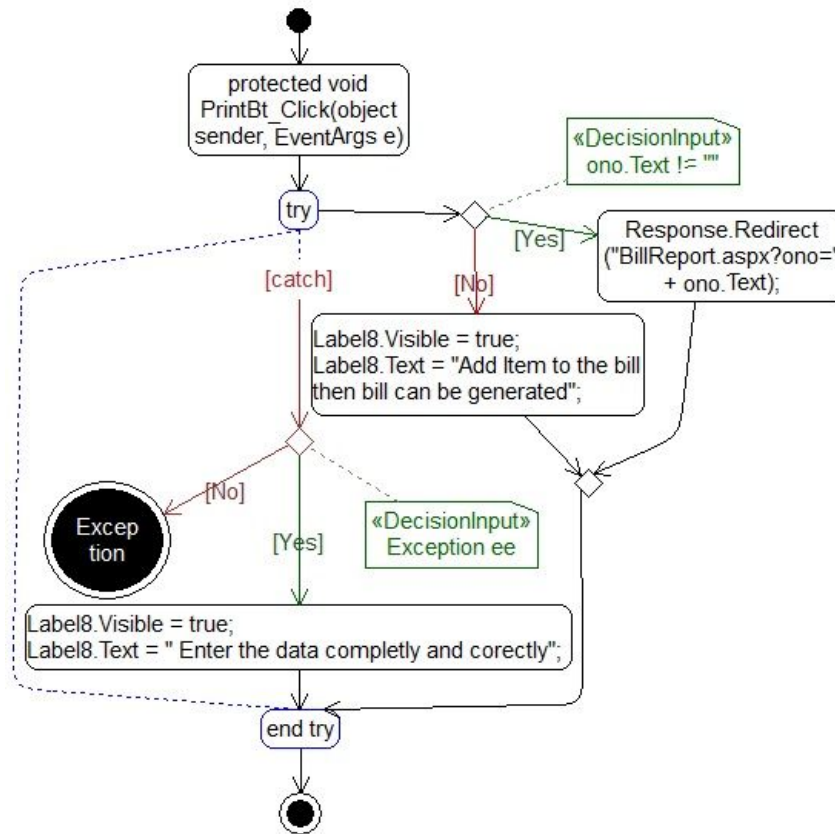


Fig.A4.13.e: UML activity diagram of the structures of [protected void PrintBt_Click(object sender, EventArgs e)] in "sale.aspx.cs "

As shown in Fig.A4.13.e: when the sale process completed and all required items were added to the bill, then the bill can be printed by just click on print buttom. Through this process if there is any error, then the system will catch that error and display message to the user to inform him that he must enter the data completely and correctly.

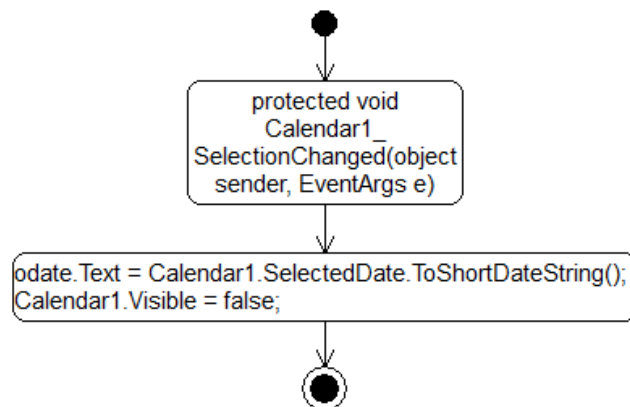


Fig.A4.13.f: UML activity diagram of the structures of [protected void Calendar1_SelectionChanged(object sender, EventArgs e)] in “sale.aspx.cs ”

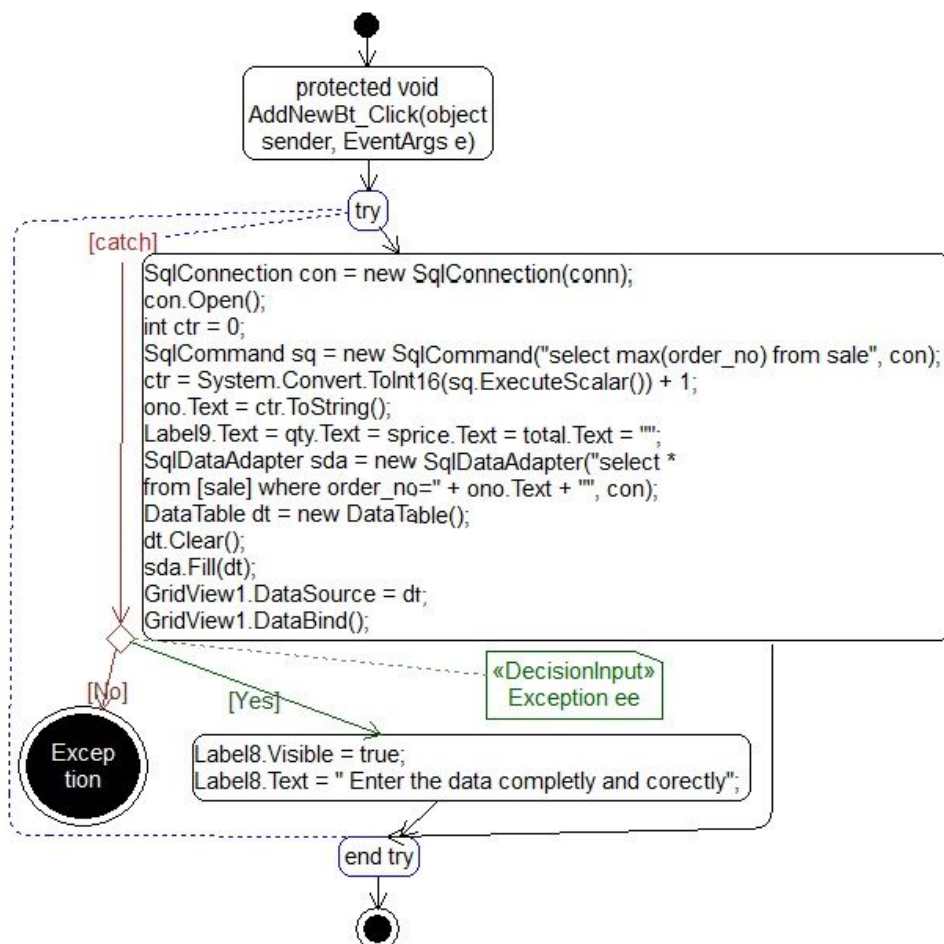
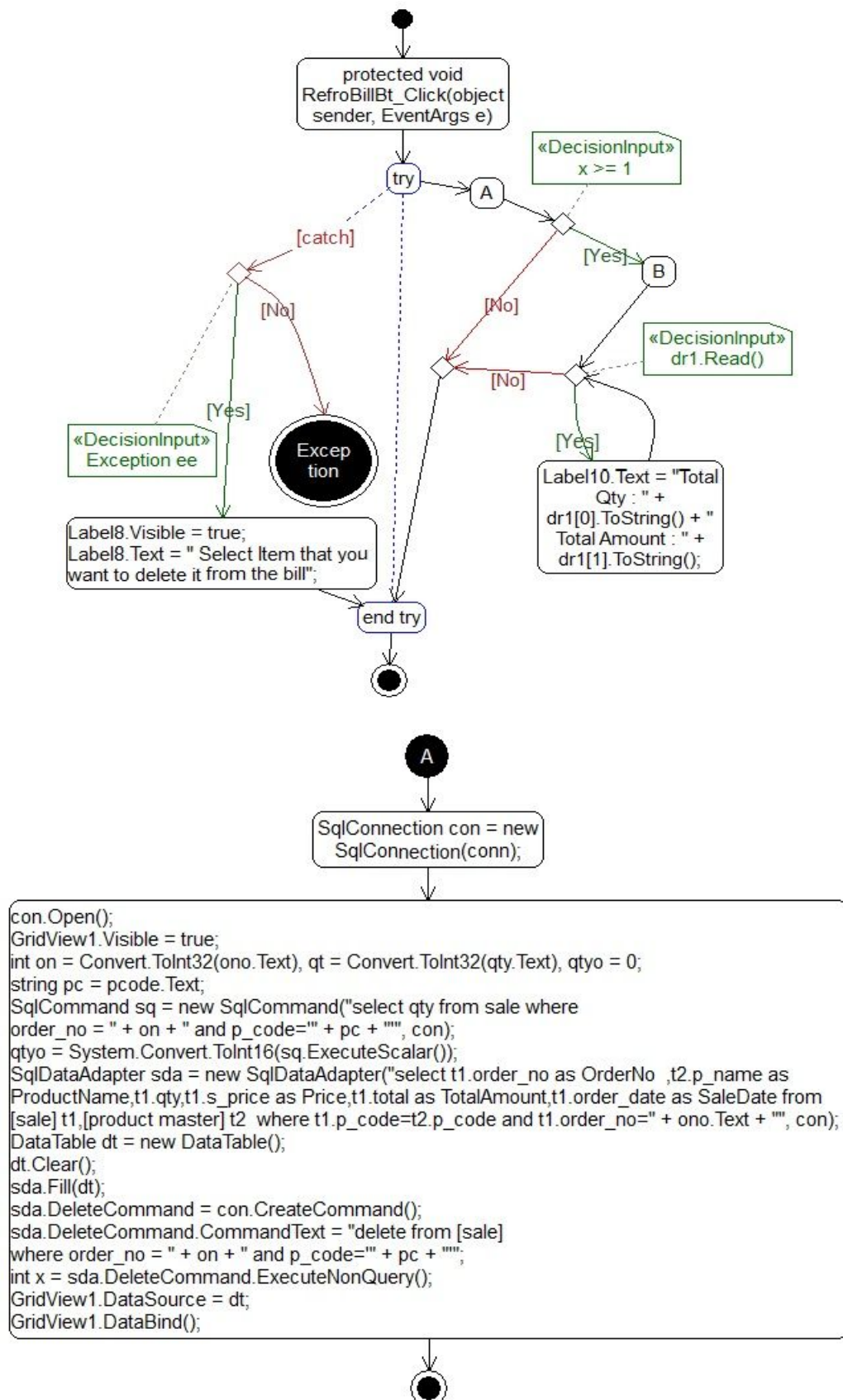


Fig.A4.13.g: UML activity diagram of the structures of [protected void AddNewBt_Click(object sender, EventArgs e)] in “sale.aspx.cs ”

As shown in Fig.A4.13.g: when the user wants to add a new order, he just clicks on ‘Add New’ buttom, the order number will be generated automaticly. Through this process if

there is any error, then the system will catch that error and display a message to the user to inform him that he must enter the data completely and correctly.



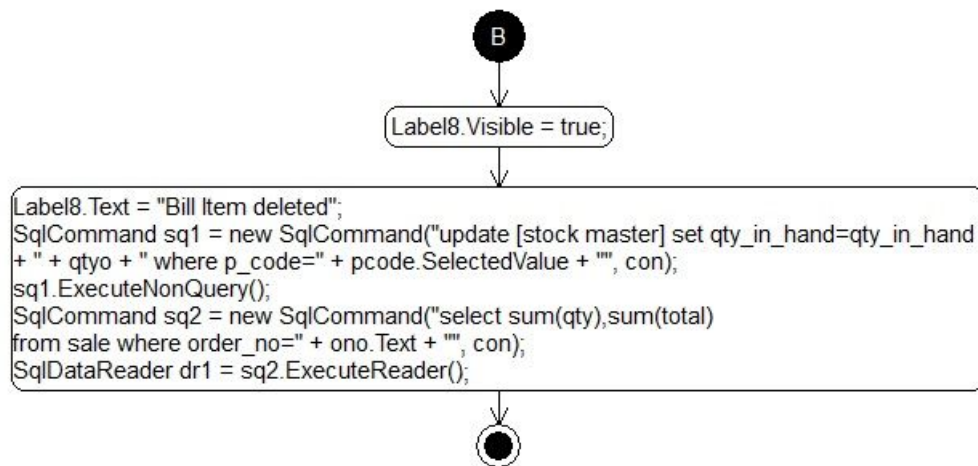
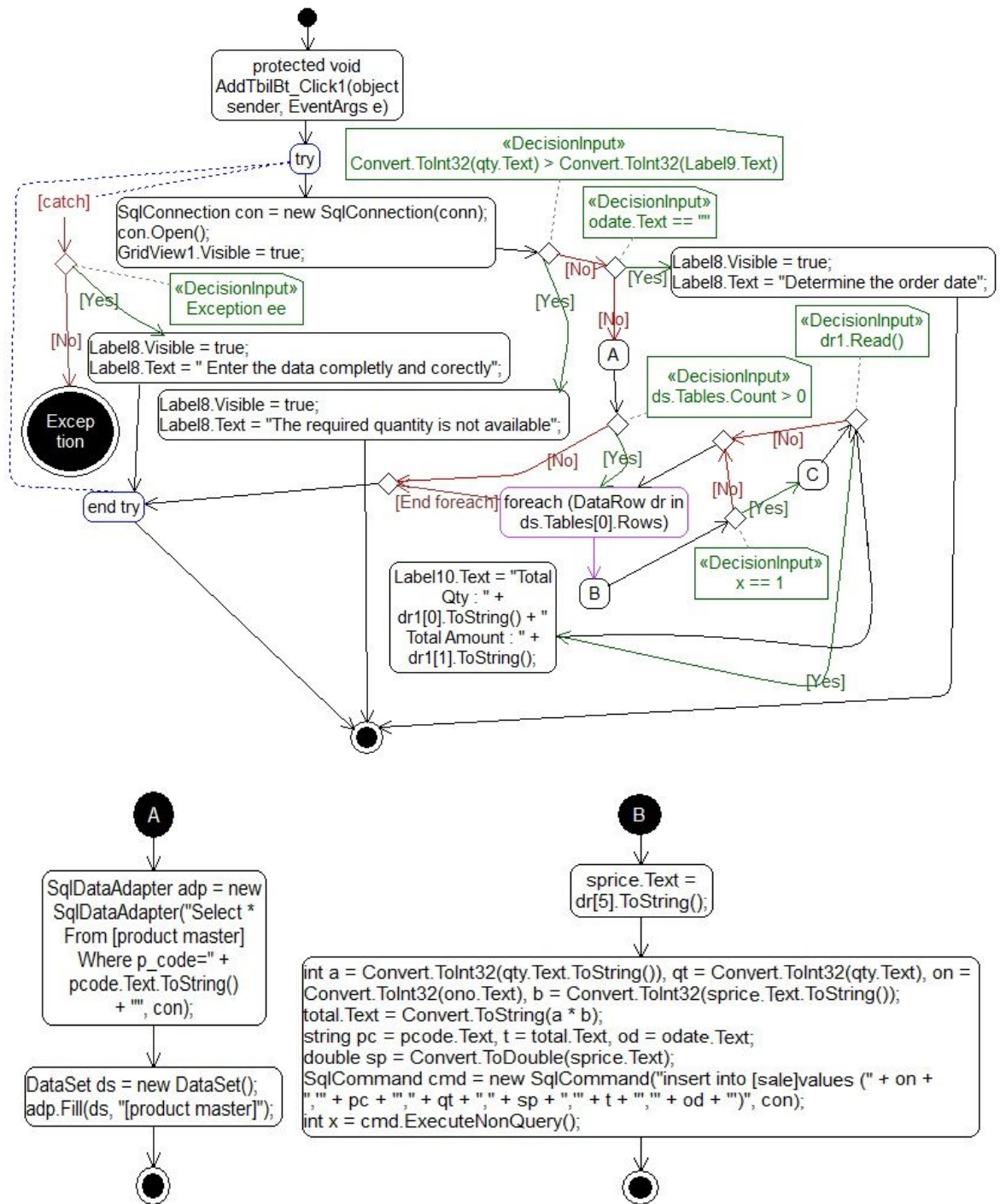


Fig.A4.13.h: UML activity diagram of the structures of [protected void RefroBillBt_Click (object sender, EventArgs e)] in "sale.aspx.cs"

As shown in Fig.A4.13.h, within the sale process and some required items were added to the bill, but the customer change his mind and he want to remove some items, then these item can be removed from the bill by just click on 'Remove from bill' buttom and update stock master. Through this process if there is any error, then the system will catch that error and display message to the user to inform him that he must select the item that he want to remove befor clicks on 'Remove from bill' button.



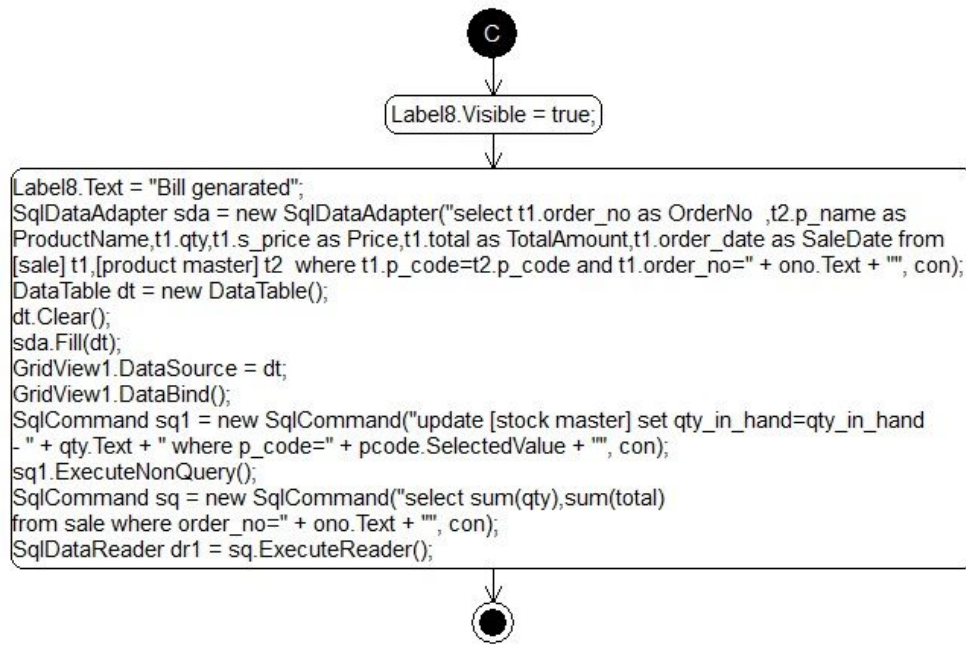


Fig.A4.13.i: UML activity diagram of the structures of [protected void AddTbilBt_Click1 (object sender, EventArgs e)] in “sale.aspx.cs”

As shown in Fig.A4.13.i: within the sale process and some required items were added to the bill, but there is another item also required by the customer, then these item can be added it to the bill by just click on ‘add to bill’ buttom, then the system will check stock master table to see if the required quantity is available or not. If it available then will be added, else message appear to the user to inform him that the required quantity is not available, Through this process if there is any error, then the system will catch that error and display a message to the user to inform him that he must enter the data completely and correctly.

The next step is to visualize the codes of the “SaleRe.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.14.a, and Fig.A4.14.b:

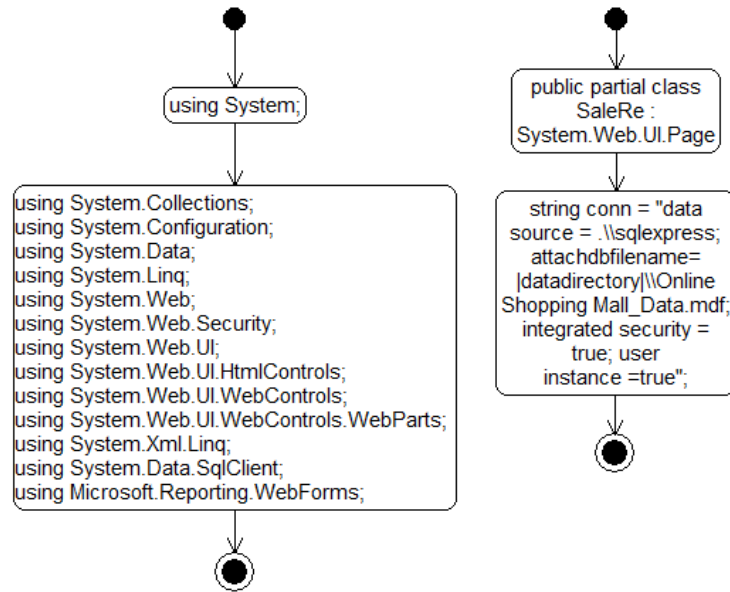


Fig.A4.14.a: UML activity diagram of the structures of [using System, public partial class SaleRe : System.Web.UI.Page] in “SaleRe.aspx.cs”

As shown in the Fig.A4.14.a, at first, all the used name spaces were determined. Then define class named as ‘SaleRe’. Then define the connection to the database.

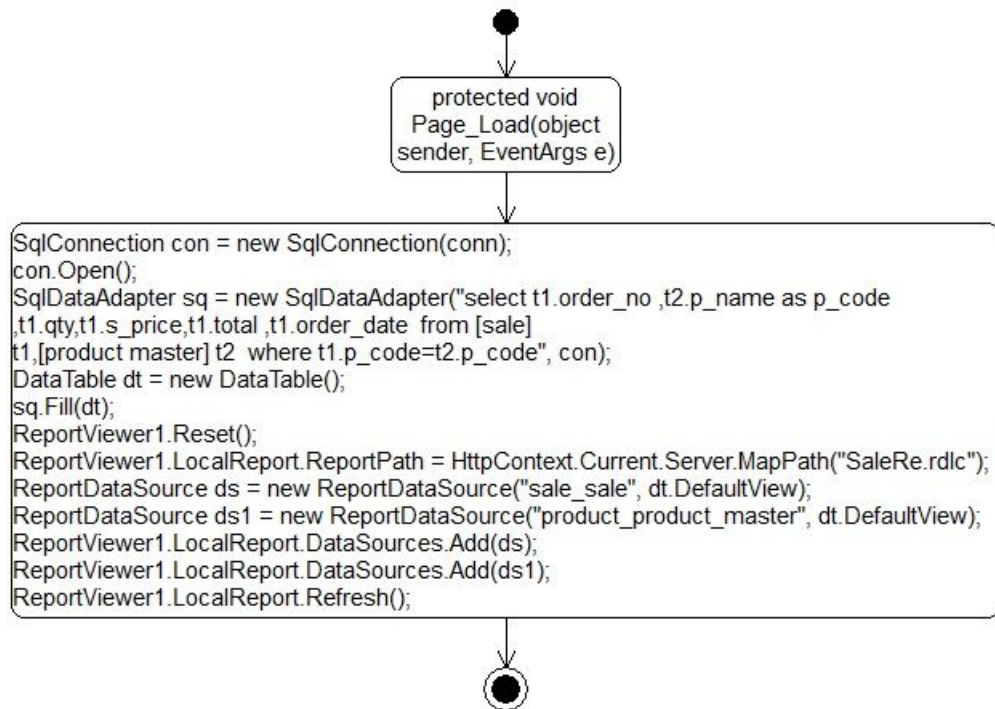


Fig.A4.14.b: UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e)] in “SaleRe.aspx.cs”

As shown in Fig.A4.14.b, when the ‘SaleRe’ page was loaded, the system connected to the database and selects the data from ‘sale’ table, then justifies report and displays it.

The next step is to visualize the codes of the “BillReport.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.15.a and Fig.A4.15.b:

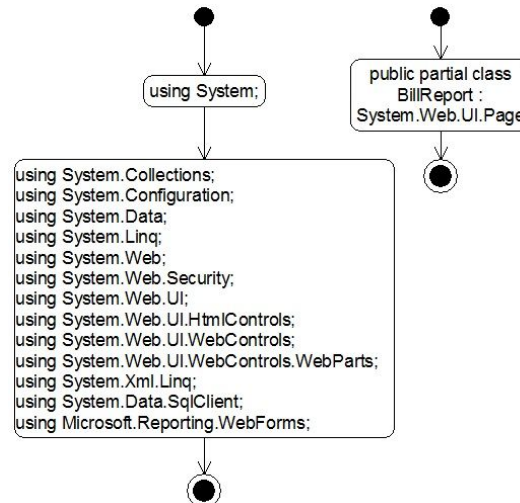


Fig.A4.15.a: UML activity diagram of the structures of [using System, public partial class BillReport: System.Web.UI.Page] in “BillReport.aspx.cs”

As shown in the Fig.A4.15.a, at first, all the used name spaces were determined. Then define class named as ‘BillReport’.

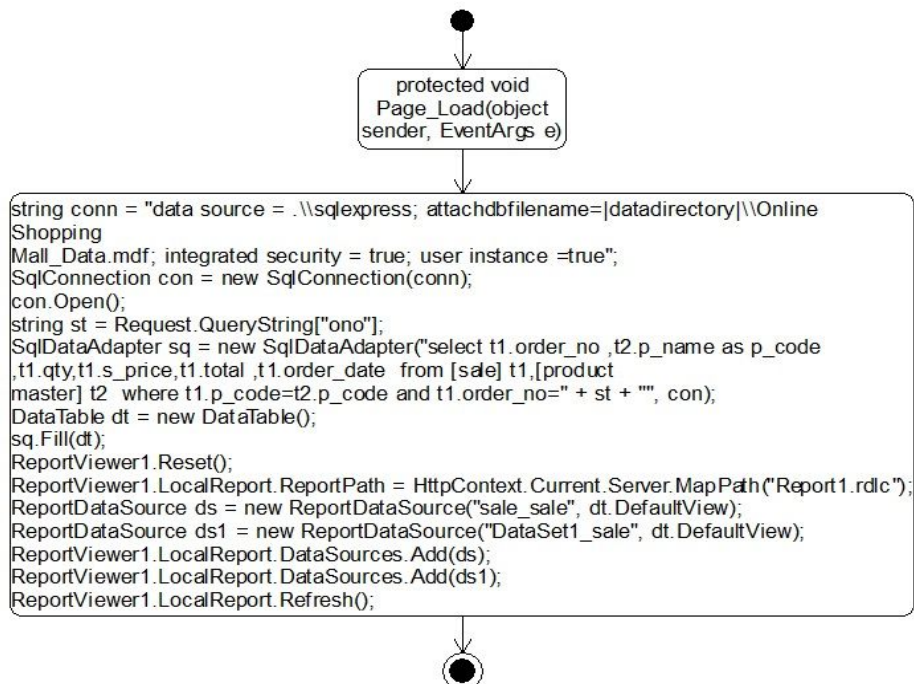


Fig.A4.15.b: UML activity diagram of the structures of [protected void Page_load(Object sender, EventArgs e)] in “BillReport.aspx.cs”

As shown in Fig.A4.15.b, when the ‘BillReport’ page is loaded, the system will be connected to the database and selected the data from ‘sale’ table, then justifies report and display it.

The next step is to visualize the codes of the “stock master.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.16.a, Fig.A4.16.b, Fig.A4.16.c, and Fig.A4.16.d:

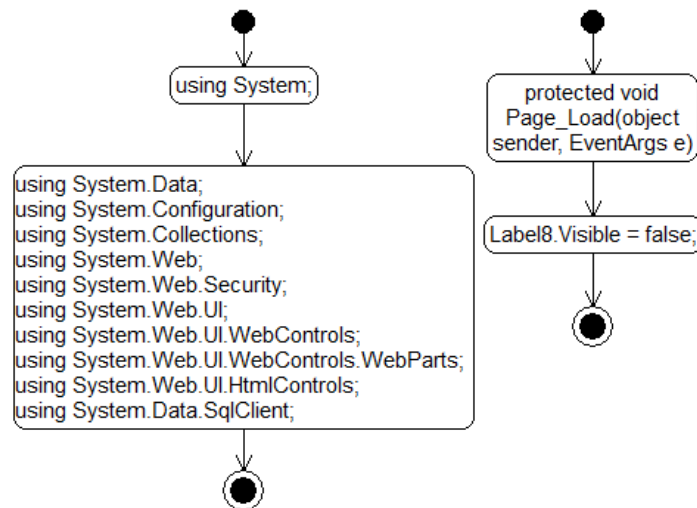


Fig.A4.16.a: UML activity diagram of the structures of [using System, protected void Page_Load (object sender, EventArgs e)] in “stock master.aspx.cs”

As shown in the Fig.A4.16.a, at first, all the used name spaces were determined. Then ‘Page_Load’ function was executed to load the stock master page and set ‘label8’ invisible.

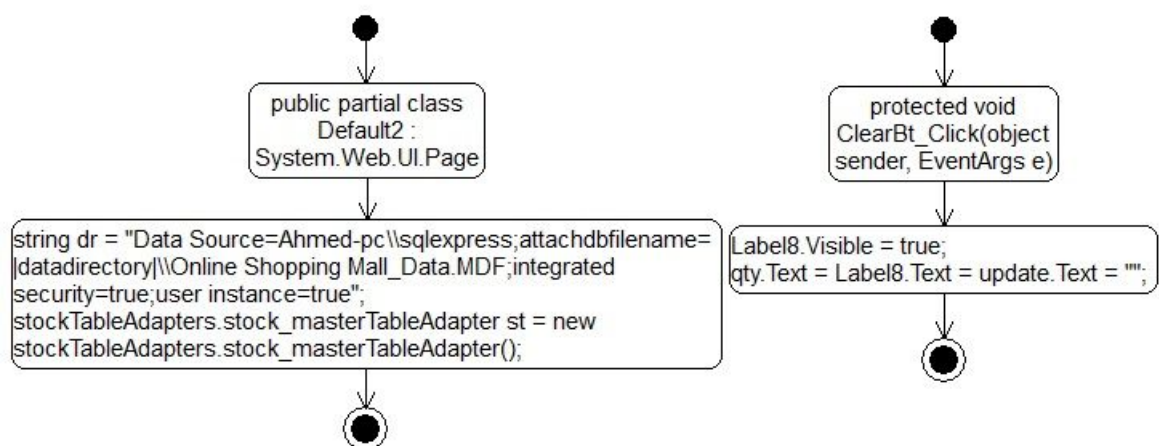


Fig.A4.16.b: UML activity diagram of the structures of [public partial class Default2: System.Web.UI.Page, protected void ClearBt_Click (object sender, EventArgs e)] in “stock master.aspx.cs”

As shown in Fig.A4.16.b, class named as ‘Default2’ was defined. If the user wants to clear the page, he has to click on the clear button, so all the textboxes will be cleared.

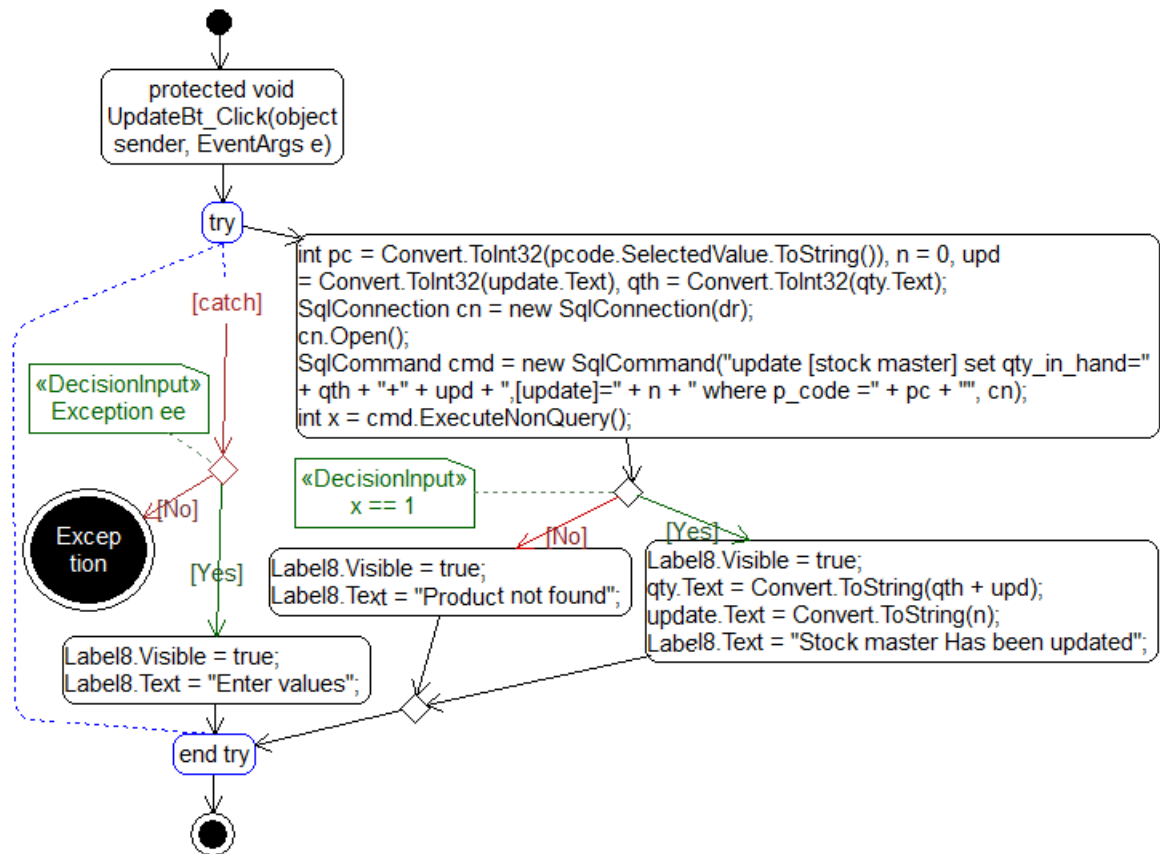


Fig.A4.16.c: UML activity diagram of the structures of [protected void UpdateBt_Click(object sender, EventArgs e)] in “stock master.aspx.cs”

As shown in Fig.A4.16.c, when the user wants to update the quantity of the specified item, he can do that by just clicking on the update button, then the system will fetch the quantity of the determined item then add this quantity to the update quantity and save the result in the stock master table. If the required product is not found, then a message will be displayed to inform the user that this product is not found.

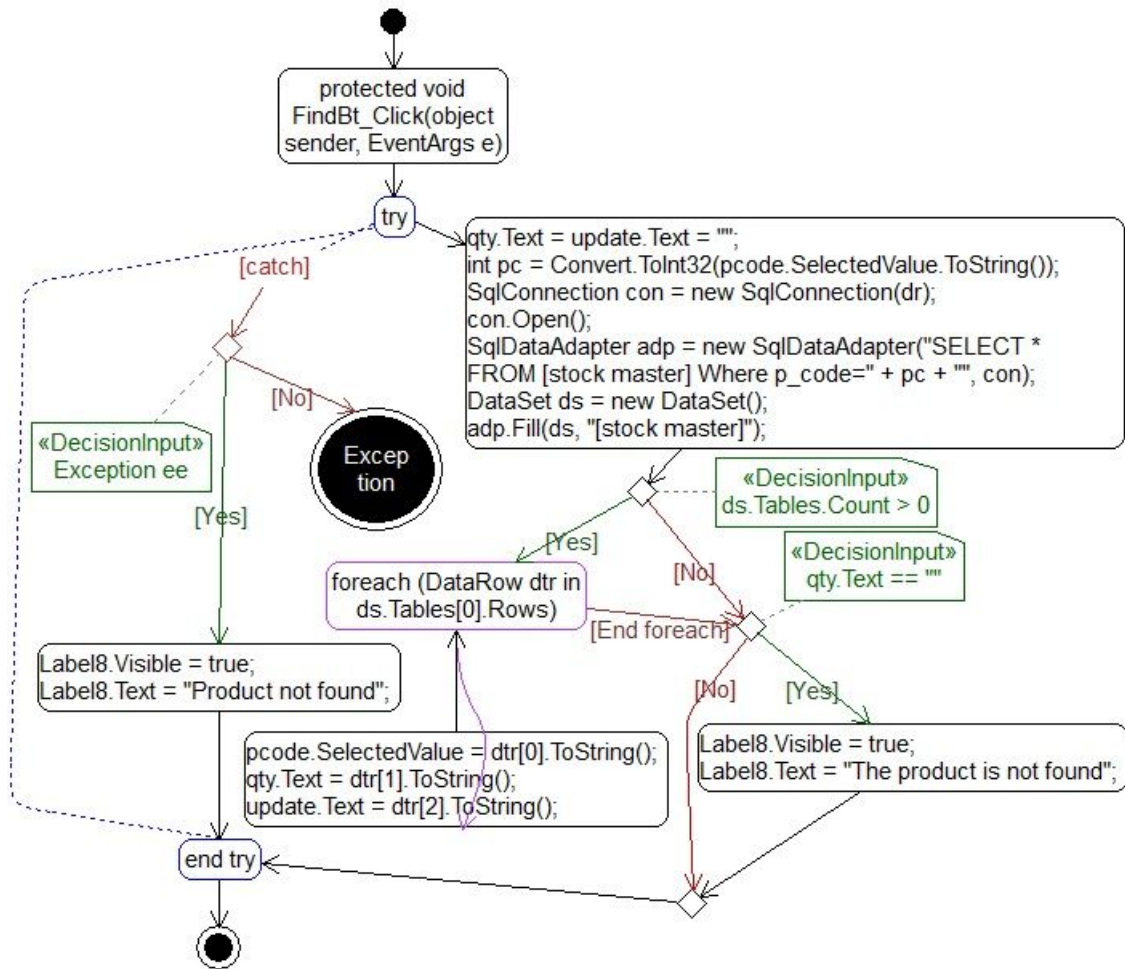


Fig.A4.16.d: UML activity diagram of the structures of [protected void FindBt_Click (object sender, EventArgs e)] in “stock master.aspx.cs”

As shown in Fig.A4.16.d: when the user want to fined the quantity of the specified item, he can do that by just clicks on find button, then the system will fetch the quantity of the determined item and display it. If the required product not founded then the message will be appeared to inform the user that this product not found.

The next step is to visualize the codes of the “Stock view.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.17:

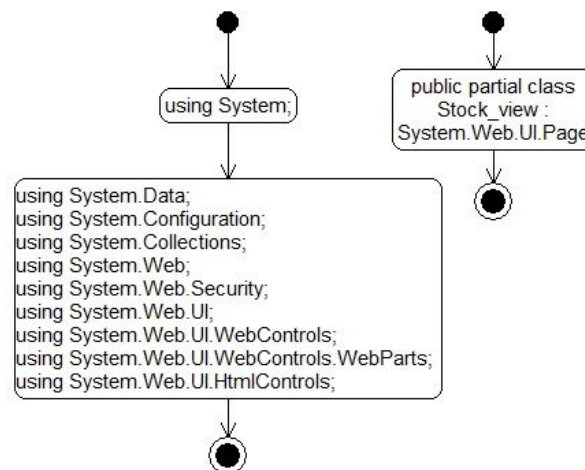


Fig.A4.17: UML activity diagram of the structures of [using System, public partial class Stock_view: System.Web.UI.Page] in “Stock view.aspx.cs”

As shown in the Fig.A4.17, at first, all the used name spaces were determined. Then define class named as ‘Stock_view’.

The next step is to visualize the codes of the “StockRe.aspx.cs” to convert this code to UML activity diagrams as shown in Fig.A4.18.a, and Fig.A4.18.b:

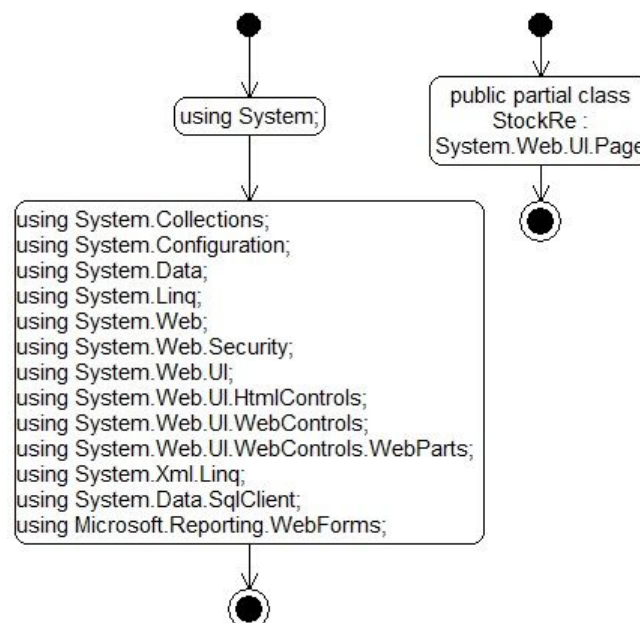


Fig.A4.18.a: UML activity diagram of the structures of [using System, public partial class StockRe : System.Web.UI.Page] in “StockRe.aspx.cs”

As shown in the Fig.A4.18.a, at first, all the used name spaces were determined. Then define class named as ‘StockRe’.

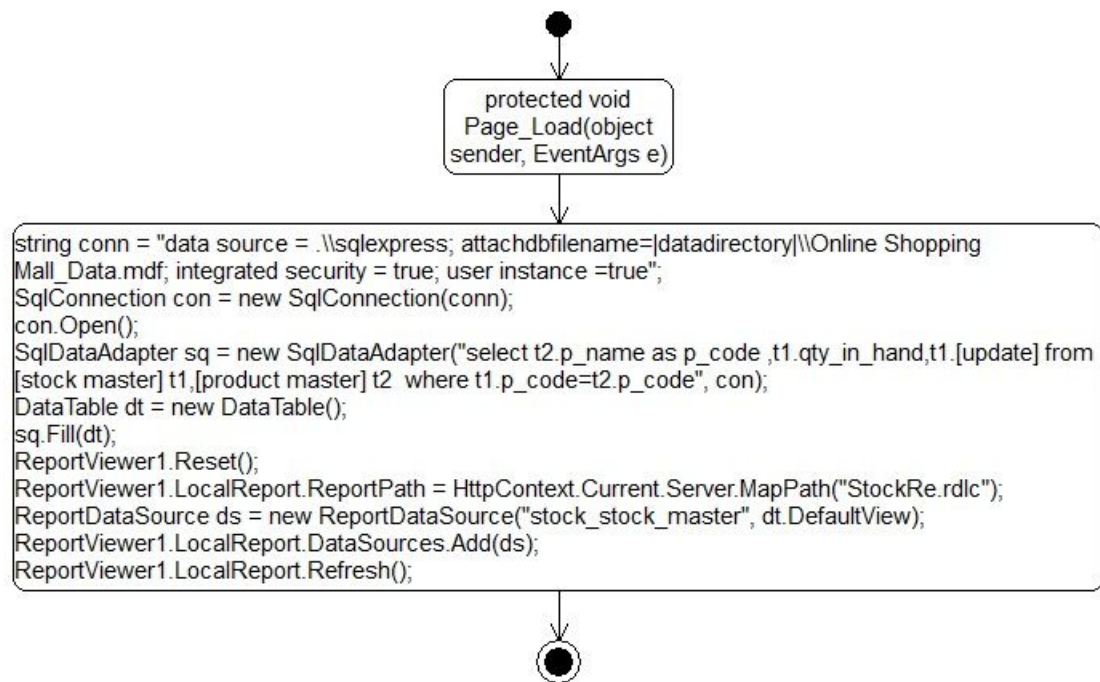


Fig.A4.18.b: UML activity diagram of the structures of [protected void Page_Load (object sender, EventArgs e)] in “StockRe.aspx.cs”

As shown in Fig.A4.18.b, when the ‘StockRe’ page was loaded, the system connected to the database and selects the data from ‘stock_master’ table, then justifies report and displays it.

Then the same process done to analyse the code of all others web forms as: “supplier information.aspx.cs”, “userlist.aspx.cs”, “UserRe.aspx.cs”, “customer information.aspx.cs”, “Department.aspx.cs”, “operator information.aspx.cs”, “product master.aspx.cs”, “product view.aspx.cs”, “ProductRe.aspx.cs”, “purchase view.aspx.cs”, “Reports.aspx.cs”, “sale view.aspx.cs”, “purchase.aspx.cs”, and “PurchaseRe.aspx.cs”.

The Need of Re-engineering in Software Engineering

Ahmed Saleem Abbas, W. Jeberson, V.V. Klinsega

Department of Computer Science & Information Technology, SHIATS University, India

ABSTRACT

In this paper we will discuss the importance of software re-engineering and the reasons behind this importance followed by a discussion on each of these reasons with examples to prove that the re-engineering process is a useful tool to convert old, obsolete systems to more efficient, streamlined systems. And the re-engineering is used to increase maintainability, interoperability, performance and testability. Also re-engineering is used to decrease personal dependency.

Key words: *Software Engineering, Software Re-engineering, restructuring, legacy systems*

1. INTRODUCTION

Software Re-engineering is the examination, analysis and alteration of an existing software system to reconstitute it in a new form, and the subsequent implementation of the new form. The process typically encompasses a combination of other processes such as reverse engineering, re-documentation, restructuring, translation, and forward engineering. The goal is to understand the existing software (specification, design, implementation)

and then to re-implement it to improve the system's functionality, performance or implementation [1].

There are four re-engineering objectives, they are: Preparation for functional enhancement, Improve maintainability, Migration, and Improve reliability [1]. Figure 1 refers to the general model of software re-engineering and illustrates the reverse engineering and forwarded engineering.

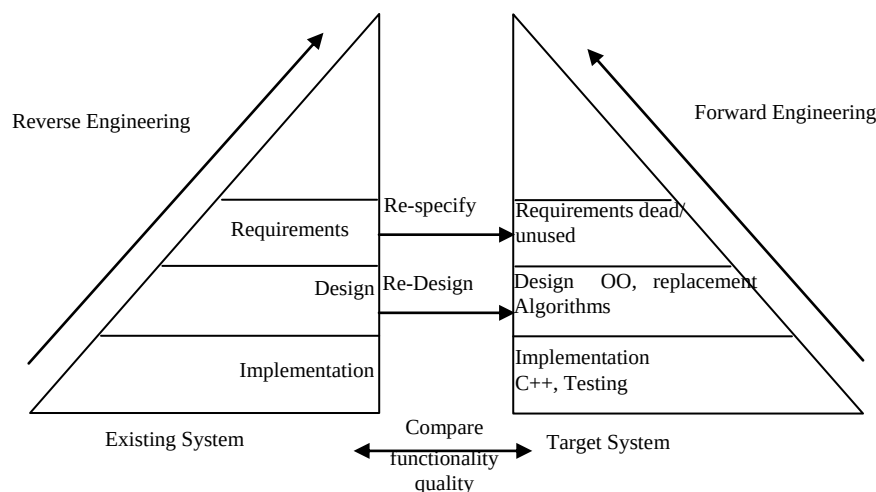


Figure 1. General Model for Software Re-engineering

For examples when one would wish to re-engineer code could be the modification of a date field (Y2K) or possibly re-modularization of code to facilitate later maintenance. In a typical Y2K scenario, one would expect to find transformation of items such as the size of the date field in a record, and other transformations that are concerned with code that is used for outputting dates.

2. WHY SOFTWARE RE-ENGINEERING?

The need of software re-engineering evolved from the following:

- Increasing legacy software.
- Emerging technologies.

- Decreasing ratio of Successful projects.
- Increasing number of competing Companies.
- More demand for quality attributes.
- Changing attitude of people.
- Insistence for Software maintenance.

3. THE BENEFITS OF SOFTWARE RE-ENGINEERING

The main benefits of software re-engineering are as follows:

- Increased maintainability [2].
- Improved performance [3].
- Increased interoperability [4].
- Decreased personal dependency [5].
- Improved testability [5].

3.1 Increased Maintainability

Many resources are spent on software maintenance. Thus, producing software that is easy to maintain may potentially save large costs. The problem of maintaining software is widely acknowledged in industry, and much has been written on how maintainability can be facilitated by e.g. tools and processes. However, you cannot control what you cannot measure, and there is yet no universal measure of maintainability. Some proposals have indeed been presented, but the very idea of measuring maintainability has inherent problems [6].

Maintainability: The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment [7].

In practice, when measuring maintainability of a system during a long sequence of changes, if it is true that the system decays continuously, the resulted graph will be similar to the one in figure 2 [6].

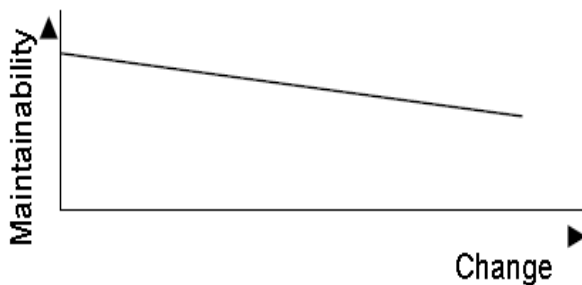


Figure 2. the system deteriorates (measured in maintainability) as changes are implemented [6].

The maintainability sometimes increases and sometimes decreases, that was expected. For example, the logical change “system restructuring” should cause the maintainability measure to increase, as Figure 3 describes [6].

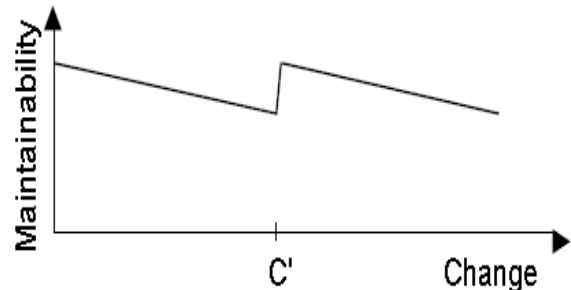


Figure 3: Change C' (a restructuring of the system) causes maintainability to (temporarily) increase [6].

As was known, the software re-engineering has many activities and the restructuring activity is one of them so the maintainability will increase when the SRE is applied.

3.2 Improved Performance

Performance (responsiveness and scalability) is a make-or-break quality for software. Poor performance costs the software industry millions of dollars annually in lost revenue, decreased productivity, increased development and hardware costs, and damaged customer relations. When performance problems occur, they must be fixed immediately. In response, the project often goes into “crisis mode” in an attempt to tune or even redesign the software to meet performance objectives. In these situations, it is vital to maximize the performance and capacity payoff of your tuning efforts [8].

There is a systematic, quantitative approach to performance tuning that helps you quickly find problems, identify potential solutions, and prioritize your efforts to achieve the greatest improvements with the least effort. The steps are:

- Figure out where you need to be.
- Determine where you are now.
- Decide whether you can achieve your objectives.
- Develop a plan for achieving your objectives.
- Conduct an economic analysis of the project.

Using this approach has been shown to provide a high payoff for a relatively small cost. Once you run into trouble, tuning the software is likely to be your only choice. However, it's important to realize that a tuned system will rarely, if ever, exhibit the level of performance that you could have achieved by re-engineering that system. The key to achieving optimum performance is to adopt a proactive approach to performance management

that anticipates potential performance problems and includes techniques for identifying and responding to those problems early in the process. With a proactive approach, you produce software that meets performance objectives and is delivered on time and within budget, and avoid the project crisis brought about by the need for tuning at the end of the project. Software performance engineering (SPE) provides a systematic, quantitative approach to proactively managing software performance [8].

Figure 4 illustrate how the software re-engineering process increases the performance of the system [9].

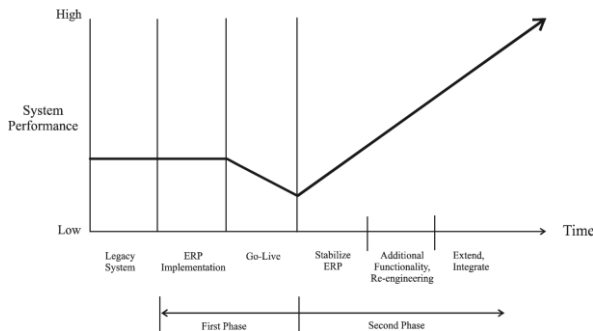


Figure 4. The effect of SRP on system performance [9].

3.3 Increased Interoperability

The IEEE Standard Glossary of Software Engineering Terminology defines interoperability as “the ability of two or more systems or components to exchange information and to use the information that has been exchanged [10].

The Department of Defense Directive defines interoperability as “the ability of systems, units or forces to provide data, information, materiel, and services, to and accept the same from other systems, units, or forces and to use the data, information, materiel, and services so exchanged to enable them to operate effectively together.” Figure 5 refer to the Levels of Information Systems Interoperability (LISI) [10].

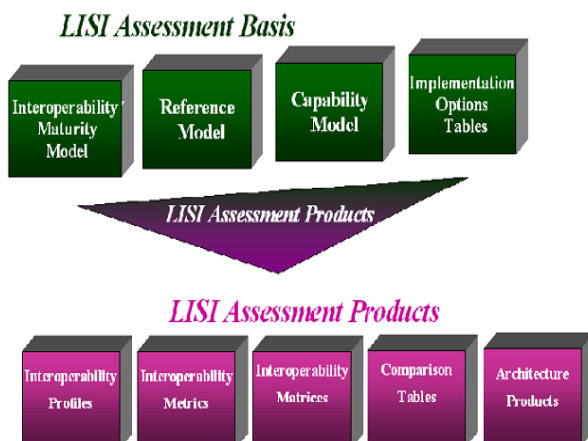


Figure 5. A Process for Interoperability [10]

System interoperability involves the efficiency of producing and refining software to interact with new and existing hardware to reliably exchange information between systems. More specifically, system interoperability encompasses the idea of software process and software modeling involving legacy systems and new hardware/software that must interact to provide reliable exchange of information between systems. Dynamic systems engineering requires that both processes and models be flexible to allow re-engineering and reevaluation of the processes and models themselves to improve the interoperability [10].

3.4 Decreased Personal Dependency

Large-scale software development requires coordination within and between very large engineering teams which may be located in different buildings, on different company campuses, and in different time zones.

Coordination between software development teams is one of the most difficult-to-improve aspects of software engineering. Kraut and Streeter argue that the software industry has been in crisis mode for its entire existence, and a root cause is the difficulty in coordinating work between teams of developers. Researchers have studied professional software development teams empirically to gain greater understanding of how software development processes, tools, and people impact coordination. The importance of intra- and inter-team coordination is a foremost concern as software development increasingly becomes globally distributed, and remains a persistent challenge in other disciplines as well.

Coordination is a decision-making that required communication, capacity and cooperation. These three components of coordination are necessary, but by themselves insufficient, for coordination to take place. Communication is necessary because person A needs to communicate to person B, in some form, what needs to be done, and B needs to understand the communication. Capacity is necessary because B needs to be able to do what is required of him. Cooperation is necessary because B needs to be willing to do what is required of him. If any of the three necessary components are lacking, the outcome will be less than ideal [11].

From that, the personal dependency is important and take a lot of effort to coordinate the software development team work, so that, Software re-engineering is used to decrease personal dependency by restructuring the legacy system in a manner that reduce the dependency of person A on person B where (both A & B) are members in the development team [5].

3.5 Improved Testability

Test costs are driven by the size and complexity of the software. Size can be measured in terms of the number of elements making up the system. These could be

statements, methods, classes, components, interfaces, files, database tables and GUIs. Complexity is measured in terms of the number of interactions between the elements. These could be associations, calls, messages, file transfers, database accesses, import, exports and events from outside. The less there are, the less there is to test [5].

Another factor which influences testability is the visibility of the data interfaces. Data passed between components can be encoded in internal data formats or it can be passed as readable character strings [5].

A very critical factor in testability is the separation of the user interface from the processing logic. This separation of presentation from processing is a prerequisite to testing the processing, i.e. the business logic, without having to enter the data in the user interface, which requires a lot of time and is difficult to automate. A final factor in reducing test effort is the separation of the data access operations from the data processing [5].

The goal of re-engineering for testability is to restructure the software in such a way that testability criteria can be met while at the same time reducing the size and the complexity of the system. Re-engineering software for testability is definitely a worthwhile effort. Identifying and removing clones, refactoring deeply nested code and restructuring the architecture are tasks that can be automated. Several tools exist which support that. By using them, re-engineering costs can be minimized. Other tasks such as algorithm optimization, merging data accesses and simplifying user interfaces can be done manually at a rather low cost. In view of the potential savings in testing costs, it is well worth it to invest in a re-engineering project running parallel to the development project. Re-engineering for testability can still be worthwhile before going into system testing. It might also mean that the development process is being complemented by a parallel re-engineering process, intended to raise the quality of the software, including testability. Figure 6 refer to parallel projects [5].



Figure 6. Parallel Projects

4. CONCLUSION

Many modern software design methods have been developed to improve the reusability and maintainability of software and to reduce the time required for the maintenance and development operations, but many companies have old or legacy software systems, these companies spend a lot of money to maintain their old systems. These systems cannot be replaced by new systems because they contain implicit information and decisions cannot be lost. For these purposes, re-engineering becomes a useful tool to convert old, obsolete systems to more efficient, streamlined systems.

REFERENCES

- [1] Dr. Linda H. Rosenberg, "Software Re-engineering" Software Assurance Technology Center, p. 2-3 Linda.Rosenberg@gsfc.nasa.gov
- [2] Fowler, M.: Refactoring—improving the design of existing code, Addison-Wesley, Reading, MA., 1999, p. 53
- [3] Sneed, H.: "Measuring the Reusability of Legacy Software Systems", Software Process – Improvement and Practice, Wiley Pub., No. 4, March, 1998, p. 43
- [4] Sneed, H., Nyary, E.: „Downsizing large Application Programs“, Journal of Software Maintenance, Vol. 6, No. 5, Oct. 1994, p. 235-248
- [5] Harry M. Sneed, Anecon GmbH, Wien, " Re-engineering for Testability", Universities of Regensburg and Passau, May, 2006
- [6] Rikard Land, "Measurements of Software Maintainability", Department of Computer Engineering, Mälardalen University.
- [7] IEEE, IEEE Standard Glossary of Software Engineering Terminology, report IEEE Std 610.12-1990, IEEE, 1990.
- [8] Lloyd G. Williams, Ph.D, Connie U. Smith, Ph.D. "Five Steps to Solving Software Performance Problems", June, 2002.
- [9] T. Hillman Willis, Ann Hillary Willis-Brown, (2002) "Extending the value of ERP", Industrial Management & Data Systems, Vol. 102 Iss: 1, pp.35 – 38
- [10] Cadet Pamela A. Sanders, Dr. John A. Hamilton, "A Process for Interoperability", Auburn University.
- [11] Christopher Poile, Andrew Begel, Nachiappan Nagappan, and Lucas Layman " Coordination in Large-Scale Software Development: Helpful and Unhelpful Behaviors", Dept. of Computer Science, North Carolina State University, September 28, 2009.

A Literature Review and Classification of Selected Software Engineering Researches

Ahmed Saleem Abbas, W. Jeberson, V.V. Klinsega

SHIATS, University, Department of Computer Science & Information Technology, India

ABSTRACT

In this paper, a literature review and classification scheme for selected software engineering researches is presented. The study shows that an increasing volume of software engineering researches have been conducted in diverse range of areas. The articles are classified and results of these are presented based on classification scheme that consist of five main categories: software development process, software management, software engineering techniques, software re-engineering, and software applications. Analyses of the selected researches are carried out and gaps in the research are identified. A comprehensive list of references is presented. This review is intended to provide impetus in research and help simulate further interest.

Keywords: *Software engineering, Software Re-engineering, Review, Software development process, Software engineering techniques.*

1. INTRODUCTION

More than 250 papers, which were published between the time period 2008-2012 in conferences and international journals like Springer, IEEE, IJSEA, IJCSI, IJCSE, IJCA, IJEST, famoosR, COMATS, and Elsevier were collected, analyzed and classified into a number of categories and subcategories. The study led to the identification of gap in software engineering researches and enabled the authors to recommend the area where there is a lot of scope for future research.

What is software engineering?

Software engineering can be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.[1] It is the application of engineering to software because it integrates significant mathematics, computer science and practices whose origins are in engineering.[2] It is also defined as a systematic approach to the analysis, design, assessment, implementation, testing, maintenance and reengineering of software, that is, the application of engineering to software.[3] The term software engineering first appeared in the 1968 NATO Software Engineering Conference, and was meant to provoke thought regarding the perceived "software crisis" at the time.[4][5]

2. IDENTIFIED SOFTWARE ENGINEERING RESEARCH AND EXTENT OF RESEARCH CARRIED OUT

The classification framework shown in figure (1) is based on the literature review and the nature of software engineering researches, which meant to give an understanding of how the subject has evolved and progressing. Since software engineering is an important area it is very essential to be clear of it's definition. There are several researchers working in this area and trying to define and specify the universally accepted list of software engineering principles and concepts and trying to apply all these principles and concepts on a real project but until now they could not do that completely and perfectly. The software engineering researches can be classified into five categories, which are; (I) software development process which account to 35 % (90 papers), (II) software management which counts to 30 % (75 papers), (III) software re-engineering which shares (36 papers) of the distribution 14 %, (IV) software engineering techniques the contribution in which is 13 % (34 papers) and (V) software applications which accounts to 8 % (19 papers). The proportion of research carried out in each category is shown in figure (2).

These main categories are further divided into a number of sub categories as in figure (1). The percentage distribution of research in the main categories is shown in figure (2) and according to the year of publishing these

Papers can be grouped as shown in figure (3). The papers are further grouped based on the journal in which they are published as shown in figure (4), and the number of

papers published in each journal as shown in table (3). All the categories and their sub-categories will be discussed in detail in the later sections with graphs and tables.

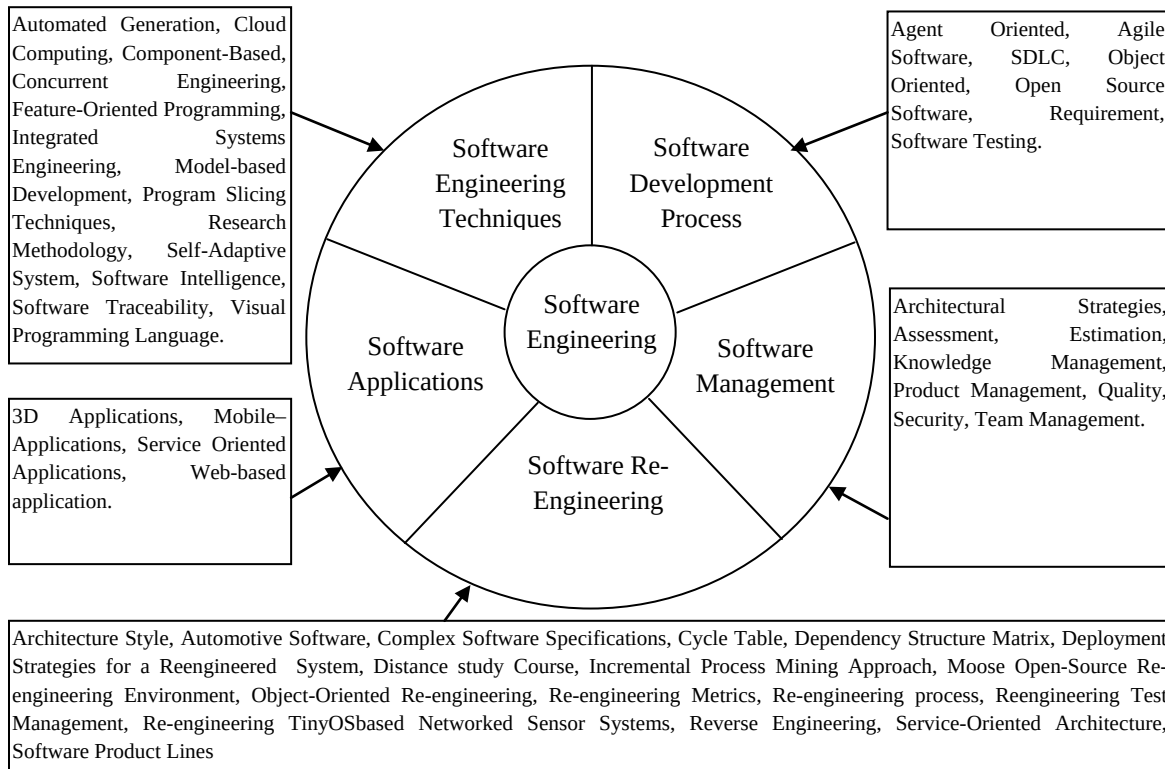


Figure (1): the literature reviews papers classification

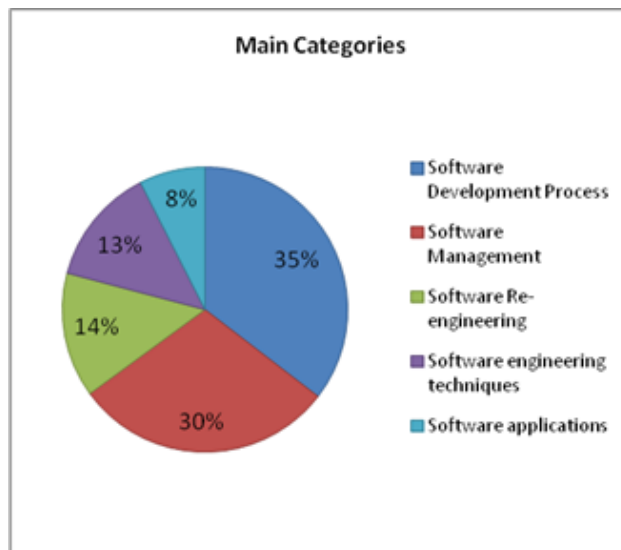


Figure (2): Percentage distribution of research in main categories

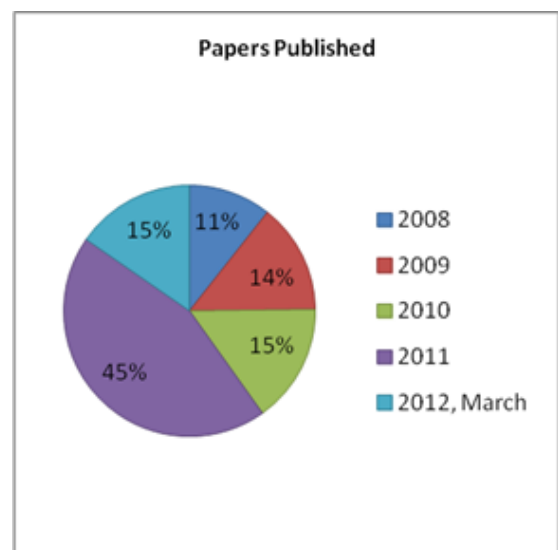


Figure (3): The papers published in a year

Table (1): Number of published papers in each category

Main Categories	No. of papers
Software Development Process	90
Software Management	75
Software Re-engineering	36

Software engineering techniques	34
Software applications	19
Total	254

Table (2): Number of published papers in each year

Year	No. of published papers

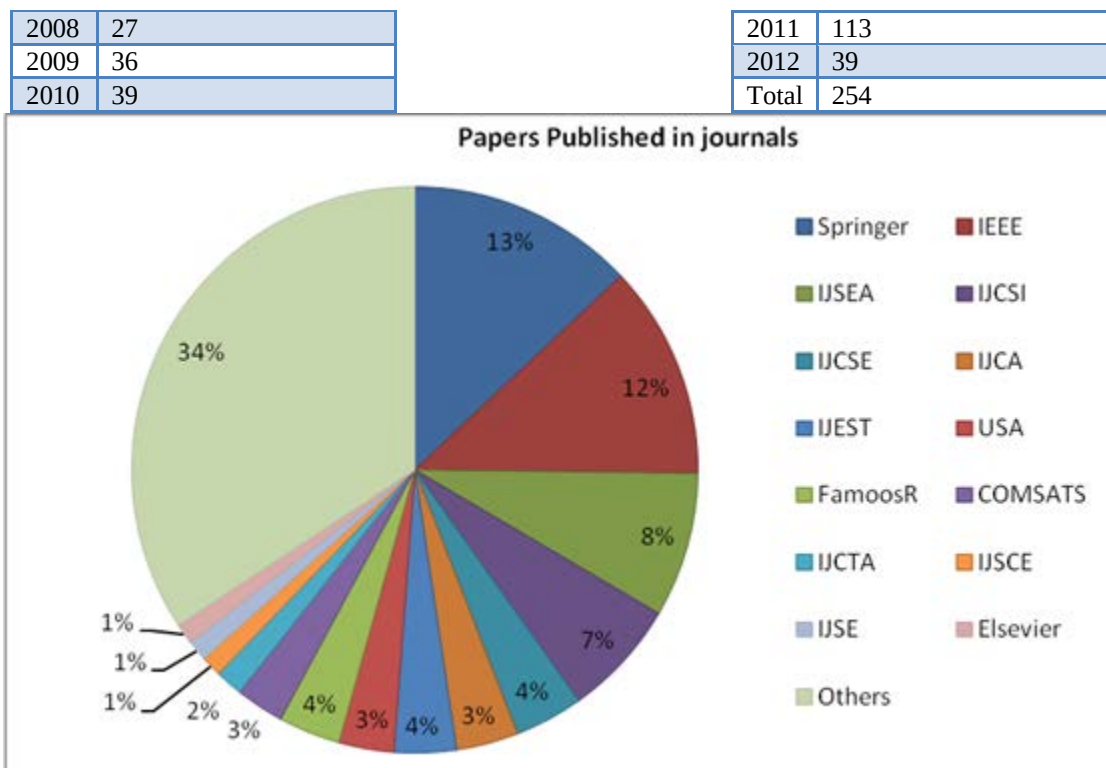


Figure (4): Papers published in various journals

Table (3): Number of papers published in various Journals

Journals / Institute	No. of published papers	Journals / Institute	No. of published papers
Springer	33	FamoosR	9
IEEE	31	COMSATS	7
IJSEA	21	IJCTA	4
IJCSI	17	IJCSCE	3
IJCSE	10	IJSE	3
IJCA	9	Elsevier	3
IJEST	9	Others	87
USA	8	Total	254

Extent of research in subcategories of each category is described in detail as follows:

2.1 Software Development Process

A software development process, also known as a software development life cycle (SDLC), is a structure imposed on the development of a software product. Similar terms include software life cycle and software process. It is often considered a subset of systems development life cycle. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Some people consider a life-cycle model a more general term

and a software development process a more specific term. For example, there are many specific software development processes that 'fit' the spiral life-cycle model. ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software [7].

There are many activities involved in software development called as "Software development activities", these activities is planning (requirements analysis), design, Implementation, testing and documenting, Deployment and maintenance.

In this research area there are 90 papers, which are analyzed and classified into number of sub-categories, which are described briefly as follows:

- a. SDLC: It is a software development life cycle and there many models like Waterfall model, V model, spiral model, incremental model, and evolutionary model [7]. In this subcategory there are 32 papers and their percentage is 34%.
- b. Requirement engineering: Requirement is a singular documented physical and functional need that a particular product or service must be or perform. It is most commonly used in a formal sense in systems engineering, software engineering, or enterprise engineering. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system for it to have value and utility to a user [7]. In this subcategory there are 15 papers and their percentage is 16%.
- c. Software testing: Software testing can be stated as the process of validating and verifying that a software program/application/product: meets the requirements that guided its design and development; works as expected; and can be implemented with the same characteristics [8]. In this subcategory there are 15 papers and their percentage is 16%.
- d. Agile Software development process: Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle [9]. In this subcategory there are 12 papers and their percentage is 13%.
- e. Open Source Software development process: Open-source software (OSS) is computer software that is available in source code form: the source code and certain other rights normally reserved for copyright holders are provided under an open-source license that permits users to study, change, improve and at times also to distribute the software. Open source software is very often developed in a public, collaborative manner. Open-source software is the most prominent example of open-source development and often compared to (technically defined) user-generated content or (legally defined) open content

movements [10]. In this subcategory there are 7 papers and their percentage is 8%.

- f. Object oriented software development process: It is a programming paradigm using "objects" – data structures consisting of data fields and methods together with their interactions – to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance. Many modern programming languages now support OOP, at least as an option [11]. In this subcategory there are 6 papers and their percentage is 6%.
- g. Agent oriented software development process: It is a programming paradigm where the construction of the software is centered around the concept of software agents. It has externally-specified agents (with interfaces and messaging capabilities) at its core. They can be thought of as abstractions of objects. Exchanged messages are interpreted by receiving "agents", in a way specific to its class of agents [12]. In this subcategory there are 3 papers and their percentage is 3%.

All these sub-categories are shown in figure (5) and the numbers of papers in each sub-category is shown in table (4). The number of published paper in this category with respect to the year of publishing is shown in figure (6).

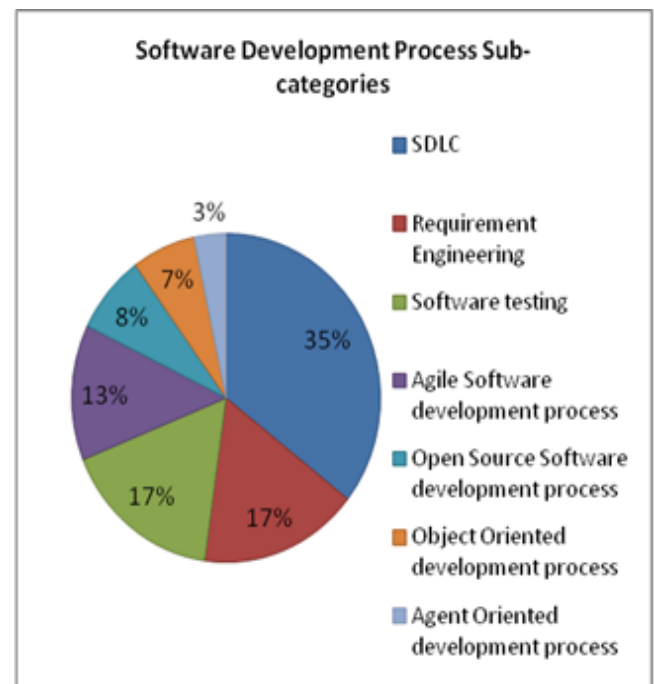
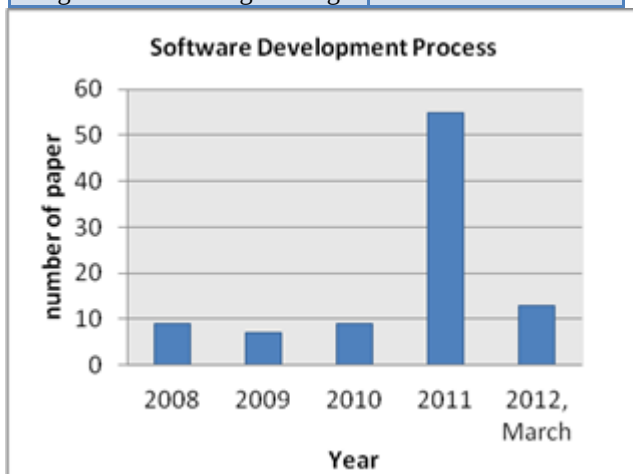


Figure (5): Software Development Process sub-categories

Table (4): Number of papers in each sub-category of Software Development Process

Software Development Process Sub-categories	No. of paper in each sub-category
SDLC	32
Requirement Engineering	15
Software testing	15
Agile Software Engineering	12
Open Source Software Engineering	7
Object Oriented Engineering	6
Agent Oriented Engineering	3

**Figure (6): Number of papers published in the software development process based on the year of publishing.**

2.2 Software Management

Software project management begins with a set of activities that are collectively called project planning. Before the project can begin, the manager and the software team must estimate the work to be done, the resources that will be required, and the time that will elapse from start to finish. Whenever estimates are made, we look into the future and accept some degree of uncertainty as a matter of course [13].

In this research area there are 75 papers, where are classified into number of sub-categories, brief description of each one is given below:

h. Assessment: Several methods have been created to define an assessment process for software. Some focus on some aspects like the maturity, the durability and the strategy of the organization around the project itself. Other methodologies add functional aspects to the assessment process [13]. In this

subcategory there are 18 papers and their percentage is 24%.

i. Quality: In the context of software engineering, software quality refers to two related but distinct notions that exist wherever quality is defined in a business context: Software functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product; Software structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly [13]. In this subcategory there are 16 papers and their percentage is 21%.

j. Architectural Strategies: As organizations enhance their architecture competency, architects will come to play a more and more active role in the strategy setting process, in addition to the crucial the role they already play in business strategy implementation. However, strategy is not part of the curriculum of the average computer science program, nor is it a competency that organizations necessarily foster in the technical community. In this primer, our goal is to provide you with a strategy for the intelligent introduction to state-of-the-art business strategy thinking and models. Our aim is to help you develop the language, sensibilities, and tools that will enable you to quickly position yourself to make a useful contribution to the strategy process. There more attention to this competency because it is emerging as a critical avenue for competitive distinction. The trend has been for technology to become more essential to basic business capabilities as well as to creating differentiating value propositions that distinguish the business in the marketplace. At the same time, technology is complex and fast moving enough that it requires its own kind of attention, and it falls to the architects to bring this insight to the strategy table. With this foundation in the formulation of the business strategy, architects can also be a more effective bridge between business strategy and the technical strategy that implements it [14]. In this subcategory there are 12 papers and their percentage is 16%.

k. Estimation: Software Estimation is responsive to the widespread problems the software industry has experienced in creating meaningful cost and schedule estimates. Software estimation is not as hard or mysterious as many people think, but the knowledge

of how to create effective estimates has not been well publicized. Software Estimation provides a comprehensive set of tips and heuristics that software developers, technical leads, and project managers can apply to create more accurate estimates. It presents fundamental estimation techniques and addresses specific estimation challenges. It explains how to avoid common pitfalls. Software Estimation doesn't avoid hairy mathematical approaches, but the non-mathematical reader will find plenty of useful guidelines without getting bogged down in complex formulas [15]. In this subcategory there are 11 papers and their percentage is 15%.

- l. Team Management: it is refers to techniques, processes and tools for organizing and coordinating a group of individuals working towards a common goal. The 'Team Development Model', identified by Bruce Tuckman, offers a foundational definition of the stages teams go through during their lifecycle. While the activities of team management are not new, many of the tools used by team managers are. The more Organizational Development-oriented practitioners often use interview-based analysis and provide reportage and insights that team leaders and their management may use to adapt team practices for higher performance. Teams can also be developed through team building activities - which can also be used simply to build relationships where team members lack cohesion due to organizational structure or physical distance. Project managers may approach team management with a focus on structure, communications and standardized practices [13]. In this subcategory there are 9 papers and their percentage is 12%.
- m. Security: Security is the degree of protection against danger, damage, loss, and crime. Security as a form of protection is structures and processes that provide or improve security as a condition. Security has to be compared to related concepts: safety, continuity, reliability. The key difference between security and reliability is that security must take into account the actions of people attempting to cause destruction [16]. In this subcategory there are 7 papers and their percentage is 9%.
- n. Knowledge management: comprises a range of strategies and practices used in an organization to identify, create, represent, distribute, and enable adoption of insights and experiences. Such insights and experiences comprise knowledge, either embodied in individuals or embedded in organizations as processes or practices. Knowledge management efforts typically focus on organizational objectives such as improved performance,

competitive advantage, innovation, the sharing of lessons learned, integration and continuous improvement of the organization. KM efforts overlap with organizational learning, and may be distinguished from that by a greater focus on the management of knowledge as a strategic asset and a focus on encouraging the sharing of knowledge [17]. In this subcategory there are 1 papers and their percentage is 1%.

- o. Product Management: it is an organizational lifecycle function within a company dealing with the planning or marketing of a product or products at all stages of the product lifecycle. Product management (inbound focused) and product marketing (outbound focused) are different yet complementary efforts with the objective of maximizing sales revenues, market share, and profit margins. The role of product management spans many activities from strategic to tactical and varies based on the organizational structure of the company [18]. In this subcategory there are 1 papers and their percentage is 1%.
- p. All these sub-categories are shown in figure (7) and the numbers of papers published in each sub-category is shown in table (5). The number of papers published in this category with respect to the year of publishing is shown in figure (8).

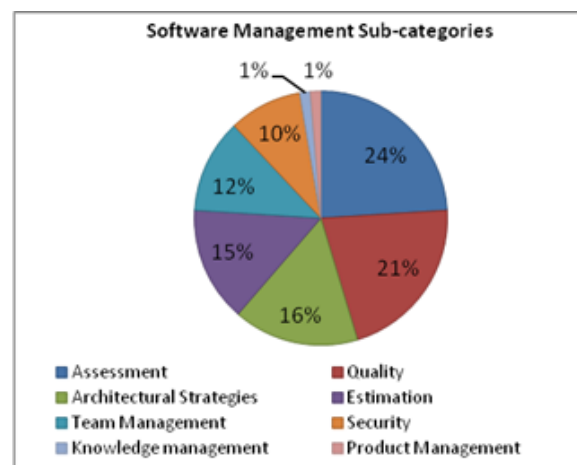


Figure (7): software management sub-categories

Table (5): Number of papers in each sub-category of software management

Software Management Sub-categories	No. of paper in each sub-category
Assessment	18
Quality	16
Architectural Strategies	12
Estimation	11

Team Management	9
Security	7
Knowledge management	1
Product Management	1

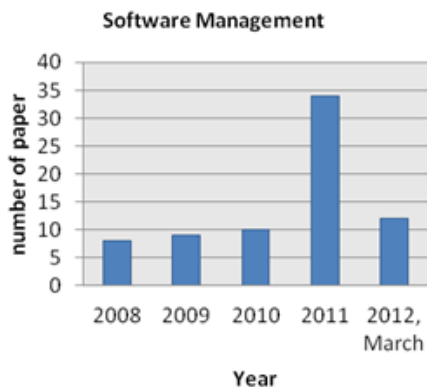


Figure (8): Number of papers published in the software management based on the year of publishing

2.3 Software Re-engineering

The reengineering of software was described by Chikofsky and Cross in their 1990 paper [19], as "The examination and alteration of a system to reconstitute it in a new form". Less formally, reengineering is the modification of a software system that takes place after it has been reverse engineered, generally to add new functionality, or to correct errors. This entire process is often erroneously referred to as reverse engineering; however, it is more accurate to say that reverse engineering is the initial examination of the system, and reengineering is the subsequent modification. Re-engineering is mostly used in the context where a legacy system is involved [20]. Software systems are evolving on high rate because there more research to make the better so therefore software system in most cases, legacy software needs to operate on a new computing platform. 'Re-engineering' is a set of activities that are carried out to re-structure a legacy system to a new system with better functionalities and conform to the hardware and software quality constraint.

In this research area there are many papers 36 papers, these papers are classified into number of sub-categories, these categories as following with brief description for each one of them:

a. Reverse Engineering: it is the process of discovering the technological principles of a device, object, or system through analysis of its structure, function, and operation. It often involves taking something (e.g., a mechanical device, electronic component, software program, or biological, chemical, or organic matter) apart and analyzing its workings in detail to be used

in maintenance, or to try to make a new device or program that does the same thing without using or simply duplicating (without understanding) the original [21]. In this sub-category there are 12 papers and their percentage is 33%.

- b. Object-Oriented Re-engineering: Object oriented development has proved its worth in today's system because its design and development is better, reliable and easier to access than the traditional methodologies. Due to updated requirements and lack of documentation in old systems has provided a motivation to revamp the systems. Rebuilding or redesigning the same system is highly expensive. To overcome this problem reverse engineering of the system is used as most suitable alternative [22]. In this subcategory there are 3 papers and their percentage is 8%.
- c. Service-Oriented Architecture: In software engineering, a Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes. SOA design principles are used during the phases of systems development and integration [23]. In this subcategory there are 3 papers and their percentage is 8%.
- d. Automotive Software: it is different from the kind of software that is usually addressed by current reengineering research. These papers give an overview of the particular conditions and challenges that in the automotive domain [24]. In this subcategory there are 2 papers and their percentage is 6%.
- e. Cycle Table: Identifying and understanding cycles in large applications is a challenging task. In these papers CycleTable which displays both direct and indirect references was presented. It shows how minimal cycles are intertwined through shared edges and allows the reengineer to devise simple strategies to remove them [25]. In this subcategory there are 2 papers and their percentage is 6%.
- f. Re-engineering Metrics: A metric is nothing more than a standard measure to assess your performance in a particular area. Metrics are at the heart of a good, customer-focused process management system and any program directed at continuous improvement. The focus on customers and performance standards show up in the form of metrics that assess your ability to meet your customers' needs and business

objectives [26]. In this subcategory there are 2 papers and their percentage is 6%.

- g. Re-engineering process: it is all the activities from the legacy software analyzing to the generation of new software. In this subcategory there are 2 papers and their percentage is 6%.
- h. There are another research in the software re-engineering field, about one paper in each sub-categories, these sub-categories are: Complex Software Specifications, Dependency Structure Matrix, Deployment Strategies for a Reengineered

System, Distance study Course, Incremental Process Mining Approach, Moose Open-Source Re-engineering Environment, Reengineering Test Management, Re-engineering TinyOSbased Networked Sensor Systems, Software Product Lines, Architecture Style.

All these sub-categories are shown in figure (9) and the numbers of papers in each sub-category is shown in table (6). The number of papers published in this category with respect to the year of publishing is shown in figure (10).

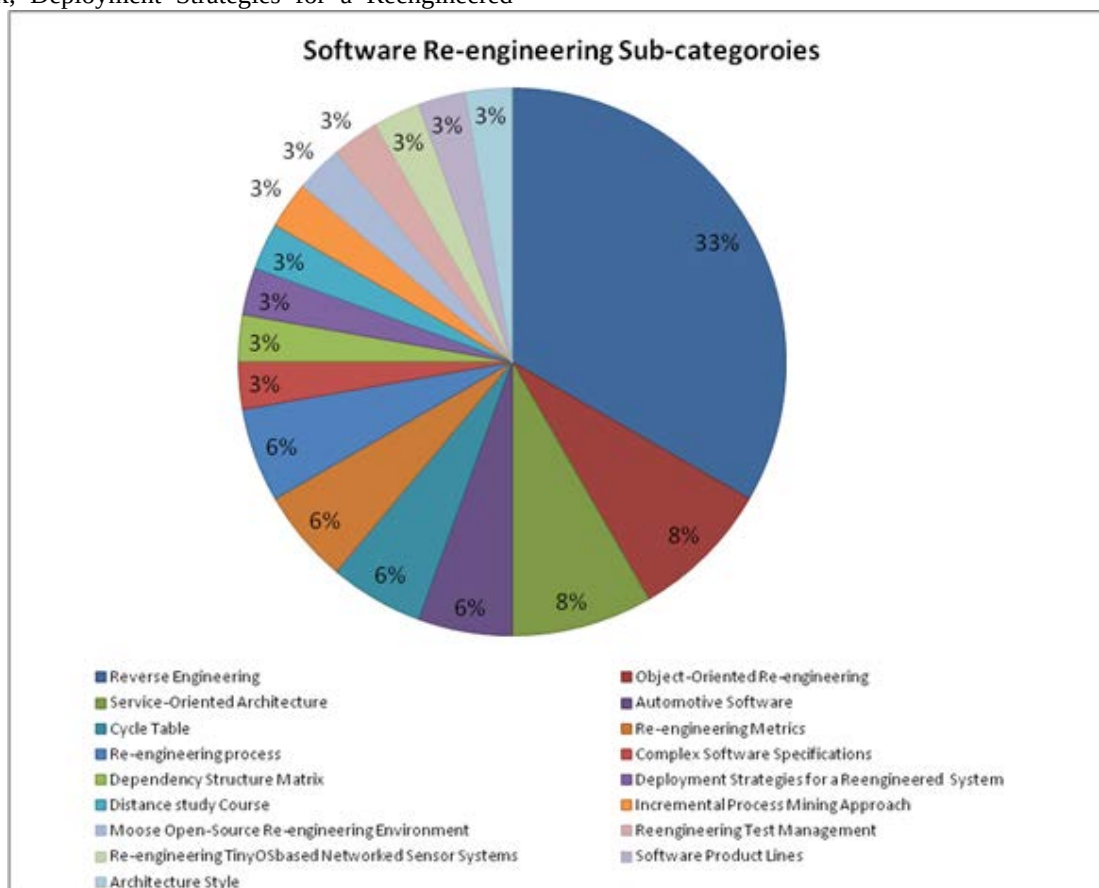


Figure (9): Software Re-engineering sub-categories

Table (6): Number of papers in each sub-category of Software Re-engineering

Software Re-engineering Sub-categories	No. of paper in each sub-category
Reverse Engineering	12
Object-Oriented Re-engineering	3
Service-Oriented Architecture	3
Automotive Software	2
Cycle Table	2
Re-engineering Metrics	2
Re-engineering process	2

Complex Software Specifications	1
Dependency Structure Matrix	1
Deployment Strategies for a Reengineered System	1
Distance study Course	1
Incremental Process Mining Approach	1
Moose Open-Source Re-engineering Environment	1
Reengineering Test Management	1
Re-engineering TinyOSbased Networked Sensor Systems	1
Software Product Lines	1
Architecture Style	1

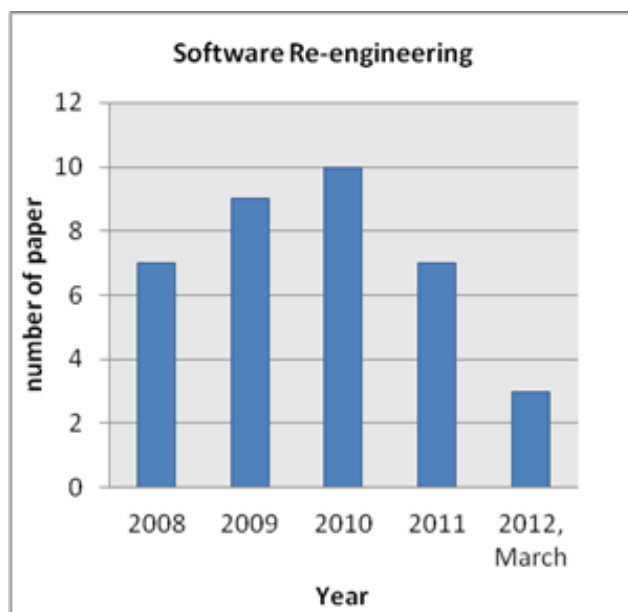


Figure (10): Papers published in the software Re-engineering.

2.4 Software Engineering Techniques

Today, software is becoming ever more complex, developed with a very short time to market, and required to cope with changing requirements. This scenario calls for increased levels of flexibility and agility, both in the technologies used and the processes followed for engineering software. So many techniques were provided that can meet these challenges and these techniques are known as Software engineering techniques.

In this research area there are 33 papers, which are classified into number of sub-categories as following with brief description for each:

a. **Component-Based:** Component-based software engineering (CBSE) (also known as component-

based development (CBD)) is a branch of software engineering that emphasizes the separation of concerns in respect of the wide-ranging functionality available throughout a given software system. It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems. This practice aims to bring about an equally wide-ranging degree of benefits in both the short-term and the long-term for the software itself and for organizations that sponsor such software [27]. In this sub-category there are 10 papers and their percentage is 29%.

- b. **Automated Generation:** is a style of computer programming that uses automated source code creation through generic frames, classes, prototypes, templates, aspects, and code generators to improve programmer productivity. It is often related to code-reuse topics such as component-based software engineering and product family engineering [28]. In this sub-category there are 7 papers and their percentage is 20%.
- c. **Concurrent Engineering:** is a work methodology based on the parallelization of tasks (i.e. performing tasks concurrently). It refers to an approach used in product development in which functions of design engineering, manufacturing engineering and other functions are integrated to reduce the elapsed time required to bring a new product to the market [29]. In this sub-category there are 3 papers and their percentage is 9%.
- d. **Research Methodology:** is a way to systematically solve the research problem. It may be understood as a science of studying how research is done scientifically. There are various steps that are generally adopted by a researcher in studying the research problem along with the logic behind them. It is necessary for the researcher to know not only the research methods/techniques but also the methodology. Researchers not only need to know how to develop certain indices or tests, how to

calculate the mean. The mode, the median or the standard deviation or chi-square, how to apply particular research techniques, but they also need to know which of these methods or techniques are relevant and which are not, and what would they mean and indicate and why. Researchers also need to understand the assumptions underlying various techniques and they need to know the criteria by which they can decide that certain techniques and procedures will be applicable to certain problems and others will not. All this means that it is necessary for the researcher to design his methodology for his problem as the same may differ from problem to problem [30]. In this sub-category there are 3 papers and their percentage is 9%.

- e. Cloud Computing: it is a metaphor used by Technology or IT Services companies for the delivery of computing requirements as a service to a heterogeneous community of end-recipients. The term cloud theoretically signifies abstraction of technology, resources and its location that are very vital in building integrated computing infrastructure (including networks, systems and applications). All Cloud computing models rely heavily on sharing of resources to achieve coherence and economies of

scale similar to a utility (like the electricity grid) over a network (typically the Internet) [31]. In this sub-category there are 2 papers and their percentage is 6%.

- f. Self-Adaptive System: the self-adaptability of systems has been studied in a wide range of disciplines, from biology to robotics, only recently has the software engineering community recognized its key role in enabling the development of future software systems that are able to self-adapt to changes that may occur in the system, its requirements, or the environment in which it is deployed [32].
- g. There are another research in the field of software engineering techniques, about one paper in each sub-categories, these sub-categories are: Feature-Oriented Programming, Integrated Systems Engineering, Model-Based Development, Program Slicing Techniques, Software Intelligence, Software Traceability, and Visual Programming Language

All these sub-categories are shown in figure (11) and the number of papers in each sub-category is shown in table (7). The number of papers published in this category with respect to the year of publishing is shown in figure (12).

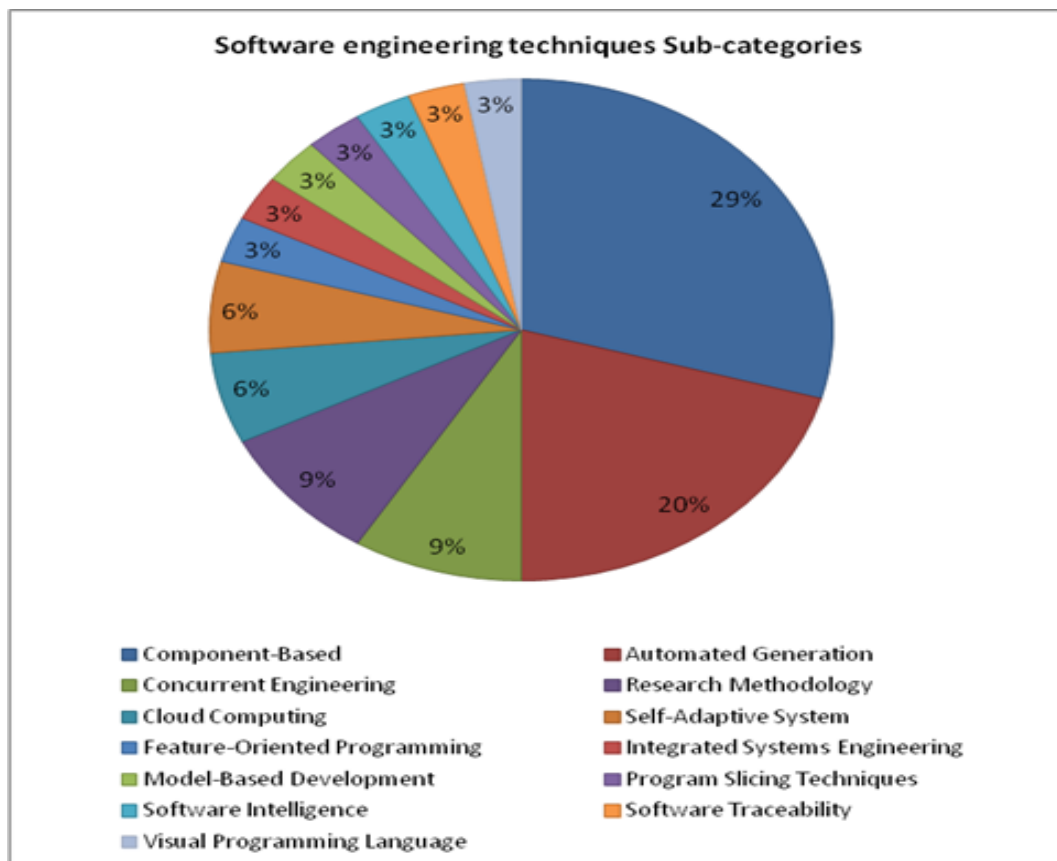
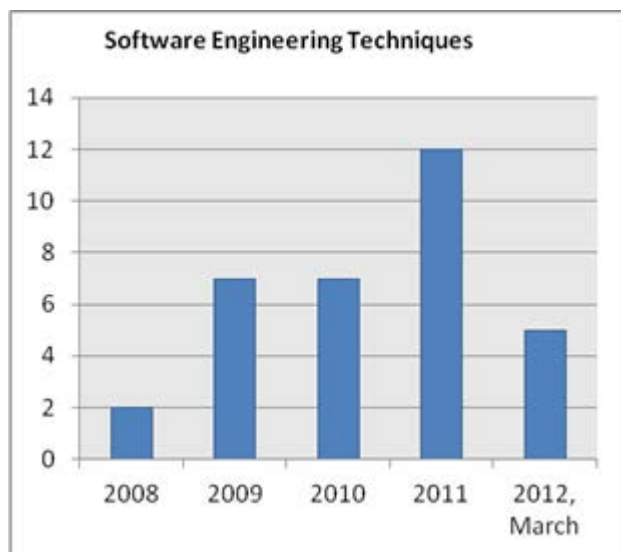


Figure (11): Software engineering techniques sub-categories

Table (7): Number of papers published in each sub-category of Software engineering techniques

Software engineering techniques Sub-categories	No. of paper in each sub-category
Component-Based	10
Automated Generation	7
Concurrent Engineering	3
Research Methodology	3
Cloud Computing	2
Self-Adaptive System	2
Feature-Oriented Programming	1
Integrated Systems Engineering	1
Model-Based Development	1
Program Slicing Techniques	1
Software Intelligence	1
Software Traceability	1
Visual Programming Language	1

**Figure (12): Papers published in the software engineering techniques.**

2.5 Software Applications

Also known as an application or an "app", is computer software designed to help the user to perform specific tasks. Examples include enterprise software, accounting software, office suites, and graphics software and media players. Many application programs deal principally with documents. Apps may be bundled with the computer and its system software, or may be published separately. Some users are satisfied with the bundled apps and need never

install one. Application software is contrasted with system software and middleware, which manage and integrate a computer's capabilities, but typically do not directly apply in the performance of tasks that benefit the user. The system software serves the application, which in turn serves the user [33].

In this research area there are many papers 19 papers, these papers are classified into number of sub-categories, these categories as following with brief description for each one of them:

- Web-based application:** A web application is an application that is accessed over a network such as the Internet or an intranet. The term may also mean a computer software application that is coded in a browser-supported language (such as JavaScript, combined with a browser-rendered markup language like HTML) and reliant on a common web browser to render the application executable. Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis and many other functions [34]. In this sub-category their are 12 papers and there percentage is 63%.
- Mobile – Applications:** It is application software that developed for small low-power handheld devices such as personal digital assistants, enterprise digital assistants or mobile phones. These applications are either pre-installed on phones during manufacture, downloaded by customers from various mobile software distribution platforms, or web applications delivered over HTTP which use server-side or client-side processing (e.g. JavaScript) to provide an "application-like" experience within a Web browser [35]. In this sub-category there are 3 papers and their percentage is 16%.
- Service Oriented Applications:** A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed. Service-oriented architectures are not a new thing. The first service-oriented architecture for many people in the

past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification [36]. In this sub-category there are 3 papers and their percentage is 16%.

- d. 3D Applications: it refers to programs used to create 3D computer-generated imagery, there are many software like 3D Studio MAX, 3D modeling application, Animation Master, and etc. [37]. In this sub-category there are 1 papers and their percentage is 5%.

These sub-categories are shown in figure (13) and the numbers of papers in each sub-category is shown in table (8). The number of papers published in this category with respect to the year of publishing is shown in figure (14).

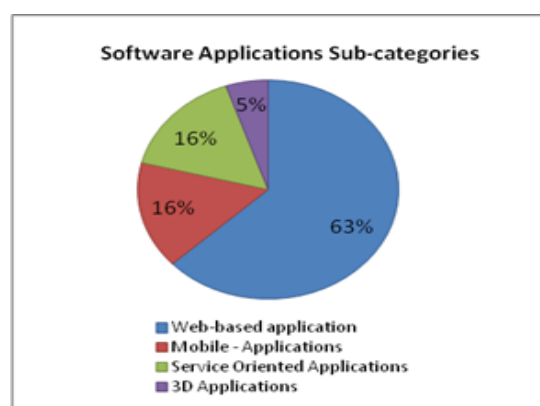


Figure (13): Software engineering techniques sub-categories

Table (8): Numbers of papers published in each sub-category of Software application

Software applications Sub-categories	No. of paper in each sub-category
Web-based application	12
Mobile - Applications	3
Service Oriented Applications	3
3D Applications	1

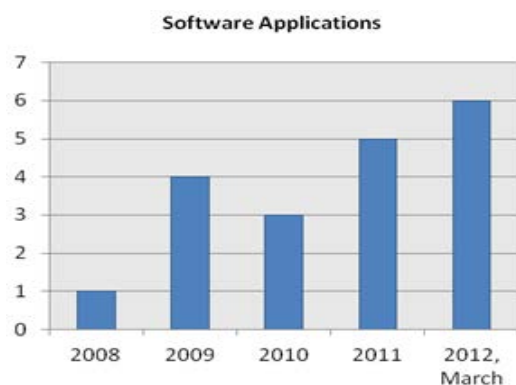


Figure (14): Papers published in the software applications category.

3. THE GAP IN SOFTWARE ENGINEERING

On classification of the researches published in the last five years and analysis of data, it gives a concrete idea that research conducted in software re-engineering, software engineering techniques and software applications is meager while a higher percent of research is carried out in rest of the two areas as shown in figure (2) and table (1). The comparison among the number of papers published in all five categories with respect to the year of publishing is shown in figure (15). From the figure, it is understood that very less research is conducted in the area of software re-engineering in the current year (2012, March), therefore, research gap exists in the area of software re-engineering which means more research is possible in the same.

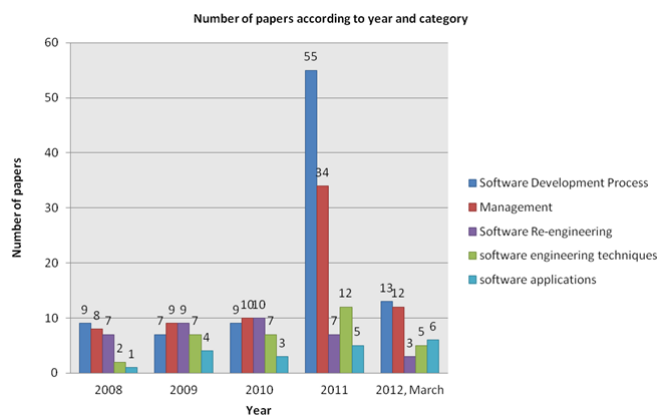


Figure (15): Papers published in each category from the year (2008 - 2012, March).

4. CONCLUSION

On reviewing more than 250 papers related to software engineering published in various reputed journals, the authors have realized that software engineering is a very important research area and it can be classified into various sub-areas like software development process, software management, software re-engineering, software engineering techniques and software applications. Research was carried out to a lesser in all five categories in the year 2008, 2009 and 2010, where as there is substantial rise in research conducted in the year 2011 especially in the first two categories, which is again followed by a sudden drop in the year 2012 until March. Further, the study also provides an insight on the insufficient research carried out in software re-engineering and hence the authors recommend greater scope in this particular area and that extensive research may be conducted to improve and enhance the software re-engineering processes.

REFERENCES

- [1] SWEBOK executive editors, Alain Abran, James W. Moore; editors, Pierre Bourque, Robert Dupuis. (2004). Pierre Bourque and Robert Dupuis. ed. Guide to the Software Engineering Body of Knowledge - 2004 Version. IEEE Computer Society. pp. 1–1. ISBN 0-7695-2330-7. <http://www.swebok.org>.
- [2] ACM (2006). "Computing Degrees & Careers". ACM. http://computingcareers.acm.org/?page_id=12. Retrieved 2010-11-23.
- [3] Laplante, Phillip (2007). What Every Engineer Should Know about Software Engineering. Boca Raton: CRC. ISBN 9780849372285. <http://books.google.com/?id=pFHYk0KWAEgC&pg=PP1&dq=What%20Every%20Engineer%20Should%20Know%20about%20Software%20Engineering.&pg=PA1#v=onepage&q&f=false>. Retrieved 2011-01-21.
- [4] Peter, Naur; Brian Randell (7–11 October 1968). "Software Engineering: Report of a conference sponsored by the NATO Science Committee" (PDF). Garmisch, Germany: Scientific Affairs Division, NATO. <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF>. Retrieved 2008-12-26.
- [5] Randell, Brian (10 August 2001). "The 1968/69 NATO Software Engineering Reports". Brian Randell's University Homepage. The School of the Computer Sciences, Newcastle University. <http://homepages.cs.ncl.ac.uk/brian.randell/NATO/NATOREports/index.html>. Retrieved 2008-10-11. "The idea for the first NATO Software Engineering Conference, and in particular that of adopting the then practically unknown term "software engineering" as its (deliberately provocative) title, I believe came originally from Professor Fritz Bauer."
- [6] Ralph, P., and Wand, Y. A Proposal for a Formal Definition of the Design Concept. In, Lyytinen, K., Loucopoulos, P., Mylopoulos, J., and Robinson, W., (eds.), Design Requirements Engineering: A Ten-Year Perspective: Springer-Verlag, 2009, pp. 103-136
- [7] Wiegers, Karl E. (2003). Software Requirements, Second Edition. Microsoft Press. ISBN 0-7356-1879-8.
- [8] Software Testing by Jiantao Pan, Carnegie Mellon University.
- [9] Beck, Kent; et al. (2001). "Manifesto for Agile Software Development". Agile Alliance. Retrieved 14 June 2010.
- [10] Verts, William T. (2008-01-13). "Open source software". World Book Online Reference Center.
- [11] McCarthy, J.; Brayton, R.; Edwards, D.; Fox, P.; Hodes, L.; Luckham, D.; Maling, K.; Park, D. et al (March 1960). LISP I Programmers Manual. Boston, Massachusetts: Artificial Intelligence Group, M.I.T. Computation Center and Research Laboratory. p. 88f. "In the local M.I.T. patois, association lists [of atomic symbols] are also referred to as "property lists", and atomic symbols are sometimes called "objects".
- [12] Shoham, Y. (1990) (Technical Report STAN-CS-90-1335). Agent-Oriented Programming. Stanford University: Computer Science Department.
- [13] Pressman, Scott (2005), Software Engineering: A Practitioner's Approach (Sixth, International ed.), McGraw-Hill Education Pressman, p. 388.
- [14] Bredemeyer, Dana and Ruth Malan, "Architecture Strategy Choreographs the Dance of Change", slides from the presentation to be given at the Enterprise Architecture Conference in Phoenix, AZ in November, 2003.
- [15] Steve McConnell's instantly indispensable Software Estimation: book, 2006.
- [16] Bruce Schneier, Beyond Fear: Thinking Sensibly about Security in an Uncertain World, Copernicus Books, pages 26-27
- [17] Addicott, Rachael; McGivern, Gerry; Ferlie, Ewan (2006). "Networks, Organizational Learning and Knowledge Management: NHS Cancer Networks". Public Money & Management 26 (2): 87–94. doi:10.1111/j.1467-9302.2006.00506.x.
- [18] Rajeev Misra, 2009, "Software Product Management", InfoPro Worldwide, Inc. 12 Perrine Road, Suite # 104 Monmouth Jn., NJ 08852, USA, Tel: 1 (609) 606 9020, Fax: 1 (609) 452 0335, Web site : www.infoproworldwide.com
- [19] Chikofsky, E. and Cross, J., 1990. Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, 7(1):13-18.
- [20] Asit Kumar Gahalaut et. al. / International Journal of Engineering Science and Technology Vol. 2(06), 2010, 2296-2303

- [21] Eilam, Eldad & Chikofsky, Elliot J. (2007). Reversing: secrets of reverse engineering. John Wiley & Sons. p. 3. ISBN 9780764574818. <http://books.google.com/books?id=K8vIf99UBYQC&pg=PA3>.
- [22] V. Carchiolo and et. Al, "Object-oriented re-engineering of manufacturing models: A case study", Springer Science, Business Media, LLC, paper, 2010
- [23] Bell, Michael (2008). "Introduction to Service-Oriented Modeling". Service-Oriented Modeling: Service Analysis, Design, and Architecture. Wiley & Sons. pp. 3. ISBN 978-0-470-14111-3.
- [24] V. S. Coerne and et al, "Automotive Software: Characteristics and Reengineering Challenges", Corporate Sector Research and Advance Engineering Software, P.O. Box 30 02 40, 70442 Stuttgart, Germany, paper, 2010.
- [25] J. Laval and et. al, "Identifying cycle causes with CycleTable", FAMOOSr, paper, 11-Oct-09
- [26] J. W. Wesner et al, Winning with Quality, Addison-Wesley, Reading, MA, 1995
- [27] Wallace, Bruce (May 19, 2010). "A hole for every component, and every component in its hole". Existential Programming. "There is no such thing as a Component".
- [28] James Wilcox, "Paying Too Much for Custom Application Development", March 2011.
- [29] Ma, Y., Chen, G. & Thimm, G.; "Paradigm Shift: Unified and Associative Feature-based Concurrent Engineering and Collaborative Engineering", Journal of Intelligent Manufacturing, DOI 10.1007/s10845-008-0128-y
- [30] C. R. Kothari, "Research Methodology: Methods and Techniques", book, second edition, 2004.
- [31] "The NIST Definition of Cloud Computing". National Institute of Science and Technology. <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Retrieved 24 July 2011.
- [32] Cheng, and et. al., "Software Engineering for Self-Adaptive Systems", book, ISBN 978-3-642-02160-2.
- [33] Ceruzzi, Paul E. (2000). A History of Modern Computing. Cambridge, Mass.: MIT Press. ISBN 0-262-03255-4.
- [34] What is a Web Application? by Daniel Nations, former About.com Guide.
- [35] Mobile Application Development, <http://www.webmapsolutions.com/mobile-arcgis-paper-gps-data-collection>
- [36] Thomas Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", book, Kindle edition,
- [37] "About Aerospace Coordinate Systems". Retrieved 2007-11-23.
- [38] Eric PLATON' and Yuichi SEP, "Security software engineering in wireless sensor networks", National Institute of Informatics, The University of Tokyo, 2008.
- [39] Simon Denier, and et. al, , "Reverse Engineering the Literature on Design Patterns and Reverse-Engineering", PTIDEJ Team DIRO, Université de Montreal and DGIGL, Ecole Polytechnique de Montreal, 2008
- [40] William M. Mongan, and et al, "Re-engineering a Reverse Engineering Portal to a Distributed SOA", Drexel University Department of Computer Science, 3141 Chestnut Street, Philadelphia, PA 19104, fwmm24, ms333, spirosg@drexel.edu, 2008.
- [41] Samuel Mann Lesley Smith, "Biomimicry as a super systems metaphor for software engineering?", Otago Polytechnic Dunedin, NZ, smann, lsmith@tekotago.ac.nz, 2008.
- [42] MEHRDAD NOJOUIMIAN and TIMOTHY C. LETHBRIDGE, "Reengineering PDF-Based Documents Targeting Complex Software Specifications", University of Waterloo, Canada, University of Ottawa, Canada, 2008.
- [43] Betty H.C. Cheng, and et. al., "Software Engineering for Self-Adaptive Systems", Dagstuhl Seminar, 2008.
- [44] Alexandre Bergel, and et. al, "Enhanced Dependency Structure Matrix for Moose", Accepted to FamooSR, 2008
- [45] Craig Niiyama, and et. al, "Service-Oriented Software Reengineering Methodology for Composite Services", Computing & Software Systems, Institute of Technology, University of Washington, Tacoma, {cniiyama, chungsa, dchinn}@u.washington.edu, Milgard School of Business, University of Washington, Tacoma, sergiod@u.washington.edu, 2008
- [46] Brian R. von Konsky and Jim Ivins, "Assessing the Capability and Maturity of Capstone Software

- Engineering Projects", Australian Computer Society, Tenth Australasian Computing Education Conference (ACE2008), Wollongong, Australia, January 2008. Conferences in Research and Practice in Information Technology, Vol. 78. Simon and Margaret Hamilton Digital Ecosystems for Business Intelligence Institute Curtin (1)University of Technology GPO Box U1987 Perth, Western Australia 6845 B.vonKonsky@acsmail.net.au, 2008.
- [47] Steve M. Easterbrook and Timothy C. Johns, "Engineering the Software for Understanding Climate Change", Dept of Computer Science, U. of Toronto, Canada, sme@cs.toronto.edu and Met Office Hadley Centre, United Kingdom, tim.johns@metoffice.gov.uk, 2008
- [48] Jurgen Ebert Volker Riediger. and Andreas Winter., "Graph Technology in Reverse Engineering, The TGraph Approach", University of Koblenz-Landau, Institute for Software Technology, Universit'atsstr. 1. 56070 Koblenz, Germany ebert,riediger@uni-koblenz.de, and Johannes-Gutenberg-University Mainz, Institute for Computer Science, Staudingerweg 9. 55128 Mainz, Germany, winter@uni-mainz.de, 2008.
- [49] Vigdis By Kampenes and et. al, "Flexibility in Research Designs in Empirical Software Engineering", Department of Software Engineering, Simula Research Laboratory, P.O. Box 134, NO-1325 Lysaker, Norway, Department of Informatics, University of Oslo, P.O. Box 1080 Blindern, NO-0316 Oslo, Norway SINTEF ICT, NO-7465 Trondheim, Norway, 2008.
- [50] Kai Petersen, and et. Al., "Systematic Mapping Studies in Software Engineering", School of Engineering, Blekinge Institute of Technology, Box 520 SE-372 25 Ronneby. (kai.petersen | robert.feldt | shahid.muftaba | michael.mattsson)@bth.se, and Ericsson AB, Box 518. SE-371 23 Karlskrona. (kai.petersen | shahid.muftaba)@ericsson.com, 2008.
- [51] Lina Tankeleviciene, "using reengineering techniques for distance study course improvement", International Book Series "Information Science and Computing, The paper is selected from Third International Conference "Modern (e-) Learning" MeL 2008, Varna, Bulgaria, June-July 2008.
- [52] Daniel Halperin, and et. al, "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses", IEEE Symposium on Security and Privacy, 2008.
- [53] Niklaus Wirth, "A Brief History of Software Engineering", IEEE, 2008
- [54] Taehyung (George) Wang, and et. al., "assessing student learning in software engineering", Consortium for Computing Sciences in Colleges CCSC, 2008.
- [55] John Karn, "An Ethnographic Study of Conflict in Software Engineering Teams", Journal of Information, Information Technology, and Organizations, 2008.
- [56] Jannik Laval, and et. al., "Assessing the Quality of your Software with MoQam", Accepted to FamoosR, 2008.
- [57] Chris Jesshope and Alex Shafarenko,, "Concurrency Engineering", Institute for Informatics, University of Amsterdam and Department of Computer Science, University of Hertfordshire, 2008.
- [58] WH Morkel Theunissen, "Software Engineering Professionalism", paper, 2008.
- [59] Eelco Visser, "WebDSL: A Case Study in Domain-Specific Language Engineering", Software Engineering Research Group Delft University of Technology, 2008.
- [60] Supannika Koolmanojwong,, nd et. al., "Incremental Commitment Model Process Guidelines for Software Engineering Class", Center of Systems and Software Engineering University of Southern California Los Angeles, CA 90089-0781 {koolmano, aroonvat, icharoen}@usc.edu, 2008.
- [61] Karthik Viswanathan, "Right Engineering SaaS: Successfully deploying Software-as-a-Service Models", Aspire Systems - Transforming Product Development E-mail :info@aspiresys.com, Web: www.aspiresys.com, 2008.
- [62] Ujjal Marjit and Sibsankar Jana, "Mashups: An Emerging Content Aggregation Web2.0 Paradigm", 7th International CALIBER-2009, Pondicherry University, Puducherry, February 25-27, 2009, © INFLIBNET Centre, Ahmedabad
- [63] nurul naslia khairuddin and khairuddin hashim, "Application of Bloom's Taxonomy in Software Engineering Assessments", Proceedings of the 8th WSEAS International Conference on applied computer science (ACS'08), 2008.
- [64] S.K.Mishra, and et. al., "Creating Reusable Software Component from Object-Oriented Legacy System

- through Reverse Engineering", Journal of object technology, Online at <http://www.jot.fm>, 2009.
- [65] Christian Bartelt, and et. al., "Orchestration of Global Software Engineering Projects", Technische Universität Clausthal, Software Systems Engineering, Julius-Albert-Str. 4, 38678 Clausthal-Zellerfeld, Germany {bartelt | rausch}@tu-clausthal.de, 2009.
- [66] Alexandre Bergel, and et. al, "SQUALE – Software QUALity Enhancement", RMoD Team, INRIA, Lille, France, Qualixo, Paris, France, and Labo IA, University of Paris 8, France <http://www.squale.org>, 2009.
- [67] Jesper Andersson, "Software Engineering for Self-Adaptive Systems", Springer-Verlag Berlin Heidelberg, 2009
- [68] Christopher Poile, and et. al., "Coordination in Large-Scale Software Development: Helpful and Unhelpful Behaviors", Dept. of Management Sciences University of Waterloo Waterloo, ON, Canada cpoile@uwaterloo.ca and Microsoft Research Redmond, WA, [fandrew.begel, naching@microsoft.com](mailto:fandrew.begel@naching@microsoft.com). and Dept. of Computer Science, North Carolina State University, Raleigh, NC, lucas.layman@ncsu.edu, 2009.
- [69] Massimo Cossentino and et. al., "ASPECS: an Agent-oriented Software Process for Engineering Complex Systems", Multiagent Systems Group, System and Transport Laboratory, University of Technology of Belfort Montbéliard, 90010 Belfort cedex, France. e-mail: fmassimo.cossentino, nicolas.gaud, vincent.hilaire, stephane.galland, abder.koukamg@utbm.fr, <http://set.utbm.fr> and ICAR Institute, National Research Council, Palermo, Italy. <http://www.pa.icar.cnr.it/cossentino>, 2009.
- [70] Fernando Olivero, and et. al., "Lumi`ere: An Infrastructure for Producing 3D Applications in Smalltalk", FAMOOSr 2009 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.
- [71] Alberto Bacchelli, and et. al., "Miler – A Tool Infrastructure to Analyze Mailing Lists", FAMOOSr 2009 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.
- [72] Daniel Ratiu, "Challenges for Domain Knowledge Driven Program Analysis", FAMOOSr 2009, 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.
- [73] Fabrizio Perin, "Driving the refactoring of Java Enterprise Applications by evaluating the distance between application elements", FAMOOSr 2009, 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.
- [74] David Rothlisberger, "Why and How to Substantiate the Good of our Reverse Engineering Tools?", FAMOOSr 2009, 3rd Workshop on FAMIX and MOOSE in Software Reengineering, 2009.
- [75] Gordana Dodig-Crnkovic, Robert Feldt, "Professional and Ethical Issues of Software Engineering Curricula", Experiences from a Swedish Academic Context, 2009.
- [76] Mahmoud O. Elish, and et. al., "Application of TreeNet in Predicting Object-Oriented Software Maintainability: A Comparative Study", In Proceedings of the 13th IEEE European Conference on Software Maintenance and Reengineering (CSMR'09), pp. 69-77, Germany, Mar. 2009.
- [77] Robert Feldt, and et. al., "Generic Skills in Software Engineering Master Thesis Projects: Towards Rubric-Based Evaluation", 978-0-7695-3539-5/09 \$25.00 © 2009 IEEE DOI 10.1109/CSEET.2009.54, 2009.
- [78] Dag Sjöberg, "Guidelines for conducting and reporting case study research in software engineering", Empir Software Eng (2009) 14:131–164, DOI 10.1007/s10664-008-9102-8 this article is published with open access at Springerlink.com, 2009.
- [79] Joseph Chao and Jennifer Brown, "Cross-Departmental Collaboration for the Community: Technical Communicators in a Service-Learning Software Engineering Course", Issues in Informing Science and Information Technology, Volume 6, 2009.
- [80] Sita Ramakrishnan, "Innovation and Scaling up Agile Software Engineering Projects", Issues in Informing Science and Information Technology, Volume 6, 2009.
- [81] Mamta Garg, and et. Al., "Reverse Engineering – Roadmap to Effective software Design", International Journal of Recent Trends in Engineering, Vol. 1, No. 2, May 2009.
- [82] Engr. Farah Naaz Raza, "Artificial Intelligence Techniques in Software Engineering (AITSE)", Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I, IMECS 2009, March 18 - 20, 2009, Hong Kong.

- [83] Rajeev Misra, "Software Product Management", InfoPro Worldwide, Inc. 12 Perrine Road, Suite # 104, Monmouth Jn., NJ 08852, USA, Tel: 1 (609) 606 9020, Fax: 1 (609) 452 0335, Web site : www.infoproworldwide.com, 25th August, 2009
- [84] Nenad Stankovic and Tammam Tillo, "Concurrent Software Engineering Project", Journal of Information Technology Education: Innovations in Practice, Volume 8, 2009
- [85] Jannik Laval, and et. al., "Identifying cycle causes with Enriched Dependency Structural Matrix", Accepted to WCRE, 2009.
- [86] Radu Calinescu and Marta, "Software Engineering Techniques for the Development of Systems of Systems", Computing Laboratory, University of Oxford Wolfson Building, Parks Road, Oxford OX1 3QD, UK, 2009.
- [87] Nina Marwede, and et. al., "Automatic Failure Diagnosis Support in Distributed Large-Scale Software Systems based on Timing Behavior Anomaly Correlation ", 13th European, Conference on Software Maintenance and Reengineering (CSMR 2009), pages 47{57. IEEE}.
- [88] TERESA F. LUNT, and et. al., "The SeaView Security Model", IEEE transactions on software engineering, VOL. 16, NO. 6, 2009.
- [89] Stefan BIFFL and Alexander schatten, "A Platform for Service-Oriented Integration of Software Engineering Environments", Institute for Software Technology and Interactive Systems (ISIS) Technische Universität Wien, A-1040 Vienna, Austria, 2009.
- [90] David A Barnhart, and et. al., "SPIDR: Integrated Systems Engineering Design-to-Simulation Software for Satellite Build", 7th Annual Conference on Systems Engineering Research 2009 (CSER 2009) Loughborough University – 20th - 23rd April 2009.
- [91] Yajing Zhao, and et. al., "Ontology Classification for Semantic-Web-Based Software Engineering", IEEE transactions on services computing, VOL. 2, NO. 4, October - December 2009.
- [92] Jasmine K.S and Dr. R. Vasantha, "Reengineering Test Management for Increasing Testing Effectiveness in Component Based Enterprise Application", International Journal of Computer Theory and Engineering, Vol. 1, No. 4, October 2009, 1793-8201.
- [93] Vincent Schulte-Coerne, and et. al., "Automotive Software: Characteristics and Reengineering Challenges", Corporate Sector Research and Advance Engineering Software P.O. Box 30 02 40, 70442 Stuttgart, Germany, 2010.
- [94] Martin Wirsing and et. al., "SENSORIA Patterns: Augmenting Service Engineering with Formal Analysis, Transformation and Dynamicity", This work has been partially sponsored by the project SENSORIA, IST-2 005-016004, 2010.
- [95] Nicholas T. Pilkington, "ESIDE: An Integrated Development Environment for Component-Based Embedded Systems", Department of Computer Science, Portland State University Portland, OR 97207, USA, 2010.
- [96] asit kumar gahalaut and padmavati khandnor, " reverse engineering: an essence for software re-engineering and program analysis.", international journal of engineering science and technology, vol. 2(06), 2010, 2296-2303.
- [97] Bharat Chhabra, et. al., "Software Engineering Issues from the Cloud Application Perspective", International Journal of Information Technology and Knowledge Management, July-December 2010, Volume 2, No. 2, pp. 669-673.
- [98] J. P. McDermott, "Fine-Grained Inspection for Higher-Assurance Software Security in Open Source", Proceedings of the 43rd Hawaii International Conference on System Sciences - 2010.
- [99] Satwinder Singh, et. al., "Re-engineering To Analyze and Measure Object Oriented Paradigms", IEEE, 2010.
- [100] André Cristiano Kalsing, and et. al., "An Incremental Process Mining Approach to Extract Knowledge from, Legacy Systems ", 2010 14th IEEE International Enterprise Distributed Object Computing Conference.
- [101] Nicolas Anquetil, and et. al., "Legacy Software Restructuring: Analyzing a Concrete Case", INRIA Lille-Nord-Europe, France, 2010.
- [102] Grace Lewis, " Basics About Cloud Computing", Software Engineering Institute, Carnegie Mellon University, 4500 Fifth Avenue, Pittsburgh, PA 15213-2612, Phone: 412-268-5800, Toll-free: 1-888-201-4479, www.sei.cmu.edu, 2010.
- [103] Todd W. Schiller and Michael D. Ernst, "Rethinking the Economics of Software Engineering", FoSER

- 2010, November 7–8, 2010, Santa Fe, New Mexico, USA.
Copyright 2010 ACM 978-1-4503-0427-6/10/11.
- [104] Ahmed E. Hassan and Tao Xie, "Software Intelligence: The Future of Mining Software Engineering Data", School of Computing, Queen's University, Kingston, ON, Canada, Department of Computer Science, North Carolina State University, Raleigh, NC, USA, 2010.
- [105] V. Carchiolo, D72and et. al., "Object-oriented re-engineering of manufacturing models: A case study", Springer Science + Business Media, LLC, 2010.
- [106] Manar Abu Talib, and et. al., "Secure Software Engineering: A New Teaching Perspective Based on the SWEBOK", Interdisciplinary Journal of Information, Knowledge, and Management Volume 5, 2010.
- [107] Philippe Dugerdil, and et. al., "Automatic Generation of Abstract Views for Legacy Software Comprehension", ISEC '10. February 25-27. 2010. Mysore. India.
- [108] Claude Y. Laporte, and et. al., "A Software Engineering Lifecycle Standard for Very Small Enterprises", École de technologie supérieure, Montréal, Canada, Centre d'Excellence en Technologies de l'Information et de la Communication, Charleroi, Belgium, and School of Computing, Dublin City University, Dublin, Ireland, 2010.
- [109] Jannik Laval, and et. al., "Supporting Simultaneous Versions for Software Evolution Assessment", Elsevier, 2010.
- [110] Ira Monarch, and et. al., "The Role of Informatics in Software Engineering: Literature Reviews, Agenda and Software Informatics", Software Engineering Institute 4500 Fifth Avenue, 2010.
- [111] Victor Pankratius, "Software Engineering in the Era of Parallelism", Multicore Software Engineering Young Investigator Group, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany, 2010.
- [112] Mark Harman, "The Relationship between Search Based Software Engineering and Predictive Modeling", University College London, Department of Computer Science, CREST Centre, Malet Place, London, WC1E 6BT, UK., 2010.
- [113] Bruce Powel Douglass and Mats Gothe, "IBM Rational Workbench for Systems and Software Engineering", Copyright IBM Corp. 2010. All rights reserved. ibm.com/redbooks
- [114] A. Begel and N. Nagappan, "Pair Programming", Microsoft Research, published in ESEM 2008.
- [115] Anthony I. Wasserman, and et. al., "Software Engineering Issues for Mobile Application Development", FoSER 2010, Santa Fe, New Mexico, USA. Copyright 2010 ACM 978-1-4503-0427-6/10/11.
- [116] Matthias Galster, "Life-cycle Assessment in Software Engineering", mgalster@ucalgary.ca, 2010.
- [117] N.Sasirekha and et. al., "program slicing techniques and its applications", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, 2011.
- [118] Csaba Nagy, and et. al., "Solutions for Reverse Engineering 4GL Applications, Recovering the Design of a Logistical Wholesale System", 2011 15th European Conference on Software Maintenance and Reengineering, IEEE computer Society.
- [119] Jörg Barthold and et. al., "Re-Engineering of a Hierarchical Product Line", 2011 15th International Software Product Line Conference, IEEE computer Society.
- [120] Jorge J. Gomez-Sanz, and et. al., "Understanding Agent Oriented Software Engineering Methodologies", GRASIA Research Group, Universidad Complutense de Madrid, Avda. Complutense, 28040 Madrid, Spain, 2011.
- [121] Nishant Dubey, "Paper Presentation on Software Development Automation by Computer Aided Software Engineering (CASE)", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011, ISSN (Online): 1694-0814, www.IJCSI.org
- [122] Karine Mordal-Manet, and et. al., "An empirical model for continuous and weighted metric aggregation", LIASD, University of Paris 8, France, RMoD Team, INRIA, Lille, France, Qualixo, Paris, France, Air France-KLM, France, and McCabe Software, US
- [123] Ashok Kumar Behera, and et. al., "Analysis and Design of Software Visualization Tool for the Behavior of Object Oriented Programming",

- International Journal of Computer Applications (0975 – 8887) Volume 13– No.1, January 2011
- [124] Jannik Laval and et. al., "Cycles Assessment with CycleTable", RMoD Team INRIA - Lille Nord Europe - USTL - CNRS UMR 8022 Lille, France, 2011.
- [125] Umesh Banodha and Kanak Saxena, "A Software Architecture Style for Medical Process Re-engineering", Proceedings of the World Congress on Engineering and Computer Science 2011 Vol I, WCECS 2011, October 19-21, 2011, San Francisco, USA
- [126] Fernando Almeida and et. al., "open standards and open source: enabling interoperability", international Journal of Software Engineering & Applications (IJSEA), Vol.2, No.1, January 2011.
- [127] Zuhail Tanrikulu and Tuna Ozce, "standardization of information systems development processes and banking industry adaptations", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.2, April 2011
- [128] Bradley Deken and Chris Cowen, "Development of Computer Aided Learning Software for Use in Electric Circuit Analysis", International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies, 2011.
- [129] Rahul Thakurta, and Subhajit Dasgupta, "impact of software requirement volatility pattern on project dynamics: evidences from a case study", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011
- [130] Hollie Boudreaux and et. al., "an algorithmic and software engineering based approach to robust video game design", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011, DOI : 10.5121/ijsea.2011.2303
- [131] Kardile Vilas Vasantrao, "Understand Need "Uncertainty Analysis" in the System Development Modeling Process", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011, DOI: 10.5121/ijsea.2011.2307.
- [132] Md. Anawarul Kabir and et. al., "unified software development model for free/open source software", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.3, July 2011, DOI :10.5121/ijsea.2011.2308.
- [133] Sk.MD.Rafi and Shaheda Akthar, "Discrete Software Reliability Growth Models with Discrete Test Effort Functions", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011, DOI : 10.5121/ijsea.2011.2402.
- [134] Pritha Guhaand, and et. al., "incorporating agile with mda case study: online polling system", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011, DOI : 10.5121/ijsea.2011.2408.
- [135] S. Arun Kumar and T.Arun Kumar, "study the impact of requirements management characteristics in global software development projects: an ontology based approach", International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011, DOI : 10.5121/ijsea.2011.2410.
- [136] Mahmoud Khraiweh, "Validation Measures in CMMI", World of Computer Science and Information Technology Journal (WCSIT), ISSN: 2221-0741, Vol. 1, No. 2, 26-33, 2011
- [137] Bimal Aklesh Kumar, "software architecture for fiji national university campus information systems", Bimal Aklesh Kumar / Indian Journal of Computer Science and Engineering (IJCSE), ISSN : 0976-5166 Vol. 2 No. 2 Apr-May 2011.
- [138] Praveen Ranjan Srivastava and Km Baby, "Automated Software Testing Using Metahuristic Technique Based on An Ant Colony Optimization", This paper appears in: Electronic System Design (ISED), 2010, International Symposium on, Issue Date: 20-22 Dec. 2010, On page(s): 235 - 240, Location: Bhubaneswar, Print ISBN: 978-1-4244-8979-4, INSPEC Accession Number: 11835766, Digital Object Identifier: 10.1109/ISED.2010.52.
- [139] Ardhendu Mandal, "BRIDGE: A Model for Modern Software Development Process to Cater the Present Software Crisis", 2009 IEEE International Advance Computing Conference (IACC 2009), Patiala, India, 6–7March 2009.
- [140] Ardhendu Mandal, "srs builder 1.0: An Upper Type CASE Tool for Requirement Specification.", Department Of Computer Science and Application, University of North Bengal (N.B.U.), INDIA, 2011.
- [141] Kardile Vilas Vasantrao, "enhance accuracy in software cost and schedule estimation by using "uncertainty analysis and assessment" in the system modeling process", International Journal of Research & Innovation in Computer Engineering, Vol 1, Issue

1, (6-18), August' 2011, www.ijrice.com ©
Innovation Science Publications.

NO. 1, FEBRUARY 2011, 0018-9359/\$26.00 ©
2010 IEEE

- [142] Badr Elmir, and et. al., "A Novel Approach for Periodic Assessment of Business Process Interoperability", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011, ISSN (Online): 1694-0814, www.IJCSI.org.
- [143] Carlos Alberto Fernández and Martín José, "Towards the integration of formal specification in the Áncora methodology", Instituto de Computación, Universidad Tecnológica de la Mixteca, carretera a Acatlima Km. 2.5 Huajuapán de León, Oax., México C.P. 69000, and División de Estudios de Posgrado, Universidad Tecnológica de la Mixteca, carretera a Acatlima Km. 2.5 Huajuapán de León, Oax., México C.P. 69000,
- [144] Asif Irshad Khan and et. al., "A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 2, July 2011, ISSN (Online): 1694-0814, www.IJCSI.org.
- [145] S.Malathi and Dr.S.Sridhar, "A Classical Fuzzy Approach for Software Effort Estimation on Machine Learning Technique", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, November 2011. ISSN (Online): 1694-0814, www.IJCSI.org, 2011.
- [146] Phuc V. Nguyen, "the study and approach of software re-engineering", Department of Cipher & Information Technology, Ho Chi Minh City's Vietnamese Communist Party Committee, Ho Chi Minh City, Vietnam, 2011
- [147] Aminat Showole, and et. al., "Layered Approach to Open Source Software Development Success", IBIMA Publishing Communications of the IBIMA, <http://www.ibimapublishing.com/journals/CIBIMA/cibima.html>, Vol. 2011 (2011), Article ID 160480, 22 pages, DOI: 10.5171/2011.160480
- [148] Yasushi Sato,, "An Inconspicuous Giant: NTT's Role in the Development of Software Engineering in Japan", IEEE Annals of the History of Computing, Published by the IEEE Computer Society, 1058-6180/11/\$26.00 @ 2011 IEEE
- [149] Smriti Srivastava, and et. al., "A Laboratory Testbed for Embedded Fuzzy Control", IEEE TRANSACTIONS ON EDUCATION, VOL. 54,
- [150] Xavier Amatriain, "Frameworks Generate Domain-Specific Languages: A Case Study in the Multimedia Domain", IEEE transactions on software engineering, VOL. 37, NO. 4, JULY/AUGUST 2011, 0098-5589/11/\$26.00 @2011 IEEE, Published by the IEEE Computer Society
- [151] Pratyusa K., and et. al., "An Attack Surface Metric", IEEE transactions on software engineering, VOL. 37, NO. 3, MAY/JUNE 2011, 0098-5589/11/\$26.00@2011 IEEE, Published by the IEEE Computer Society
- [152] Vladan Devedžić and Saša R. Milenković, "Teaching Agile Software Development: A Case Study", IEEE TRANSACTIONS ON EDUCATION, VOL. 54, NO. 2, MAY 2011.
- [153] Yue Jia, and Mark Harman, "An Analysis and Survey of the Development of Mutation Testing", IEEE transactions on software engineering, VOL. 37, NO. 5, SEPTEMBER/OCTOBER 2011, 0098-5589/11/\$26.00 2011 IEEE, Published by the IEEE Computer Society
- [154] Moonzoo Kim and Hotae Kim, "A Comparative Study of Software Model Checkers as Unit Testing Tools: An Industrial Case Study", IEEE transactions on software engineering, VOL. 37, NO. 2, MARCH/APRIL 2011, 0098-5589/11/\$26.00 @2011 IEEE, Published by the IEEE Computer Society
- [155] Magne Jørgensen and Stein Grimstad, "The Impact of Irrelevant and Misleading Information on Software Development Effort Estimates: A Randomized Controlled Field Experiment", IEEE transactions on software engineering, VOL. 37, NO. 5, SEPTEMBER/OCTOBER 2011, 0098-5589/11/\$26.00 @2011 IEEE, Published by the IEEE Computer Society
- [156] Yonghee Shin and et. al., "Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities", IEEE transactions on software engineering, VOL. 37, NO. 6, NOVEMBER/DECEMBER 2011, 0098-5589/11/\$26.00 @2011 IEEE, Published by the IEEE Computer Society
- [157] Ivica Crnković and Michel R.V., "A Classification Framework for Software Component Models", IEEE transactions on software engineering, VOL. 37, NO. 5, SEPTEMBER/OCTOBER 2011, 0098-

5589/11/\$26.00 @ 2011 IEEE, Published by the IEEE Computer Society

- [158] Andrea De Lucia, and et. al., "Improving Source Code Lexicon via Traceability and Information Retrieval", IEEE transactions on software engineering, VOL. 37, NO. 2, MARCH/APRIL 2011, 0098-5589/11/\$26.00 @ 2011 IEEE, Published by the IEEE Computer Society
- [159] Radu Calinescu, and et. al., "Dynamic QoS Management and Optimization in Service-Based Systems", IEEE Transactions on software engineering, VOL. 37, NO. 3, MAY/JUNE 2011, 0098-5589/11/\$26.00 @ 2011 IEEE, Published by the IEEE Computer Society
- [160] W. Ke and P. Zhang, "Effects of Empowerment on Performance in Open-Source Software Projects", IEEE transactions on engineering management, VOL. 58, NO. 2, MAY 2011
- [161] P. K. Kapur and et. al., "A Unified Approach for Developing Software Reliability Growth Models in the Presence of Imperfect Debugging and Error Generation", IEEE TRANSACTIONS ON RELIABILITY, VOL. 60, NO. 1, MARCH 2011
- [162] Cleidson R.B. de Souza and David F. Redmiles, "The Awareness Network, To Whom Should I Display My Actions? And, Whose Actions Should I Monitor?", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 37, NO. 3, MAY/JUNE 2011
- [163] Anandasivam Gopal, and et. al., "Coordination and Performance in Global Software Service Delivery: The Vendor's Perspective", IEEE transactions on engineering management, VOL. 58, NO. 4, NOVEMBER 2011.
- [164] Kavita Choudhary, "Parametric Estimation of Software Systems", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-1, Issue-2, May 2011
- [165] Mazni Omar and et. al., "The Impact of Agile Approach on Software Engineering Teams", American Journal of Economics and Business Administration 3 (1): 12-17, 2011, ISSN 1945-5488 , 2010 Science Publications.
- [166] Martin Nordio and et. al., "Collaborative Software Development on the Web", ETH Zurich, Switzerland, 2011.
- [167] Bruno Antunes and et. al., "Context Capture in Software Development", Knowledge and Intelligent Systems Laboratory Cognitive and Media Systems Group, Centre for Informatics and Systems of the University of Coimbra, Coimbra, Portugal, 2011.
- [168] Damien Cassou Emilie Balland Charles Consel and et. al., "Leveraging Software Architectures to Guide and Verify the Development of Sense/Compute/Control Applications", ICSE '11, May 21–28, 2011, Waikiki, Honolulu, HI, USA.
- [169] Eleni Constantinou and et. al., "Open Source Software: How Can Design Metrics Facilitate Architecture Recovery?", Computer Science Department, Aristotle University of Thessaloniki, Thessaloniki, Greece, econst@csd.auth.gr, stamelos@csd.auth.gr, and Computer Science Department, Aristotle University of Thessaloniki, Thessaloniki, Greece, and Computer Science and Telecom. Department T.E.I. of Larisa, Greece, gkakaron@teilar.g
- [170] Maurice H. ter Beek and et. al., "Product Lines for Service Oriented Applications - PL for SOA", Istituto di Scienza e Tecnologie dell'Informazione, ISTI-CNR, Pisa, Italy, and IMT Institute for Advanced Studies Lucca, Lucca, Italy, 2011.
- [171] Robert Da browski and et. al., "Software is a directed multigraph (and so is software process)", Institute of Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, Poland, 2011.
- [172] P. A. Judas and L. E. Prokop, "A historical compilation of software metrics with applicability to NASA's Orion spacecraft flight software sizing", NASA-Lyndon B. Johnson Space Center, 2101 NASA Parkway, 2011.
- [173] Laurie McLeod and et. al., "Qualitative research on software development: a longitudinal case study methodology", Springer Science+Business Media, LLC 2011.
- [174] Steve Adolph and et. al., "Using grounded theory to study the experience of software development", Springer Science+Business Media, LLC 2011.
- [175] Jana Samalikova and et. al., "Toward objective software process information: experiences from a case study", The Author(s) 2010. This article is published with open access at Springerlink.com
- [176] Faheem Ahmed and Luiz Fernando Capretz, "A business maturity model of software product line engineering", Springer Science+Business Media, LLC 2010.

- [177] T. A. W. Jarratt, and et. al., "Engineering change: an overview and perspective on the literature", Springer-Verlag London Limited 2010.
- [178] Natalia Juristo and Sira Vegas, "The role of non-exact replications in software engineering experiments", Springer Science+Business Media, LLC 2010.
- [179] Anne Martens, and et. al., "From monolithic to component-based performance evaluation of software architectures", Springer Science+Business Media, LLC 2010
- [180] Tomaz Dogs̃a and et. al., "The effectiveness of test-driven development: an industrial case study", Springer Science+Business Media, LLC 2011.
- [181] Marko Rosenmüller, and et. al., "Flexible feature binding in software product lines", Springer Science+Business Media, LLC 2011.
- [182] Yao-Jen Chang and et. al., "Student Engineers as Agents of Change: Combining Social Inclusion in the Professional Development of Electrical and Computer Engineering Students", Springer Science+Business Media, LLC 2010
- [183] Andy Zaidman and et. al., "Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining", This article is published with open access at Springerlink.com, 2010.
- [184] Jeronimo Castrillon and et. al., "Component-based waveform development: the Nucleus tool flow for efficient and portable software defined radio", Springer Science+Business Media, LLC 2011
- [185] Cuauhtemoc Lopez-Martin, "Applying a general regression neural network for predicting development effort of short-scale programs", Springer-Verlag London Limited 2010
- [186] Stephen Gilmore and et. al., "Non-functional properties in the model-driven development of service-oriented systems", Springer-Verlag 2010.
- [187] Zude Li and et. al., "Characteristics of multiple-component defects and architectural hotspots: a large system case study", Springer Science+Business Media, LLC 2011.
- [188] Alistair Sutcliffe and et. al., "Experience with user-centred requirements engineering", Springer-Verlag London Limited 2011.
- [189] Eran Rubin and Hillel Rubin, "Supporting agile software development through active documentation", Springer-Verlag London Limited 2010
- [190] Kristina Pitula and Thiruvengadam Radhakrishnan, "On eliciting requirements from end-users in the ICT4D domain", Springer-Verlag London Limited 2011.
- [191] Steve Bannerman and Andrew Martin, "A multiple comparative study of test-with development product changes and their effects on team speed and product quality", Springer Science+Business Media, LLC 2010.
- [192] Faheem Ahmed and Luiz Fernando , "An architecture process maturity model of software product line engineering", Springer-Verlag London Limited 2011
- [193] Jitendra S. Kushwah and Mahendra S. Yadav, "testing for object oriented software", Indian Journal of Computer Science and Engineering (IJCSE), 2011.
- [194] B.Bharathi and G.Kulanthaivel, "towards developing a performance evaluator for component based software architectures", Indian Journal of Computer Science and Engineering (IJCSE), 2011.
- [195] Bimal Aklesh Kumar, "software architecture for fiji national university campus information systems", Indian Journal of Computer Science and Engineering (IJCSE), 2011.
- [196] Arti Chhikara and R.S.Chhillar, "Impact of Aspect Orientation on Object Oriented Software Metrics", Indian Journal of Computer Science and Engineering (IJCSE), 2011.
- [197] Arashdeep Kaur and Sunil Gulati, "A Framework for Analyzing Software Quality using Hierarchical Clustering", International Journal on Computer Science and Engineering (IJCSE), 2011
- [198] Ghazal Keshavarz and et. al., "Metric for Early Measurement of Software Complexity", International Journal on Computer Science and Engineering (IJCSE), 2011.
- [199] Sanjana and Dr. P.K. Suri, "Simulator for Software Project Reliability Estimation", International Journal on Computer Science and Engineering (IJCSE), 2011.
- [200] Nishant Dubey, "A Paper Presentation on Software Development Automation by Computer Aided Software Engineering (CASE)", IJCSI International

- Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011.
- [201] Osama Shata, "Teaching Software Engineering: Problems and Suggestions", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011.
- [202] Rashmi Yadav, and et. al., "Software Engineering for Practiced Software Enhancement", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.
- [203] Dr. Satya Prasad Ravi, and et. al., "SPC for Software Reliability: Imperfect Software Debugging Model", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011
- [204] Rajender Singh Chhillar and Parveen Kajla, "A New - Knot Model for Component Based Software Development", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011, ISSN (Online): 1694-0814, www.IJCSI.org
- [205] Narender Singh and Nasib Singh Gill, "Aspect-Oriented Requirements Engineering for Advanced Separation of Concerns: A Review", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 2, September 2011, ISSN (Online): 1694-0814, www.IJCSI.org
- [206] Nor Azliana Akmal Jamaludin and Shamsul Sahibuddin, "Development Strategy using Cognitive Domain in e-Requirement Engineering Learning System", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 2, September 2011, ISSN (Online): 1694-0814, www.IJCSI.org
- [207] Mohammed Abdul Bari and Dr. Shahanawaj Ahamad, "Process of Reverse Engineering of Enterprise Information System Architecture", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 3, September 2011 ISSN (Online): 1694-0814 www.IJCSI.org, 2011.
- [208] Ashish B. Sasankar and Dr Vinay Chavan, "SWOT Analysis of Software Development Process Models", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 5, No 3, September 2011, ISSN (Online): 1694-0814, www.IJCSI.org
- [209] R. Jeyakarthish, "Requirements Engineering In Current Web Engineering Methodologies", R.Jeyakarthish, Int. J. Comp. Tech. Appl., Vol 2 (3), 490-497, 2011.
- [210] Rituraj Jain, "Improvement in Software Development Process and Software Product through Knowledge Management", Rituraj Jain et al, Int. J. Comp. Tech. Appl., Vol 2 (5), 1557-1562, 2011.
- [211] Sheikh Umar Farooq, and et. al., "software measurements and metrics: role in effective software testing", Sheikh Umar Farooq et al. / International Journal of Engineering Science and Technology (IJEST), 2011.
- [212] Dr.S.S.Riaz Ahamed, "integrated approach and strategic case issues of project engineering: a comprehensive study", International Journal of Engineering Science and Technology (IJEST), 2011.
- [213] Dr.S.S.Riaz Ahamed , "the influence of scope and timing of quality assurance in the planned and systematic pattern of software development process: a novel study", International Journal of Engineering Science and Technology (IJEST), 2011.
- [214] OLGA Fedotova, and et. al., "software effort prediction: an empirical evaluation of methods to treat missing values with rapid miner", International Journal of Engineering Science and Technology (IJEST), 2011.
- [215] R.Amuthakkannan, "Effective Software Assembly for the Real time systems using Multi-level Genetic Algorithm", International Journal of Engineering Science and Technology (IJEST), 2011.
- [216] DR. NACHAMAI and et. al., "enacted software development process based on agile and agent methodologies", International Journal of Engineering Science and Technology (IJEST), 2011.
- [217] Jyothi R and Dr. V. K. Agrawal, "process quality analysis of perfective maintainability for component-based software systems using aspect-oriented programming techniques", International Journal of Engineering Science and Technology (IJEST), 2011.
- [218] Suchetha Vijay and Annapoorna Shetty, "A study on different approaches towards Aspect Oriented Requirements Engineering", Indian Journal of Computer Science and Engineering (IJCSE), Aug - Sep 2011.
- [219] Ashok Kumar Behera and Rajesh Tiwari, "Analysis and Design of Software Visualization Tool for the Behavior of Object Oriented Programming", International Journal of Computer Applications (0975 – 8887), 2011.

- [220] Dharendra Pandey, and et. al., "Security Requirement Engineering Issues in Risk Management", International Journal of Computer Applications (0975 – 8887), 2011.
- [221] R.Chinnaiyan, and et. al., "An Experimental Study on Reliability Estimation of GNU Compiler Components – A Review", International Journal of Computer Applications (0975 – 8887), 2011.
- [222] M. Aqeel Iqbal, and et. al., "Critical Aspects of Project Management Causing Major Impacts on Software Development", International Journal of Computer Applications (0975 – 8887), 2011.
- [223] Bandla Srinivasa Rao and et. al., "Monitoring Software Reliability using Statistical Process Control: An Ordered Statistics Approach", International Journal of Computer Applications (0975 – 8887), 2011.
- [224] Niharika Aggarwal and Aditya Garg, "Using Re-Usable, Secure Software Engineering Principles for Designing User Focused Mixed Reality Systems", International Journal of Computer Applications (0975 – 8887), 2011.
- [225] Aurangzeb Khan, and et. al., "Managing Requirement Risks in Global Software Development", International Journal of Computer Applications (0975 – 8887), 2011.
- [226] Elizabeth Bjarnason, and et. al., "Requirements Are Slipping Through the Gaps - A Case Study on Causes & Effects of Communication Gaps in Large-Scale Software Development", 2011 IEEE 19th International Requirements Engineering Conference, 2011.
- [227] Francesca Arcelli Fontana and Marco Zanoni, "A tool for design pattern detection and software architecture reconstruction", Elsevier, 2011.
- [228] Krzysztof Wnuk and et. al., "Scaling Up Requirements Engineering –Exploring the Challenges of Increasing Size and Complexity in Market- Driven Software Development", Department of Computer Science, Lund University, Sweden, and Siemens Corporate Research, Inc, USA, 2011.
- [229] Manik Sharma and et. al., "A Comparative Study of Static Object Oriented Metrics", International Journal of Advancements in Technology, <http://ijict.org/> , ISSN 0976-4860, 2012.
- [230] Jordan Vrtanoski and Toni Stojanovski, "Order Handling in Convergent Environments", 2011 2nd International Conference on e-Education, e-Business, e-Management and E-Learning (IC4E 2011)
- [231] Toni Stojanovski PhD and Tomislav Dzekov PhD, "Rapid Application Development Using Software Factories", Faculty of Informatics, European University, Republic of Macedonia, 2012.
- [232] Fatima-Zahra Belouadha and et. al., "A MDA approach for defining WS-Policy semantic non-functional properties", International Journal of Engineering Science and Technology, Vol. 2(6), 2010, 2164-2171
- [233] Faudziah Ahmad and et. al., "investigating the awareness of applying the important web application development and measurement practices in small software firms", International Journal of Computer Science & Information Technology (IJCSIT) Vol 3, No 6, Dec 2011.
- [234] Ch Ram Mohan Reddy and et. al., "General Methodology for developing UML models from UI", International Journal on Web Service Computing (IJWSC), Vol.2, No.4, December 2011.
- [235] Chintan Amrit, and et. al., "Identifying Coordination Problems in Software Development: Finding Mismatches between Software and Project Team Structures", 2012.
- [236] Rizwan Jameel Qureshi, "requirements and the baseline plan", Department of Computer Science, COMSATS Institute of Information Technology, Defence Road, Lahore, 2012.
- [237] Ananda Rao and M.Gopichand, "Four Layered Approach to Non-Functional Requirements Analysis", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 2, November 2011, ISSN (Online): 1694-0814, www.IJCSI.org.
- [238] Ravi S. Iyer, "Improve the Practice of Software Development by Having a Software Development Career Track in CS & IT Academia", Software Consultant, Puttaparthi, Andhra Pradesh, India, 2012.
- [239] Murtaza Hussain Shaikh, "collaboration for enhancing the system development process in open source diligence.", Advanced Computing: An International Journal (ACIJ), Vol.3, No.1, January 2012.
- [240] Youssef Bassil, "building sustainable ecosystem-oriented architectures", International Journal in Foundations of Computer Science & Technology, Vol. 2, No.1, January 2012.

- [241] M. Rizwan Jameel Qureshi and M. E. Sandhu, "Survey-Based Analysis of the Proposed Component-Based Development Process", Dept. of Computer Science, COMSATS Institute of Information Technology, Lahore. National College of Business Administration & Economics 40 E/I Gulberg III Lahore, Pakistan, 2012.
- [242] M. R. J. Qureshi, "An Evaluation of the Improved XP Software Development Process Model", Dept. of Computer Science, COMSATS Institute of Information Technology Lahore Pakistan Defence Road, Off Raiwind Road Lahore Pakistan
- [243] M. R. J. Qureshi and S.A. Hussain, "An Improved XP Software Development Process Model", Dept. of Computer Science, COMSATS Institute of Information Technology Lahore Pakistan, 2008.
- [244] Nayyar Iqbal and M. Rizwan Jameel Qureshi, "Improvement of Key Problems of Software Testing in Quality Assurance", Department of Computer Science, COMSATS Institute of Information Technology, Lahore Pakistan, 2009.
- [245] M. Rizwan Jameel Qureshi and M. E. Sandhu, "A Validation of the Proposed Component-Based Development Process", Dept. of Computer Science, COMSATS Institute of Information Technology, Lahore. National College of Business Administration & Economics 40 E/I Gulberg III Lahore, Pakistan, 2009.
- [246] M. Rizwan Jameel Qureshia and et. al., "A new teaching model for the subject of software project management", Department of Computer Science, COMSATS Institute of Information Technology M. A. Jinnah Campus, Defence Road off Raiwind Road, Lahore Pakistan, 2012.
- [247] M. Nadeem, and et. al., "A step forward to component-based software cost estimation in object-oriented environment", Pakistan Journal of Science (Vol. 62 No. 4 December, 2010)
- [248] M. Rizwan Jameel Quresh, "Empirical Evaluation of the Proposed eXSCRUM Model: Results of a Case Study", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011, ISSN (Online): 1694-0814, www.IJCSI.org.
- [249] Asif Irshad Khan and et. al., "A Comprehensive Study of Commonly Practiced Heavy & Light Weight Software Methodologies", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 4, No 1, July 2011, ISSN (Online): 1694-0814, www.IJCSI.org.
- [250] Ahmed Barnawi, and et. al., "A Framework for Next Generation Mobile and Wireless Networks Application Development using Hybrid Component Based Development Model", International Journal of Research and Reviews in Next Generation Networks (IJRRNGN), Vol. 1, No. 2, December 2011, ISSN: 2046-6897, © Science Academy Publisher, United Kingdom, www.sciacademypublisher.com
- [251] Ahmed Barnawi, and et. al., "novel component-based development model for sip-based mobile application", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [252] Lorina Negreanu, and et. al., "A Formal Approach for the Development of Service-Oriented Applications", Computer Science Department, "Politehnica" University of Bucharest Bucharest, Romania, 2012.
- [253] Rashmi Gupta and Shalini Raghav, "Risk Assessment Techniques and Survey Method for COTS Components", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [254] Toni Stojanovski, and et. al., "Scalability of Data Binding in ASP.NET Web Applications", Faculty of Informatics, European University, Bld. Kliment Ohridski 68, 1000 Skopje, MACEDONIA, 2012
- [255] Isabella and Emi Retna, "study paper on test case generation for gui based testing", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [256] Dr. Reena Dadhich and Ujala Chauhan, "integrating cmmi maturity level-3 in traditional software development process", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [257] Sandhya Tarar and Dr. Ela Kumar, "Design Paradigm and Risk Assessment of Hybrid Re-engineering with an approach for development of Re-engineering Metrics ", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [258] V.Dattatreya and et. al., "agile programming and design patterns in web development -a case study",

- International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [259] Vikash Lalsing, and et. al., "people factors in agile software development and project management", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012
- [260] Isabella and Emi Retna, "study paper on test case generation for gui based testing", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012
- [261] S. Babu and Dr.R.M.S. Parvathi., "Development of Dynamic Coupling Measurement of Distributed Object Oriented Software Based on Trace Events",
- [262] B.Jalender, and et. al., "designing code level reusable software components", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012
- [263] Mar'ia-Cruz Valiente, and et. al., "Applying Ontology-Based Models for Supporting Integrated Software Development and IT Service Management Processes", IEEE transactions on systems, man, and cybernetics—part c: applications and reviews, VOL. 42, NO. 1, January 2012.
- [264] Benjamin Heitmann and et. al., "An Empirically Grounded Conceptual Architecture for Applications on the Web of Data", IEEE transactions on systems, man, and cybernetics—part c: applications and reviews, VOL. 42, NO. 1, January 2012 51.
- [265] Holger M. Kienle, "Open Data: Reverse Engineering and Maintenance Perspective", arXiv:1202.1656v1 [cs.SE] 8 Feb 2012.
- [266] Drazen Lucanin and Ivan Fabek, "A Visual Programming Language for Drawing and Executing Flowcharts", arXiv:1202.2284v1 [cs.PL] 10 Feb 2012.
- [267] Geeta Nagpal and et. al., "A Hybrid Technique using Grey Relational Analysis and Regression for Software Effort Estimation using Feature Selection", International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-1, Issue-6, January 2012.
- [268] Brendan Murphy, "The difficulties of building generic reliability models for software", Springer Science+Business Media, LLC 2011.
- [269] Jesús M. González-Barahona: and Gregorio Robles, "On the reproducibility of empirical software engineering studies based on data retrieved from development repositories", This article is published with open access at Springerlink.com, 2011.
- [270] Ernst Sikora and et. al., "Industry needs and research directions in requirements engineering for embedded systems", Springer-Verlag London Limited 2011.
- [271] Martin Ivarsson and Tony Gorschek, "Tool support for disseminating and improving development practices", Springer Science+Business Media, LLC 2011.
- [272] Dirk Riehle, "The single-vendor commercial open course business model", Springer-Verlag 2010.
- [273] Marsha Chechik and et. al., "A relationship-based approach to model integration", Springer-Verlag London Limited 2011.
- [274] Irene Luque Ruiz and Miguel Ángel Gómez-Nieto, "Rolling: A new technique for the practical teaching in computer science university degree", Springer Science+Business Media, LLC 2010.
- [275] Yosi Ben Asher and et. al., "Refactoring techniques for aggressive object inlining in Java applications", Springer Science+Business Media, LLC 2011.
- [276] Raphael Mannadiar and Hans Vangheluwe: , "Modular artifact synthesis from domain-specific models", Springer-Verlag London Limited 2011.
- [277] Veerapaneni Esther Jyothi and K. Nageswara Rao, "Effective Implementation of Agile Practices – Incoordination with Lean Kanban", International Journal on Computer Science and Engineering (IJCSSE), 2012.
- [278] Abhiraja Sharma, and et. al., "Defect Prevention Technique in Test Case of Software Process for Quality Improvement", Int.J.Comp.Tech.Appl, Vol 3 (1), 56-61, IJCTA | JAN-FEB 2012, Available online@www.ijcta.com, ISSN:2229-6093.
- [279] Lata Nautiyal, and et. al., "Elite: A New Component-Based Software Development Model", Int.J.Computer Technology & Applications, Vol 3 (1),119-124, IJCTA | JAN-FEB 2012, Available online@www.ijcta.com, ISSN:2229-6093.
- [280] Mrinal Singh Rawat and et. al., "Survey on Impact of Software Metrics on Software Quality", (IJACSA) International Journal of Advanced Computer Science

- and Applications, Vol. 3, No. 1, 2012, www.ijacsa.thesai.org.
- [281] M.R. Dube, S.K. Dixit: , "Modeling Theories and Model Transformation Scenario for Complex System Development", International Journal of Computer Applications (0975 – 8887) Volume 38– No.7, January 2012.
- [282] Hongli Liu, and et. al., "An object-oriented serial implementation of a DSMC simulation package", Published by Elsevier Ltd., Computers & Fluids, journal homepage: www.elsevier.com/locate/complfluid, Computers & Fluids 57 (2012) 66–75
- [283] Musawwer Khan, and et. al., "Deployment Strategies for a Reengineered Information System in Context of Legacy System", Research Journal of Applied Sciences, Engineering and Technology 4(3): 178-185, 2012, ISSN: 2040-7467, © Maxwell Scientific Organization, 2012.
- [284] Sunil Kumar Jangir, and et. al., "evolution and melioration of software management processes", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [285] A.Nirmal Kuma and Dr.B.G.Geetha, "Achieving Software Engineering Knowledge Items with an Unit Testing Approach", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-2, Issue-1, March 2012.
- [286] Muthu Ramachandran, "Guidelines Based Software Engineering for Developing Software Components", Journal of Software Engineering and Applications, 2012, 5, 1-6 doi:10.4236/jsea.2012.51001 Published Online January 2012 (<http://www.SciRP.org/journal/jsea>).
- [287] Saleh H. Al-Daajeh and et. al., "A Tactic-Based Framework to Evaluate the Relationships between the Software Product Quality Attributes", Int.J. of Software Engineering, IJSE Vol.5 No.1 January 2012.
- [288] Devesh Kumar Srivastava, and et. al., "VRS Model: A Model for Estimation of Efforts and Time Duration in Development of IVR Software System", Int.J. of Software Engineering, IJSE Vol.5 No.1 January 2012.
- [289] Hisham M. Haddad, and et. al., "Framework for Instituting Software Metrics in Small Software Organizations", Int.J. of Software Engineering, IJSE Vol.5 No.1 January 2012.



Proposed Software Re-engineering Process That Combine Traditional Software Re engineering Process With Spiral Model

Ahmed Saleem Abbas*

Department of Computer Science & IT
Karbala University & SHIATS University Babylon, Iraq
ahmed_saleem@yahoo.com

DR. W. Jeberson

Department of Computer Science & IT
SHIATS University Allahabad, India
jeberson@rediffmail.com

V.V. Klinsega

Department of Computer Science & IT
SHIATS University Allahabad, India
segavijayakumar@rediffmail.com

Abstract: Software re-engineering, a recent research area includes reverse engineering, forward engineering and reengineering tools while re-engineering process modification to get new enhanced process until now does not discussed in a serious manner, so in this paper the traditional software reengineering process is discussed and the "proposed software reengineering process" is suggested. The proposed process incorporates both the reverse engineering process and forward engineering process integrated with spiral model where reverse engineering applies to existing system code to extract design & requirements, although this is often used as means to mitigate risks & reduced costs of operation and maintaining the software system. This paper briefly describes traditional re-engineering then discusses the emerging process of proposed-reengineering which is often used as means to simplify the complex tasks. The paper represents how quality is going to be effect with the help of given software reengineering process. An analysis of various possible risks, their impact and mapping with various attributes is correspondingly presented. This paper is also presenting the way to reduce the impact of most of these risks by using proposed re-engineering through get all the useful features of spiral model.

Keywords: Software Engineering, Software Re-engineering Process, Proposed Re-engineering Process, Reverses Engineering; Forward Engineering, Risk Assessment, Software Metrics.

I. INTRODUCTION

Re-engineering is the examination, analysis and alteration of an existing software system to reconstitute it in a new form, and the subsequent implementation of the new form [1]. Proposed Reengineering "is a re-engineering process that uses not just the traditional six activities, but a combination of these activities and a set of framework activities of the spiral model to transition an existing system to a target system.

There are a lot of number of risks associated with various principles such as Re-think & Re-specify, Re-Design, Re-Code, Re-Test i.e. Technical risk, known risk & project risk belonging to various sub categories of risk such as selection of code translation, operational, line by line translation is not possible, quality, reliability, external, interfacing of COTS with legacy system, schedule, COTS will perform up to the standards [2], because of these risks the thinking about planning and risk analysis activity of the spiral model is discussed as a solution to determine and reduce risks and their effects. The cyclic approach of spiral model is used within the proposed software reengineering to get the benefit of it; incrementally growing a system degree of definition and implementation while it decreasing its degree of risk.

The process typically encompasses a combination of other processes such as reverse engineering, re-

documentation, restructuring, translation and forward engineering. As the software industry moves to a new era, many new software design methodologies are developed, improving software reusability and maintainability, and decreasing development and maintenance time. But most companies have legacy systems that are costly to maintain. These systems cannot just be replaced with new systems. They contain corporate information and implied decisions that would be lost. They also are an investment, and were too costly to develop and evolve just to discard [2].

For these purposes, re-engineering becomes a useful tool to convert old, obsolete systems to more efficient, streamlined systems. But project development is always short on time and money, making the need to look at alternatives necessary. The use of COTS packages is seen as a way to increase reliability while decreasing development and test time. Translation of code is a means of decreasing time and cost. This has resulted in a combination of the development methods into a form of proposed re-engineering. In the proposed software re-engineering process there are a set of anchor point milestones for ensuring stakeholders commitment to feasible and mutually satisfactory system solution.

- a. Re-engineering Model: The goal is to understand the existing software (specification, design and implementation) and then to re-implement it to improve the functionality, performance or implementation of the system and it is used to maintain the existing functionality and prepare for functionality to be added later, achieving greater

reliability, preparation for functional enhancement, improve maintainability & migration [1]

- b. System Re-engineering: Re-structuring or re-writing part or all of a legacy system without changing its functionality considered as system re-engineering [3]. It applies where subsystem of a larger system require frequent maintenance [4]. Re-engineering involves adding effort to make functionality of those systems easier to maintain. The system may be restructured and re-documented [5]. Generally, there are following phases of software re-engineering, Source code translation, Reverse engineering, Program structure improvement, Program modularization and Data re-engineering.

II. RELATED WORK

Re-engineering is generally discussed as “business process change”. Such change imposes new requirements on systems. Pooley et. al. include re-engineering in business process change not only changes over time within one organization but also the situation presenting many of the same problems in which a system developed in one organization and to be used in another[6]. Expert in re-engineering are much rarer than are experts in design and most of the engineers do not have much research experience in this area [5]. The problems with legacy systems had posed everywhere in the world. Brodie et. al. define a legacy

system as one that significantly resists modification and evolution to meet new and constantly changing business requirements regardless of the technology used to design it [7]. The legacy system is replaced by a new system with the same or improved functionality [5].

In 2001, Alessandro Bianchi et al, they published paper titled "Iterative Reengineering of Legacy Functions", this paper describes a process of gradual reengineering of the procedural components of a legacy system. The proposed method enables the legacy system to be gradually emptied into the reengineered system, without needing to either duplicate the legacy system or freeze it. The process consists of evolving the legacy system components toward firstly a restored system and then toward the reengineered system. Meanwhile, the legacy system can coexist with both the restored and the reengineered parts. By the end of the process, a single system will be in existence: the reengineered one. The method has been applied to reengineer a real system and demonstrated its ability to: support gradual reengineering, maintain the system at work during the process, and minimize the need to freeze maintenance requests, renew the operative environment of the reengineered system with respect to the legacy system and, finally, eliminate all the system's aging symptoms This reengineering process is shown in figure 1, the various phases each being enclosed in a box, while the label on the arrow linking two successive phases indicates the output produced by the relative phase [8].

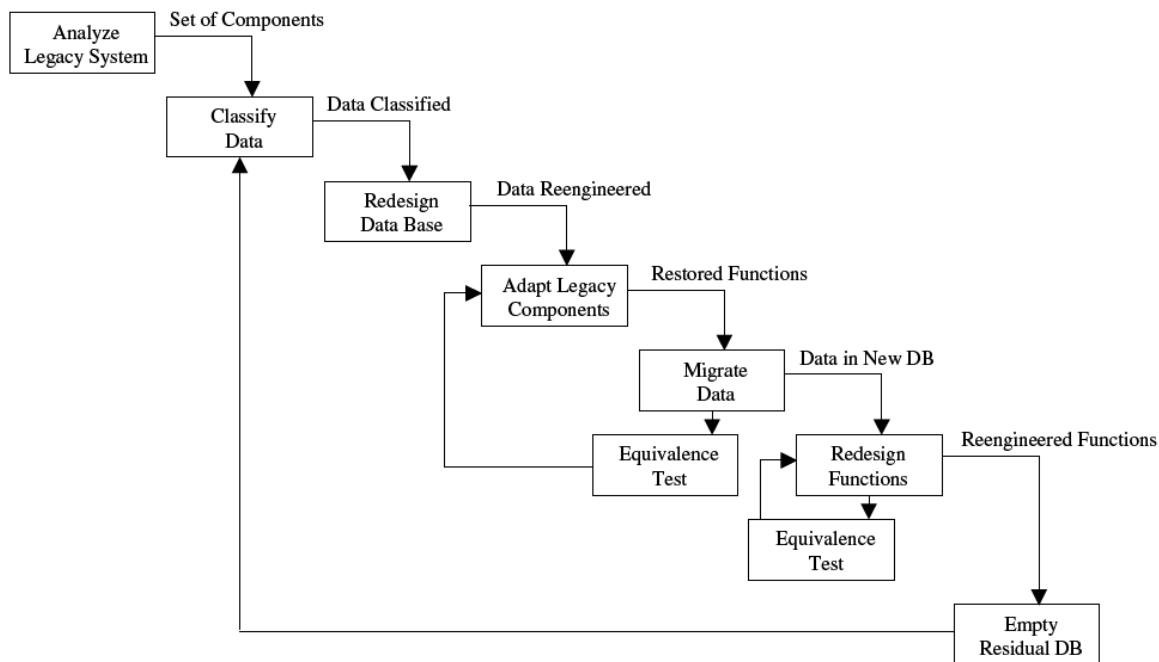


Figure 1. Iterative reengineering process [8].

In 2005, Xiaohu Yang et al. they published paper titled "A Dual-Spiral Reengineering Model for Legacy System". This paper presents a dual-spiral reengineering model, which performs as a cyclic approach. The main workflow in Dual-Spiral Reengineering Model requires that the two systems (legacy one and target one) work together, and

move the functionality (not the modules) from the legacy system to the target system step by step, as in the spiral model. During the entire process, the active functionality in the legacy system is in a decremental pattern, and the active functionality in the new target system is in an incremental pattern, as Figure (2) [9].

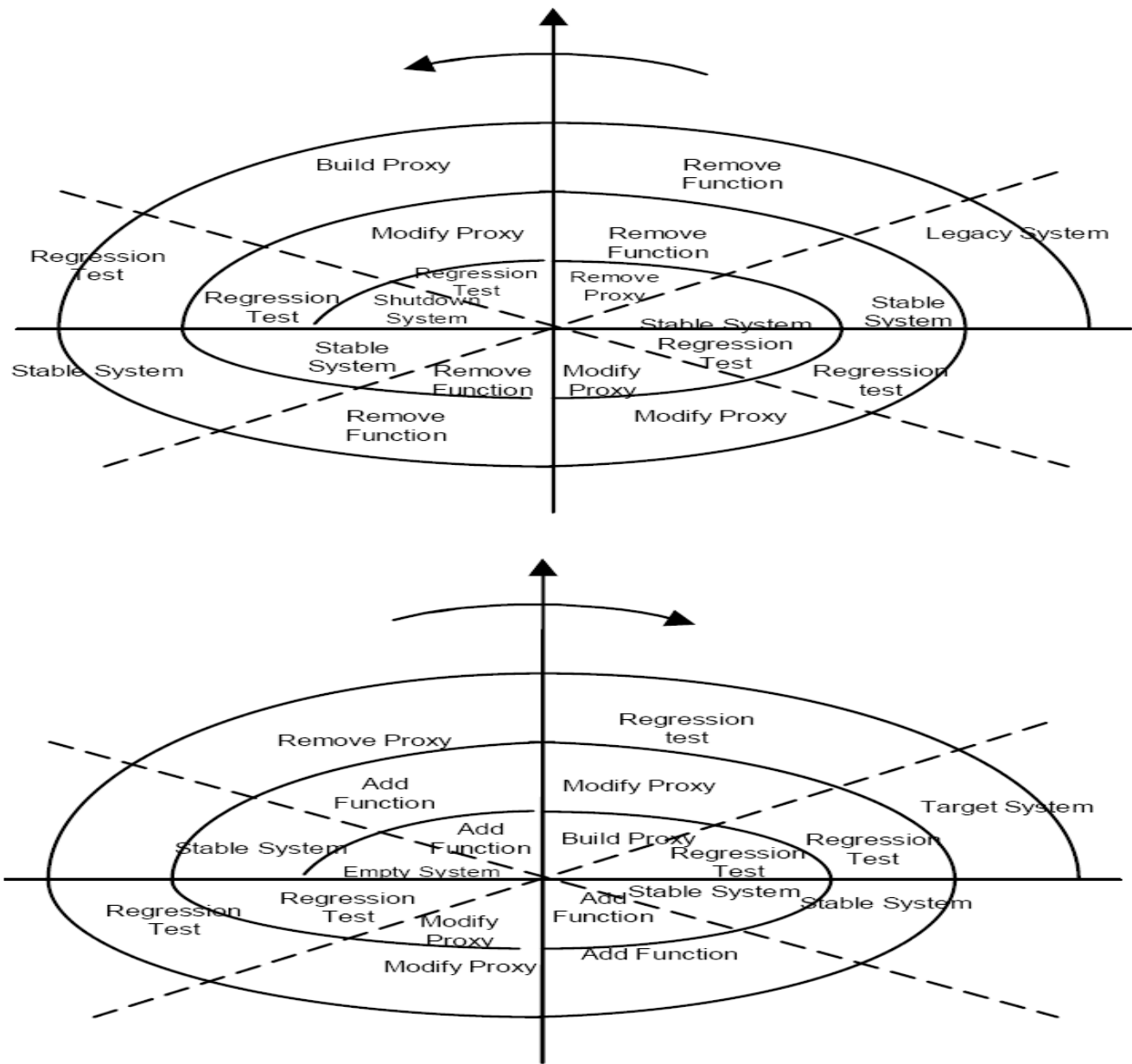


Figure 2. Decremental Legacy system Pattern and Incremental Target system Pattern in Dual-Spiral Model [9]

In 2012, Sandhya Tarar and Dr. Ela Kumar they design the Hybrid reengineering paradigm [2]. Hybrid re-engineering is a re-engineering process that uses not just a single, but a combination of abstraction levels and alteration methods to transition an existing system to a target system. In order to do the hybrid reengineering the reengineering model is mapped on with the software component library that can be Commercial off the shelf (COTS) or Math description engineering (MDE). Because hybrid reengineering uses COTS or MDE libraries through which design and requirements can be specified much faster thereby reducing effort, time and increasing reliability.

COTS have some risk associated with it i.e. A package will not perform an anticipated or advertised, or that will be unreliable, immature or incomplete. Also COTS product may limit further enhancements to the system because changes in COTS provided functions may not be possible due to legal or contractual issues. All the main principles of hybrid reengineering approach namely re-specify, redesign, recode & retest the main task remains of integrating all the work of these principles together to make an effective model or system. Any specific principle is useless unless there is no integration with the other [2]. The Hybrid model is shown in figure (3).

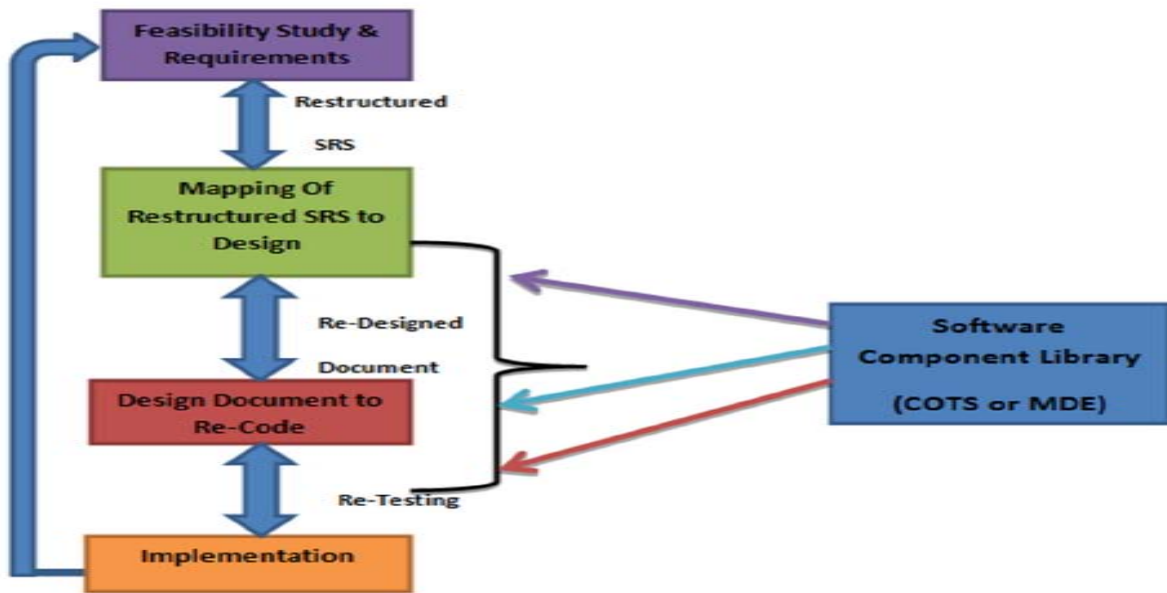


Figure 3. Hybrid Re-Engineering Model [2]

III. SPIRAL MODEL [10]

The spiral model is an evolutionary software process model that combines the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. This model was shown in figure (4).

Spiral model is Prevalent in OO design methodologies, also encompasses all the essential development phases: Requirements analysis, Design, Code, Test and Maintenance. This model Marked by use of the control and structure of the

more linear, sequential waterfall process, but with a series of “evolutionary releases”, it is also known as the “hybrid model”, Explicitly addresses the issue of quality assurance by performing the development process in a “step-wise refinement” method. Each step produces a “deliverable” which embodies the structure or sequential nature of the process. This process naturally focuses on a high degree of customer or stakeholder involvement during the development process.

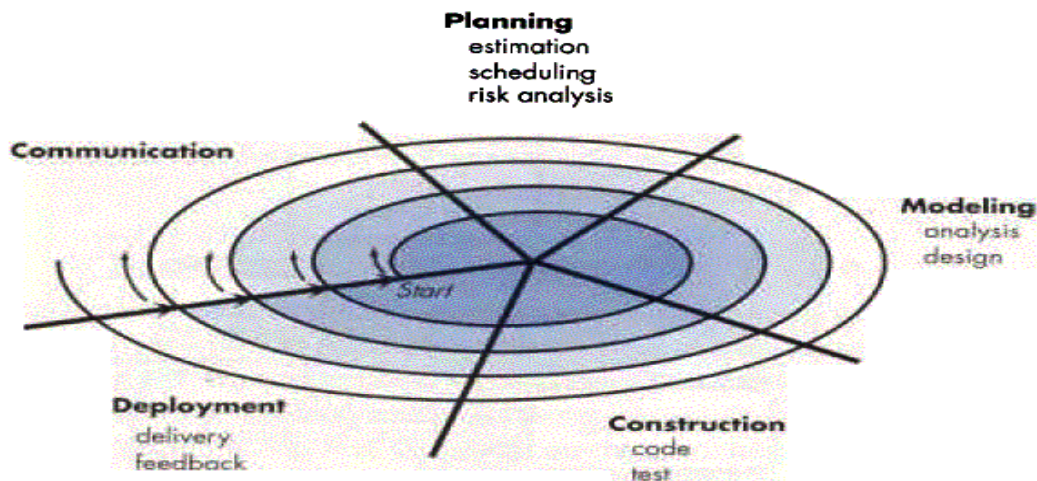


Figure 4. spiral model [10].

In this diagram, “Communication” refers to the Requirements and analysis process, “Planning” corresponds to preliminary design and scheduling, “Modeling” is the detailed design, “Construction” refers to code/debug/integration, and “Deployment” is the delivery to

the customer and the feedback process. Software is developed in a series of evolutionary releases

- During early iterations, the release might be a paper model or prototype
- During later iterations, increasingly more complete versions of the engineered system are produced

- c. The final iteration produces the complete software product

First circuit around the spiral might result in the development of the product specification; might result in a CoDR review by the customer. Next iteration might produce a prototype, containing the GUI, for example; the customer might want to see this, so there could be a PDR and/or CDR at this time. Third time around might be used to fill in more detailed functionality, and release a preliminary working model. Fourth circuit might result in a complete alpha release, which the customer could “hammer on” for a while to test robustness and provide feedback to the solution provider about the product’s strengths and weaknesses. Fifth iteration might be a beta test, or it could be the final build for initial release (if the previous circuit was satisfactory enough to warrant this).

IV. TRADITIONAL SOFTWARE REENGINEERING PROCESS:[10]

Reengineering takes time; it costs significant amounts of money; and it absorbs resources that might be otherwise occupied on immediate concerns. For all of these reasons, reengineering is not accomplished in a few months or even a few years. Reengineering of information systems is an activity that will absorb information technology resources for many years. That’s why every organization needs a pragmatic strategy for software reengineering. A workable strategy is encompassed in a reengineering process model.. Reengineering is a rebuilding activity, to implement it apply a software reengineering process model that defines six activities, shown in Figure (5). In some cases, these activities occur in a linear sequence, but this is not always the case. For example, it may be that reverse engineering (understanding the internal workings of a program) may have to occur before document restructuring can commence. The reengineering paradigm shown in the figure is a cyclical model. This means that each of the activities presented as a part of the paradigm may be revisited. For any particular cycle, the process can terminate after any one of these activities.

- a. **Inventory analysis:** Every software organization should have an inventory of all applications. The inventory can be nothing more than a spreadsheet model containing information that provides a detailed description (e.g., size, age, business criticality) of every active application.
- b. **Document Restructuring:** Weak documentation is the trademark of many legacy systems. But What are available options to solve it?
 - (a). Creating documentation is far too time consuming. If the system works, we’ll live with what we have.
 - (b). Documentation must be updated, but there are limited resources. We’ll use a “document when touched” approach.

- (c). The system is business critical and must be fully redocumented.
- c. **Reverse Engineering:** the Reverse engineering is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than source code. Reverse engineering is a process of design recovery. Reverse engineering tools extract data, architectural, and procedural design information from an existing program.
- d. **Code Restructuring:** The most common type of reengineering (actually, the use of the term reengineering is questionable in this case) is code restructuring. Some legacy systems have relatively solid program architecture, but individual modules were coded in a way that makes them difficult to understand, test, and maintain. In such cases, the code within the suspect modules can be restructured. To accomplish this activity, the source code is analyzed using a restructuring tool. Violations of structured programming constructs are noted and code is then restructured (this can be done automatically). The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced. Internal code documentation is updated.
- e. **Data Restructuring:** A program with weak data architecture will be difficult to adapt and enhance. In fact, for many applications, data architecture has more to do with the long-term viability of a program than the source code itself. Unlike code restructuring, which occurs at a relatively low level of abstraction, data structuring is a full-scale reengineering activity. In most cases, data restructuring begins with a reverse engineering activity. Data objects and attributes are identified, and existing data structures are reviewed for quality. When data structure is weak (e.g., flat files are currently implemented, when a relational approach would greatly simplify processing), the data are reengineered. Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.
- f. **Forward Engineering:** In an ideal world, applications would be rebuilt using a automated “reengineering engine.” The old program would be fed into the engine, analyzed, restructured, and then regenerated in a form that exhibited the best aspects of software quality. In the short term, it is unlikely that such an “engine” will appear, but CASE vendors have introduced tools that provide a limited subset of these capabilities that addresses specific application domains (e.g., applications that are implemented using a specific database system). More important, these reengineering tools are becoming increasingly more sophisticated.

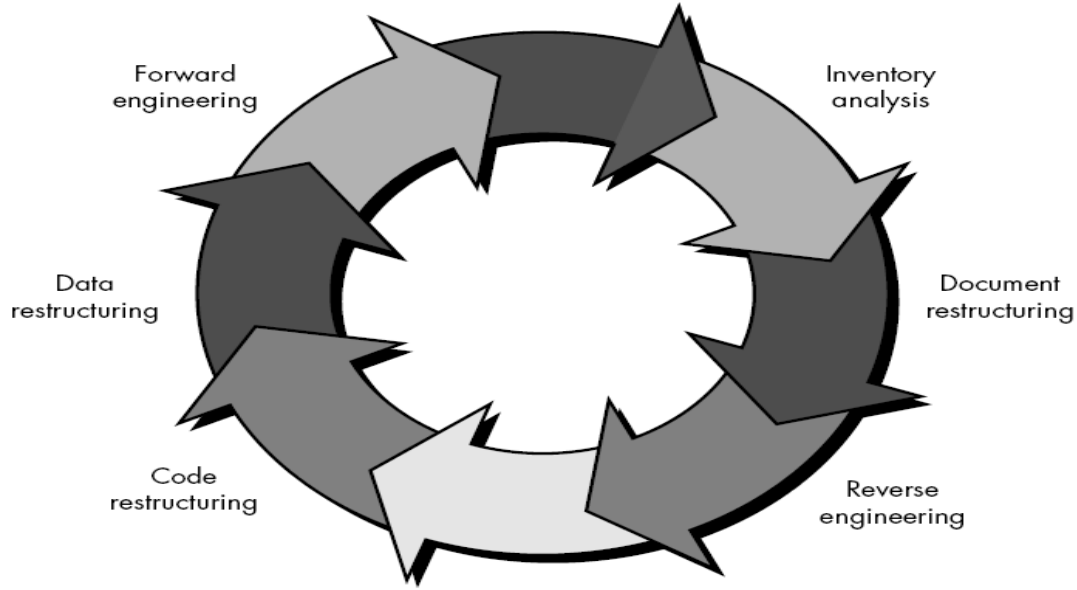


Figure 5. A software reengineering process model [10]

V. PROPOSED SOFTWARE REENGINEERING PROCESS

In this section a new reengineering process was suggested and named as a "Proposed Reengineering Process". It is a re-engineering process that uses not just the six activities of traditional re-engineering process, but a combination of these activities and a set of framework activities of the spiral model to perform the transition of an existing system (legacy system) to a target system (new system). The proposed reengineering process consists of eight task regions, these tasks regions as follows:

- a. **Communication & Detection the Frailer Reasons of Legacy SW:-** Communication with customer in order to re-specify the requirements as per the new need of the user, and then map these requirements with the SRS. The frailer reasons of the selected legacy software were detected within this first step and the feasibility study is done.
- b. **Planning:-** it is the activity that required to define resources, timelines, cost and other project related information, then risk analysis is accomplished to assess both technical and management risks.
- c. **Reverse Engineering:-** Reverse engineering is the process that starts from the implementation phase and moving towards the coding, design and requirement phase, through this region the data, architectural and procedural design information was extracted from existing legacy software.
- d. **Document Restructures:** in this phase the mapping of restructured SRS to the Design document is being done i.e. integration of new SRS to design in order to get the redesigned document which is the output of this phase. As SRS changes the design structure consisting of

DFD/ ER diagrams/UML diagrams need to be changed as per the extent of changed requirements.

- e. **Code Restructures:** To accomplish this activity, the source code is analyzed using a restructuring tool. The resultant restructured code is reviewed and tested to ensure that no Distortions have been introduced. Internal code documentation is updated.
- f. **Data Restructures:** In most cases, data restructuring begins with a reverse engineering activity. Data objects and attributes are identified, and existing data structures are reviewed for quality. When data structure is weak, the data are reengineered. Because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.
- g. **Forward Engineering:** it is the process which starts from requirement to the implementation phase. Forward engineering, also called renovation or reclamation, not only recovers design information from existing software, but uses this information to alter or reconstitute the existing system in an effort to improve its overall quality. In most cases, reengineered software re-implements the function of the existing system and also adds new functions and/or improves overall performance.
- h. **Deployment:** it is the delivery of reengineered software to the customer and then feedback process is done to obtain customer evaluation of that software.

These eight task regions are shown in the (figure 6) and they are very important to reengineer the legacy software; this proposed process model integrated the iteration nature of spiral model with traditional software reengineering process. There are basics task that have to be accomplished

by any software reengineering process but the diagnostic features of the proposed process are the following:

- a) The millstones are inserted after each iteration to check if the process is performed within schedule time or not.
- b) The reengineering process can be finished after any iteration.
- c) The reasons that cause the failure of the legacy software are determined at first step to remedy these reasons.
- d) In the proposed reengineering process there is planning task regions that make feasibility study, scheduling, estimation and risk analysis to determine all the expected benefit and estimation of the required cost and effort, so according this the developer can make his decision to reengineer that legacy software or rebuild it from scratch.
- e) The third, fourth, fifth, sixth, seventh are the same of that activities that done in traditional software reengineering process except that in the proposed there are a non determined number of iterations that allowed to reach to the final project with new features, capabilities and high quality software.
- f) Deployment and waiting for the customer feedback it is also diagnostic activity of the proposed reengineering process. According this feedback the decision is made if the reengineering process is finished or not. If the customer dose not satisfy the next iteration will began.

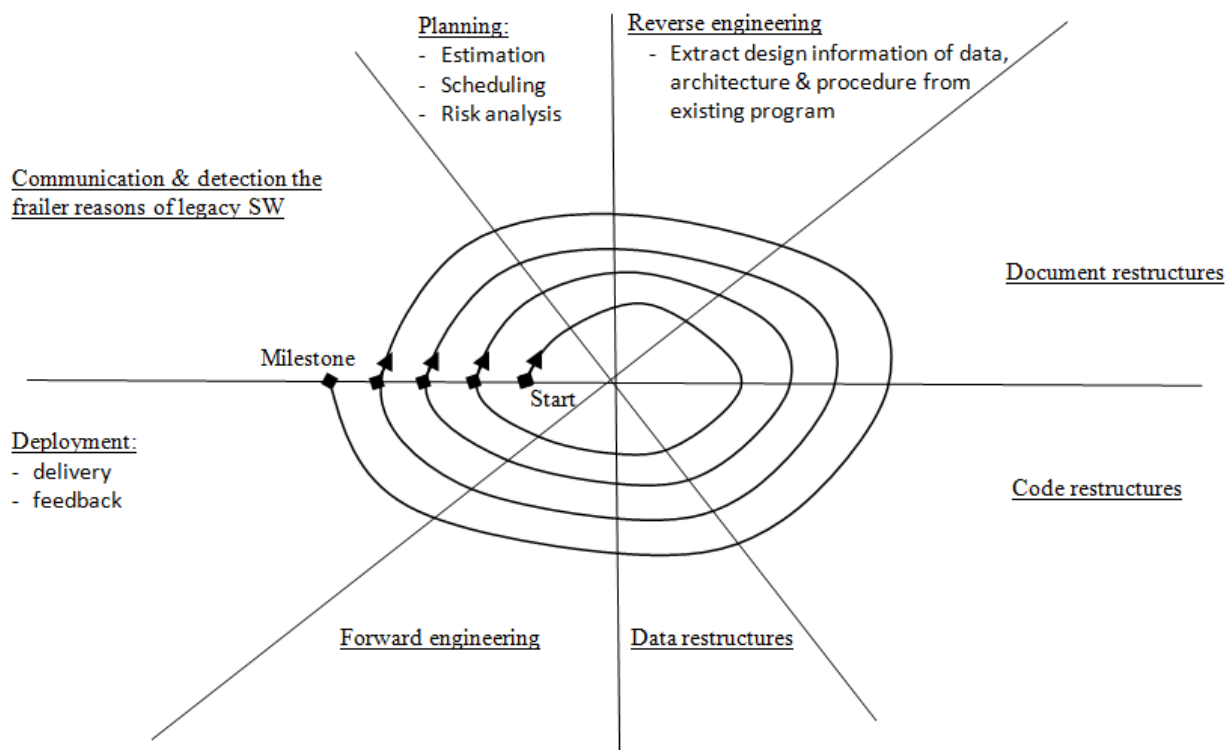


Figure 6: Proposed reengineering process

A. **Benefits of Proposed software reengineering process:**

Proposed re-engineering has the advantage that it determine the causes of failure to overcome it and prevent it for the future, also it is reduces the development schedule time and reduced the costs. This process can be used to produce many versions of modern software by using the legacy software as a baseline.

B. **Limitations of Proposed software reengineering process:**

Since the proposed re-engineering is a new approach in the re-engineering area although a very cost efficient approach as

it reduces development time & cost but there are no exact metrics available for this in order to measure the scalability & performance and most of these metrics depend on estimation values such as human effort (person/ hour), this metric is deferent from person to other according to experience and skills of that person, also cost and time are depend on human effort metrics and the complexity of the software, so this is a main limitation that face as a developer.

VI. EXPECTED RISKS

There are number of risk that can be recognized and the reengineering teams have to take care about them, they are:

- (a). Development Environment: risks associated the availability and quality of the COTS/MDE tools.
- (b). Product Size: risks impact is directly proportional to the overall product size that has to be re-engineered/re-build/re-modulated.
- (c). Size and experience; risks associated with the overall technical and project experience of the software engineers who will do the task of re-engineering.
- (d). Customer characteristics: risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- (e). Technology to be built: risks associated with the complexity of the system to be reengineered and the "newness" of the technology that is packaged by the system.
- (f). Process definition: risks associated with respect to the process followed by the organization to perform re-engineering process.
- (g). Business impact: risks associated with constraints imposed by management or the marketplace and also the Domain of the business causes risks for performing reengineering process.

The reengineering risk can be distributed into six risk areas as shown in figure (7). These risk areas as following [11]:

a. Technology risks:

- a) Recovered information is not useful or used
- b) Reverse engineering to representations that cannot be shared
- c) Reengineering technology inadequate to accomplish reengineering goals

b. Tool risks:

- a) Dependence on tools that do not perform as advertised
- b) Not using installed tools

c. Strategy risks:

- a) Premature commitment to a reengineering solution for an entire system
- b) Failure to have a long-term vision with interim goals
- c) Lack of global view: code, data, process reengineering
- d) No plan for using reengineering tools

d. Application risks:

- a) Reengineering with no local application experts available
- b) Existing business knowledge embedded in source code is lost
- c) Reengineered system does not perform adequately

e. Process risks

- a) Extremely high manual reengineering costs
- b) Cost benefits not realized in required time frame
- c) Cannot economically justify the reengineering effort
- d) Reengineering effort drifts
- e) Lack of management commitment to ongoing reengineering solution

f. Personnel risks:

- a) Programmers inhibiting the start of reengineering

b) Programmers performing less effectively to make an unpopular reengineering project look less effective



Figure (7): Re-engineering Risks area [11]

VII. SOFTWARE METRICS

Metrics is a quantitative measurement of its attributes. Various parameters such as Cost, Reliability, Quality, Object-Oriented Measures, Reusability, Complexity, and Portability can be used to measure the software re-engineering in quantitative form. As shown in table (1) corresponding metrics leads to various factors responsible for the measurement of the software re-engineering [2][10].

Table I. Software Reengineering Metrics

Complexity	• Size (LOC,FP) • Design structure • Data analysis
Cost	• Effort (person, months) • Time • Productivity
Reliability	• Interoperability • Availability • MTTR • MTTF
Reusability	• Retrievability • Integrability • Testability • Generality • Portability • Modifiability
Quality	• Functionality • DRE • Complexity • Reliability • Reusability • Performance • Security
Object oriented measures	• Inheritance • Encapsulation • Polymorphism • Abstraction

VIII. CONCLUSION

The new era is recognized by the very fast changes in the computer industry that introduce new hardware and software, making older systems suffers from the market competition and the difficulty in maintenance. Software re-engineering provides reduced risk level. There is a high risk in new software

development. There may be development problems, staffing problems and specification problems which are reduced by the use of reengineering process. Another advantage of software re-engineering is reduced cost. The cost of re-engineering is often significantly less than the costs of developing new software because some part i.e. sub-systems are used as it is and some sub-systems are changed. Based on previous history, project development is usually short on time and money, making it necessary to look at another solution. "Proposed Re-engineering" plays as a solution in this scenario. The use of re-engineering approaches provides a way to increase reliability and quality of the system while decreasing development efforts. Translation of code is a means of decreasing time and cost. This results in a combination of development methods of traditional software reengineering with the Spiral software development model into a form of proposed re-engineering process model.

IX. ACKNOWLEDGMENT

We thank everyone at the SHIATS/department of computer science and information technology who participated in this research for many stimulating discussions. Also special thanks to Karbala University/department of computer science for them great support.

X. REFERENCES

- [1] L. Manzella. Mutafelija.: Concept of re-engineering Life Cycle, ICSI Second International Conference On System Integration. IEEE. 1992.
- [2] Sandhya Tarar, Dr. Ela Kumar, "Design Paradigm and Risk Assessment of Hybrid Re-engineering with an approach for development of Re-engineering Metrics", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
- [3] Linda H. Rosen Berg, Lawrence E.Haytt. : Hybrid Re-engineering, Software technology conference. Utah, April 1997.
- [4] MindTree Ltd.: Rehosting & Re-engineering from Mainframe to Wintel, Case Study, 2010.
- [5] Shekhar Singh, Significant role of COTS to design Software Reengineering Patterns, International Conference on Software Engineering and Applications(ICSEA),2009.
- [6] 15. Pooley R., Stevens P., Systems Reengineering Patterns, CSG internal report, 1998.
- [7] Brodie, Michael L., and Stonebraker, Michael, "Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach", Morgan-Kaufman Publishers, 1995.
- [8] Alessandro Bianchi et al, " Iterative Reengineering of Legacy Functions", IEEE International Conference on ISBN: 07695118 Year: 2001 Pages: 632-641 Provider: IEEE Publisher: IEEE.
- [9] Xiaohu Yang et al, "A Dual-Spiral Reengineering Model for Legacy System", TENCON 2005 - 2005 IEEE Region 10 Conference ISBN: 0780393112 Year: 2005 Pages: 1-5 Provider: IEEE Publisher: IEEE.
- [10] Pressman, Scott (2005), Software Engineering: A Practitioner's Approach (Sixth, International ed.), McGraw-Hill Education Pressman.
- [11] Ian Sommerville, "Software Engineering, 6th edition", 2000.

IMPLEMENTATION OF FUSION MODEL TO RE-ENGINEER LEGACY SOFTWARE

AHMED SALEEM ABBAS¹, W. JEBERSON² & V. V. KLINSEGA³

¹Doing Ph.D at SHIATS & Assistant Lecturer at Department of Computer Science, Karbala University, Iraq

^{2,3}Associate Professor, Department of Computer Science & IT, SHIATS University, Allahabad, India

ABSTRACT

Software re-engineering, is a recent research area which includes reverse engineering, forward engineering and reengineering. In this paper, the software reengineering process model proposed by the same authors in another paper titled “Proposed Software Re-engineering Process That Combine Traditional Software Re-engineering Process with Spiral Model” [1] is validated by implementing it on a legacy system, and its name is renamed as FUSION model. This new model incorporates both the reverse engineering process and forward engineering process integrated with spiral model where reverse engineering applies to existing system code to extract design & requirements, although this is often used as means to mitigate risks & reduce operation and maintenance cost of the software system. This paper briefly describes the FUSION model which can be used to simplify the complex tasks. The paper represents how the legacy supermarket management system can be re-engineered using the new approach then the result of comparison between the legacy software and the re-engineered software will be presented. The features of the FUSION model are also presented in later section of the paper.

KEYWORDS: Cyclomatic Complexity, Quality Metrics, Reverse Engineering, Risk Assessment, Software Engineering, Software Re-Engineering Process

INTRODUCTION

Re-engineering is the examination, analysis and alteration of an existing software system to reconstitute it in a new form, and the subsequent implementation of the new form [2]. FUSION Reengineering model “is a re-engineering process that uses not just the traditional six activities. (Inventory analysis, Document restructuring, Reverse engineering, Code restructuring, Data restructuring, Forward engineering), but a combination of these activities and a set of framework activities of the spiral model to transform an existing system to a target system.

There are a number of risks associated with various principles such as Re-think & Re-specify, Re-Design, Re-Code, Re-Test i.e. technical risk, known risk & project risk belonging to various sub categories of risk [3], because of these risks the planning and risk analysis activity of the spiral model is integrated with FUSION model as a solution to determine and minimize risks. The cyclic approach of spiral model is used within the FUSION software reengineering model to get the benefit of its incrementally growing system degree of definition and implementation while decreasing the degree of risk.

As the software industry moves to a new era, many new software design methodologies are being developed, improving software reusability and maintainability, and decreasing development and maintenance time. But most companies have legacy systems that are costly to maintain. These systems cannot just be replaced with new systems. They contain corporate information and implied decisions that would be lost. They also are an investment, and were too costly to develop and evolve just to discard [3].

For these purposes, re-engineering becomes a useful tool to convert old, obsolete systems to more efficient,

streamlined systems. FUSION model is used to convert the legacy supermarket management system that was developed by using visual basic 6 to the target system according to the feedback of the customer, where the legacy system was used as a base line for this reengineering process. In the next sections a quick review of the basic concepts of reengineering will be done. After that the FUSION model will be discussed step by step and how these steps were applied on the supermarket legacy system, which will be followed by a discussion on the tools that helped us to apply and evaluate this proposed reengineering process.

RELATED WORK

Re-engineering is generally discussed as “business process change”. Such change imposes new requirements on systems. Pooley et. al. include re-engineering in business process change not only changes over time within one organization but also the situation presenting many of the same problems in which a system developed in one organization and to be used in another [5]. Expert in re-engineering are much rarer than are experts in design and most of the engineers do not have much research experience in this area [4]. The problems with legacy systems had posed everywhere in the world. Brodie et. al. define a legacy system as one that significantly resists modification and evolution to meet new and constantly changing business requirements regardless of the technology used to design it [6]. The legacy system is replaced by a new system with the same or improved functionality [4].

The paper published by Alessandro Bianchi et al on Iterative Reengineering of Legacy Functions describes a process of gradual reengineering of the procedural components of a legacy system. The proposed method enables the legacy system to be gradually emptied into the reengineered system, without having the need to either duplicate the legacy system or freeze it. The process consists of evolving the legacy system components firstly towards a restored system and then toward the reengineered system. Meanwhile, the legacy system can coexist with both the restored and the reengineered parts. By the end of the process, a single system will be in existence, which is the reengineered one. The method has been applied to reengineer a real system and demonstrated its ability to: support gradual reengineering, maintain the system at work during the process, and minimize the need to freeze maintenance requests, renew the operative environment of the reengineered system with respect to the legacy system and, finally, eliminate all the system’s aging symptoms [7].

A Dual-Spiral Reengineering Model for Legacy System proposed by Xiaohu Yang et al. performs as a cyclic approach. The main workflow in Dual-Spiral Reengineering Model requires that the two systems (legacy one and target one) work together, and move the functionality (not the modules) from the legacy system to the target system step by step, as in the spiral model. During the entire process, the active functionality in the legacy system is in a decremental pattern, and the active functionality in the new target system is in an incremental pattern [8].

The Hybrid reengineering paradigm [3] proposed by Sandhya Tarar and Dr. Ela Kumar. Hybrid re-engineering is a re-engineering process that uses not just a single, but a combination of abstraction levels and alteration methods to transform an existing system to a target system. Here the reengineering model is mapped on with the software component library that can be Commercial Off The Shelf (COTS) or Math Description Engineering (MDE). Because hybrid reengineering uses COTS or MDE libraries through which design and requirements can be specified much faster thereby reducing effort, time and increasing reliability. COTS have some risk associated with it i.e. a package will not perform an anticipated or advertised, or that will be unreliable, immature or incomplete. Also COTS product may limit further enhancements to the system because changes in COTS provided functions may not be possible due to legal or contractual issues. The main task in hybrid reengineering is to integrate its main approach namely re-specifying, redesigning, recoding & retesting to work together to make an effective model or system. Any specific principle is useless unless there is no integration with the other [3].

BASIC CONCEPTS OF FUSION MODEL

The FUSION model enables cyclic reengineering of a legacy system, where the various components will go through different states during this process. Each component may be in one of the following states during the various steps of the reengineering process:

- Legacy, i.e. components of the legacy system that have not yet been reengineered.
- Restored, i.e. functions with the same structure as in the legacy system, but access data through the new data banker, that alone recognizes the physical structure of the database.
- Reengineered, i.e. components of the legacy system that have already been modified, and whose quality has reached the desired levels.
- New, i.e. components that did not exist in the legacy system, but have been added to introduce new functions in the same application domain.

The FUSION model is a re-engineering process model that uses not just the six activities of traditional re-engineering process, but a combination of these activities and a set of framework activities of the spiral model to perform the transition of an existing system (legacy system) to a target system (new system). As shown in the figure 1, there are eight task regions, which were applied by the software engineering team on legacy software to get the target re-engineered software, for this purpose we have chosen a legacy software known as supermarket management system that was developed in visual basic before many years and all the steps required to convert it to the target system will be discussed as follows:

- **Identification Failure Reasons of Legacy SW:** The software re-engineering team should meet with customer in order to re-specify the requirements as per the current expectations of the user. In the supermarket system the new requirements are identified such as convert the system from windows based to web based, create two levels of authentication as administrator level and employee level, add the facility to generate many types of reports about the product, stock master, supplier, sale and purchase information, and feedback report, followed by mapping these requirements with the SRS. The reasons for failure of the selected legacy software were detected within this first step, some of which are that the legacy software don't have any type of security, there is weak error handling procedure that does not cover all the possible error cases, and that the old system can work on only one computer which results in poor service with only one counter in the supermarket to generate the bill for the customer. After that, the feasibility study was conducted,

In this sense we express the *mean time to maintenance request* for the i-th component (MTTMR_i) with the following formula:

$$MTTMR_i = \sum_{j=1}^n \frac{\Delta t_{j,i}}{n_i}$$

, where:

- n_i is the total number of requests for maintenance made for the i-th component;
- $\Delta t_{j,i} = t_{j,i} - t_{j-1,i}$ is the time interval between two successive requests for maintenance of the i-th component.

The values for n_i and $\Delta t_{i,1}$ are obtained from the historical archives of the system. When the expected time for reengineering the i -th component, RT_i , is less than $MTTMR_i$, then it is reasonable to suppose that it is unlikely to get requests for maintenance of this component during the time it is being reengineered. On vice versa, if RT_i is higher than $MTTMR_i$, then the component must be subdivided into several subcomponents, to prevent the requests for maintenance that are likely to be made during this time from being held up for very long time, and according to this analysis the decision to continue in re-engineering process on the legacy software was made

- **Planning:** it is the activity of defining resources, timelines, cost, and other project related information, to carry out the re-engineering process. It was decided to use ASP.net with C# programming language to design and implement all web forms of the system, and to use Microsoft SQL server 2008 to redesign the database and implement it. The duration of the whole process was estimated to be three months, and the cost benefit of re-engineering was calculated according to Sneed [9] formula, followed by risk analysis which was accomplished to assess both technical and management risks.
- **Reverse Engineering:** Reverse engineering is the process that starts from the implementation phase and moving towards the coding, design and requirement phase, as a result of which the data, architectural and procedural design information was extracted from existing legacy software. In this case study the “Visustin v7” analysis tool was used to Visualize the code of the legacy supermarket manangement system with flow charts! Visustin converts source code to flow charts and UML activity diagrams to understand existing code.
- **Document Restructures:** In this phase the mapping of restructured SRS to the Design document is being done i.e. integration of new SRS to design in order to get the redesigned document which is the output of this phase. As SRS changes the design structure consisting of DFD/ ER diagrams/UML diagrams need to be changed as per the extent of changed requirements, the algorithms have to be reviewed and modified and program logic should be verified. In this case study the “Visustin v7” analysis tool was used to restructure the document of complex functions of legacy software.
- **Code Restructures:** To accomplish this activity, we used “Visustin v7” restructuring tool and the resultant restructured code is reviewed and tested to ensure that no distortions have been introduced and the internal code documentation is updated.
- **Data Restructures:** In most cases, data restructuring begins with a reverse engineering activity. Data objects and attributes are identified, existing data structures are reviewed for quality and the data are reengineered when data structure is weak, because data architecture has a strong influence on program architecture and the algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes. In this case study all variables are reviewed and we found that the data structure was weak and so we restructured it, adding some new variables to be used in the new functions. From a conceptual point of view, the legacy data are partitioned into two classes:
 - Essential data, needed to carry out the application’s business functions; The essential data contain:
 - Conceptual data, that is specific to the application domain, and for which they have a precise conceptual meaning;
 - Necessary structural data, that do not belong to the above class but constitute the primary keys to the tables making up the database.

- Residual data, that are not necessary for carrying out the business functions, but are used by the legacy system and must therefore remain in the database until the procedures that use them have been reengineered. Instead, the residual data class contains all the data that existed in the legacy database, and which should ultimately be eliminated to improve the software quality. They are classified as:
 - Control data, that communicate to a procedure that an event occurred during execution of other procedures, thus controlling the behavior of the former one.
 - Redundant structural data, used to organize and support the data structures of the legacy system, but which are not strictly required, and can be removed with a better design of the database.
 - Semantically redundant data, whose definition domain is the same as, or is contained in, the definition domain of other data, while each equal value in the two definition domains is interpreted in the same way.
 - Computationally redundant *data* which can be computed starting from a different set of data included in the same database.

In this case study there are seven tables in the database of the legacy supermarket management system they are customer bill, customer detail, department, employee, items, transaction, and vendor and there are three tables were added to the reengineered database as per the customer need which are user login, product master, and feedback.

- Forward engineering: It is the process which starts from requirement to the implementation phase. Forward engineering, also called renovation or reclamation, not only recovers design information from existing software, but uses this information to alter or reconstitute the existing system in an effort to improve its overall quality. In most cases, reengineered software re-implements the functions of the existing system and also adds new functions to improve overall performance. In this case study we use a Rapid Application Development tool, Microsoft visual studio 2008 with C# and some web site templates to renovate the selected legacy software.
- Deployment: It is the delivery of reengineered software to the customer. Before this step the system should be tested by using any testing technique. But with the proposed software reengineering process the recommended testing techniques are Alpha testing and then Beta testing which is conducted when the project is delivered to the customer. The feedback process is necessary to obtain customer evaluation of that software, according to which the re-engineering process iterates to meet the customer demand.

The eight task regions are shown in the (figure 1) and they are very important to reengineer the legacy software. The FUSION model integrates the iteration nature of spiral model with traditional software reengineering process.

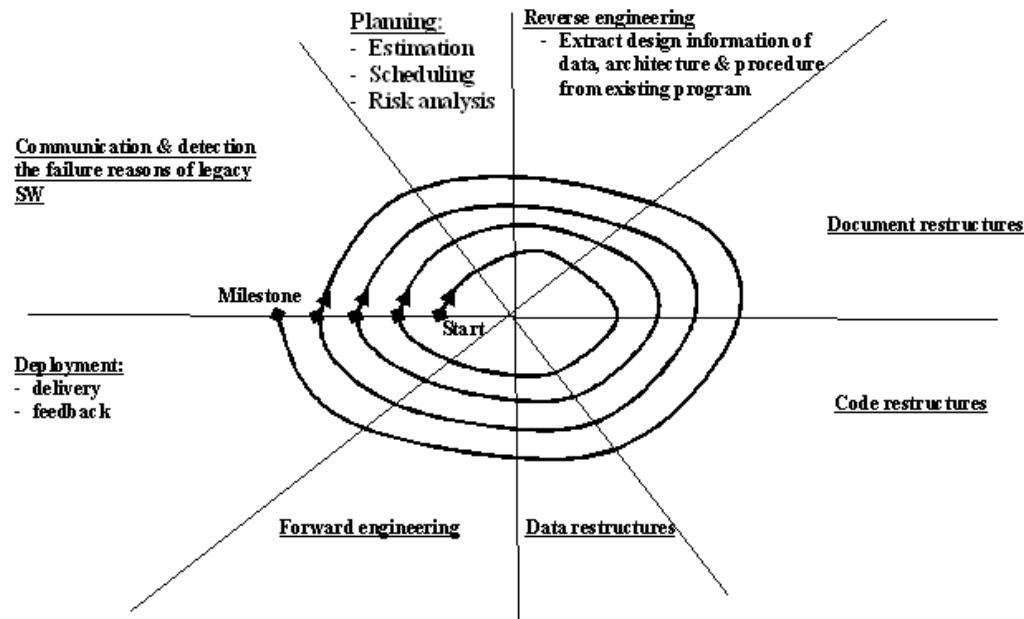


Figure 1: FUSION Reengineering Process

DIFFERENCES BETWEEN FUSION RE-ENGINEERING MODEL AND OTHER MODELS

There are many basic tasks that have to be accomplished by any software reengineering process but the diagnostic features of the proposed process are the following:

- The milestones are inserted after each iteration to check if the process is performed within scheduled time or not.
- The reengineering process can stop after any iteration.
- The reasons that cause the failure of the legacy software are determined at first step to remedy these reasons.
- In the proposed reengineering process there is planning task that includes feasibility study, scheduling, estimation and risk analysis to determine the expected benefit and estimation of the cost and effort, so that the developer can make his decision on whether to reengineer that legacy software or rebuild it from scratch.
- The third, fourth, fifth, sixth and seventh activities are the same as in traditional software reengineering process except that in the proposed model there is a non deterministic number of iterations that lead to the final reengineered product with new features, capabilities and high quality.
- Deployment and waiting for customer feedback is also a diagnostic activity of the proposed reengineering process. According to this feedback the decision is made if the reengineering process is completed or not. If the customer is not satisfied the next iteration will begin.

BENEFITS OF PROPOSED SOFTWARE REENGINEERING PROCESS AND THE COST BENEFIT FORMULA

Proposed re-engineering model has the advantage that it determines the causes of failure before any decision can be made, in order to overcome and prevent the failures in the future which in turn reduces the development time and cost. This model can be used to produce many versions of modern software by using the legacy software as a baseline.

According to the cost/benefit analysis model for reengineering, proposed by Sneed [9], the cost benefit for the proposed model can be calculated with respect to nine parameters as defined below:

P_1 = current annual maintenance cost for an application

P_2 = current annual operation cost for an application

P_3 = current annual business value of an application

P_4 = predicted annual maintenance cost after reengineering

P_5 = predicted annual operations cost after reengineering

P_6 = predicted annual business value after reengineering

P_7 = estimated reengineering costs

P_8 = estimated reengineering calendar time

P_9 = reengineering risk factor ($P_9 = 1.0$ is nominal)

L = expected life of the system

From the above, the following cost values can be obtained:

- (C_{maint}) the cost associated with the continuous maintenance of a candidate application (i.e. if reengineering is not performed) defined as

$$C_{\text{maint}} = [P_3 - (P_1 + P_2)] * L$$

- (C_{reeng}) that is the cost associated with reengineering a candidate application defined as

$$C_{\text{reeng}} = [P_6 - (P_4 + P_5)] * (L - P_8) - (P_7 * P_9)$$

- Using equations 1 and 2 above, the overall benefit of reengineering can be computed thus:

$$\text{Cost-benefit} = C_{\text{reeng}} - C_{\text{maint}}$$

For a software organization, the cost-benefit presented above can be performed for all high priority applications identified during inventory analysis and those that show high cost-benefit can be targeted for reengineering, while others can be postponed until resources are available. The model does not specify how to compute P_4 to P_9 hence relies on costs estimation models like Constructive Cost Model II (COCOMO II), Halstead, Expert Judgment, Function Point, and Lines-of-Code.

Those applications that show the highest cost/benefit can be targeted for reengineering, while work on others can be postponed until resources are available.

In this case study the predicted parameters to calculate the cost benefit of reengineering the legacy supermarket system are as follows:

$$P_1 = 400\$, P_2 = 150\$, P_3 = 600\$,$$

$$P_4 = 100\$, P_5 = 100\$, P_6 = 1500\$,$$

$$P_7 = 150\$, P_8 = 3 \text{ months} = 0.25 \text{ years},$$

$$P_9 = 1, L = 4 \text{ years}.$$

The cost associated with the continuous maintenance of the legacy supermarket system before reengineering is equal to 200\$ and the cost associated with reengineering of the legacy supermarket system is equal to 600\$, and the overall benefit of reengineering can be computed from these two values and it is equal to 400\$.

EXPECTED RISKS

There are a number of risks that can be recognized and the reengineering teams have to take care of them, some of which are:

- **Development Environment:** Risks may be associated with the availability and quality of COTS (Common-Off-The-Shelf) tools. In this case study such risks are minimized by using Asp.net tools as RAD (Rapid application development) from Microsoft, given that the Microsoft tools are most popular and trusted tools.
- **Product Size:** Impact of risks is directly proportional to the overall size of the product that has to be re-engineered/re-build/re-modulated. In this case study the size of the legacy software is 2411 SLOC (Source Line of Code) and the number of modules is 212.
- **Experience:** Risks associated with the overall technical and project experience of the software engineers who are involved in the process of re-engineering. In this case study this risk was eliminated by the engineers taking advance technical course in ASP.net and SQL server.
- **Customer Characteristics:** Risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.
- **Technology to be built:** Risks associated with the complexity of the system to be reengineered and the "newness" of the technology that is packaged by the system. The complexity of the system can be determined through the Cyclomatic Complexity, which it is a measurement of complexity of a program code and can be calculated by using the control flow graph of the program. In this case study the total Cyclomatic complexity of the legacy supermarket management system is 578, the Mean Cyclomatic complexity is 2.726, and the decision density is 0.2397. All these values are calculated by using "Visustin v7" analysis tool and according to the following formulas:
 - Code Cyclomatic Complexity = Number of graph edges – Number of Graph Nodes + Number of connected components
 - Mean Cyclomatic complexity= Cyclomatic Complexity / Number of modules
 - Decision density= Cyclomatic complexity / Number of lines

The calculated values show that, the legacy supermarket software is a simple program, without much risk. The following table (table 1), displays the results published by SEI (Software Engineering Institute) and they are being used widely to determine the health of the code base and to evaluate the risk associated with the application [10].

Table 1: Risk Evaluation According to Mean Cyclomatic Complexity

Mean Cyclomatic Complexity	Risk Evaluation
1-10	A simple program, without much risk
11-20	More complex, moderate risk
21-50	Complex, high risk program
Greater than 50	Un testable program (very high risk)

- **Process Definition:** Risks associated with respect to the process followed by the organization to perform re-engineering process. In this case study the used model for reengineering is the FUSION model that has the cyclic features and that depends on the feedback of the customer obtained at the final step of each iteration to evaluate and regenerate next versions of the reengineered system, reducing the risks.
- **Business Impact:** The constraints imposed by management or the market place and also the domain of the business are likely to cause risks in performing reengineering process. In this case such risks are minimized through fine planning at the planning phase.

RESULTS AND DISCUSSIONS

In this paper the FUSION reengineering model is applied on legacy supermarket management system to convert it to a new system with more functionality, high security, and good quality. The FUSION reengineering model is applied to get and prove the benefit of reengineering, and the software metrics is used to make comparison between the legacy system and new system, and the result of comparison is shown in table (2). Some metrics are calculated by using “Visustin v7” analysis tool such as Size (LOC), Number of modules, Cyclomatic Complexity, Mean Cyclomatic Complexity, and Decision Density.

Three levels are used to evaluate the metrics and these levels are Low (L), Medium (M), and High (H). Result of technical comparison between legacy supermarket management system and the re-engineered system is shown in Table (3) and this comparison includes the difference in Platform, Application, Programming Language, Front end, Backend, Database, Number of tables, Number of generated reports, Execution time, Coding, Making dll (dynamic link library), Authentication levels, Stock management and Customer Feedback.

From the data in table (2) and table (3), it is proved that the reengineered system is better than the legacy system and it meets the customer need.

Table 2: Metrics Comparison between Legacy Supermarket Management System and the Re-Engineered System

		Legacy Supermarket Management System				Re-Engineered Supermarket Management System			
Complexity	Size (LOC)	2411				2323			
	Number of modules	212				139			
	Cyclomatic Complexity	578				354			
	Mean Cyclomatic complexity	2.726				2.546			
	Decision density	0.239				0.152			
Reusability	Integrability		L	M	H		L	M	H
	Testability		L	M	H		L	M	H
	Portability		L	M	H		L	M	H
	Modifiability		L	M	H		L	M	H
Quality	Functionality		L	M	H		L	M	H
	Reliability		L	M	H		L	M	H
	Performance		L	M	H		L	M	H
	Security		L	M	H		L	M	H

Table 3: Technical Comparison between Legacy Supermarket Management System and the Re-Engineered System

	Legacy Supermarket Management System	Re-Engineered Supermarket Management System
Platform	Windows	Windows & linux
Application	Windows based	Web Based
Language	VB	C#
Front end	VB	Asp.net
Backend	Sqlserver	Sqlxpress
Database Re-engineering	All tables are dependent upon the reference constraint which results in changing the condition difficult as it can lead to the removal of old data that do not match the condition.	The constraints are implemented through c# application codes which results in data independency and thereby eases alteration of condition by just changing the code with no effect on the tables and database.
Execution time	The legacy system with VB as its front end takes more execution time, because it includes the execution of both the control and its backend code.	The asp.net with active-x controls takes less time to execute as it requires the execution of only the control and not the back end code.
Coding	Visual Basic programming language which supports the structured programming paradigm is used and hence leads to rewriting the codes since VB does not support grouping control.	C# which supports OOP is used as the programming language and thus enables the programmer to reuse the predefined objects and classes in an easy manner without having any need to write them again.
Making dll (dynamic link library) library	Not supported in VB	The ASP.Net supports the development of dll files that can be used easily when needed, reducing time and effort.
Number of tables	7	10
Authentication levels	None	Tow levels, Administrator and Employee
Number of reports	3	9
Stock management	Not available	Available
Customer Feedback	Not available	Available

CONCLUSIONS

The new era is marked by the rapid changes in the computer industry that introduces new hardware and software, making older systems suffer from market competition and difficulty in maintenance. From the tabulated values in table (2) and table(3), we found that the FUSION Software re-engineering process model reduces risk level, cost, cyclomatic complexity, and effort. The use of this re-engineering approach provides a way to increase testability, portability, modifiability, functionality, reliability, performance, and security of legacy software. These results prove that the FUSION model is an effective process to perform the re-engineering of legacy software.

ACKNOWLEDGMENTS

We thank everyone at the SHIATS/department of computer science and information technology who participated in this research for many stimulating discussions. Also special thanks to Karbala University/department of computer science for their great support.

REFERENCES

1. Ahmed S. Abbas et al, "Proposed Software Re-engineering Process That Combine Traditional Software Reengineering Process With Spiral Model", International Journal of Advanced Research in Computer Science, Volume 4, No. 2, Jan-Feb 2013.

2. L. Manzella. Mutafelija.: Concept of re-engineering Life Cycle, ICSI Second International Conference On System Integration. IEEE. 1992.
3. Sandhya Tarar, Dr. Ela Kumar, "Design Paradigm and Risk Assessment of Hybrid Re-engineering with an approach for development of Re-engineering Metrics", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.1, January 2012.
4. Shekhar Singh, Significant role of COTS to design Software Reengineering Patterns, International Conference on Software Engineering and Applications(ICSEA),2009.
5. 15. Pooley R., Stevens P., Systems Reengineering Patterns, CSG internal report, 1998.
6. Brodie, Michael L., and Stonebraker, Michael, "Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach", Morgan-Kaufman Publishers, 1995.
7. Alessandro Bianchi et al, " Iterative Reengineering of Legacy Functions", IEEE International Conference on ISBN: 07695118 Year: 2001 Pages: 632-641 Provider: IEEE Publisher: IEEE.
8. Xiaohu Yang et al, "A Dual-Spiral Reengineering Model for Legacy System", TENCON 2005 - 2005 IEEE Region 10 Conference ISBN: 0780393112 Year: 2005 Pages: 1-5 Provider: IEEE Publisher: IEEE.
9. Sneed, H., "Planning the Reengineering of Legacy Systems," *IEEE Software*, January 1995, pp. 24–25.
10. SEI (Software Engineering Institute), <http://www.sei.cmu.edu/>,