



**SUDAN UNIVERSITY OF SCIENCE AND
TECHNOLOGY
COLLEGE OF GRADUATE STUDIES**



A Concerns-Based Reverse Engineering Methodology for Partial Software Architecture Visualization

**منهجية الهندسة العكسية القائمة على الإهتمامات للتصور الجزئي
لمعمارية البرمجيات**

A thesis Submitted to the College of Graduate Studies,
College of Computer Science and Information Technology,
Sudan University of Science and Technology
In Partial Fulfillment of the Requirements for the degree of
DOCTOR OF PHILOSOPHY in Computer Science

By

Hind Alamin Mohamed Hassan

Supervisor

Dr.Hany Hussein Ammar, Professor

October 2020



Approval Page

(To be completed after the college council approval)

Name of Candidate:

Thesis title: A Concerns-Based Reverse Engineering
Methodology for Partial Software Architecture
Visualization
.....
.....
.....
.....
.....
.....

Degree Examined for:

Approved by:

1. External Examiner

Name:

Signature: Date:

2. Internal Examiner

Name:

Signature: Date:

3. Supervisor

Name:

Signature: Date:

**Declaration of the Status of Thesis
(By Student)**

I signing here-under, and declare that the contents of this dissertation represent *my own work*, and that *the dissertation has not previously been submitted for academic examination* towards any qualification. Furthermore, it represents my own opinions, which is an original intellectual work.

Candidate's Name: ***Hind Alamin Mohamed Hassan.***

Candidate's Signature: *Hind A. M.*

Date: ***October 2020***

Declaration of the Status of Thesis
(By Main Supervisor)

I signing here-under, and declare that I'm the supervisor of the sole author of the Ph.D. dissertation entitled:

**A Concerns-Based Reverse Engineering Methodology
for Partial Software Architecture Visualization**

Supervisor's Name: *Prof. Hany Hussein Ammar.*

Supervisor's Signature: *Hany H. Ammar*

Date: *October 2020*

Assigning the copy-right to CGS

I signing here-under, and declare that I'm the sole author of the Ph.D. dissertation entitled:

A Concerns-Based Reverse Engineering Methodology for Partial Software Architecture Visualization

This is an original intellectual work. Willingly, I assign the copyright of this work to the College of Graduate Studies (CGS), Sudan University of Science and Technology (SUST).

Accordingly, SUST has all the rights to publish this work for scientific purposes.

Candidate's Name: ***Hind Alamin Mohamed Hassan***

Candidate's Signature: *Hind A. M.*

Date: ***October 2020***

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

فَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا
وَلْيَسِّرْ لَنَا ذُرِّيَّتَنَا

صَدَقَ اللَّهُ الْعَظِيمَ

سورة المجادلة الآية (11)

الحمد لله

الحمد لله حمداً يليق بجلاله و عظيم سلطانه وكمال صفاته، وبنعمته علينا بدين الإسلام و الصلاة والسلام على خاتم المرسلين الذي أرسل للناس كافة بشيراً ونذيراً و على آله و صحبه و من إهتدى بهديه، و إستن بسنته الى يوم الدين، الحمد لله الذي علم بالقلم، علم الإنسان ما لم يعلم، الحمد لله الذي أنعم علينا بنعمة العقل والسمع و البصر، و الحمد لله الذي بنعمته تتم الصالحات.. والحمد لله دائماً وأبداً..

DEDICATION

To those whom always remember me to believe in myself and continuous to provide me their constant love, unlimited support and encouragement for advancement and success throughout my PhD journey and my life in general. I could not have done it without your support. I am extremely grateful to

My Heartbeat (Mum and Dad),

My Sweet Sisters (Hiba, Sara and Salam),

My Lovely Husband and little Angle (My Son, Abdulrahman),

All My Big Family,

*Special People in my heart we lost them, but their souls are always with us
(Uncle Ahmed Abdalla, Uncle Tahar Osman and Uncle Mohamed Abdalla).*

To those who enlighten our way with knowledge since the first steps and throughout the whole of the education stages.

My Supervisor and My Teachers..

To those whom we spent with them all the enjoyable moments, happy times and been touch with all meaning of Friendship and Encouragement.

My best friends and My Colleagues..

ACKNOWLEDGMENT

Firstly, “*Alhamdulillah*”..

I would like to extend my sincere thanks and appreciation to a number of people for their participation and cooperation to accomplish this research:

Firstly, special thanks to my supervisor **Prof. Hany Ammar**, Who had afforded and guided me with all the needed instructions and information, inspiration, encouragement, guidance, reading drafts and many technical discussions which have improved the expected outcomes, and providing feedback to complete the aim of this research. Honestly; it was really great honor for me to work under his supervision.

My gratitude and appreciation to a number of people (*Dr.Ibrahim Abdallah Mohamed, Dr.Yahia Abdallah Mohamed, Dr.Hisham M.Abushama, Dr.Wafaa F.Mukhtar, Dr.Mohamed Mustafa Ali, Dr.Afaf Madani, Miss.Intsar Ibrahim, Dr.Amar Ibrahim and Dr.Hitham A.Moneim*) for their encouragement and continuous support me throughout my PhD journey.

I am extremely grateful to *my colleagues and all members at the College of computer science and information technology* at SUDAN University of Science and Technology for their unlimited support and encouragement.

Moreover; I place my sincere thanks to **Scientific Affair and College of Graduation Studies** at SUNAN University for their supporting, coordinating and following up through my PhD scholarship.

Last but not least, I would like to thank my special friends (*Dr.Amal Hassan, Dr.Asma, Wafaa Ali, Hind Ahmed, Hiba A.Maki, Bodoor Ali, Nahla Murtada, Sara Abdalla, Hyba Abdu, Eman A.moneim, and Bothyna Moneer*) and all my friends around the world and my colleagues for their moral help and support, and for their understanding during my PhD journey. I am lucky to have such special friends and colleagues! Thank you all.

Hind Alamin Mohamed

ABSTRACT

The use of reverse engineering (RE) is increasingly spreading and becoming one of the essential engineering trends for software evolution and maintenance. RE is used to support the process of analyzing and recapturing the design information in legacy systems or complex systems during the maintenance phase. The major problem stakeholders might face in understanding the architecture of existing software systems is that the knowledge of software architecture information is difficult to obtain because of the size of the system, and the existing architecture document often missing or does not match the current implementation of the source code of software system. Therefore, much more effort and time are needed from multiple stakeholders such as developers, maintainers and architects for obtaining and re-documenting and visualizing the architecture of a target system from its source code files. Hence, most of the current work is mainly focused on the developer's viewpoint.

To contribute in solving the mentioned problems for obtaining and re-documenting the architecture of target system; this research presents a RE methodology for visualizing architectural information for multiple stakeholders and viewpoints by applying a reverse engineering process on specific parts of the source code. The process is driven by eliciting stakeholders' concerns on specific architectural viewpoints to obtain and visualize the architectural information related to these concerns. In this research the proposed RE methodology's phases have been illustrated and validated using a case study of a legacy web application system.

The main contributions of this research are three folds: firstly; the RE methodology is based on IEEE1471 standard for architectural description and supports concerns of stakeholder including the end-user and maintainer; secondly; it supports the visualization of a particular part of the target system by providing a visual model of the architectural representation which highlights the main components needed to execute specific functionality of the target system and finally; the methodology also uses architecture styles to organize the visual architecture information, this architectural

representation helps stakeholders to inspect the dependencies of the different parts of the architecture obtained from specific source code segments of the target system.

المستخلص

إنتشر إستخدام الهندسة العكسية بشكل متزايد وأصبح يُمثل أحد الاتجاهات الهندسية الأساسية لتطوير البرمجيات وصيانتها. حيث تُستخدم الهندسة العكسية في دعم عملية التحليل وإستعادة معلومات التصميم للأنظمة القديمة أو الأنظمة المعقدة التي تم تطويرها خلال مرحلة الصيانة لها. تتمثل المشكلة الرئيسية التي يتم مواجهتها من قبل أصحاب المصلحة و المستفيدين من النظام في صعوبة الحصول ومعرفة المعلومات الخاصة بمعمارية النظام وذلك بسبب حجم النظام، كما أنه غالبًا ما يكون التوثيق للمعمارية الحالية للنظام مفقود أو أنه لا يتوافق مع التنفيذ الحالي لمصدر(أو شفرة) البرنامج المتوفر للنظام. لذلك نجد أنه يتم بذل المزيد من الجهد والوقت من جانب العديد من أصحاب المصلحة مثل: المطورين ومهندسي الصيانة والمعمارين في الحصول على معمارية النظام وإعادة التوثيق والتصور لبنية ومعمارية النظام المستهدف من خلال التركيز على ملفات مصدر(شفرة) البرنامج الخاصة بالنظام. نجد أن الأعمال المقدمة والحالية لعكس معمارية النظام المستهدف تُركز وبشكل أساسي على وجهة نظر المطور أو المبرمج للنظام.

للمساهمة في حل المشاكل المذكورة للحصول على معمارية النظام وإعادة التوثيق لمعمارية النظام المستهدف؛ يقدم هذا البحث منهجية تعتمد على إستخدام الهندسة العكسية لتصوير المعلومات المعمارية للعديد من أصحاب المصلحة بناءً على وجهات النظر الخاصة بهم من خلال تطبيق عملية الهندسة العكسية على أجزاء محددة من مصدر أو شفرة البرنامج. هذه العملية تكون موجهة من خلال إستخلاص الاهتمامات الخاصة بأصحاب المصلحة وتحديد وجهات نظر محددة ليطم اعتمادها في وضع التصور لجزء محدد من المعلومات المعمارية المتعلقة بهذه الاهتمامات التي تم تحديدها. تم في هذا البحث توضيح مراحل المنهجية المقترحة والتحقق منها بإستخدام دراسة حالة وتطبيقها على أحد أنظمة تطبيقات الويب القديمة.

من أهم الإسهامات الرئيسية لهذه المنهجية المقترحة هي أولاً؛ تستند منهجية الهندسة العكسية على معيار أساسي هو IEEE1471 المستخدم لوصف معمارية الأنظمة البرمجية التي يتم تطويرها، كما تدعم اهتمامات أصحاب المصلحة بما في ذلك المستخدم النهائي ومهندس الصيانة لهذه الأنظمة. ثانياً؛ تدعم المنهجية التصور لجزء محدد من النظام المستهدف من خلال توفير نموذج توضيحي للتمثيل المعماري و الذى يستعرض المكونات الرئيسية اللازمة في تنفيذ الوظائف المحددة للنظام المستهدف، وختاماً؛ إستخدمت المنهجية المقترحة طريقة و نمط محدد لتنظيم وتمثيل المعلومات المعمارية حيث يساعد هذا التمثيل أصحاب المصلحة على فهم ومتابعة هذه المعلومات المعمارية التي تم الحصول عليها من خلال التركيز على أجزاء ومقاطع محددة من مصدر/ شفرة البرنامج وفقاً لطبيعة النظام المستهدف.

TABLE OF CONTENTS

الْحَمْدُ لِلَّهِ	i
DEDICATION.....	ii
ACKNOWLEDGMENT	iii
ABSTRACT	iv
المستخلص.....	vi
TABLE OF CONTENTS.....	vii
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xiv
LIST OF PUBLICATIONS.....	xv
LIST OF APPENDICES	xvi
CHAPTER I.....	1
INTRODUCTION.....	1
1.1. Introduction	1
1.2. Problem Statement.....	2
1.3. Research Questions	3
1.4. Research Hypotheses.....	3
1.5. Research Objectives	4
1.6. Research Scope.....	4
1.7. Research methodology	5
1.8. Research Contributions	5
1.9. Thesis Organization.....	6

CHAPTER II	7
BACKGROUND AND RELATEDWORK.....	7
2.1. Introduction	7
2.2. Reverse Engineering Definitions.....	7
2.3. Software Architecture Definition	8
2.4. Literature Review	9
2.4.1. Reverse Engineering for Understanding Software Artifacts.....	10
2.4.2. Model Driven Reverse Engineering (MDRE).....	11
2.4.3. Documenting of Architectural Design Decisions (ADDs)	12
2.4.3.1. Collecting of Architectural Design Decisions	13
2.4.3.2. Scenario-Based Documentation and Evaluation Method	14
2.4.3.3. UML Metamodel.....	14
2.4.4. Comparison of Existing Architectural Design Decisions Models ...	15
2.5. Comparison with Related Work	16
2.6. OPEN ISSUES	22
2.7. Chapter Summary	23
CHAPTER III.....	24
CONCERNS-BASED RE METHODOLOGY FOR EXTRACTING	
PARTIAL SOFTWARE ARCHITECTURE	24
3.1. Introduction	24
3.2. Overview of the Proposed RE Methodology	24
3.3. The Principles of RE Methodology.....	25
3.4. The Main Phases of RE Methodology	27

3.4.1. Define stakeholders concerns based on architectural viewpoint	28
3.4.1.1. Select viewpoint from a given catalog.....	28
3.4.1.2. Categorize stakeholder’s concerns related to selected viewpoint.	30
3.4.2. Elicit specific stakeholder concern.....	32
3.4.3. Extract related requirement information based on elicited concern	34
3.4.3.1. Extraction of related requirement information	37
3.4.3.1.1. Define Full-Text Index.....	37
3.4.3.1.2. Select Full-Text searching mode:	38
3.4.3.1.3. Relevance in Full-Text searching:	38
3.4.3.2. Traceability among specific concern and its related information.	41
3.4.4. RE for extracting particular architectural information.....	41
3.4.4.1. RE process for extracting specific source code files	41
3.4.4.2. Representation of the particular architectural information	43
3.4.4.2.1. Mapping extracted code into a component architecture	44
3.4.4.2.2. Visualizing architectural information using ArcheType	44
3.5. Chapter Summary	47
CHAPTER IV	49
IMPLEMENTATION OF RE METHODOLOGY TO CASE STUDY	49
4.1. Introduction	49
4.2. Selecting Software System for a Case Study	49
4.3. Applying RE Methodology Phases to the Case Study	51
4.3.1. Define a set of stakeholders concerns	51
4.3.2. Elicit a specific stakeholders concern:	52

4.3.3. Extract requirements information based on elicited concern.....	52
4.3.4. Extracting architectural information	54
4.3.4.1. Extracting specific source code files.....	54
4.3.4.2. Representation and Visualization of architectural information	56
4.4. The Main Results and Discussion	60
4.5. Chapter Summary	62
CONCLUSION AND FUTURE WORK	63
APPENDICES	66
APPENDIX A	66
APPENDIX B	71
APPENDIX C	73
APPENDIX D	75
REFERENCES.....	76

LIST OF TABLES

Table 2.1 Examples of some methodologies and approaches for documenting software architecture	18
Table 2.2 Summarization of important related approaches and methodologies	21
Table 3.1 Functional Viewpoint Catalog	30
Table 3.2 Functional viewpoint: Stakeholders and Concerns	31
Table 3.3 Summary of <i>Phase(1)</i> of RE Methodology	32
Table 3.4 The application archetypes summary from MICROSOFT Architecture guide	45
Table 4.1 TMS source code overview	51

LIST OF FIGURES

Figure 2.1 RE thorough Complementary Software View	10
Figure 2.2 MoDisco Framework's Architecture	12
Figure 2.3 Triple View Model Framework	13
Figure 3.1 Overview of RE Methodology	25
Figure 3.2 IEEE1471 Conceptual Framework	26
Figure 3.3 The RE Methodology's Phases	27
Figure 3.4 The Viewpoints Catalog	29
Figure 3.5 The <i>Phase(2)</i> of RE Methodology	32
Figure 3.6 Elicitation specific stakeholder's concern(s)	33
Figure 3.7 The <i>Phase(3)</i> of RE Methodology	34
Figure 3.8 Example of the Requirement repository information	35
Figure 3.9 Tracing Specific Concern to its Related Requirement Information	36
Figure 3.10 Calculation formula for the weight in MySQL Database	39
Figure 3.11 The <i>Phase(4)</i> of RE Methodology	41
Figure 3.12 The Code Analyzer Process	42
Figure 3.13 Example of Mapping Code's Element into Component	44

LIST OF FIGURES

Figure 3.14 Example of Visualizing Architectural Information using Layered Architecture Model	47
Figure 4.1 An overview of the main contents of TMS system	50
Figure 4.2 Elicit a specific stakeholder functional concern	52
Figure 4.3 Extraction of related Requirements Information	53
Figure 4.4 Traceability among specific concerns and related Requirement Information	54
Figure 4.5 Applying Code Analyzer Process	55
Figure 4.6 Extracted Call Graph for executing “ <i>TMS_Req2.20</i> ” functionality	55
Figure 4.7 Mapping extracted code elements into Components Architecture	56
Figure 4.8 Visualizing particular architectural information using Layered Architecture Model	58
Figure 4.9 Representation of particular Architectural Information based on stakeholder’s Functional Concern	59
Figure 4.10 Example of the main components that implement the core functionality	60
Figure 4.11 Example of how to determine and decide a new feature for a target system	61

LIST OF ABBREVIATIONS

RE	Reveres Engineering
RQs	Research Questions
ROs	Research Goals
MDRE	Model Driven Reverse Engineering
ADDs	Architectural Design Decisions
UML	Unified Modeling Language
SBSE	Search Based Software Engineering
TVM	Triple View Model
EA tool	Enterprise Architect tool
ATM	Automatic Teller Machine
TMS	Timetable Management System
GUI	Graphical User Interface
CID	Functional Concern ID
PML	Process Modelling Language
FR	Functional Requirement
IEEE Std1471-2000	Institute of Electrical and Electronics Engineers Standard 1471-2000 (also known as Conceptual framework for architectural description)
MySQL Database	MySQL is one of the most popular open-source database systems. It based on a universal language known as SQL (Structured Query Language) which has been developed and accepted as the definitive model for relational database management systems (RDBMS).
PHP	Personal Home Page is an open source technology that is supported by large community of users and developers and becoming one of the most popular server side scripting language for creating dynamic web pages.

LIST OF PUBLICATIONS

- Hind Alamin M. and Hany. H Ammar, "*Reverse Engineering for Documenting Software Architectures, a Literature Review*", International Journal of Computer Applications Technology and Research, vol. 3(2014), no.12, pp. 785 - 790, Dec 2014, ISSN: 2319-8656 (Online).

- Hind Alamin M. and Hany. H Ammar, "*Concerns-Based Reverse Engineering for Partial Software Architecture Visualization*", International Journal on Informatics Visualization, vol. 4(2020), no.2, pp. 58 - 68, Apr 2020, ISSN: 2549-9610 (Online).

LIST OF APPENDICES

APPENDIX A	The Execution of GUI Prototype Tool	67
APPENDIX B	Timetable Management System (TMS): The detailed description of the Privileges of System's Users	72
APPENDIX C	Viewpoint Catalog (Rozanski & Woods, 2011)	74
APPENDIX D	Certification of Publication	76

CHAPTER I

INTRODUCTION

1.1. Introduction

The use of reverse engineering (RE) is increasingly spreading and becoming one of the essential engineering trends for software evolution and maintenance. Generally; RE is defined as the way of analyzing an existing software system to identify its current components and the dependencies between these components to recover design information, and create new forms of system representations (Chikofsky & James, 1990) (Rosenberg & Lawrence, 1996) (Penta & Massimiliano, 2008) (Garg & Jindal, 2009).

Furthermore; software architecture is defined by the recommended practice of (ANSI/IEEE Std1471-2000) as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution. Therefore, software architecture focuses on how the major elements and components within software application are used by or interact with other elements and components (Institute of Electrical and Electronics Engineers, 2000) (R.Hilliard, D. Emery, M. Maier, 2007).

The core of RE consists of extracting information from the available software artifacts (such as: source code) and representing it into visual models to be understandable by stakeholders (Harman, et al., 2013). The main objectives of RE are focused on generating alternative views of system's architecture, recapture design information, re-documentation of software system, facilitate software system's reuse, and represent software systems at higher level of abstractions (by putting the system's users in the maintenance loop so that users can give feedback on the information related to the target system) (Garg & Jindal, 2009).

The software documentation is essential for the system's stakeholders (such as: developers, end-users, testers, maintainers, architects, system administrators) to decide on activities in order to evolve and maintain the software system. For example source code is considered as the detailed documentation for the software system implementation, and in most cases, it is the only source of information that up to date and available for legacy software systems (Rosenberg & Lawrence, 1996) (Garg & Jindal, 2009).

Recovering and documenting software architectures (either fully or partially) has been an area of active research where programmers, architects, maintainers, testers and software engineers spend a lot of time using their expertise in resolving such problems of mapping existing source code of a target system into architecture components and for supporting the understand-ability and maintainability of software systems.

1.2. Problem Statement

The RE is used to support recapturing the design information for restructuring the architecture into more maintainable architecture. Hence, most of the companies rely on reengineering the legacy systems which are important for their business process and keep them in operations (Rosenberg & Lawrence, 1996) (Harman, et al., 2013).

The major problem stakeholders might face in understanding the architecture of existing software systems is that the knowledge of software architecture information is difficult to obtain because of the size of the system, and the existing architecture document often missing or does not match the current implementation of the source code of software system. Therefore, much more efforts and time are needed from multiple stakeholders such as developers, maintainers and architects for obtaining and re-documenting and visualizing the architecture of a target system from its source code files. Hence, most of the current work is mainly focused on the developer viewpoint.

Previous research made great progress to overcome the problems of documenting and recovering software architectures to reflect the system's changes at the code level. However to deal with complex legacy systems, there is a significant need to develop new RE approaches or methods for documenting the partial architecture of the target system in order to simplify and visualize the available information of complex architectures.

Additionally, these approaches should be based on stakeholders concerns and their decisions about the architecture of the target system. Hence, it's important to determine what to look for and focus in obtaining specific information on the architecture of the implemented software system.

1.3. Research Questions

The following are the research questions (RQs) of this research:

- **RQ(1):** Which Industry standard that will be used for visualizing the architectural Information of software system?
- **RQ(2):** How this standard could be used to develop RE Methodology for architecture description?
- **RQ(3):** How to visualize particular architectural information, so it can support such a stakeholder's concern for end-user and maintainer?

1.4. Research Hypotheses

The hypothesis behind the proposed methodology of this research concerns with the possibility to represent a flexible reverse engineering methodology for obtaining and re-documenting the architecture of a target system from its source code files. The methodology will focus on extracting and visualizing the architectural information for multiple stakeholders and viewpoints based on applying the RE process on specific parts of the implemented source code of the target software.

Accordingly, the extraction and visualization of information documenting the partial of the architecture in order to simplify and visualize the available information of

complex architectures. As the result, this visual architecture information will support the understand-ability and the maintainability process for particular parts of the software system.

1.5. Research Objectives

The general goal is mainly to design an RE methodology for extracting a particular architectural information based on applying the RE process on existing source code of a target system. Additionally; the other specific objectives are highlighted as follows:

- Investigate the current state of existing work related to the reverse engineering methodologies and the documenting approaches for software's architecture.
- Develop a methodology based on the industry standard of architecture description to visualize the architectural information.
- Implement and validate a methodology' phases in a case study of a legacy system.

1.6. Research Scope

This research will propose a methodology that focuses on using the RE approach on source code for visualizing and re-documenting the architecture of software. Besides that adapting of the extraction of the architectural information on specific stakeholders' concerns about the target system. The proposed methodology will be based on the industry standard of architecture description to visualize the architectural information, and validate by applying the proposed methodology's phases using a case study of a legacy web application system. The reason for choosing these types of applications is that it became well known and most of existing applications were developed based on them.

1.7. Research methodology

This research starts to investigate the existing work related to the reverse engineering methodologies and approaches and the documenting approaches for software's architecture; then present a survey to determine the gaps and the suggested challenges that will need to focus on as a research area.

Based on the results of a survey, propose RE methodology as an alternative solution for extracting and visualizing the architectural information of the target system from its source code files. The proposed RE methodology will base on the conceptual framework in IEEE1471-2000 standard for the architectural description; also known as the conceptual framework for architectural description (Institute of Electrical and Electronics Engineers, 2000).

Generally; the extraction process of the proposed methodology will be totally driven by addressing specific stakeholder's concern about the target system for extracting a particular architectural information. Furthermore; the proposed methodology will extend additional stakeholders beside the developer viewpoints to supports the understandability and maintainability of legacy software systems. Finally, validate the main phases of proposed methodology using a case study in a legacy software system.

1.8. Research Contributions

The main contributions of this research can be summarized as follows:

- A new methodology that supports the IEEE1471 standard for architectural description and, supports the concerns of stakeholder including End-user and Maintainer.
- Prototype tool to support the main phases of methodology.
- Verification of the methodology using a legacy web application system, (called Timetables Management System).

1.9. Thesis Organization

The rest of this thesis is organized as the following chapters: *Chapter II* explains the main concepts of the reverse engineering and the software architecture definition, and outlines the main objectives of RE, investigates some of the related works on the reverse engineering from different perspective and highlights the summarization of important related work. *Chapter III* presents an overview of the proposed methodology (*RE Methodology*); discusses the principles of the proposed methodology, and describes the detailed design of the main phases of the methodology. *Chapter IV* describes how to apply the proposed RE methodology's phases to a practical case study, and discusses the main results from applying the methodology's phases to a practical case study. *Chapter VI* concludes with the main contributions and highlights the future research based on the methodology verification results. Finally, the end of thesis presents the references and the appendices.

CHAPTER II

BACKGROUND AND RELATEDWORK

2.1. Introduction

This chapter describes an overview of the reverse engineering definitions, software architecture definition, and outlines the main objectives of RE. The following section presents a literature review of common existing research on the reverse engineering from different perspectives that form the current state of the art in documenting software architectures. Finally; the last section highlights the new research areas as open issues for future works, and concludes with summarizing the main contributions and the future research.

2.2. Reverse Engineering Definitions

The reverse engineering (RE) has become one of the major engineering trends for software evolution. RE is defined as the process of analyzing an existing system to determine its current components and the relationship between them. This process extracts and creates the design information and new forms of system representations at a higher level of abstraction (Garg & Jindal, 2009).

According to the main RE concepts; some of the researchers classified RE into two types: *hardware* and *software* reverse engineering. Hardware reverse engineering is based on expertise and concerns with taking a part of the device to show how it works, with respect to the copyright and trade secrets with the original design. While software reverse engineering concerns with studying how the program performs its operations, investigate and correct errors or limitations. Furthermore, software reverse engineering allows the retrieving and generating the source code of program in case the code is lost or for recapturing the design information of a target system (Garg & Jindal, 2009; Rosenberg & Lawrence, 1996).

Garg et al. categorized the software engineering into forward engineering and reverse engineering; and both of these types are essential in the software

development life cycle. The forward engineering refers to the traditional process for developing software which includes: gathering requirements, designing and coding process till reach the testing phase to ensure that the developed software satisfied the required needs. While reverse engineering defined as the way of analyzing an existing system (without changing its overall functionality) to identify its current components and the dependencies between these components to recover the design information, and other forms of system representations (Garg & Jindal, 2009). However, some of the researches suggested integrating the reverse and forward engineering processes for large systems to achieve long term evolution and increase the productivity of these systems as discussed in (Chikofsky & James, 1990; Penta & Massimiliano, 2008; Rosenberg & Lawrence, 1996).

Legacy systems are old existing systems which are important for business process. Companies rely on these legacy systems and keep them in operations. Therefore, reverse engineering is used to support the software engineers in the process of analyzing and recapturing the design information of complex and legacy systems during the maintenance phase (Rosenberg & Lawrence, 1996; Harman, et al., 2013).

2.3. Software Architecture Definition

Software architecture is defined by the recommended practice (ANSI/IEEE Std1471-2000) as: the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution (Institute of Electrical and Electronics Engineers, 2000; R.Hilliard et al., 2007).

The recovering and documenting software architectures (either fully or partially) has been an area of active research where programmers, architects, maintainers, testers and software engineers spend a lot of time using their expertise in resolving such problems of mapping existing source code of a target system into architecture components and for supporting the understand-ability and maintainability of software systems.

Furthermore; previous research such as (Chikofsky & James, 1990; Garg & Jindal, 2009; Harman, et al., 2013; Kumar, 2013) made great progress to overcome the problems of documenting and recovering software architectures to reflect the system's changes at the code level. However to deal with complex legacy systems, there is a significant need to develop a new RE approaches or methods for documenting the only part of the architecture in order to simplify and visualize the available information of complex architectures. This should be based on stakeholders concerns and their decisions about the architecture of the target system. Hence, it's important to determine what to look for and focus in obtaining specific information on the architecture of the implemented software system.

The main objectives of RE are focused on generating alternative views of system's architecture, recapture design information, re-documentation of software system, facilitate software system's reuse, and represent software systems at higher level of abstractions (by putting the system's users in the maintenance loop so that users can give feedback on the information related the target system). Furthermore; RE is used to support recapturing the design information for restructuring the architecture into more maintainable architecture (Chikofsky & James, 1990; Garg & Jindal, 2009; Harman, et al., 2013).

The following sections of this chapter present a literature review of the common existing researches on reverse engineering from different perspectives, and highlights the new research areas as open issues for future works. Finally, the last section concludes with summarizing the main contribution and the future research.

2.4. Literature Review

Program understanding plays a vital role in most of software engineering tasks. In fact; the developers use the software documentation to understand the structure and behavior of existing systems (Harman et al., 2014; Kumar, 2013). However, the main problem that developers face is that the design document or others software artifacts were out-of-date to reflect the system's changes. As a result, more effort and time needed for understanding the software rather that modifying it.

The following sub sections will introduce the most common reverse engineering approaches that focused in documenting the architecture of software from different perspectives.

2.4.1. Reverse Engineering for Understanding Software Artifacts

The developers should understand the source code based on the static information and dynamic information as described in (Kumar, 2013). The static information explained the structural characteristic of the system. While dynamic information explained the dynamic characteristics or behaviors of the system. Hence, these details help the developers on understanding the source code in order to maintain or evaluate the system. However, Kumar clarified that few reverse engineering tools supported both of dynamic and static information. Therefore, presented alternative methodology to extract the static and dynamic information from existing source code.

This methodology focused on using one of the RE tools; namely, *Enterprise Architect* (EA) to extract the static and dynamic views. Additionally, all of the extracted information was represented in form of Unified Modeling Language (UML) models. The main purpose was to get the complementary views of software in form of state diagrams and communication diagrams. The stages of this methodology are summarized and shown in Figure 2.1.

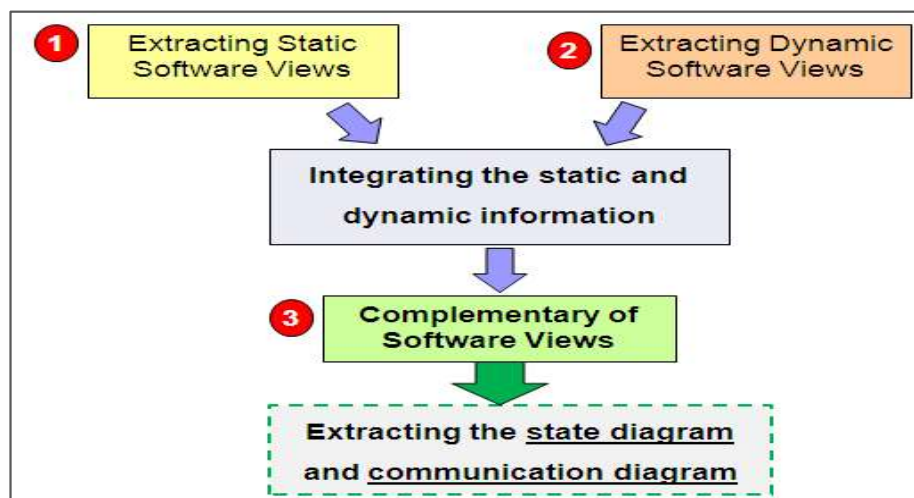


Figure 2.1 RE through Complementary Software Views (Kumar, 2013)

The Kumar's proposed methodology was very useful for supporting developers to understand the software artifacts of existing software systems. However, the methodology needs to support additional stakeholder beside the developers in order to identify the stakeholders' concerns and their decisions about the whole system.

2.4.2. Model Driven Reverse Engineering (MDRE)

MDRE was proposed as described in (Hugo, et al., 2014) to improve the traditional reverse engineering activities and legacy technologies. It is used to describe the representation of derived models from legacy systems to understand their contents. However, most of MDRE solutions focused on addressing several types of legacy system scenarios, but these solutions are not complete and they do not cover the full range of legacy systems. The work also introduced several reverse engineering processes such as: the technical/functional migration, processes of MDRE (Hugo, et al., 2014).

Recently, Hugo et al. presented a generic and extensible MDRE framework called "*MoDisco*". This framework is applicable to different refactoring and re-documentation techniques (Hugo, et al., 2014).

The architecture of *MoDisco* is represented in three layers, each layer is comprised of one or more components (see Figure 2.2). The components of each layers provided high adaptability because they are based on the nature of legacy system technologies and the scenario based on reverse engineering.

However, *MoDisco* framework was limited to traditional technologies such as: JAVA, JEE (including JSP) and XML. This framework needs to be extended to support additional technologies and to add more advanced components to improve the system comprehension, and expose the key architecture design decisions.

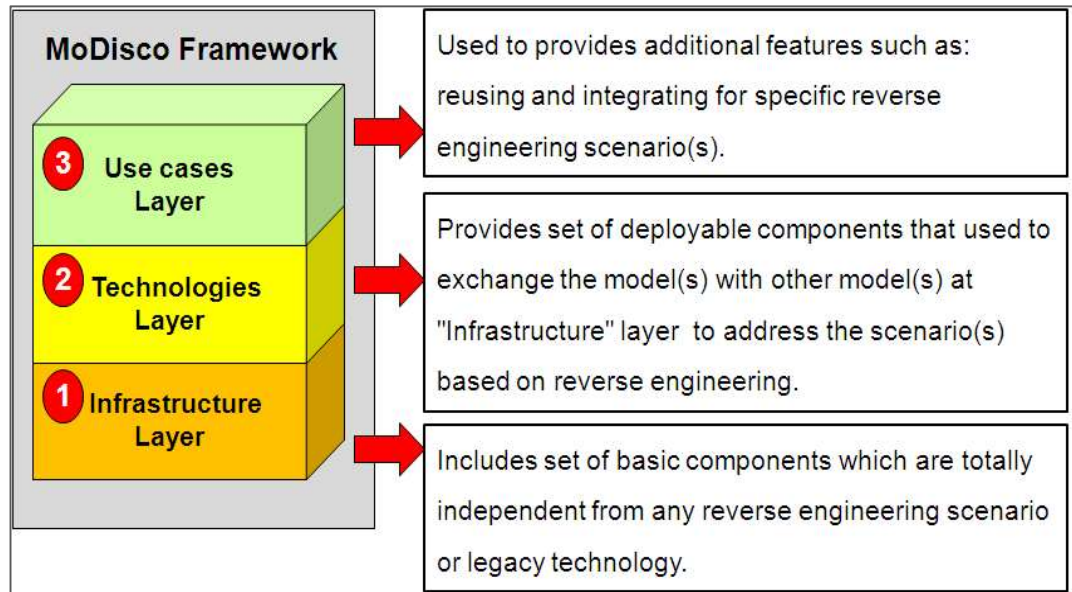


Figure 2.2 MoDisco Framework's Architecture (Hugo, et al., 2014, p.9)

2.4.3. Documenting of Architectural Design Decisions (ADDs)

Historically, Shaw and Garlan introduced the concepts of software architecture and defined the system in terms of computational components and interactions between these components as indicated in (Nicholas, 2005). Furthermore, Perry and Wolf defined software architecture in terms of elements, their properties, and the relationships among these elements. They suggested that the software architecture description is the consequence of early design decisions as indicated in (Nicholas, 2005).

The software architecture development is based on a set of architectural design decisions (ADDs). This is considered as one of the important factors in achieving the functional and non-functional requirements of the system as introduced in (Che, 2013). Che explained that the process of capturing and representing ADDs is very useful for organizing the architecture knowledge and reducing the possibility of missing this knowledge (Che, 2013).

Furthermore, the previous research focused on developing tools and approaches for capturing, representing and sharing of the ADDs. However, Che clarified that most of the previous research proposed different methods for documenting

ADDs, and these methods rarely support architecture evaluation and knowledge evaluation in practice (Che, 2013).

Furthermore, (Che & Dewayne, 2011) presented an alternative approach for documenting and evaluating ADDs. This approach proposed solutions described in the following subsections:

- Collecting of Architectural Design Decisions
- Scenario-Based Documentation and Evaluation Method
- UML Metamodel

2.4.3.1. Collecting of Architectural Design Decisions

The first solution focused on creating a general architectural framework for documenting ADDs called the Triple View Model (TVM). The framework includes three different views for describing the notation of ADDs as shown in Figure 2.3. It also covers the features of the architecture development process.

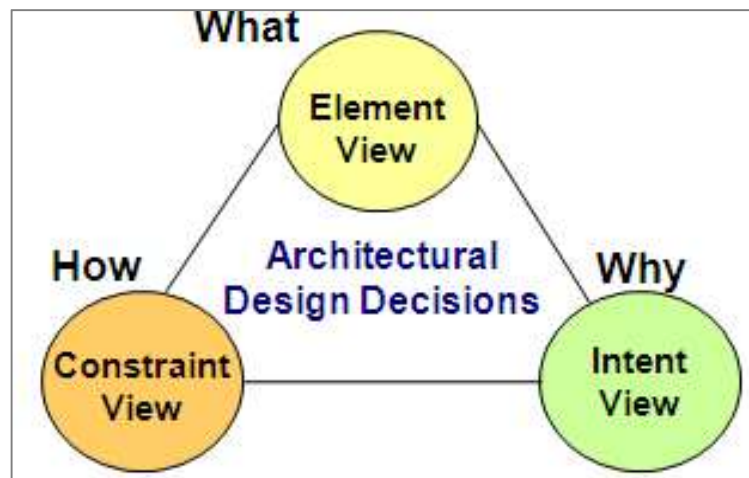


Figure 2.3 Triple View Model Framework (Che, 2013, p.1374)

As it shown in Figure 2.3; the *Element View* describes the elements that should be defined to develop the architecture; such as: computation elements, data elements, and connector elements. The *Constraint View* explains how the elements interact with each other by defining what the system should do

and not to do, the constraint(s) on each element of the element view. Additionally, define the constraints on the interaction and configuration among the elements.

Finally, the *Intent View* includes the rationale decision that made after analyzing all the available decisions, Moreover, the selection of styles and patterns for the architecture and the design of the system (Che, 2013).

2.4.3.2. Scenario-Based Documentation and Evaluation Method

The second solution called *SceMethod* is based on the TVM framework. The main purpose is to apply the TVM framework by specifying its views through the end-user scenarios; then manage the documentation and the evaluation needs for ADDs as discussed in (Che & Dewayne, 2012; Che, 2013).

2.4.3.3. UML Metamodel

The third solution is focused on developing the UML Metamodel for the TVM framework. The main purpose was to make each view of TVM specified by classes and a set of attributes for describing ADD information. Accordingly, this solution provided the following features as discussed in (Che, 2013): a) establish traceable evaluation of ADDs, b) apply the evaluation related to the specified attributes, c) support multiple ways on documenting during the architecture process and allow explicit evaluation knowledge of ADDs.

Furthermore, TVM and SceMethod solution was validated in using a case study to ensure the applicability and the effectiveness. Supporting the ADD documentation and evaluation in geographically separated software development (GSD) is currently work in progress as mentioned and stated in (Che, 2013).

2.4.4. Comparison of Existing Architectural Design Decisions Models

Researchers made a great of effort to present related tools and models for capturing, managing, and sharing the ADDs. These proposed models were based on the concept of architectural knowledge to promote the interaction between the stakeholders and improve the architecture of the system as mentioned in (Shahin, et al., 2009; Che, 2013).

Accordingly (Shahin, et al., 2009) presented a comparison study that is based on surveying and comparing the existing architectural design decisions models. Their comparison included nine ADD models and used six criteria based on desired features as discussed in (Shahin, et al., 2009). The main reason was to investigate the ADD models to decide if there are similarities and differences in capturing the ADDs. Moreover, the study aimed at finding the desired features that were missed according to the architecture needs.

The authors in (Shahin, et al., 2009) classified the ADD elements into two categories: *major elements* and *minor elements*. The major elements refer to the consensus on capturing and documenting ADDs based on the constraints, rationale, and alternative decisions. While the minor elements refer to the elements that used without consensus on capturing and documenting the ADDs, such as: stakeholders, problem, group, status, dependency, artifacts, and phase/iteration.

The main observations of this comparison study in (Shahin, et al., 2009) are highlighted as the following points: 1) all of the selected ADD models included the major elements and used different terms to express similar concepts of the architecture design; 2) most of ADD models used different minor elements for capturing and documenting ADDs; 3) all the selected ADD models deal with the architecture design as a decision making process; 4) While not all of them are supported by tools, some were based on only textual templates for capturing and documenting ADDs; 5)The main important observation was that most of existing ADD tools do not provide support for ADD personalization which

refers to the ability of stakeholders to communicate with the stored knowledge of ADD in (Shahin, et al., 2009) that based on their own profile.

2.5. Comparison with Related Work

The core of RE consists of extracting information from the available software artifacts such as source code and translating it into abstract representations to be understandable by the stakeholders (Chikofsky & James, 1990; Rosenberg & Lawrence, 1996; Harman, et al., 2013).

Accordingly; (Stringfellow et al., 2006) discussed that reverse architecting is a specific type of reverse engineering, and stated that the RE process should consist of three phases starting with an extraction phase where we extract information from the source code and document it in documentation, and documented system history. The process also include an abstraction phase which abstracts the extracted information based on the objectives of RE activity, then elicits the extracted information into a manageable amount of information. And finally a presentation phase that represents the abstracted data in a way suitable for the stakeholders.

Software architecture consists of the description of components and their relationships and interactions, both statically and behaviorally as described in (Clements, et al., 2010; Riva & Yang, 2002; Stringfellow, et al., 2006; Che, 2013).

Chikofsky et al. discussed that the RE process helps to generate the documentation to recover the design information of the system by analyzing the software to identify the components and the interrelationships between these components, and to create a representations of the software system (Chikofsky & James, 1990).

Previous research made great strides to overcome the problem of documenting and recovering the software architecture to reflect the system's changes. Therefore, several approaches, methods, frameworks and RE methodologies have been proposed form different perspectives such as the following works (Harman, et al., 2013; Clements, et al., 2010; Riva & Yang, 2002; Stringfellow, et al., 2006; Len Bass & Celements, 2003; Lau & Tran, 2012; Panas , et al., n.d.; Razavizadeh , et al.,

2009; Demeyer, et al., 2008; Arshad & Lau , 2017). The most important of these proposed approaches were based on the concept of architectural knowledge as discussed in (Che, 2013; Shahin, et al., 2009). They promote the interactions between the stakeholders to improve the architecture of the software system.

Moreover; some of the recent approaches and techniques considered the perspective of getting the executable architecture from existing source code of software system as in (Lau & Tran, 2012; Maras, et al., 2009; Arshad & Lau , 2017). These techniques considered every line of code for extracting the architecture of a target system. However, these extracted architecture were reflected every functionality exists in the original source code. For example; Arshad et al. proposed a RE model called (X-MAN) for extracting executable architecture in form of component model based on object oriented source code (Arshad & Lau , 2017).

The executable architecture contains structural and behavioral aspects of software system in analyzed manner, and the extracted components can be used to support the re-usability of component and integrated them with other systems as described in (Lau & Tran, 2012; Arshad & Lau , 2017).

For further information; we presented a survey paper indicated in (Alamin & Ammar, 2014). This survey paper reflects the current state of art in documenting and recovering software architectures using RE techniques. We highlighted and compared set of existing RE methods and approaches based on their findings and limitations (for more information see Table 2.1). However, the main observation indicates that most of these existing methods and approaches are mainly focused on the developer viewpoint as the main stakeholder; and based to reflect the whole architecture of software system(see Table 2.2 the summarization of important approaches and methodologies).

Furthermore, the recent approaches and methods discussed the need for alternative solutions to extend additional stakeholders. The solutions should focus to communicate with the stored architectural information by applying the scenario based documentation through stakeholders' scenarios and managing the

architecture's documentation of software system. However; these issues should simplify and classify the architectural information based on identifying stakeholders' concerns and viewpoints about the target system, and visualize the architectural information in a proper level of abstractions based on these stakeholders' concerns.

Table 2.1 Examples of some related methodologies and approaches for documenting software architecture adapted from (Alamin & Ammar, 2014, p.788)

(year)	Problem Statement	Proposed Solution(s)	Results and Findings	Limitation(s)
Kumar (2013)	RE method for understanding the software artifacts	Alternative methodology to extract the static and dynamic information from the source code. The main purpose is to get complementary views of software systems.	Supports the <i>developers</i> to achieve RE goals in order to understand the artifacts of software systems.	This methodology needs to support additional stakeholder beside the developers in order to identify the stakeholders' concerns and their decisions about the whole system.
Hugo et al. (2014)	Understanding the contents of the legacy systems using model driven reverse engineering (MDRE)	Generic and extensible MDRE framework called "MoDisco". This framework is applicable to different types of legacy systems.	MoDisco provided high adaptability because it is based on the nature of legacy system technologies and scenario(s) based on RE.	MoDisco should extend to support additional technologies and include more advanced components to improve system comprehension.
Che et al. (2011)	Collecting architectural design decisions (ADDs)	Triple View Model (TVM) an architecture framework for documenting ADDs.	TVM framework includes three different views for describing the notation of ADDs. TVM covers the main features of the architecture process.	TVM framework should extend to manage the evaluation and documentation of ADDs by specifying its views through the stakeholders' scenarios.

Table 2.1 Examples of some related methodologies and approaches for documenting software architecture adapted from (Alamin & Ammar, 2014, p.788)

(year)	Problem Statement	Proposed Solution(s)	Results and Findings	Limitation(s)
Che et al.(2012)	Managing the documentation and evolution of the architectural design decisions	Scenario based method (<i>SceMethod</i>) for documenting and evaluating ADDs. This solution is based on TVM. The main purpose is to apply TVM for specifying its views through end-user scenario(s).	Manage documentation and the evaluation needs for ADDs through stakeholders' scenario(s).	There is a need to support multiple ways on managing and documenting the ADDs during the architecture process.
Che (2013)	Documenting and evolving the architectural design decisions	Developed UML Metamodel for the TVM framework. The main purpose was to make each view of TVM specified by classes and a set of attributes for describing ADDs information.	Apply the evaluation related to the specified attributes and establish traceable evaluation of ADDs. Allow explicit evaluation knowledge of ADDs. Support multiple ways for documenting ADDs during the architecture process.	This solution is focused on the developers view point and their work is currently in progress to support the ADD documentation and evaluation in geographically separated software development.

Table 2.1 Examples of some related methodologies and approaches for documenting software architecture adapted from (Alamin & Ammar, 2014, p.788)

year	Problem Statement	Proposed Solution(s)	Results and Findings	Limitation(s)
Shahin et al. (2009)	A survey of architectural design decision models and tools	<p>The purpose of this survey was to investigate ADD models to decide if there are any similar concepts or differences on capturing ADD.</p> <p>The survey classified ADD concept into two categories: <i>Major elements</i> refer to the consensus on capturing and documenting ADD based on the constraint, rationale and alternative of decision. While the <i>Minor elements</i> refer to the elements that used without consensus on capturing and documenting ADD.</p> <p>Moreover, to clarify the desired features that are missed according to the architecture needs</p>	<ul style="list-style-type: none"> - All selected ADD models include the <i>major elements</i>. - Most of ADD models are based on using different <i>minor elements</i> for capturing and documenting the ADD. - All selected ADD models deal with the architecture design as the decision making process. - Not all models were supported by tools. Hence, some of these ADD based on <i>text template</i> for capturing and documenting ADDs. <p>However, most of existing ADD tools do not support the ability of stakeholders to communicate with the stored knowledge of ADD.</p>	There is a need to focus on stakeholder to communicate with the stored knowledge of ADDs. This could be achieved by applying the scenario based documentation and evaluation methods through stakeholders' scenario(s) to manage the documentation and the evaluation needs for ADDs.

Table 2.2 Summarization of important related approaches and methodologies adapted from (Alamin & Ammar, 2020, p.67)

Author (year)	General Description	Documenting Architecture <i>Whole/ Particular</i>	Addressing stakeholder concern	Organizing Extracted information
Maras et al. (2009)	PHPModeler tool for legacy PHP Web applications	Whole Architecture	<i>developer concern</i>	Static UML diagrams (such as: dependency models for representing resources of the current page, its functions and dependencies).
Razavizadeh et al. (2009)	Framework for extracting the architectural views from object-oriented source code.	Whole Architecture	<i>developer concern</i>	Conceptual model for representing the architectures' viewpoints.
Che et al. (2011)	An approach for collecting the architectural design decisions (ADDs)	Whole Architecture	<i>developer and Architect concern</i>	Using triple view model framework (TVM) which includes three different views for describing the notation of ADDs.
Che et al. (2012)	An approach for managing the documentation and evolution of architectural design decisions	Whole Architecture	<i>developer and architect concern</i>	TVM framework for specifying its views through the end-user scenario(s).

Table 2.2 Summarization of important related approaches and methodologies adapted from (Alamin & Ammar, 2020, p.67)

Author (year)	General Description	Documenting Architecture <i>Whole/ Particular</i>	Addressing stakeholder concern	Organizing Extracted information
Kumar (2013)	RE methodology for understanding software artifacts.	Whole Architecture	<i>developer concern</i>	UML models such as (state diagram and communication diagram).
Hugo et al. (2014)	Framework for understanding the contents of legacy systems using model driven RE.	Whole Architecture	<i>developer concern</i>	By three layers and the components of each layer are based on the nature of legacy system technologies.
Arshad et al. (2017)	RE model for extracting the architecture of object oriented source code.	Whole/ Particular Architecture	<i>Developer concern</i>	Component model for representing the architecture.
Starke et al. (2017)	Arc24 Template for documentation of software and system architecture	Whole Architecture	<i>Developer and architect concern</i>	Textual document includes several sections: underlying business goals, essential features and functional requirements for the system, quality goals, the relevant stakeholders and their expectations.

2.6. OPEN ISSUES

This section describes the open issues that require further research based on the research work, these issues are described in our survey paper (Alamin & Ammar, 2014) as follows:

- There is a significant need to develop alternative approaches of reverse engineering for documenting the architectures that should simplify and classify all of the available information based on identifying the stakeholders' concerns and their decisions about the system.
- Improve the system's comprehension by establishing more advanced approaches for understanding the software artifacts. These approaches should help in documenting the architecture at different levels of abstractions and granularities based on the stakeholders concerns.
- Finally, it's important to support multiple methods and guidelines on how to use the general ADDs framework in the architecting process. These methods should base on the architecture needs, context and challenges in order to evaluate the ADDs in the architecture development and evolution processes.

2.7. Chapter Summary

This chapter described an overview of the reverse engineering definitions, software architecture definition, and outlines the main objectives of RE process. The second section presented a literature review of common existing research on reverse engineering from different perspectives that form the current state of the art in documenting software architectures. The next section compared the important related work based on the findings and limitations. Finally; the last section highlighted several open issues for future work.

CHAPTER III

CONCERNS-BASED RE METHODOLOGY FOR EXTRACTING PARTIAL SOFTWARE ARCHITECTURE

3.1. Introduction

Generally, this chapter represents an overview of the proposed reverse engineering methodology (*RE Methodology*); discusses the principles of proposed methodology, and describes the detailed design of the main phases of the methodology. The last section of this chapter concludes with a summary that highlights the main activities of each phase of proposed RE methodology.

3.2. Overview of the Proposed RE Methodology

The main goal of the proposed methodology is to design a reverse a RE methodology for extracting particular architectural information based on applying the RE process on implemented source code to support the understand-ability and maintainability of a target system.

The RE methodology is based on three main concepts in IEEE1471-2000 standard for architectural description such as (stakeholder, viewpoint and concern). The main idea is to elicit stakeholders' concern on specific architectural viewpoint of target system; then apply RE process to extract and document a particular architectural information about the target software system driven by the elicited concern that held by one or more stakeholder(s).

The extraction process of RE methodology is driven by addressing the specific concern by stakeholder(s) for extracting only partial architectural information. Therefore, it's doesn't address the RE of the whole architecture of target software system.

The general overview of RE methodology is shown in Figure 3.1; the inputs are the source code and documentation as well as the stakeholders concerns regarding the software system. The output is a model of a particular architectural information based on the specific concerns.

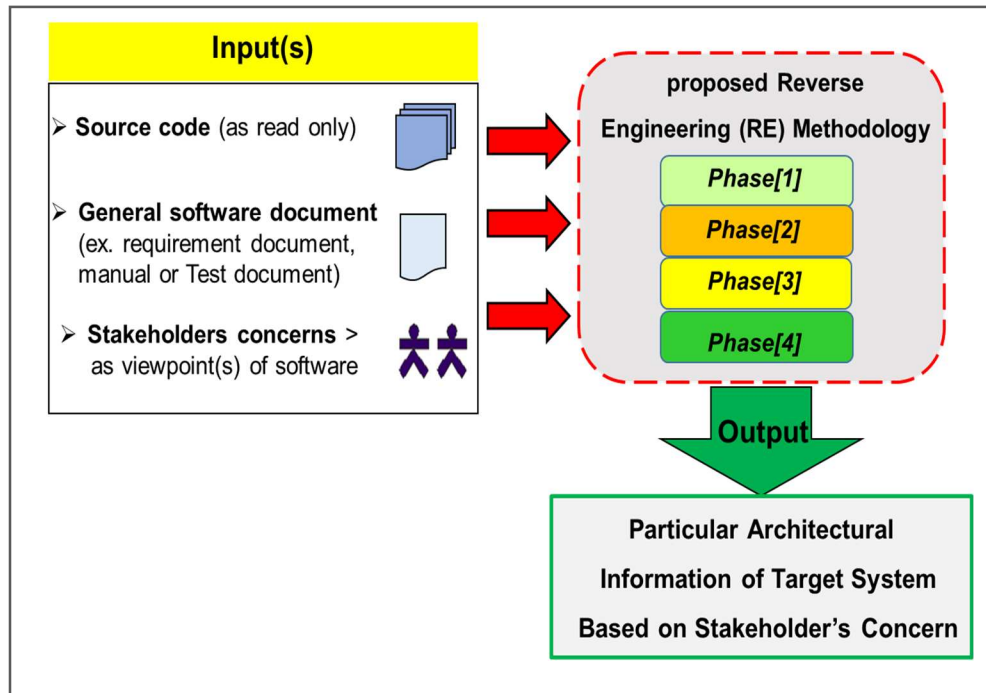


Figure 3.2 Overview of RE Methodology

3.3. The Principles of RE Methodology

The principles of RE methodology are summarized as:

➤ RE methodology is based on three concepts defined in the IEEE1471-2000 standard for architectural description (see Figure 3.2). These concepts are described in (Institute of Electrical and Electronics Engineers 2000, R.Hilliard et al. 2007, Clements et al. 2010) as:

- **Stakeholder** is a person, group or entity with an interest in the realization of the architecture.
- **Concern** is related to specific functional or non-functional requirements of the software system is defined as: a concern to a

requirement, an objective, an intention, or aspiration which a stakeholder has for the software system.

- **Viewpoint** defines the perspective from which the view is taken; and each viewpoint covers a set of concerns related to one or more stakeholder(s).
- RE methodology extends additional stakeholders such as: end-user, maintainer, analyst, architect and tester.
- The RE methodology supports the understand-ability and maintainability of legacy software systems.

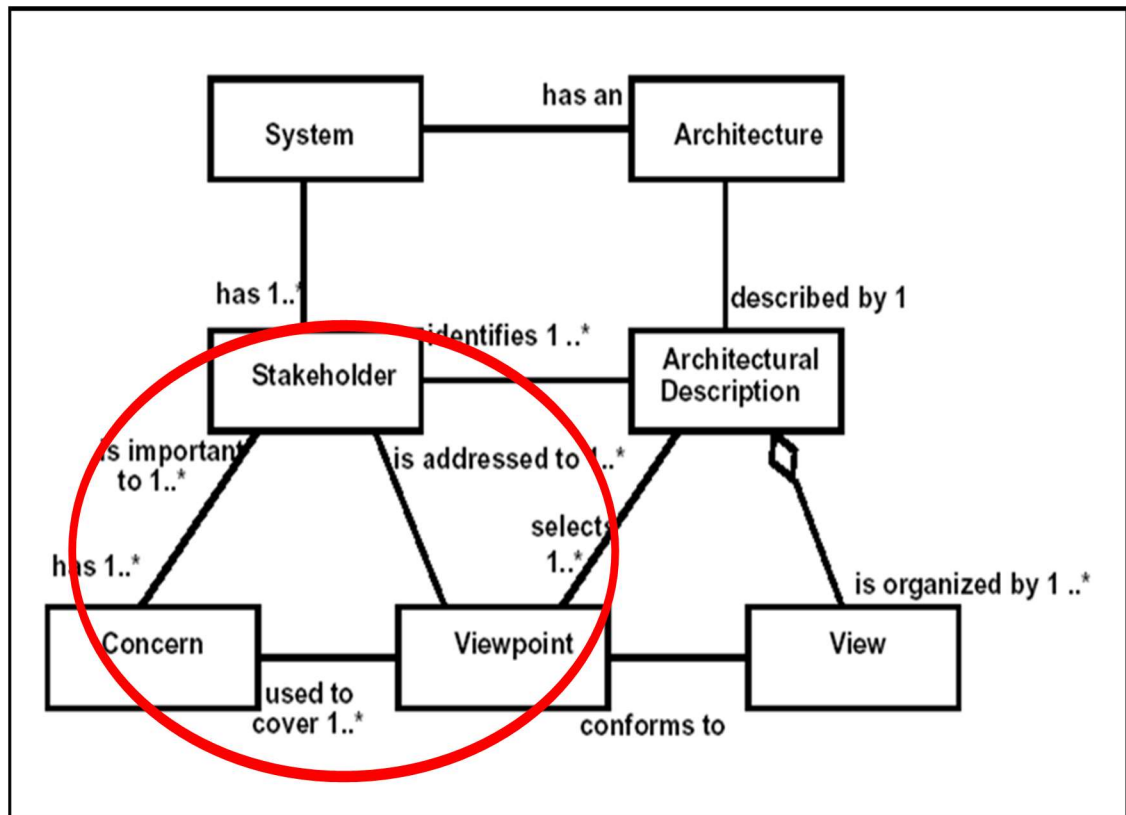


Figure 3.2 IEEE1471 Conceptual Framework. Adapted from (Institute of Electrical and Electronics Engineers, 2000, p.15)

3.4. The Main Phases of RE Methodology

The RE methodology consists of four phases (see Figure 3.3) described as follows:

- **Phase(1):** Define stakeholders concerns based on the architectural viewpoints.
- **Phase(2):** Elicit specific stakeholder's concern.
- **Phase(3):** Extract related requirement information based on the elicited concern.
- **Phase(4):** Apply RE process for extracting the particular architectural information driven by the extracted requirement information.

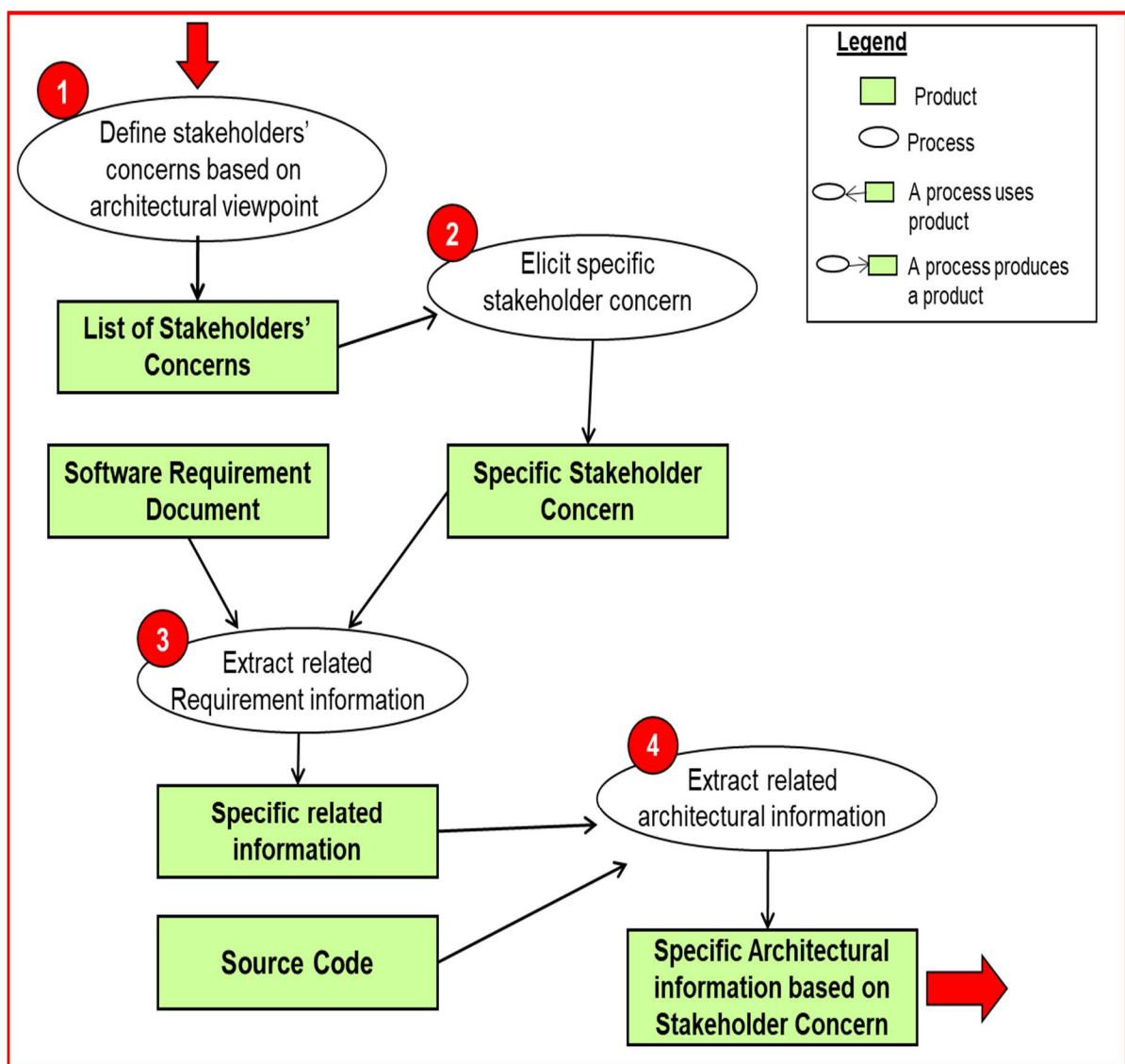


Figure 3.3 The RE Methodology's Phases

As shown in Figure 3.3; the phases of RE methodology is described using a process modelling language. The following paragraphs of this chapter elaborate on the detailed design of each phase of the proposed RE methodology.

3.4.1. Define stakeholders concerns based on architectural viewpoint

This phase is based on the definition of “*stakeholders*” and “*concerns*” in IEEE1471-2000 standard for architectural description. The phase follows the classification of architectural viewpoints that are presented in literature. The activities in this phase includes the following two steps:

- Select viewpoint from a given catalog which describes specific architectural viewpoint for the target software system.
- Categorize common stakeholders related to the selected viewpoint.

3.4.1.1. Select viewpoint from a given catalog

The definitions of stakeholders’ concerns are based on a set of architectural viewpoints about software system. These viewpoints have been considered by several researchers from different perspectives (Kruchten 1995), (Riva & Yang 2002), (Woods 2004), (Nicholas 2005), (Rozanski & Woods, 2005), (Clements 2005), (Henk & Vliet 2006), (R.Hilliard et al. 2007), (Clements et al. 2010), (Rozanski & Woods 2011).

The selection step is based on the classification of viewpoints catalog that were presented by (Rozanski & Woods 2005, 2011). They developed a set of core viewpoints which are based on extending the well-known “4+1” standard view model of software architectures (Logical, Process, Physical, and Development) that was defined by Philippe Kruchten in (Kruchten, 1995).

The viewpoint catalog includes six core viewpoints for information systems architecture, namely: Functional viewpoint, Information viewpoint, Concurrency viewpoint, Development viewpoint, Deployment viewpoint,

and Operational viewpoint (see Figure 3.4). Each one of these viewpoint defines a set of concerns related to one or more stakeholder(s).

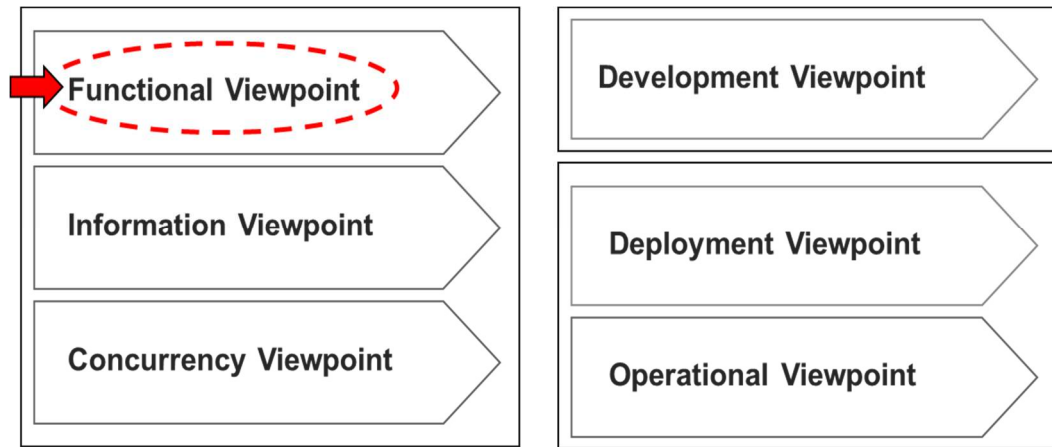


Figure 3.4 The Viewpoints Catalog (Rozanski & Woods 2005, 2011)

Summarized the viewpoints catalog in Figure 3.4; the first three viewpoints: *Functional viewpoint*, *Information viewpoint* and *Concurrency viewpoint* characterize the fundamental organization of the software system. The *development viewpoint* exists to support the system's construction. The *deployment* and *operational viewpoints* characterize the system's runtime environment (Rozanski & Woods 2005, 2011). The last three viewpoints mainly covers the concerns of the developers and maintainers stakeholders.

The RE methodology is focused on the “*Functional viewpoint*” from the catalog of (Rozanski & Woods, 2011). The justification for selecting the “*Functional viewpoint*” is that it is applicable to all types of software systems; and reflects the essential architectural information for most of the stakeholders (such as: maintainer, end-user, developer, system administrator, tester, acquirer, assessor and communicator).

Furthermore, the functional viewpoint includes a set of general stakeholders' concerns which reflect and realize the essential and basic architectural information about the software system. This information include the internal structure which determines the main elements of software system, the

responsibilities of each element and primary interactions between elements, the functional capabilities that defines what the specific action(s) that system should take in a given situation, and the functional design philosophy that reflects how the system will work step by step from the user’s perspective as represented in Table 3.1.

Table 3.1 Functional Viewpoint Catalog (Rozanski & Woods, 2011)








Functional Viewpoint	
Description	Describes the system’s runtime functional elements and their responsibilities, interfaces, and primary interactions between these elements.
General Concerns	<ul style="list-style-type: none"> ▪ Internal structure ▪ Functional capabilities ▪ Functional design philosophy ▪ The external interfaces
Related Stakeholders	<ul style="list-style-type: none"> ▪ End-User, ▪ Maintainer, ▪ Developer, ▪ Tester, ▪ Acquirer, ▪ System Administrator, ▪ Assessor, ▪ Communicator.

3.4.1.2. Categorize stakeholder’s concerns related to selected viewpoint

This step includes the categorization of common stakeholders and their architectural concerns based on selected viewpoint catalog. The main idea is to address the following points: who are the stakeholders of target software system; and which concerns do they have according to the selected

viewpoint. Table 3.2 represents the categorization of stakeholder's and their architectural concerns based on the selected functional viewpoint catalog as following:

Table 3.2 Functional Viewpoint: Stakeholders and Concerns adapted from (Rozanski & Woods, 2011)

Functional viewpoint > Categorize the Stakeholder Concern(s)	
 Acquirer  Users	<ul style="list-style-type: none"> ▪ Internal structure: (The main elements of system, responsibilities of each elements and primary interactions between elements). ▪ Functional capabilities:(define what the specific action(s) the system should taken in a given situation). ▪ The external interfaces
 System Administrator	<ul style="list-style-type: none"> ▪ Internal structure ▪ Functional design philosophy
 Maintainer  Tester  Communicator  Developer	<ul style="list-style-type: none"> ▪ Internal structure ▪ Functional capabilities ▪ Functional design philosophy ▪ The external interfaces

To summarize; the Phase(1) includes two key points, the first one is to select specific architectural viewpoint; and the second one is to categorize common stakeholders related to the selected functional viewpoint, accordingly the main output of this phase is the list of the stakeholders' concerns (see Table 3.3).

Table 3.3 Summary of the *Phase(1)* of RE Methodology

Phase[1]	Define Stakeholders' Concern(s) based on Architectural Viewpoint								
Description	The phase includes the following steps: <ul style="list-style-type: none"> ➤ Select Viewpoint from <u>a given Catalogue</u> which describes specific architectural viewpoint about software system, ➤ Categorize the common stakeholders and general architectural concerns related to selected viewpoint catalogue. 								
Main Output	<p>List of Stakeholders' Concern(s)</p> <div style="display: flex; align-items: center;"> <div style="border: 2px dashed green; padding: 5px; margin-right: 10px;"> Functional viewpoint </div> <div style="border: 1px solid black; padding: 5px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr style="background-color: #ffffcc;"> <th colspan="2">Functional viewpoint: [2,6,7]</th> </tr> </thead> <tbody> <tr> <td style="width: 20%;">Description</td> <td>Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions between these elements.</td> </tr> <tr> <td>General Concerns</td> <td> <ul style="list-style-type: none"> Internal structure: (The main elements of system, responsibilities of each elements and primary interactions between elements). Functional capabilities: (define what the specific action(s) the system should taken in a given situation). Functional design philosophy: (how the system will work "step by step" from user's perspective?). The external interfaces </td> </tr> <tr> <td>Related Stakeholders</td> <td>Acquirer, User, Tester, System Administrator, Assessor, Communicator and Developer.</td> </tr> </tbody> </table> </div> </div>	Functional viewpoint: [2,6,7]		Description	Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions between these elements.	General Concerns	<ul style="list-style-type: none"> Internal structure: (The main elements of system, responsibilities of each elements and primary interactions between elements). Functional capabilities: (define what the specific action(s) the system should taken in a given situation). Functional design philosophy: (how the system will work "step by step" from user's perspective?). The external interfaces 	Related Stakeholders	Acquirer, User, Tester, System Administrator, Assessor, Communicator and Developer.
Functional viewpoint: [2,6,7]									
Description	Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions between these elements.								
General Concerns	<ul style="list-style-type: none"> Internal structure: (The main elements of system, responsibilities of each elements and primary interactions between elements). Functional capabilities: (define what the specific action(s) the system should taken in a given situation). Functional design philosophy: (how the system will work "step by step" from user's perspective?). The external interfaces 								
Related Stakeholders	Acquirer, User, Tester, System Administrator, Assessor, Communicator and Developer.								

3.4.2. Elicit specific stakeholder concern

This phase is called “*Phase(2)*” which includes the elicitation process for specific concern that needed to take and decide where to handle such a concern from the general architectural concerns that addressed in methodology *Phase(1)*.

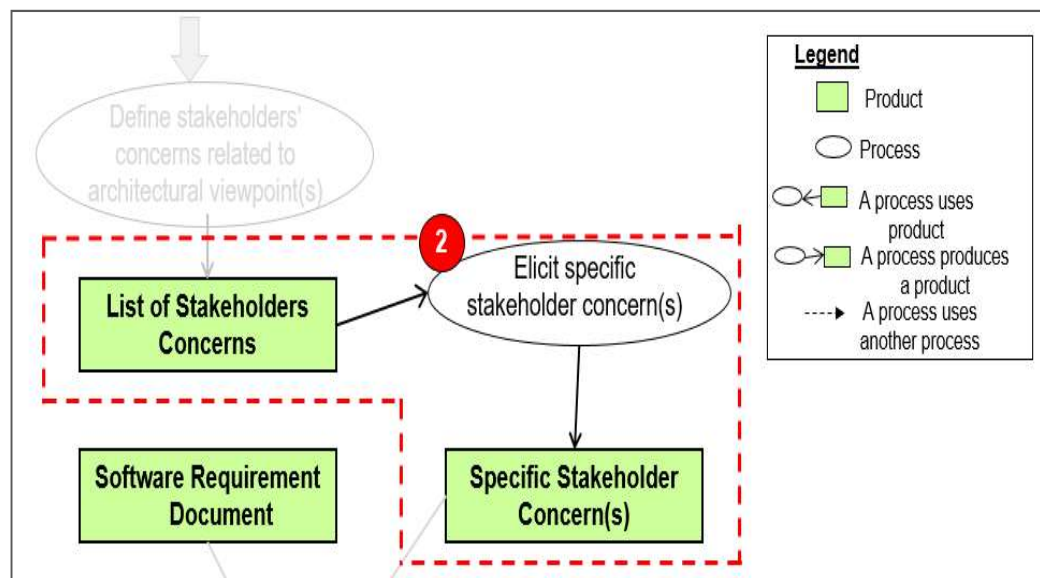


Figure 3.5 The *Phase(2)* of RE Methodology

The elicitation process performs by eliciting specific concern of stakeholder from the functional requirements of a target software system (that addressed in use case diagram). Accordingly, each elicited concern should be in a form of question format and has two elements (see Figure 3.6):

- **CIDn**: refers to concern ID (where n is an integer number), which written in dotted diamond box.
- **Question**: refers to elicited concern from the functional scenario of a target software system, and written in dotted rectangular box.

Figure 3.6 describes the association between the functional requirement (FR) and elicited concern appears with dotted lines in the use case diagram of the target system. Moreover, it's possible to have multiple elicited concerns for one FR which are numbered as *CID1*, *CID2*, ..., *CIDn*.

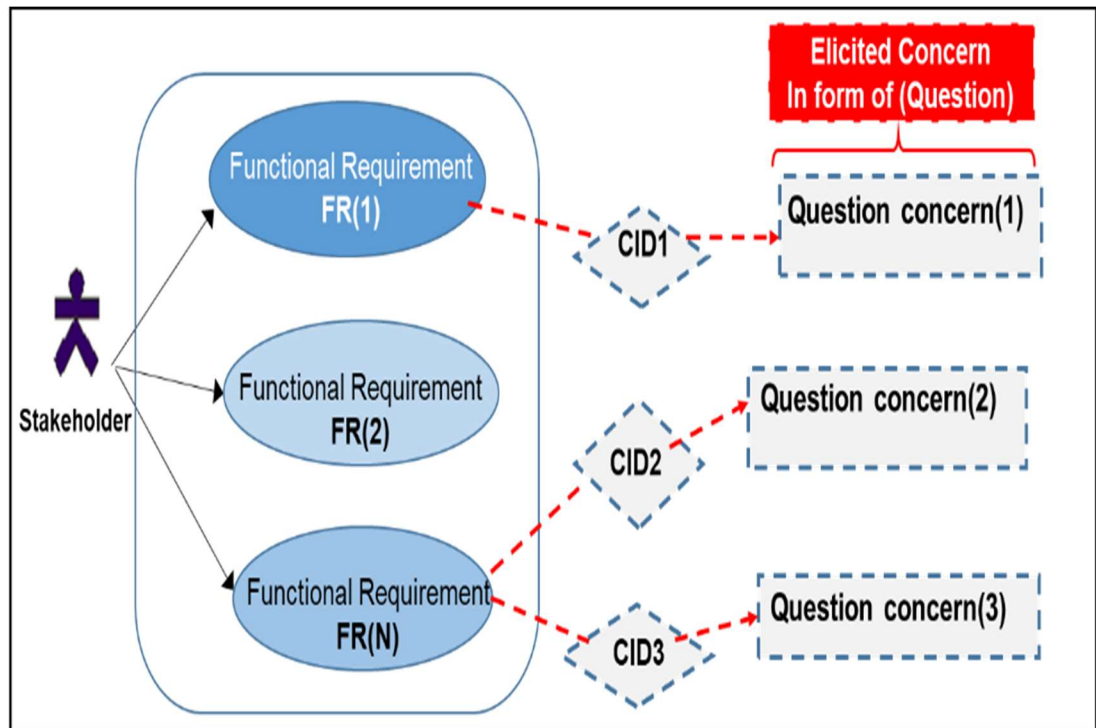


Figure 3.6 Elicitation Specific Stakeholder's Concern(s)

3.4.3. Extract related requirement information based on elicited concern

In “*Phase(3)*” which describes how to extract the related requirement information related to the elicited functional concern produced in *Phase(2)*. The stakeholder’s functional concern should be focused on the functionality offered by the target software system, as follow:

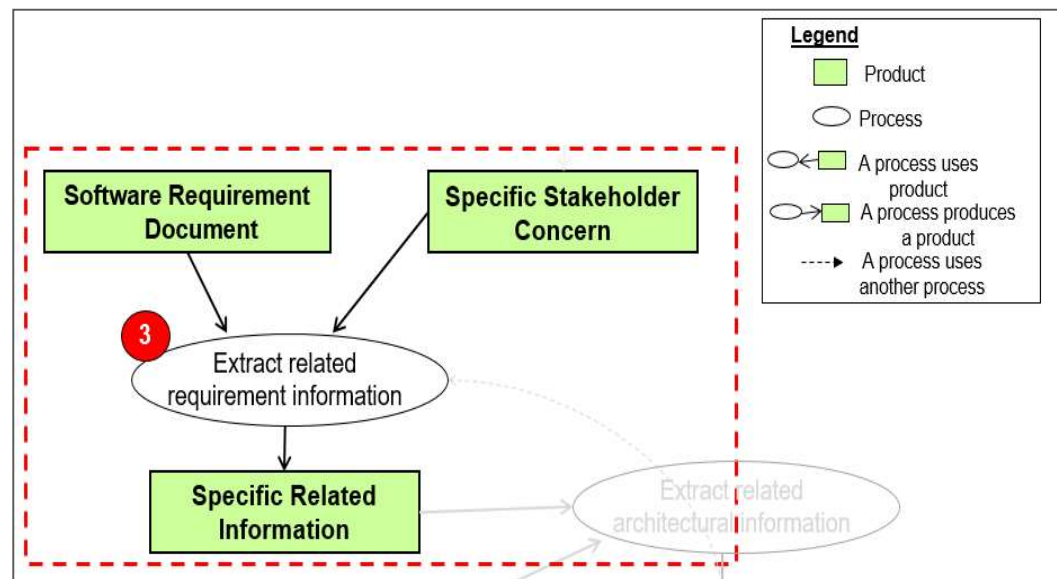


Figure 3.7 The *Phase(3)* of RE Methodology

Therefore, it’s important to note that, this phase assumes that all of system’s requirements are already existed in a requirement repository. The requirement repository contains detailed descriptions for all the requirements of software system as follow:

- RID (Requirement code)
- Description of requirement
- Type of Requirement: either “F > Functional” or ”NF >Non Functional”
- The creation date of requirement in form of (dd-mm-yyyy)
- Status
- Author
- Additional comments

localhost > db_tracelink > requirement_repository "Requirements Repository Information"

Showing rows 0 - 29 (35 total, Query took 0.0010 sec)

```

SELECT *
FROM 'requirement_repository'
LIMIT 0, 30

```

Options

No	RID	ReqName	Description	Type	Author	DateCreated	RelatedTo	Status
1	TMS_Req2.1		Using the web interface, the College admin can log...	F	Hind Alamin	2008-12-05	College Admin	Active
2	TMS_Req2.2		Allow to change the password of College's admin a...	F	Hind Amin	2008-12-05	College Admin	Active
3	TMS_Req2.3		Insert the list of Teachers (Staff) for each colle...	F	Hind Amin	2008-12-05	College Admin	Active
4	TMS_Req2.4		Update/delete the registered Teachers per semester	F	Hind Amin	2008-12-05	College Admin	Active
5	TMS_Req2.5		Insert the departments of the college	F	Hind Amin	2008-12-05	College Admin	Active
6	TMS_Req2.6		Insert the academic programs and classes on each d...	F	Hind Amin	2008-12-05	College Admin	Active
7	TMS_Req2.7		Update/delete the academic programs on department	F	Hind Amin	2008-12-05	College Admin	Active
8	TMS_Req2.8		Register courses per semester for each academic pr...	F	Hind Amin	2008-12-05	College Admin	Active
9	TMS_Req2.9		Update/ delete the registered courses on each seme...	F	Hind Amin	2008-12-05	College Admin	Active
10	TMS_Req2.10		Insert Academic year before the starting of the fi...	F	Hind Amin	2008-12-05	College Admin	Active
11	TMS_Req2.11		Insert the numbers of students on each academic pr...	F	Hind Amin	2008-12-05	College Admin	Active
12	TMS_Req2.12		Select the lecture rooms at College on each semest	F	Hind Amin	2008-12-05	College Admin	Active

Figure 3.8 Example of the Requirement Repository Information

To support the activities of this phase, the development of a prototype tool is adopted which has a graphical user interface (GUI). The tool allows stakeholders to enter a specific concern in form of a “*query*”. The specific concern will be elicited from the functional requirements repository assumed to be available for the target software system.

The tool extracts a set of related requirement information based on elicited concern, and creates a trace link between elicited concern and its relevant information (see APPENDIX A for more detailed about the execution of GUI prototype tool). The Figure 3.9 shows screen shots of GUI prototype tool described as follow:

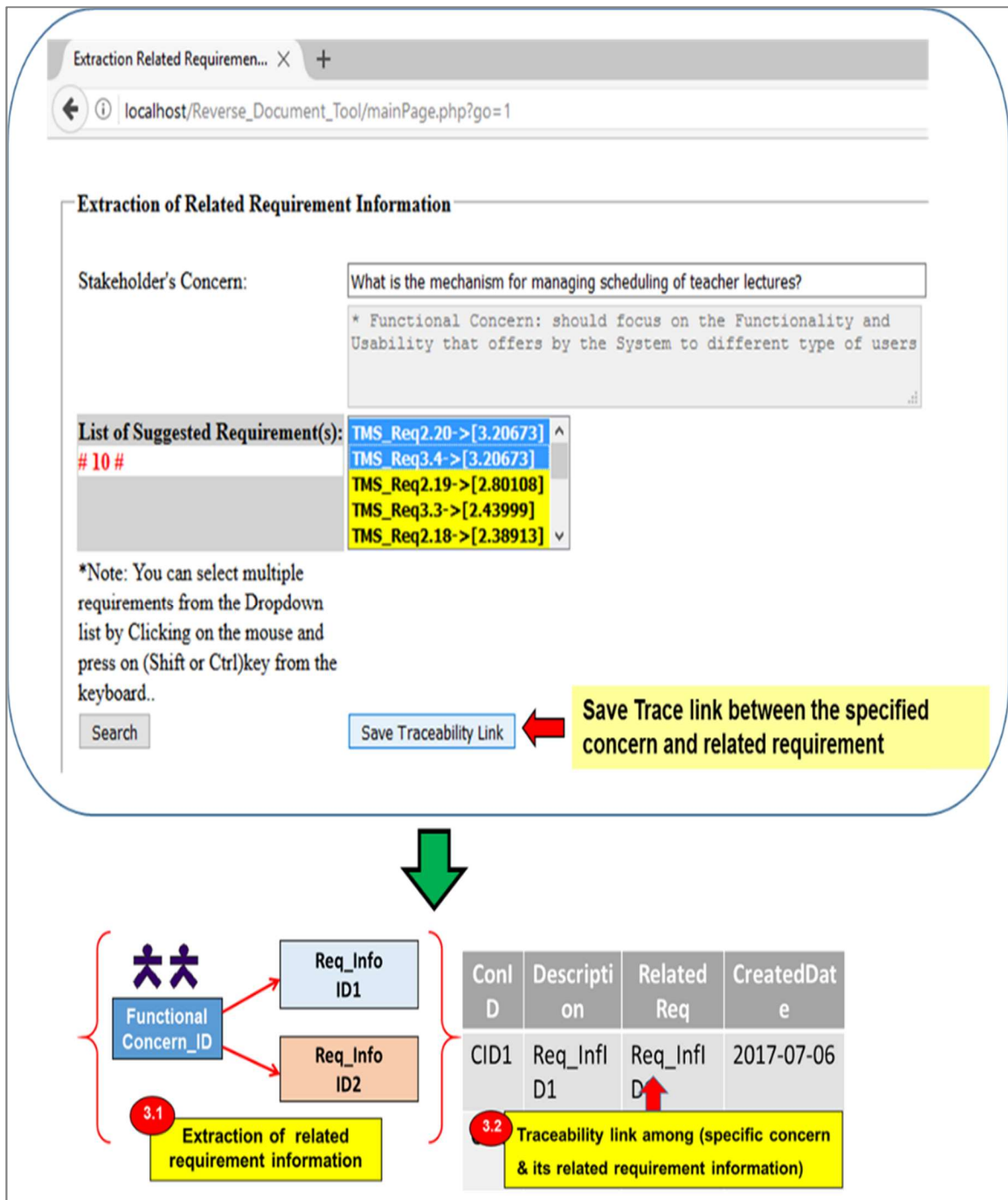


Figure 3.9 Tracing Specific Concern to its Related Requirement Information

The following paragraphs elaborate on the detailed of the main activities of the *Phase(3)* as follow:

- Extraction of related requirement information, and
- Traceability among specific concern and its related information.

3.4.3.1. Extraction of related requirement information

The extraction process starts by accessing the requirement repository and filtering all relevant information related the specified concern. Furthermore, the extraction process is achieved using the Full-Text indexing and searching mode technique.

Furthermore, the Full-Text indexing and searching technique is supported by MySQL database since version 3.23.23 and above. It allows to implement keyword based filtering and sorting; and provides several searches mode such as (Natural Language, Boolean and Query expansion).

The implementation techniques to adopt the extraction process requires the following key steps which are adapted and configured from (MySQL 5.7 Reference Manual Document, 2017). The following paragraphs describe the detailed design of these key steps:

- Define Full-Text Index
- Select Full-Text searching mode
- Relevance in Full-Text searching

3.4.3.1.1. Define Full-Text Index

The definition of Full_Text index is compulsory in the MySQL database before executing of the Full_Text query. Technically, Full-Text index in MySQL is an index of type “FULLTEXT”; this index is used with MyISAM tables, and can be created only for CHAR, VARCHAR, or TEXT columns.

In case of large data sets in the database, it is much faster to load the data into a table that has no FULLTEXT index and then create the index after that, than to load data into a table that has an existing FULLTEXT index. Accordingly; the FULLTEXT index had been created on the requirements repository after load the data into the table according the

adapted instructions from (MySQL 5.7 Reference Manual Document, 2017, section 12.9.1).

3.4.3.1.2. Select Full-Text searching mode:

The Full-Text searching is performed using the following syntax:

```
MATCH (coll, col2, ...) AGAINST (expr [ search_modifier ] )
```

Where

- *MATCH*: a function takes the name of the column(s) to be searched.
- *AGAINST(expr)*: this function takes a string to search for. The search modifier indicates what type of search from the following options:
 - *Natural language search*: interprets the search for string as a phrase in natural human language.
 - *Boolean search*: interprets the search for string using the rules of a special query language. The string contains the words to search for and additional operators that used to determine the present or absent of word in matching rows.
 - *Query expansion search*: the search string is used to perform a natural language search. Then words from the most relevant rows returned by the search are added to the search string and the search is done again.

3.4.3.1.3. Relevance in Full-Text searching:

MySQL database uses the ranking with Vector spaces technique for ordinary Full-Text queries adapted from (MySQL 5.7 Reference Manual Document, 2017). The relevance (R) is a number that describes how the match of text is, the basic formula for R which stands for either rank or relevance as follow: $R = w * qf$

Where

- *w*: is the weight, which goes up if the term occurs more often in a row, and goes down if the term occurs in many rows on a target table. This is depending on whether the number of unique words in a row is fewer or more than average.
- *qf*: is the number of times the term appears in the AGAINST expression.

Additionally, the term *weight(w)* is what MySQL stores in the index, and the calculation of weight is done using the following formula:

$$w = \underbrace{(\log(\text{dtf})+1)/\text{sumdtf}}_{\text{Base part}} * \underbrace{U/(1+0.0115*U)}_{\text{Normalization factor}} * \underbrace{\log((N-\text{nf})/\text{nf})}_{\text{Global multiplier}}$$

Where:

- **dtf** is the number of times the term appears in the document
- **sumdtf** is the sum of $(\log(\text{dtf})+1)$'s for all terms in the same document.
- **U** is the number of Unique terms in the document
- **N** is the total number of documents
- **nf** is the number of documents that contain the term

Figure 3.10 Calculation formula for *weight* in MySQL. Adapted from (MySQL Reference Manual Document 2017, section 10.7)

Generally, the calculation formula in Figure 3.10 has three parts:

- **Base part**: is the left part of the formula; the idea of base part is totally depends on two values: the number of times that the term appears in the document and the summation of all terms which appear in the document.

- **Normalization factor:** is the middle part, the idea of this factor is that: if the document is *shorter than average* length then weight goes up, if *its average length* then weight stays the same, and if *it longer than average length* then weight goes down. The constant **0.0115** is a pivot value that uses in MySQL source code, which known as *pivoted unique normalization factor*, and the measure of document length is based on the unique terms in the document.
- **Global multiplier:** is the final part and it used to make a better guess of the probability that a term will be relevant.

According to these mentioned technical key points; the searching techniques of GUI prototype tool is achieved using the *natural language searching mode* which interprets the search for specific functional concern (in form of user query); then performs filtering process and ranking of the relevant information related to the specified concern.

The main results from the GUI prototype are displayed in a dropdown menu and sorted into three categories and each one highlighted with a specific color as follow:

- **High weight:** appears in green color and represents highly relevant requirement information related the specified functional concern,
- **Medium weight:** appears in yellow color and represents the medium relevance requirement information related the specified functional concern,
- **Low weight:** represents low relevance values of requirement information, and appears in red color.

3.4.3.2. Traceability among specific concern and its related information

The traceability process is performed after the extraction process. The main idea is to create a trace link among the extracted concerns and its relevant information using the tool as shown in Figure 3.9.

3.4.4. RE for extracting particular architectural information

The final phase, “*Phase(4)*” is based on using the extracted requirement information that produced from the previous phases as shown in Figure 3.11.

This phase includes two key activities as follows:

- RE process for extracting specific source code files,
- Representation of the particular architectural information based on the extracted code files.

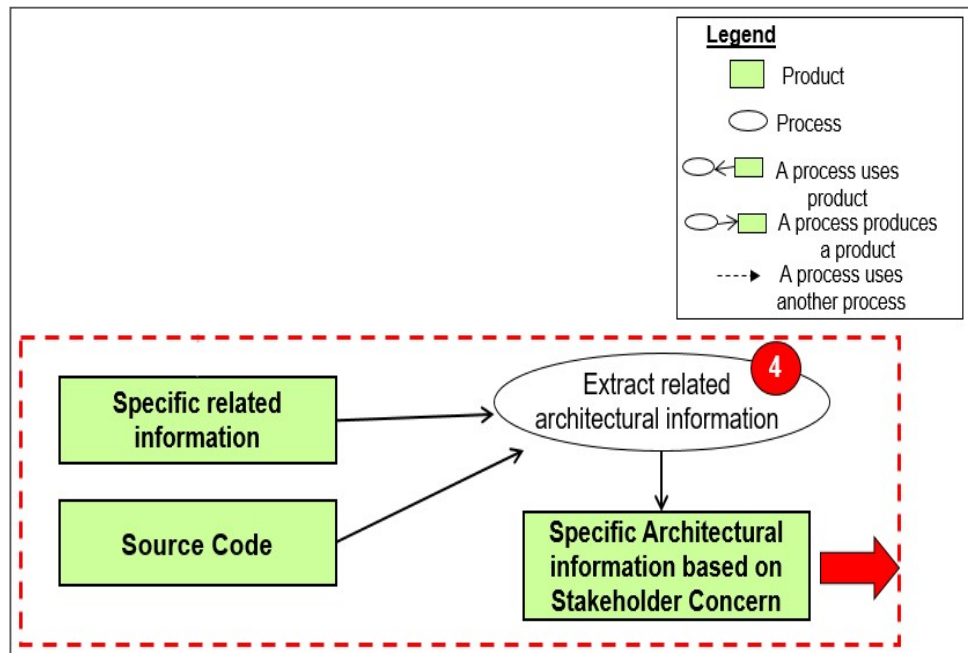


Figure 3.11 The *Phase(4)* of RE Methodology

3.4.4.1. RE process for extracting specific source code files

The RE process is achieved by applying a code analyzer process which performs static analysis on source code files to determine and trace which set of code files are used to implement specific functionality reflected by the

extracted requirement information in *Phase(3)*. The code analyzer process includes three key steps (see Figure 3.12).

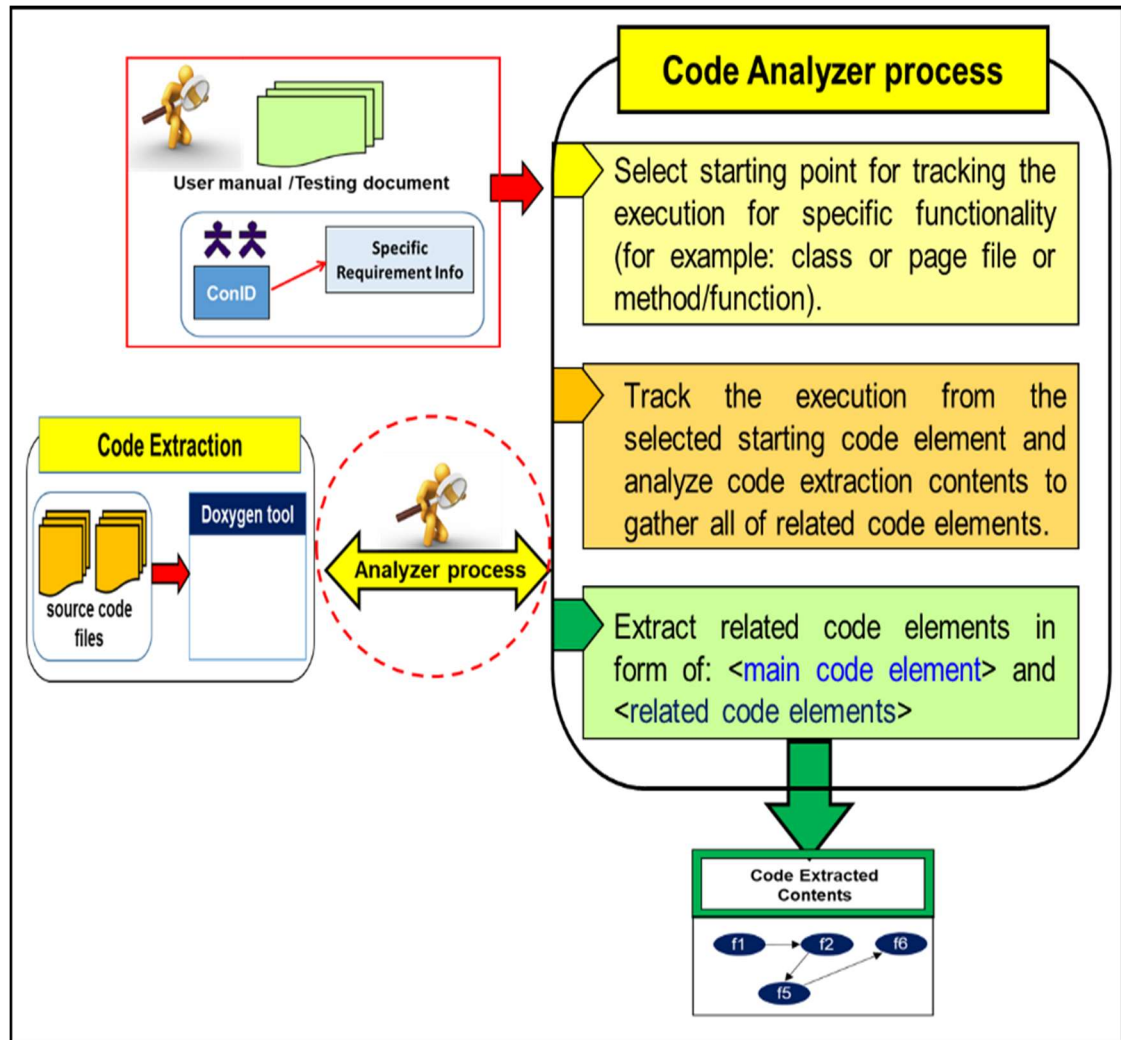


Figure 3.12 The Code Analyzer Process

The following paragraphs describe these three key steps (see Figure 3.12):

- Select the starting point for tracking the execution of a specific functionality represented by extracted requirement information. For examples: page file, class, method or function from code elements. Notably, the selection of a starting point can be performed by using references from existing documents such as the user manual, or the software testing document.

- Track the execution of selected starting code element and analyze the code extraction contents and gather all related code elements.
- Extract related code elements in form of main code element and its related elements. The relation between code elements can be describes as following:
 - *require relation* is used to describe the relations between code files and show the dependences of these files within the software system, or
 - *contain relation* is used to describe that code file contains a set of functions that are used to execute specific functionality of the system, or
 - *call relation* is used to describe the relation between code elements and how different functions interact with each other.

As summarized; the whole process of code analyzer is achieved by using a static analyzer tool called *Doxygen* tool. The *Doxygen* tool is used to extract code structure from the existing source code files, and visualize the relations between various code elements according the type of source code of target software system in the form of function call graphs, or dependency graphs, or inheritance diagrams, or collaboration diagrams, which are all generated automatically by the tool (Doxygen Reference Manual Document, 2016).

3.4.4.2. Representation of the particular architectural information

The representation process includes two key steps; mapping the extracted code elements into a component model; and visualizing the architectural information using architecture styles. The following paragraphs describe the details of these steps:

3.4.4.2.1. Mapping extracted code into a component architecture

This step involves the process of organizing the extracted code elements into a *component model* to make an explicit mapping between software architecture and the code elements of the target system. It is important to note that this process assumes that the term “*component*” can be associated with a code element such as a code file, a webpage file, a class, a class method, a function, or either as a group of related methods or functions which are used frequently together in the execution of specific system’s functionality.

For example, suppose the given code element is a webpage source file called *page_Layout.php*, this webpage file can be mapped into a “*Page Layout*” component which contains the set of functions or methods that are used to execute specific system’s functionality as in the following example shown in Figure 3.12.

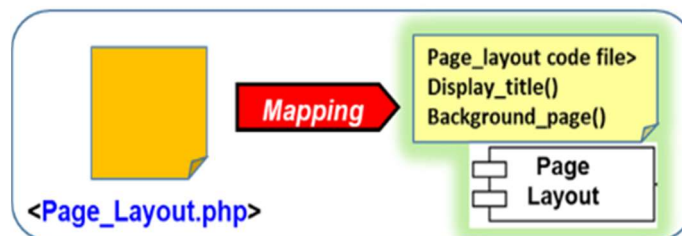


Figure 3.13 Example of Mapping Code’s Element into Component

3.4.4.2.2. Visualizing architectural information using ArcheType

The whole purpose of this process is to create a logical model, so that the architectural information is visualized and represented in the form of logical component model which helps the stakeholders to gain insight of the architecture information related to their functional concerns about a target system.

The visualization process starts by selecting the structure of the architecture which is mainly based on the application’s type called

archetypes. The Microsoft guide for application architecture defines these archetypes as in Table 3.4.

Table 3.4 Application Archetypes Summary adapted from (MICROSOFT Architecture Guide, 2009, P.226).

Application Type	Description
Web applications	The applications of this type are typically support connected scenarios and can support different browsers running on a range of operating systems and platforms.
Rich client applications	The applications are usually developed as standalone applications with a graphical user interface that displays data using a range of controls. Those applications can be designed for disconnected and occasionally connected scenarios if they need to access remote data or functionality.
Rich Internet applications	The applications of this type can be developed to support multiple platforms and multiple browsers, displaying rich media or graphical content. Rich Internet applications run in a browser sandbox that restricts access to some features of the client.
Service applications	The services expose shared business functionality and allow clients to access them from a local or a remote system. Service operations are called using messages, based on XML schemas, passed over a transport channel. The goal of this type of application is to achieve loose coupling between the client and server.
Mobile applications	Applications of this type can be developed as thin client or rich client applications. Rich client mobile applications can support disconnected or occasionally connected scenarios. Web or thin client applications support connected scenarios only. Device resources may prove to be a constraint when designing mobile applications.

As summarized in Table 3.4; the application archetypes includes the architecture’s structure for common types of applications such as web applications, rich client applications, rich internet applications, service applications and mobile applications. However, beside these archetypes,

the Microsoft's guide also contains details of some specialized application types such as hosted and cloud services, and office business applications.

The architecture of each of the archetype application can be defined using architecture styles. For example, the guide (MICROSOFT® Architecture guide, 2009) describes a layered architecture style for web applications. The visualization process is performed using these architectural styles. This is based, for example, on grouping related components in web applications as a three-layered architecture which consists of a *presentation layer*, *business layer* and *data layer* as the shown example in Figure 3.14. Each layer should include specific components described as follows:

- **Presentation Layer:** responsible for managing user interaction with software system, and generally consists of components that provide a common bridge into the core business logic that encapsulated in the business layer.
- **Business Layer:** which implements the core functionality of software system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use.
- **Data Access Layer:** provides access to data hosted within the system, and data exposed by other networked systems; perhaps accessed through services.

To summarize; *Phase(4)* includes two key steps. The *first step* deals with organizing the extracted code elements into a component model to make an explicit mapping between the system's architecture and code elements. The *second step* deals with using archetypes and architecture styles to visualize the

architecture model and give an example of a layered architectural style for web applications.

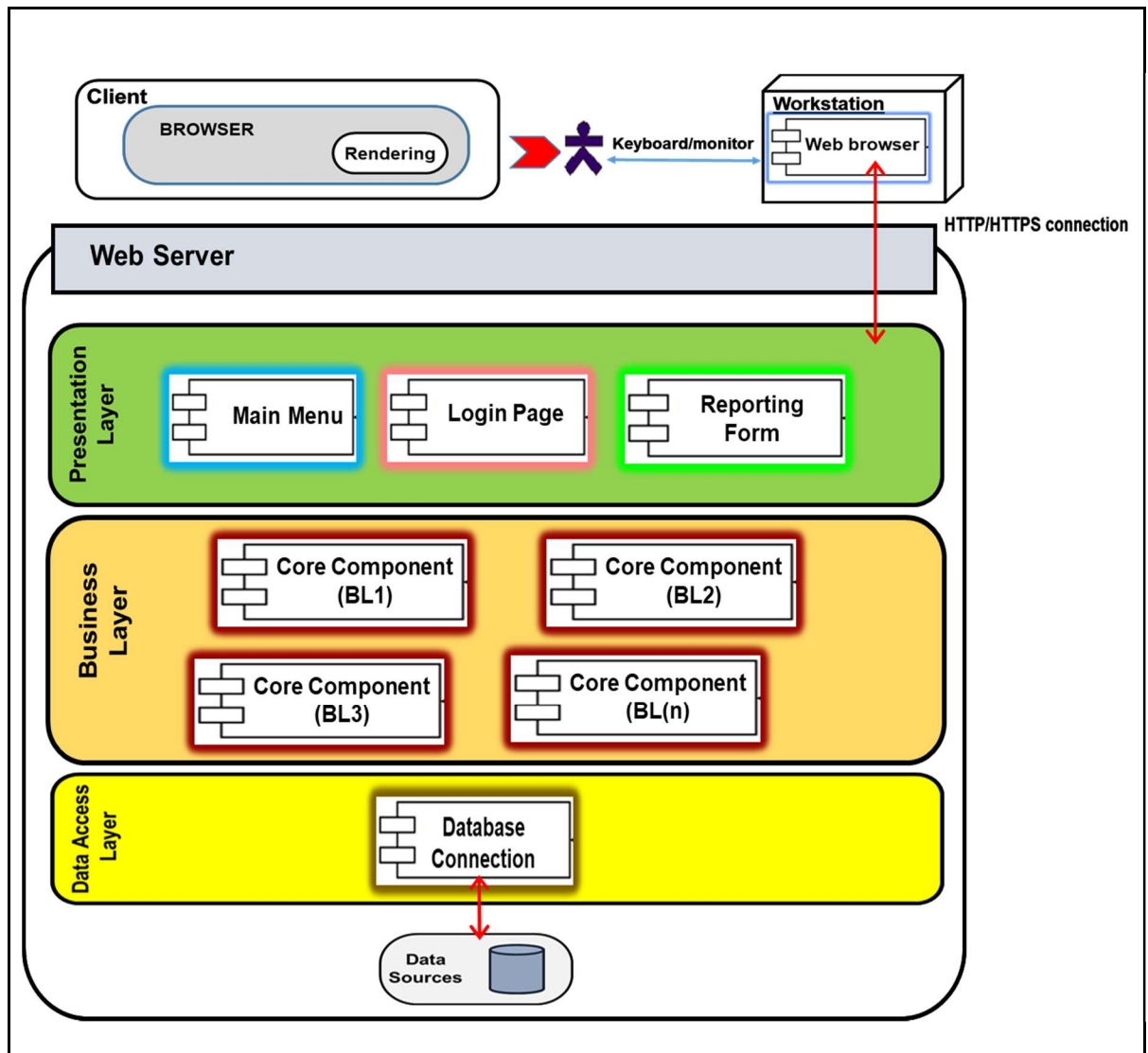


Figure 3.14 Example of Visualizing Architectural Information using Layered Architecture Model

3.5. Chapter Summary

This chapter represented an overview of the proposed RE methodology; then discussed the principles of proposed methodology and described the detailed design of the main phases of proposed methodology. To summarize the main activities of each RE Methodology phase: *Phase(1)* includes the selection of specific

architectural viewpoint and categorize common stakeholders related to the selected viewpoint, accordingly the main output is the list of the stakeholders' concerns. **Phase(2)** performs the elicitation process for specific concern that needed to take and decide where to handle such a concern from the general architectural concerns that addressed in previous phase.

In **Phase(3)** the development of a GUI prototype tool is adopted. The tool allows stakeholders to enter a specific concern in form of a “query”. The specific concern will be elicited from the functional requirements repository assumed to be available for the target software system. The tool extracts a set of related requirement information based on elicited concern, and creates a trace link between elicited concern and its relevant information.

The final phase; **Phase(4)** involves the organization of the extracted code elements into a component model and using archetypes and architecture styles to visualize the architecture model. As a result, the visual model represents the extraction of the partial architectural information in the form of a logical model. This architectural information helps stakeholders to answer their architectural concerns about a target system. The next chapter will describe how to apply the methodology phases to a practical case study.

CHAPTER IV

IMPLEMENTATION OF RE METHODOLOGY TO CASE STUDY

4.1. Introduction

Generally, this chapter describes how to implement the RE methodology phases using a legacy web application as a practical case study. The first section starts by giving an overview of the selected software system, and describes the main reasons for selecting this system. The second section describes the details of applying each phase of the methodology to the case study. The third section represents main benefits of the extracted architectural information for stakeholders; and the last section concludes with the chapter summary.

4.2. Selecting Software System for a Case Study

A practical case study had been implemented in a web application system called Timetable Management System (TMS). TMS was developed by the Computer Center at Sudan University of Science and Technology (SUST) in 2008.

TMS is a Web-based open source system which was built for Sudanese Universities using MySQL database and PHP web page language with Arabic interface; and it provides high flexible features for managing and controlling the scheduling of lectures' times for students at Sudanese universities (adapted from TMS Manual Document, 2009).

Moreover; TMS is flexible to accept changes that occur in schedules for all colleges at the university during the academic year without an overlap in specified slot times between these colleges. The main reports are the extraction timetables and schedules for students according the academic year, the scheduling timeslots for the lecture rooms and laboratories per week and the timetable for teachers per semester (see Appendix B for more details about the privileges of system's users).

The selection of TMS for the following reasons; TMS software is a diverse software implemented as a combination of both front-end PHP, JavaScript and HTML code plus a back-end MySQL database. It is an example of an application with multiple components implemented with different technologies. TMS is considered to be a legacy system implemented with more than 10 years old technologies (since 2008).

The documentation of TMS's architecture is missing, and the system documentation needs to reflect its current architectural representation in order to be reengineered with new technologies. Recovering the particular architectural information of the system is essential to support the system's understand-ability and maintainability. The following Figure 4.1 represents the general description of the main contents of TMS.

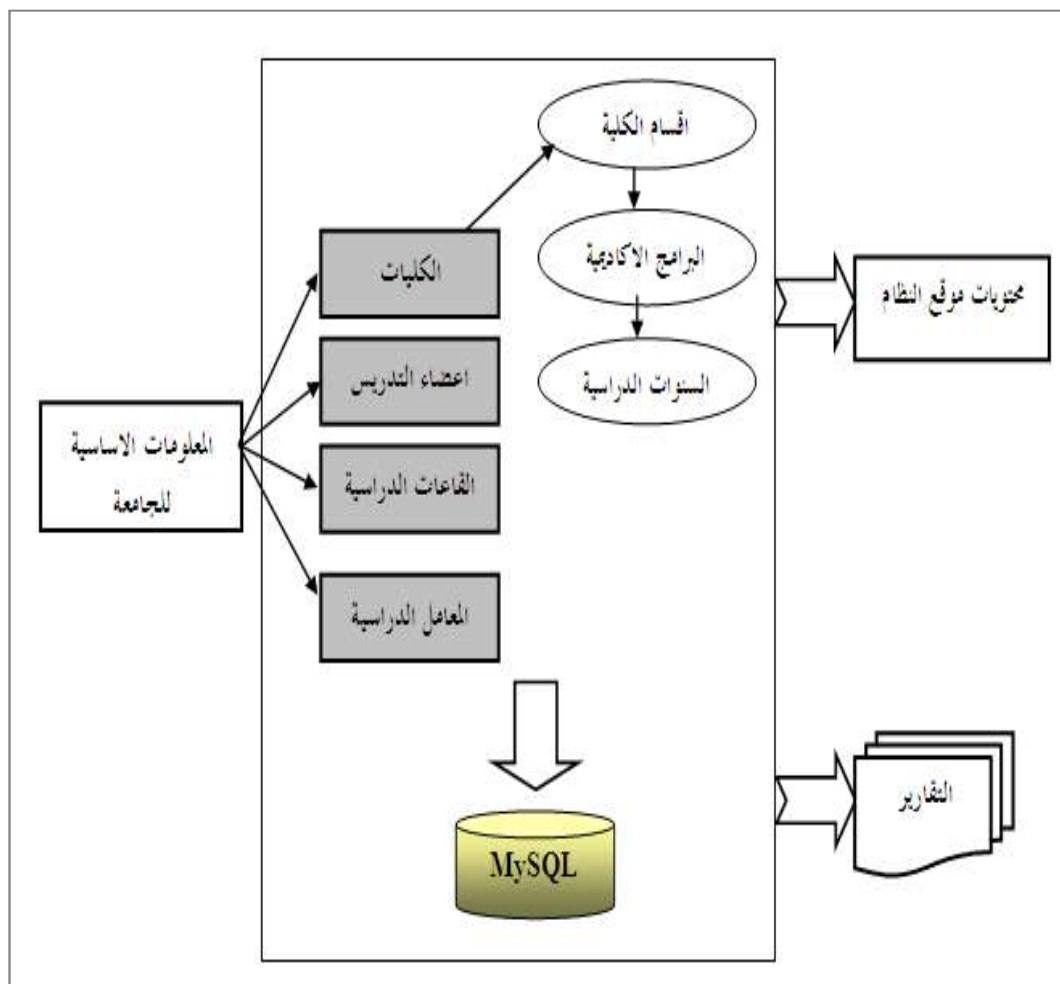


Figure 4.1 an overview of the main contents of TMS, adapted from (Developer's Documentation, 2009)

The general description about TMS’s source code contents represented in Table 4.1 as follow:

Table 4.1 TMS Source Code Overview

System Name	Timetable Management System(TMS)
Description	The core of source code is mainly PHP webpage source files (written with PHP procedural function code style, and its non-object oriented code style).
PHP Source Files	110
Total LOC	30364
Number of Functions Code	148

4.3. Applying RE Methodology Phases to the Case Study

The following paragraphs elaborate on the details of applying each phase of RE methodology:

4.3.1. Define a set of stakeholders concerns

Apply *Phase(1)* to define a set of stakeholders’ concerns base on the *Functional viewpoint* of the TMS system. The primary TMS’s stakeholders are:

- End-User: who defines the system’s functionality and ultimately make use of it. TMS has three end-users such as (*College Admin, Teachers, and Students*).
- Maintainer: who manages the reengineering and improvements of the TMS system.

4.3.2. Elicit a specific stakeholders concern:

The elicitation process is focused on a selecting a particular functional concern related to a use-case or a major functionality offered by the system to different type of users. The main idea is to elicit a specific concern such as “CID1” shows in the Figure 4.2 bellow.

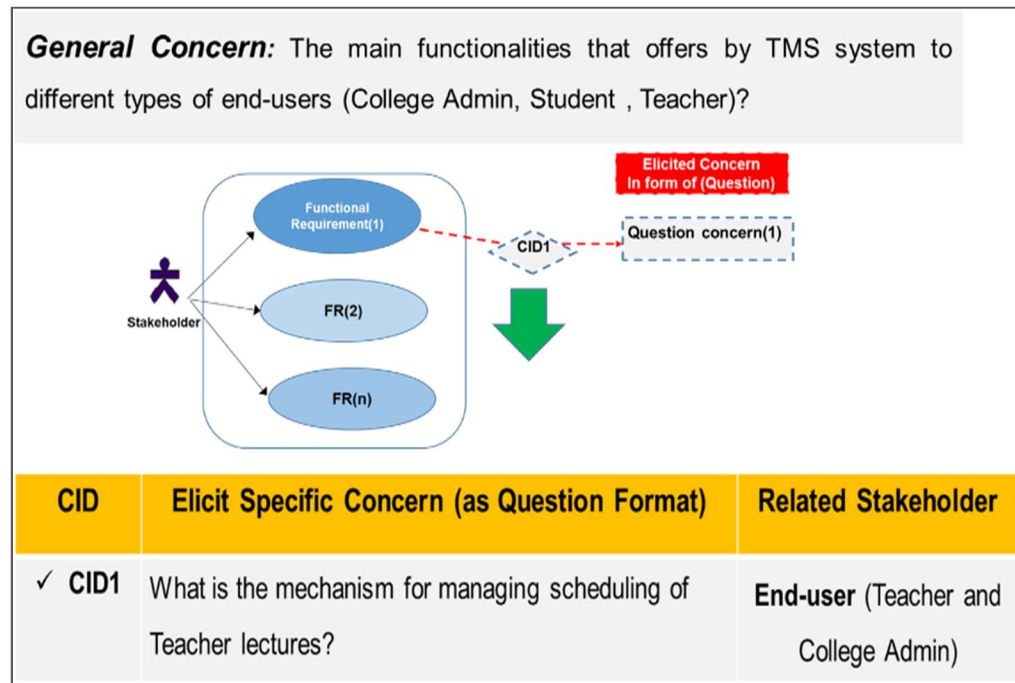


Figure 4.2 Elicit a specific stakeholder functional concern

4.3.3. Extract related requirements information based on elicited concern

TMS has 34 functional requirements; this phase assumes that all of TMS’s functional requirements are already existed in a “requirement repository” (see Figure 3.8). The extraction process starts by accessing the requirement repository and filtering all of relevant information based on the elicited concern. Then create a trace link between its relevant information. The phase is achieved by using the tool as described in *section 3.4.3.1* and *section 3.4.3.2* in the previous chapter.

Using the developed prototype tool and obtain the results shown in Figure 4.3. The results of the search shows *ten requirements information* that displayed in a dropdown menu and sorted by ranking using three following categories as:

- *High weight (2)* appears in green color,
- *Medium weight (7)* appears in yellow color, and
- *Low weight (1)* appears in red color.

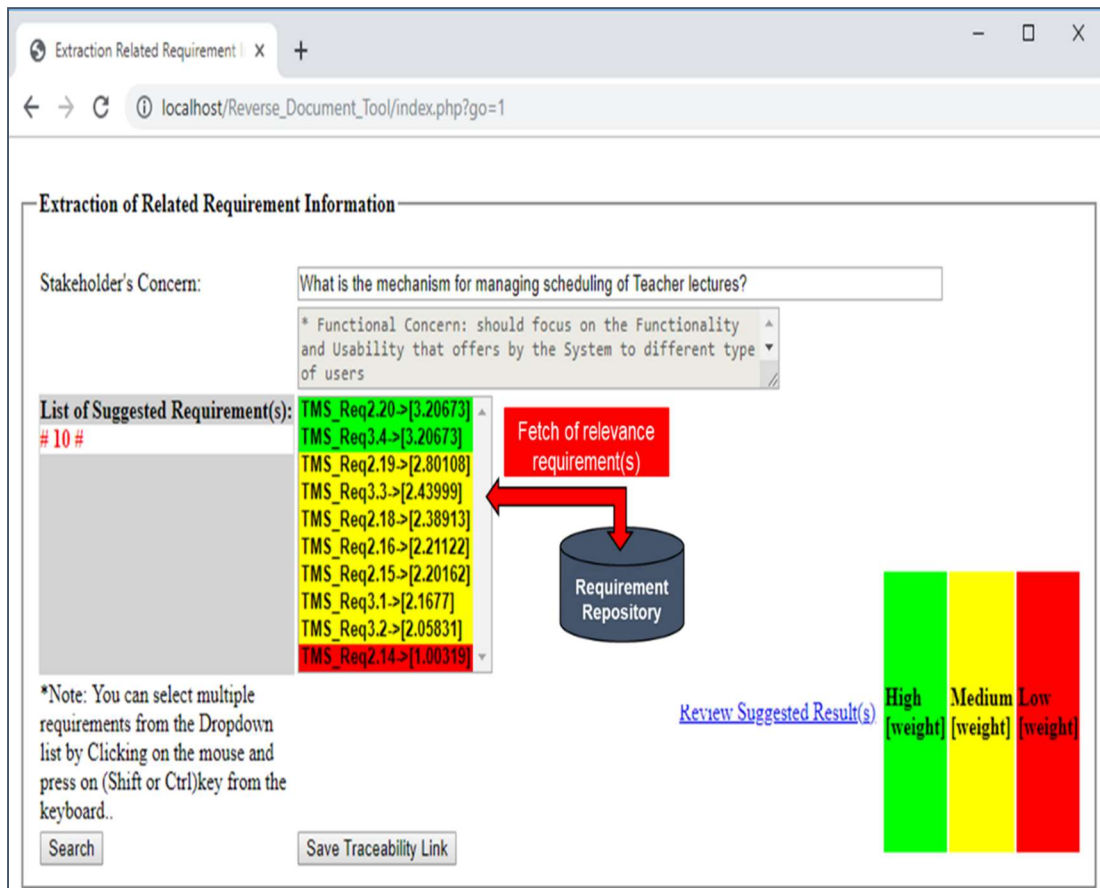


Figure 4.3 Extraction of related Requirements Information

Additionally, the creation of a trace link is performed in order to link the elicited concern with its relevant information produced from the extraction process in Figure 4.4 as follow:

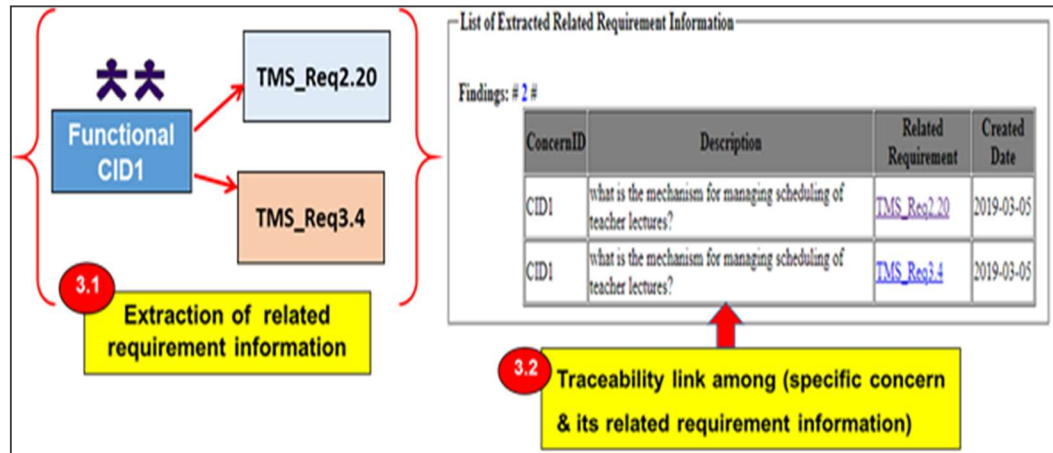


Figure 4.4 Traceability among specific concerns and their related Requirement Information

4.3.4. Extracting architectural information

This final phase is achieved by applying the RE process at code level to perform following key steps:

- Extracting specific source code files
- Representation and visualization of architectural information

4.3.4.1. Extracting specific source code files

The code extraction process is performed by using a static code analyzer as described in *section 3.4.4.1*. Using the existing TMS source code file, to determine which set of source code files are used to implement the specific functionality of the system specified in the previous steps.

Notably, the selection of a starting point for the extraction process is performed by returning to TMS’s user manual document in order to track the starting point for “*TMS_Req2.20*” execution. The main output of this process is to extract the call graph to obtain and visualize the dependencies between the function elements which are used to execute specific functionality in the system as described in Figure 4.5 and Figure 4.6 below.

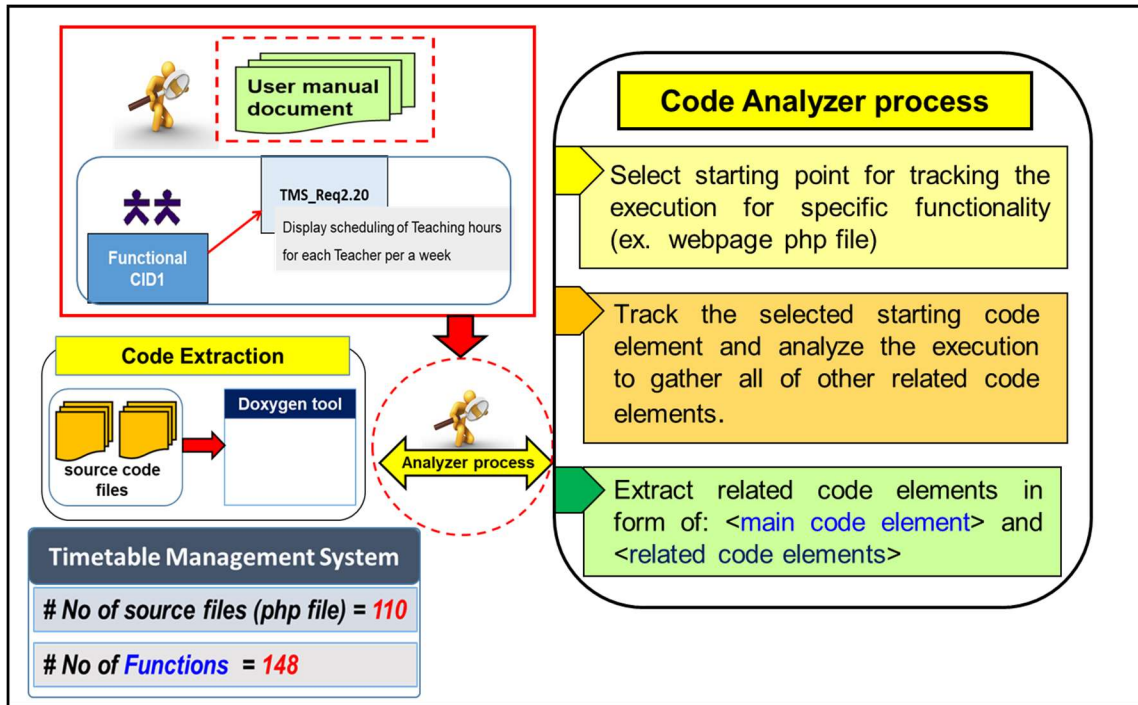


Figure 4.5 Applying Code Analyzer Process

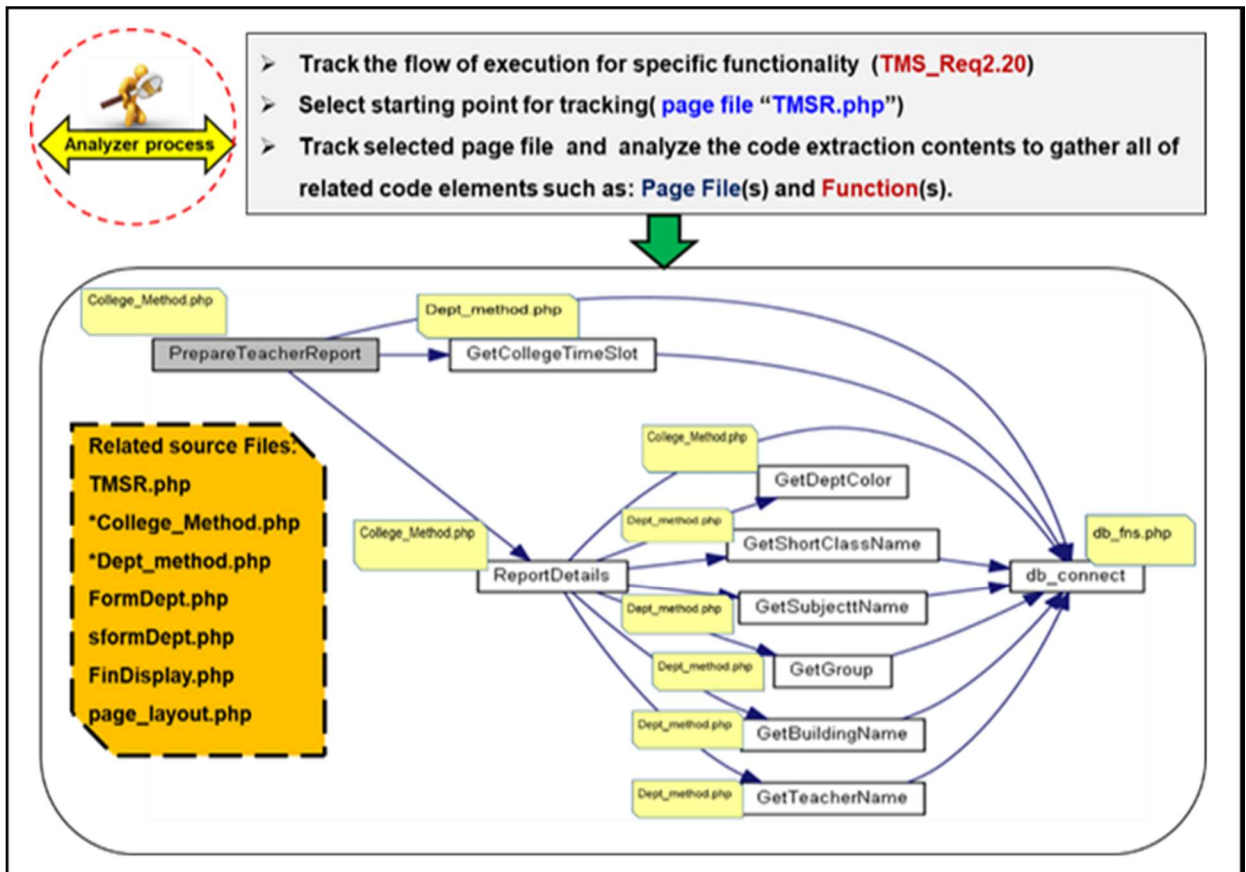


Figure 4.6 Extracted Call Graph for executing "TMS_Req2.20" functionality

4.3.4.2. Representation and Visualization of architectural information

This process includes two steps: The first step deals with mapping the extracted code elements into architectural components. The selected code elements in Figure 4.7 (for webpages and functions) are mapped into thirteen components architecture as following:

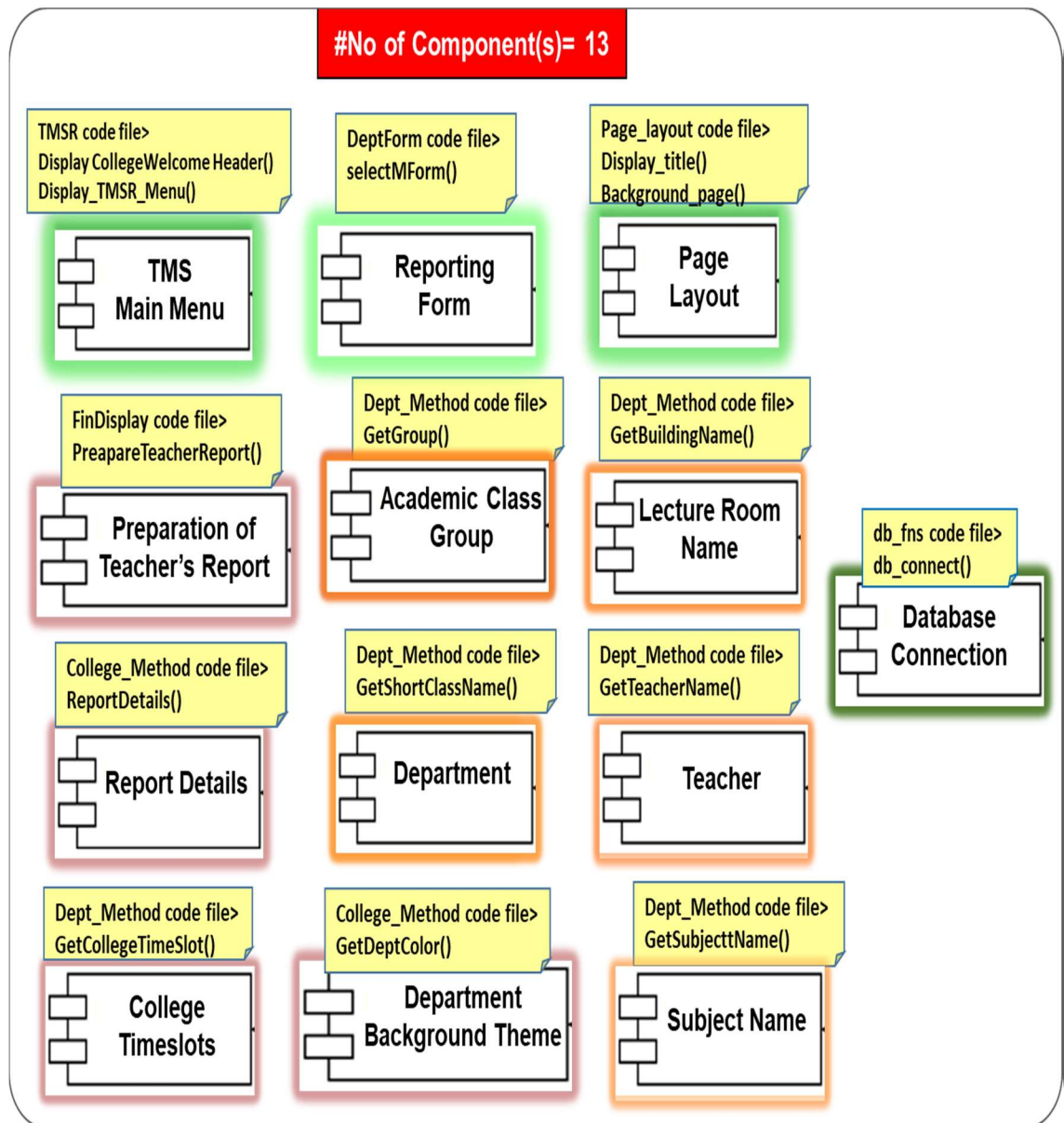


Figure 4.7 Mapping extracted code elements into Components Architecture

The second step is visualizing and representing particular architectural information using a web application layered architecture style.

The selection of the architecture type is based on Web Application Archetype which is applicable with the TMS system. The core of the Web application is the server-side logic which is visualized in a three-layer architecture.

Figure 4.8 shows the main components in each layer that are used to describe and represent “*TMS_Req2.20*” functionality as following:

- **Presentation layer** includes three components such as (*TMS Main Menu, Reporting Form and Page Layout component*). These components are responsible for managing the End-user interaction with TMS system.
- **Business layer** includes nine components which implement the core functionality of TMS system. The first four components such as (*Preparation of Teacher Report, College Timeslots, Report Detail component and DeptBackground Theme component*). These Components are concerned with the retrieval, processing, transformation, and management of TMS’s data; business rules and policies. The others five components called “business entities” which encapsulate the business logic and data necessary to present the real world elements within TMS system, such as (*Academic Class Group, Lecture Room, Teacher, Subject and Department*).
- **Data access layer** consists of the *database connection component* which provides access to the data hosted within TMS system.

To summarized *Phase(4)*, the layered architecture model is used to visualize and represent the extraction of particular architectural information into a graphical model for stakeholders which helps to answer their architectural concerns about specific functionality of the TMS system.

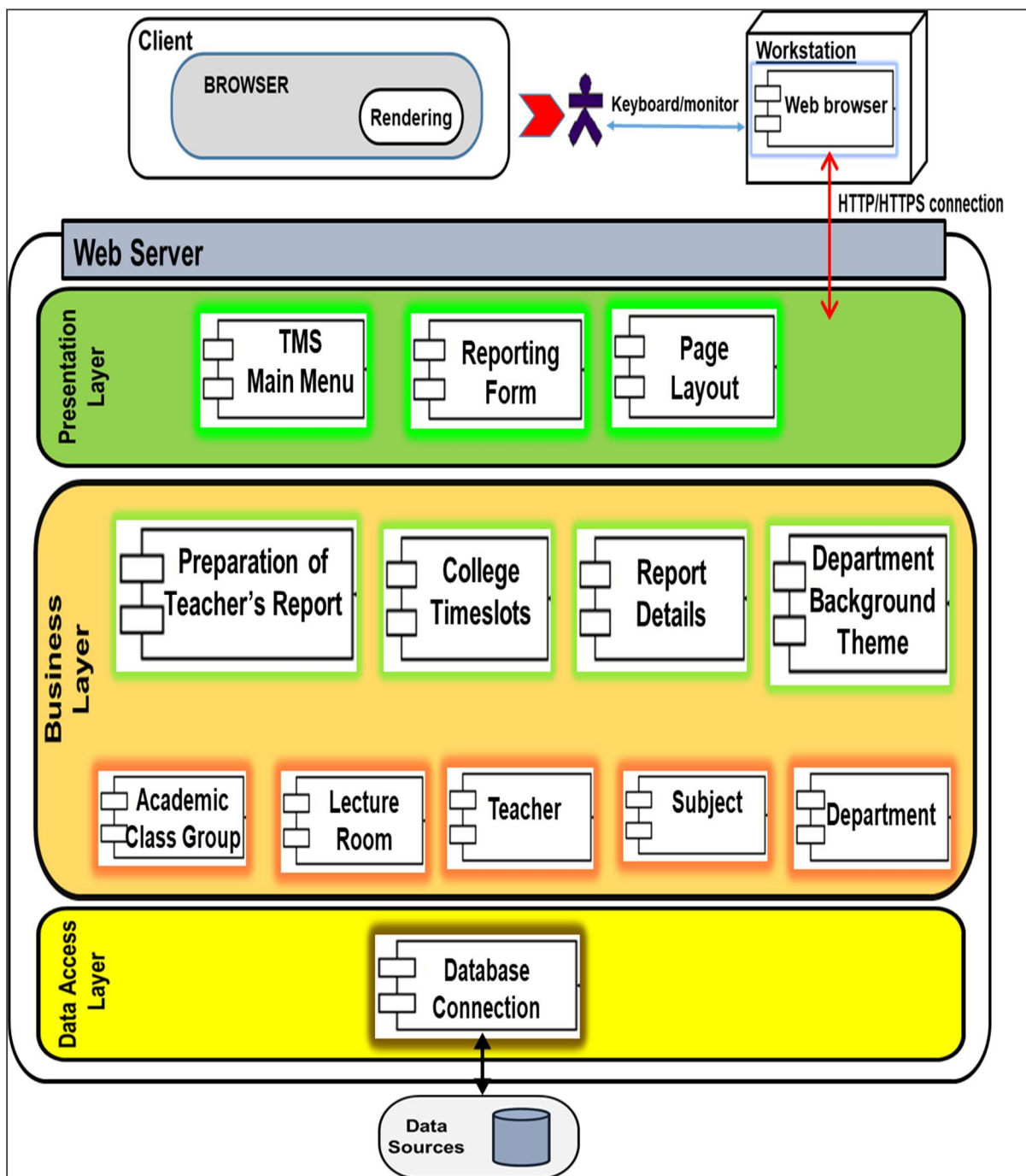


Figure 4.8 Visualizing particular Architectural Information using Layered Architecture Model

Moreover, the architectural model provides an abstract level of architectural representation for stakeholders which highlights which set of components are needed to execute specific functionality of the system. This is shown here as the functionality of the mechanism for managing the scheduling of Teachers lectures as in Figure 4.9.

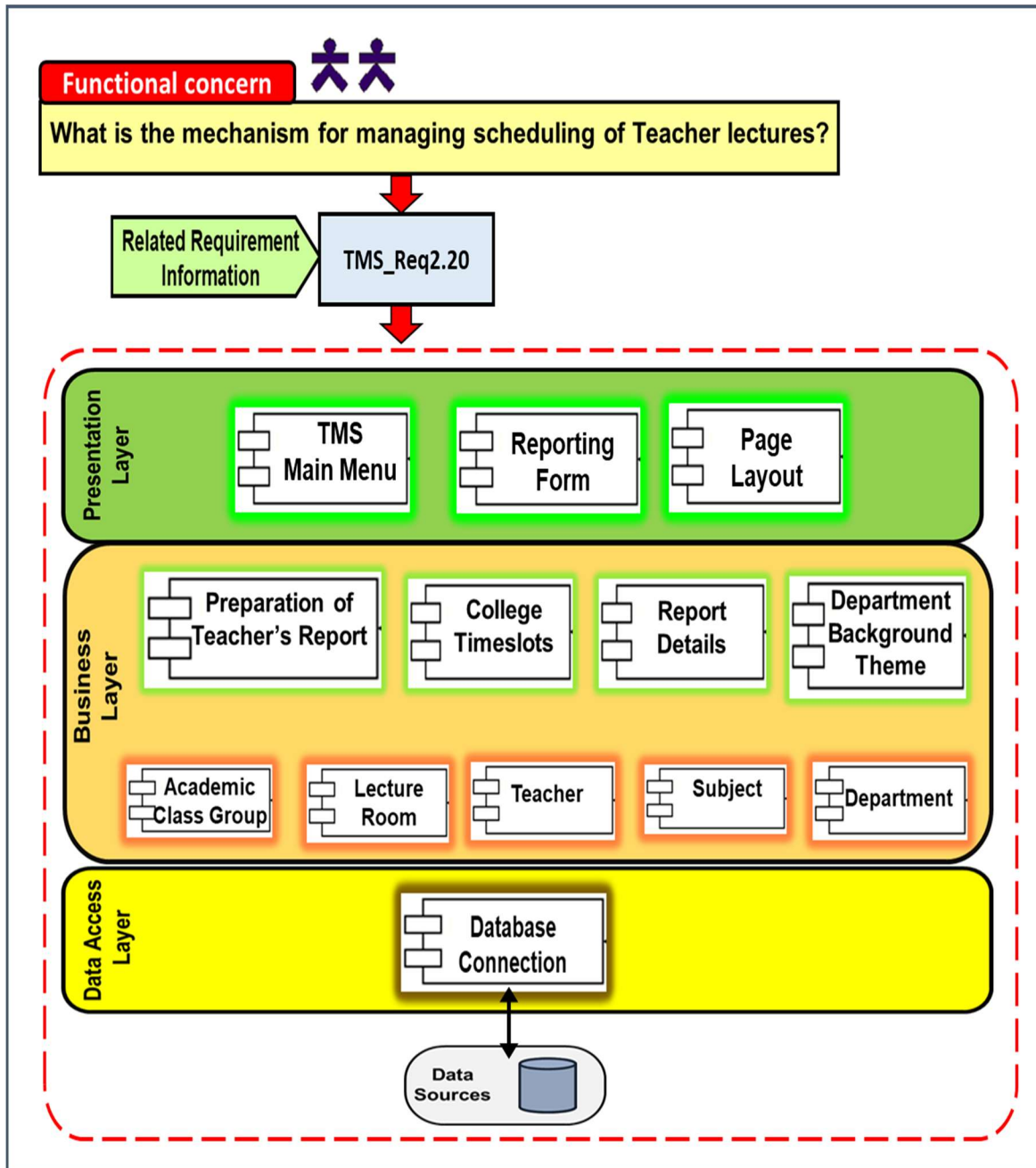


Figure 4.9 Representation of particular Architectural Information Based on Stakeholder's Functional Concern

4.4. The Main Results and Discussion

The extraction of architectural representation is very useful and helps the stakeholders especially (developer, maintainer, architect and tester) for obtaining the as built architecture of implemented software system based on its existing source code, and supporting the understand-ability and maintainability phase for the target systems.

For example; the architectural representation can be used by the *maintainer* to support the understand-ability for particular part of the system; by tracing the related requirement information through its implemented code elements and highlighted which components were needed to represent specific functionality of the target system as described in Figure 4.9.

Furthermore; in case of improving or re-engineering the legacy software system into new technology such as (object oriented system or cloud based application system); the architectural representation helps the maintainer to identify which set of components that implement the core functionality of legacy system, and encapsulate the relevant business logic, or either to decide how to manage and migrate the executable components into cloud based environment, as the following example:

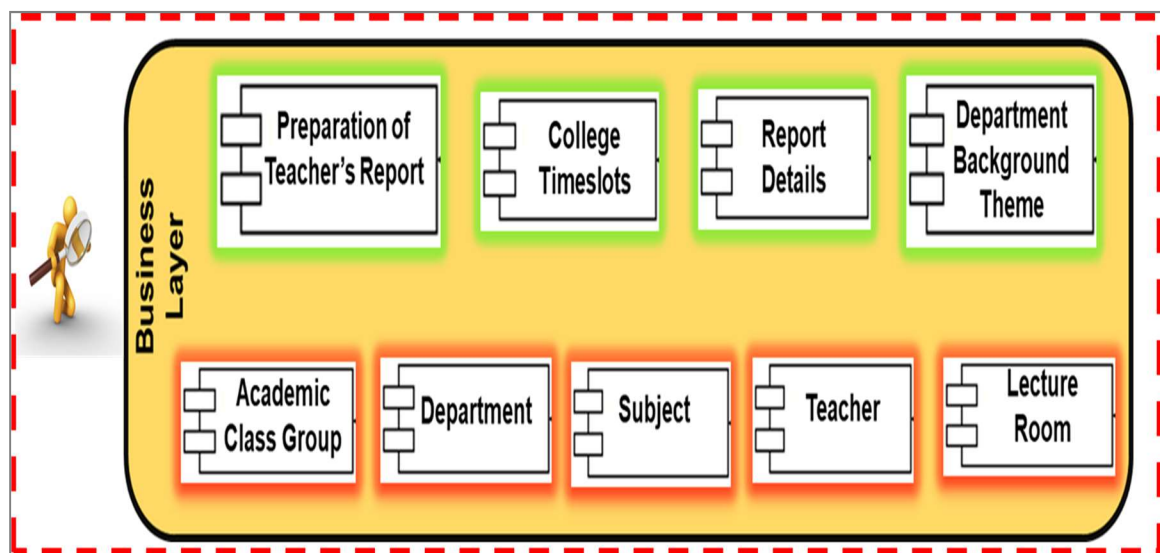


Figure 4.10 Example of the main components that implement the core functionality

Additionally, the extracted architectural information can be used by the end-user to support the understand-ability for particular part of the system by providing a proper level of architectural diagram that highlighted which components are needed to describe specific functionality. Actually, this is very important point by putting the end-user in the maintenance loop so that end-user can give feedback on the information related the target system, or either to determine and decide in case of re-engineering specific functionality of legacy software system through adding new features for the target system.

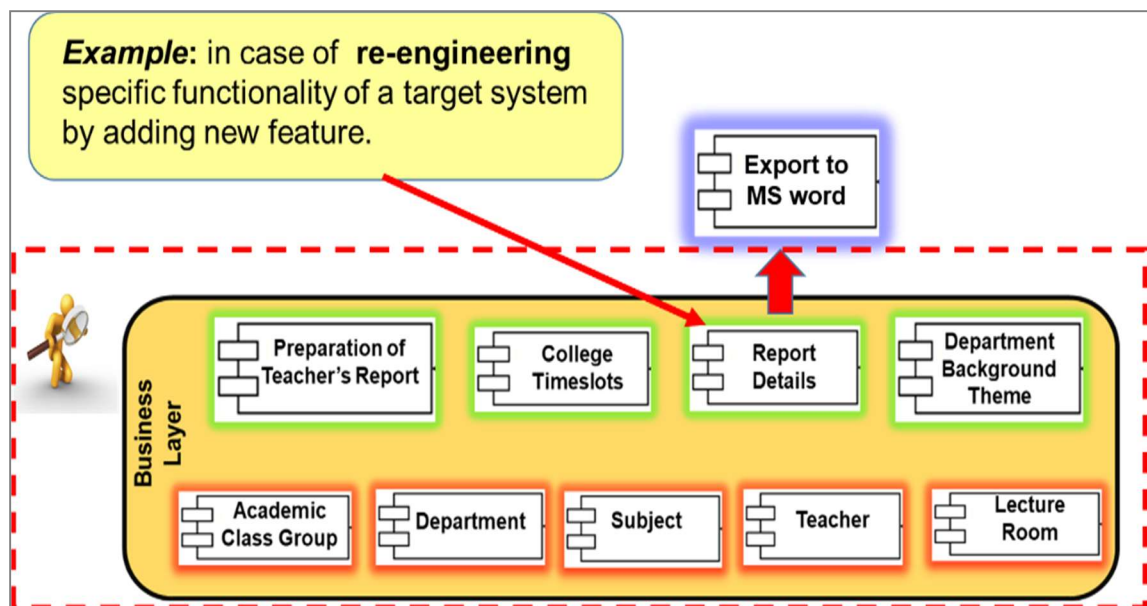


Figure 4.11 Example of how to determine and decide a new feature for a target system

To summarize; the extraction of architectural representation is very useful and helps the stakeholders for obtaining the as built architecture of implemented software system based on its existing source code, and supporting the understand-ability and maintainability phase for these existing or legacy systems. Generally; the main benefits of this extracted architectural representation summarized as follow:

- Basis for re-documenting the architecture document of legacy software systems, in case of the document is out of date or the nonexistent of document;

- Determine what to look for and focus in the extracted architectural information and help in identifying which set of components can be used for reuse; for example in case of reusing specific components to others software system.
- Starting point for re-engineering the legacy systems to a new desired architecture, or managing and upgrading them to a new technology.

4.5. Chapter Summary

This chapter described how to implement the RE methodology phases using a practical case study. The first section presented an overview of the selected software system, and described the main reasons for selecting the system. The second section described the details of applying each phase of the proposed RE methodology to a case study. The final section highlighted the main benefits of the extracted architectural information for stakeholders and discussed the main contributions of applying the proposed RE methodology.

CONCLUSION AND FUTURE WORK

The main contributions drawn from this research are: firstly; a new RE Methodology follows IEEE 1471 standard of architectural description and support concerns of stakeholder including end-user and maintainer. Secondly; GUI prototype tool to support the steps of Methodology. It supports the visualization of a particular part of the target system by providing a visual model of the architectural representation which highlights the main components needed to execute specific functionality of the target system. Finally; the verification of the methodology using legacy web application system.

Further information; the extraction of architectural representation helps stakeholders especially (maintainer, end-user, architect, tester and developer) for obtaining the as built architecture from its implemented source code elements, and supporting the understand-ability and maintainability phase for the target system.

For example; the architectural representation can be used by the maintainer to support the understand-ability for particular part of the system; by tracing the related requirement information through its implemented code elements and highlighted which components were needed to represent specific functionality of the target system as described in a case study.

Furthermore; in case of improving or re-engineering the legacy software system into new technology such as (object oriented system or cloud based application system); the architectural representation helps the maintainer to identify which set of components that implement the core functionality of legacy system, and encapsulate the relevant business logic, or either to decide how to manage and migrate the executable components into cloud based environment.

The extracted architectural information can be used by the end-user to support the understand-ability for particular part of the system by providing a proper level of

architectural diagram that highlighted which components are needed to describe specific functionality. Actually, this is very important point by putting the end-user in the maintenance loop so that end-user can give feedback on the information related the target system, or either to determine and decide in case of re-engineering specific functionality of legacy software system through adding new features for the target system.

RECOMMENDATIONS FOR FUTURE WORK

This section presents some recommendations for future work. While many issues related to this area of research remain to be explore. Therefore, this thesis could be extended in several directions to cover additional related issues. These issues can be highlighted as the following key points:

- There is a need to extend the proposed RE methodology to support and apply additional architectural viewpoints beside the selected “*Functional viewpoint*” based on a given classification of viewpoints catalog (such as: the information viewpoint, the deployment viewpoint, and the operational viewpoint),
- The development of automated tool is needed to support and include the whole phases of the proposed RE methodology,
- Furthermore; the proposed RE methodology can be apply in different application domains beside the legacy software systems such as: the robotics systems and smart object systems to support the understand-ability and maintainability process for the particular parts of these systems,
- The proposed RE methodology can be extremely important for iterative migration of legacy systems. Accordingly; the extraction of architectural representations from the proposed RE methodology were based on the layered architecture model. This model is used to visualize and represent the extraction of particular architectural information into a graphical model to answer the stakeholder's concerns about specific functionality of the target

system. Furthermore; this architectural representation will help stakeholders such as the maintainer, architect, tester and developer to inspect the dependencies of the different parts of the architecture obtained from specific source code segments of the legacy system. This will support the understand-ability process by identifying which set of components implement the core functionality of the target legacy system. The visual models also encapsulate the relevant business logic. This information is needed to manage and migrate the executable components into the desired cloud based environment or either into mobile based environment.

- The important concepts of the container technology and the microservices architecture style show the importance of the proposed RE methodology in migrating legacy systems architectures to scalable cloud applications architectures. Accordingly; the layered architecture model from the proposed RE methodology will support the understand-ability process by identifying which set of components that implement the core functionality of the target legacy system can be migrated as microservices in containers. This will help the stakeholders decide how these components can be factored out as microservices and allocated to different containers.

APPENDICES

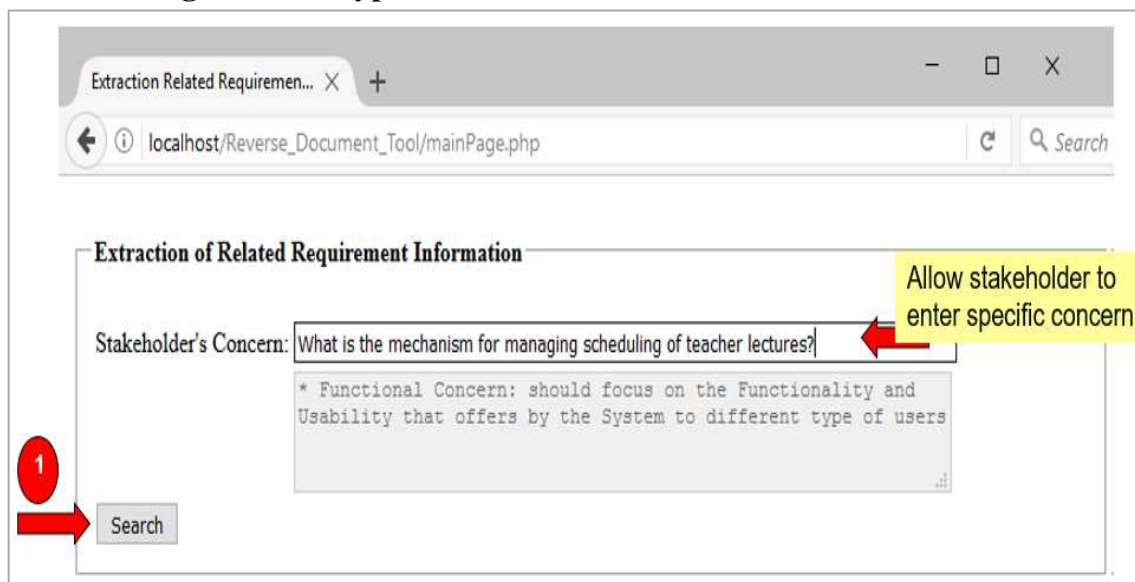
APPENDIX A

The Execution of GUI Prototype Tool

The prototype tool is a Web-based tool which has a graphical user interface (GUI), and built by using MySQL database and PHP web page language. The prototype tool allows stakeholders to enter a specific concern in form of a “query”. The specific concern will be elicited from the functional requirements repository assumed to be available for the target software system. The main functions of prototype tool is to extract a set of related requirement information based on the elicited concern, and create a trace link between elicited concern and its relevant information. The following is the execution of the prototype tool:

- A.1: Main Page of Prototype Tool
- A.2: Extraction of Related Requirement information
- A.3: Review the suggested Results
- A.4: Create a trace link

A.1: Main Page of Prototype Tool



A.2: Extraction of Related Requirement information

The screenshot displays a web browser window with the address bar showing `localhost/Reverse_Document_Tool/mainPage.php?go=1`. The page title is "Extraction Related Requirement Information".

Stakeholder's Concern: What is the mechanism for managing scheduling of teacher lectures?
* Functional Concern: should focus on the Functionality and Usability that offers by the System to different type of users

List of Suggested Requirement(s):

# 10 #	TMS_Req2.20->[3.20673]
	TMS_Req3.4->[3.20673]
	TMS_Req2.19->[2.80108]
	TMS_Req3.3->[2.43999]
	TMS_Req2.18->[2.38913]

10/ [35 Reqs]

*Note: You can select multiple requirements from the Dropdown list by Clicking on the mouse and press on (Shift or Ctrl)key from the keyboard..

Search Save Traceability Link

Fetch the related Requirement(s)

DB Repository

- Requirement repository
- Source_file(.php)_repository

Review Suggested Result(s)

High	Medium	Low
[weight]	[weight]	[weight]

A.3: Review the suggested Results

Extraction of Related Requirement Information

Stakeholder's Concern:

* Functional Concern: should focus on the Functionality and Usability that offers by the System to different type of users

List of Suggested Requirement(s):

# 10 #	TMS_Req2.20->[3.20673]
	TMS_Req3.4->[3.20673]
	TMS_Req2.19->[2.80108]
	TMS_Req3.3->[2.43999]
	TMS_Req2.18->[2.38913]

*Note: You can select multiple requirements from the Dropdown list by Clicking on the mouse and press on (Shift or Ctrl)key from the

[Review Suggested Result\(s\)](#)

High [weight] Medium [weight] Low [weight]

2 Click to review all of suggested results

ملحوظة هامة: النتائج التي تم إستخلاصها يكون ترتيبها وفقاً لدرجة تقاربها للإستفسار المدخل من قبل المستخدم. ويتم ذلك من خلال قيمة [وزن] يتم حسابه بهذه المرحلة بهدف تحديد درجة التقارب كما يلي:

- اللون الأخضر < أعلى قيمة لدرجة التقارب
- اللون الأصفر < قيمة متوسطة لدرجة التقارب
- اللون الأحمر < أقل قيمة لدرجة التقارب بالمقارنة مع القيمة الأعلى والمتوسطة.

2 [Review Suggested Result\(s\)](#)



High [weight]	Medium [weight]	Low [weight]
------------------	--------------------	-----------------

Extraction Related Requiremen... X Review Suggested Results X +

localhost/Reverse_Document_Tool/SuggestResults.php?concern=What is the mechanism for managing scheduling of t Search

Review Suggested Result(s) # 10 #

[weight]	Requirement_Code	Description
[3.20673]	TMS_Req2.20	Display the scheduling of Teaching hours for each Teacher per a week
[3.20673]	TMS_Req3.4	Display the scheduling of Teaching hours for each Teacher per a week.
[2.80108]	TMS_Req2.19	Display the scheduling of all lectures and labs for each academic program per semester, allow college admin to edit or delete the scheduling slots
[2.43999]	TMS_Req3.3	Display the scheduling of all lectures and labs for each academic program per semester.
[2.38913]	TMS_Req2.18	Update/Delete the scheduling of lectures and labs for each academic program class per semester
[2.21122]	TMS_Req2.16	Reserve a laboratory for scheduling lectures lab of the academic class and extend an existing laboratory reservation per semester
[2.20162]	TMS_Req2.15	Reserve a lecture room for scheduling lectures of the academic program and extend an existing room reservation per semester
[2.1677]	TMS_Req3.1	Using the web interface, Teacher and student allowed to access the University page (without login)
[2.05831]	TMS_Req3.2	Display the scheduling of all lectures and labs on each semester per a week. If there are many Colleges that used the specific lecture room during the week, then the displaying of scheduled slots will appear with specific color for each college

A.4: Create a trace link

Extraction Related Requiremen... X +

localhost/Reverse_Document_Tool/mainPage.php?go=1

Extraction of Related Requirement Information

Stakeholder's Concern:

* Functional Concern: should focus on the Functionality and Usability that offers by the System to different type of users

List of Suggested Requirement(s):

10

TMS_Req2.20->[3.20673]

TMS_Req3.4->[3.20673]

TMS_Req2.19->[2.80108]

TMS_Req3.3->[2.43999]

TMS_Req2.18->[2.38913]

*Note: You can select multiple requirements from the Dropdown list by Clicking on the mouse and press on (Shift or Ctrl)key from the keyboard..

List of Extracted Related Requirement Information

Display all specified link between concerns and Requirements

Findings: # 2 #

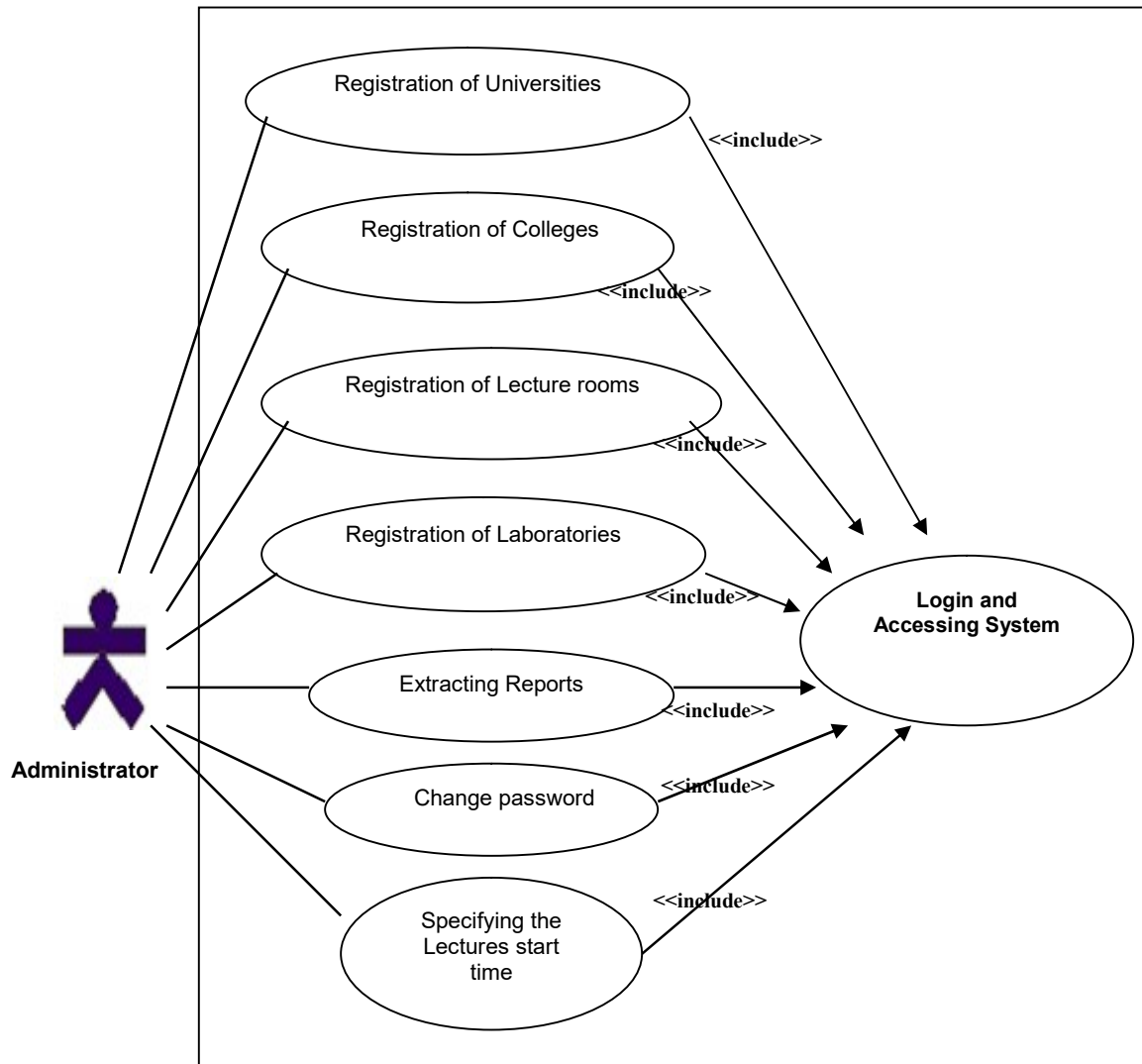
ConcernID	Description	Related Requirement	Created Date
CID1	What is the mechanism for managing scheduling of teacher lectures?	TMS_Req2.20	2017-05-02
CID1	What is the mechanism for managing scheduling of teacher lectures?	TMS_Req3.4	2017-05-02

APPENDIX B

Timetable Management System (TMS)

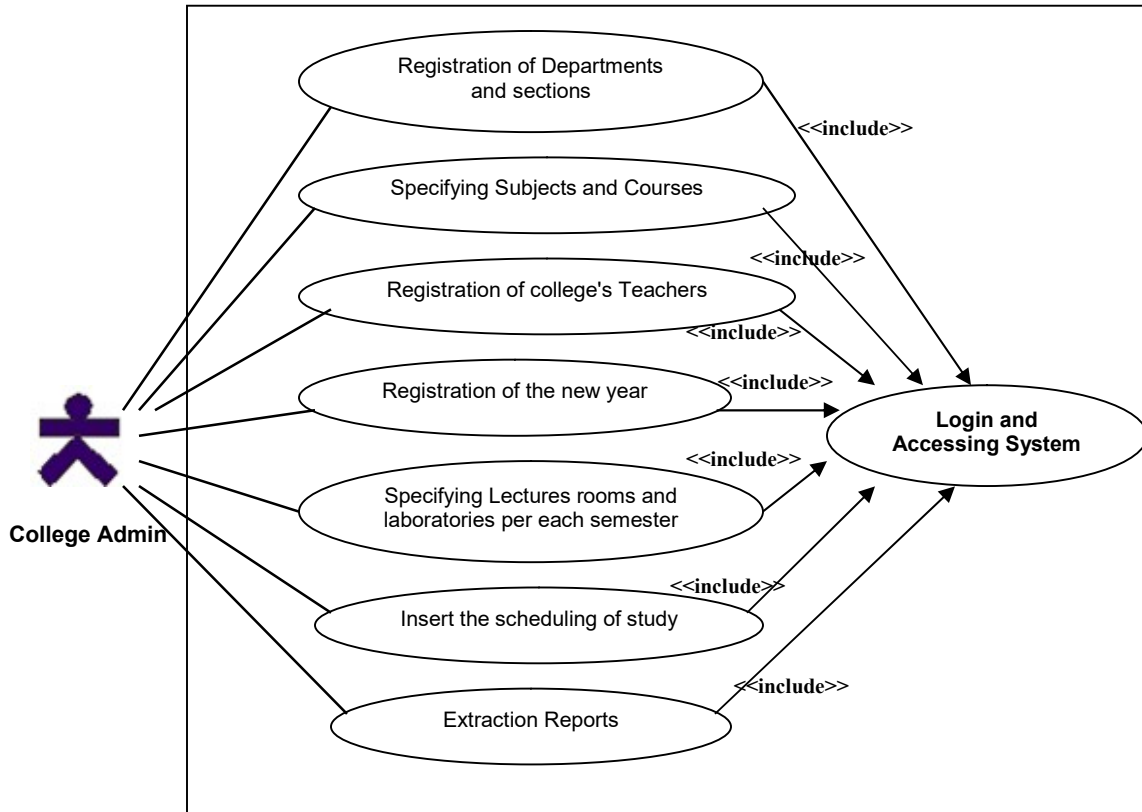
➤ The Privileges of System's Users:

B.1: System Administrator

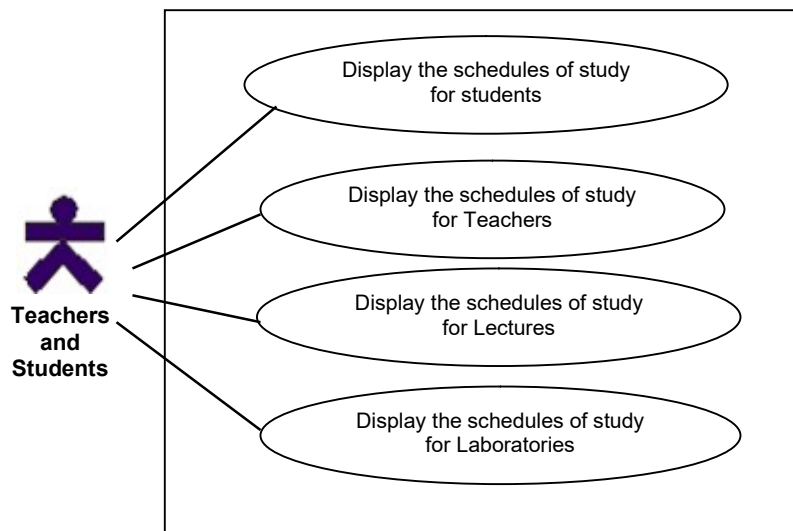


➤ **The privileges of System's Users:**

B.2: Administrator of College (CollegeAdmin)

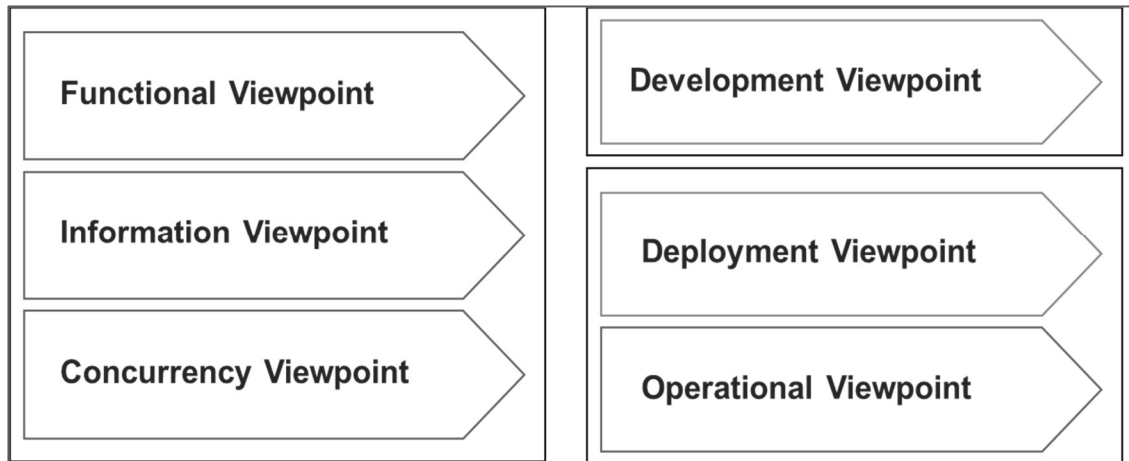


B.3: Teachers and Students



APPENDIX C

Viewpoint Catalog (Rozanski & Woods, 2011)

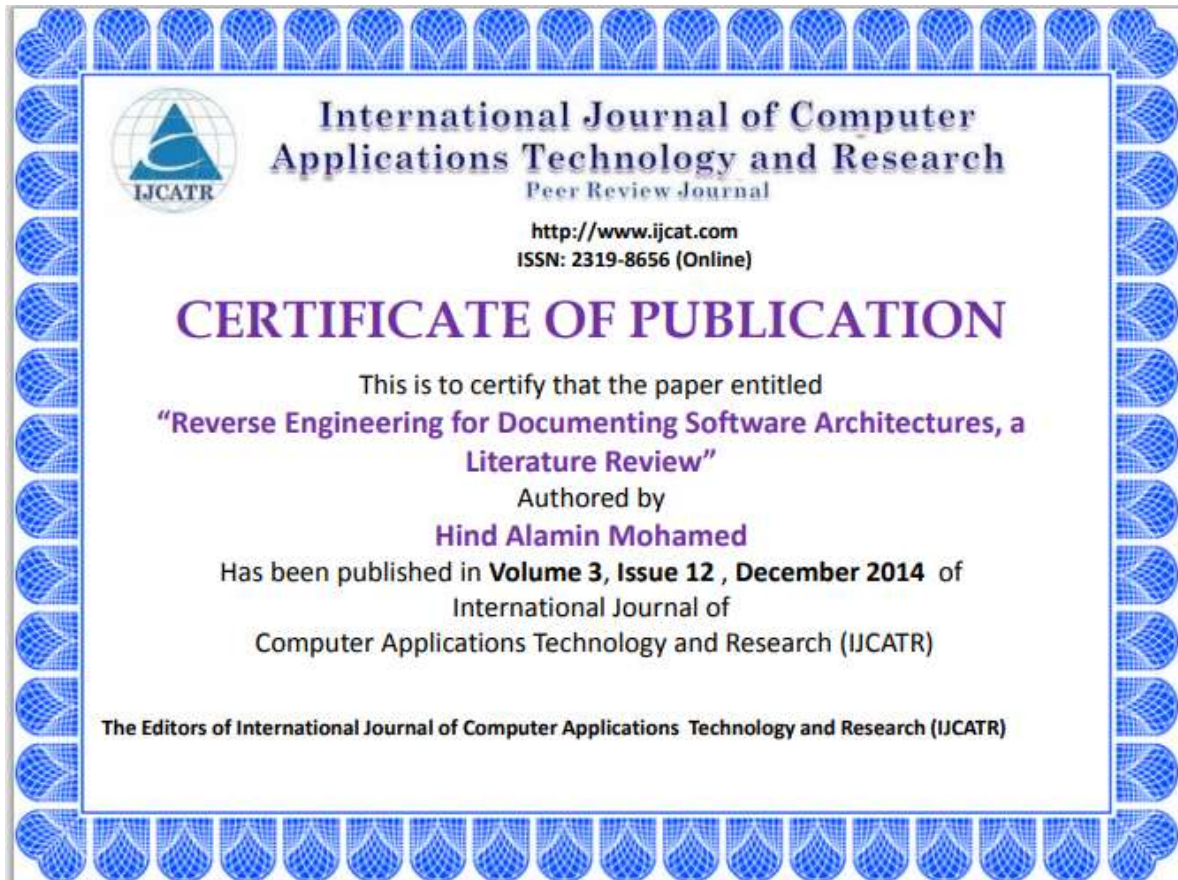


#	Description
Functional	Describes the system’s functional elements, their responsibilities, interfaces, and primary interactions. A Functional view is the cornerstone of most ADs and is often the first part of the description that stakeholders try to read. It drives the shape of other system structures such as the information structure, concurrency structure, deployment structure, and so on. It also has a significant impact on the system’s quality properties such as its ability to change, its ability to be secured, and its runtime performance.
Information	Describes the way that the architecture stores, manipulates, manages, and distributes information. The ultimate purpose of virtually any computer system is to manipulate information in some form, and this viewpoint develops a complete but high-level view of static data structure and information flow. The objective of this analysis is to answer the big questions around content, structure, ownership, latency, references, and data migration.

#	Description
Concurrency	Describes the concurrency structure of the system and maps functional elements to concurrency units to clearly identify the parts of the system that can execute concurrently and how this is coordinated and controlled. This entails the creation of models that show the process and thread structures that the system will use and the interprocess communication mechanisms used to coordinate their operation.
Development	Describes the architecture that supports the software development process. Development views communicate the aspects of the architecture of interest to those stakeholders involved in building, testing, maintaining, and enhancing the system.
Deployment	Describes the environment into which the system will be deployed, including capturing the dependencies the system has on its runtime environment. This view captures the hardware environment that your system needs (primarily the processing nodes, network interconnections, and disk storage facilities required), the technical environment requirements for each element, and the mapping of the software elements to the runtime environment that will execute them.
Operational	Describes how the system will be operated, administered, and supported when it is running in its production environment. For all but the simplest systems, installing, managing, and operating the system is a significant task that must be considered and planned at design time. The aim of the Operational viewpoint is to identify system-wide strategies for addressing the operational concerns of the system's stakeholders and to identify solutions that address these.

APPENDIX D

Certification of Publication



REFERENCES

Alamin, H. & Ammar, H., 2014. Reverse Engineering for Documenting Software Architectures, a Literature Review. *International Journal of Computer Applications Technology and Research*, Dec, 3(12), pp. 785-790, ISSN: 2319-8656 (Online).

Alamin, H. & Ammar, H., 2020. Concerns-Based Reverse Engineering for Partial Software Architecture Visualization, *International Journal on Informatics Visualization*, vol. 4(2020), no.2, Apr 2020, pp. 58 - 68, ISSN: 2549-9610 (Online).

Anquetil , N. & C. Lethbridge, T., 1999. Recovering Software Architecture from the Names of Source Files. *Journal of Software Maintenance: Research and Practice*, 3(11), pp. 201-221.

Arshad, R. & Lau , K. K., 2017. Extracting Executable Architecture From Legacy Code Using Static Reverse Engineering. s.l., s.n., pp. 55-59.

Che, M., 2013. *An Approach to Documenting and Evolving Architectural Design Decisions*. San Francisco, CA, USA, IEEE, pp. 1373-1376.

Che, M. & Dewayne, E. P., 2011. Scenario-based architectural design decisions documentation and evolution. In *Proceedings of Engineering of Computer Based Systems (ECBS'11)*, IEEE, 2011, 27-29 Aprilpp. 216-225.

Che, M. & Dewayne, E. P., 2012. Managing architectural design decisions documentation and evolution. In *Proceedings of 6th International Journal of Computers*, pp. 137-148.

Chikofsky, E. J. & James, H. C., 1990. Reverse Engineering and Design Recovery: A Taxonomy. *IEEE Software*, 13-17 January, 7(1), pp. 13-17.

Clements, P., 2005. Comparing the SEI's Views and Beyond Approach for Documenting Software Architectures with ANSI-IEEE 1471-2000, Carnegie Mellon University: Software Engineering Institute.

Clements, P., Bachmann, F., Bass, L. & Ga, D., 2010. Prologue: Software Architectures and Documentation. [Online] [Accessed 26 April 2014].

Demeyer, S., Stéphane, D. & Nierstrasz, 2008. Object Oriented Reengineering Patterns. In: Switzerland: Square Bracket Associates, pp.338.

Garg, M. & Jindal, M. K., 2009. Reverse Engineering Roadmap to Effective Software Design. International Journal of Recent Trends in Engineering, May.1(2).

Harman, M., Langdon, W. B. & Weimer, W., 2013. Genetic Programming for Reverse Engineering. Koblenz, Germany, IEEE.

Heesch, D. v., Copyright © 1997-2016. Doxygen Tool website and Doxygen Documentation. [Online] Available at: <http://www.doxygen.org/download.html>

Henk, K. & Vliet, H. V., 2006. A method for defining IEEE Std 1471 viewpoints. Journal of Systems and Software, ELSEVIER, 79(1), pp. 120-131.

Hugo, B., Cabot, J., Dupé, G. & Madiot, F., 2014. MoDisco: A Model Driven Reverse Engineering Framework. Information and Software Technology, 56(8), pp. 1012-1032.

Imam Ya'u , B. & Yusuf, M. N., February 2018. Building Software Component Architecture Directly from User Requirements. International Journal Of Engineering And Computer Science, 7(2), pp. 23557-23566.

Institute of Electrical and Electronics Engineers, 2000. IEEE Recommended Practice for Architectural Description of Software Intensive Systems. [Online] Available at: <http://cabibbo.dia.uniroma3.it/ids/altrui/ieee1471.pdf> [Accessed 9 July 2014].

Khadka, R., A., S., S., J. & J., H., 2013. A structured legacy to SOA migration process and its evaluation in practice. 2013 IEEE 7th International Symposium in Maintenance and Evolution of Service-Oriented and CloudBased Systems (MESOCA), pp. 2-11.

Kim, W., Chung, S. & Endicott Popovsky, B., October 15–18, 2014, ACM. Software Architecture Model Driven Reverse Engineering Approach to Open Source Software Development. Atlanta, Georgia, USA, ACM, pp. 9-14.

Kruchten, P., 1995. Architectural Blueprints: The “4+1” View Model of Software Architecture. IEEE Software, 6(12), p. 42–50.

Kumar, N., 2013. An approach for Reverse Engineering thorough Complementary Software Views. In Proceedings of International Conference on Emerging Research in Computing, Information, Communication and Applications (ERCICA'13), pp. 229-234.

Lau, K. & Tran, C. M., 2012. X-man: An mde tool for Component based System Development. IEEE, IEEE, pp. 158-165.

Len Bass, R. K. & Celements, P., 2003. Software Architecture in Practice. 2nd ed. s.l.:Addison Wesley Professional.

Liang, G. & Yu, L., December, 2013. Quality Driven Re-engineering Framework, Sweden: Blekinge Institute of Technology.

M. Harman, W. B. Langdon, and W. Weimer, 2013. Genetic Programming for Reverse Engineering. Koblenz, Germany, IEEE.

M. Harman, Yue J., W. B. Langdon, J. Petke, Iman H. Moghadam, Shin Y. and F. Wu., 2014. Genetic Improvement for Adaptive Software Engineering. s.l., In Proceedings of 9th International Symposium on Software Engineering for Adaptive and Self-Managing System.

Maras, J., Štula , M. & Crnkovic, I., 2009. PHPModeler- a Web Model Extractor. IEEE Computer Society (Nov2009), s.n., pp. 660-661.

MI C R O S O F T ® Architecture guide, 2009. MI C R O S O F T ® Application Architecture Guide(patterns & practices Developer Center), Application ArcheTypes, Chapter 20: Choosing an Application Type. [Online] Available at: <https://msdn.microsoft.com/en-us/library/ee658104.aspx>

MySQL 5.7 Reference Manual Document, 2017. MySQL 5.7 Reference Manual Document /Full-Text Search Functions/Full-Text Restricitions. [Online] Available at: <https://dev.mysql.com/doc/refman/5.7/en/> [Accessed 28 March 2017].

MySQL 5.7 Reference Manual Document, 2017. MySQL 5.7 Reference Manual Document/Full-Text Search Functions/Natural Language Full-Text Searches. [Online] Available at: <https://dev.mysql.com/doc/refman/5.7/en/fulltext-natural-language.html> [Accessed 25 March 2017 at 8:00AM].

MySQL Reference Manual, n.d. MySQL Reference Manual/Important Algorithms and Structures/10.7-Full-TextSearch. [Online] Available at: <https://dev.mysql.com/doc/internals/en/full-text-search.html> [Accessed 1 April 2017 at 05:30PM].

Nicholas, M., 2005. A survey of Software Architecture Viewpoint Models. s.l., s.n., pp. 13-24.

Panas , T., Lowe, W. & Aßmann , U., n.d. Towards the Unified Recovery Architecture for Reverse Engineering. [Online] Available at: <https://ai2-s2-pdfs.s3.amazonaws.com/b8e1/c9bd8cf3360b82de68e8049b281a1e2f4a25.pdf> [Accessed 30 October 2017].

Penta, G. C. & Massimiliano , D., 2008. Frontiers of Reverse Engineering: a Conceptual Model. pp. 38-47.

R.Hilliard, D. Emery, M. Maier, 2007. All About IEEE Std 1471. [Online] Available at: <http://www.csee.wvu.edu/~ammar/CU/swarch/lectureslides/slidesstandards/all-about-ieee-1471.pdf>

Razavizadeh, A., Verjus, H., Cimpan, S. & Ducasse, S., 2009. Multiple Viewpoints Architecture Extraction. 2009 IEEE/IFIP WICSA/ECSA, pp. 329-332.

Riva, C. & Yang, Y., 2002. Generation of architectural documentation using XML. IEEE Computer Society Press, In Proceedings of the 9th Working Conference on Reverse Engineering (WCRE02), pp. 161-169.

Rosenberg, L. H. & Lawrence, E. H., 1996. Software re-engineering. [Online] Available at: <http://www.scribd.com/doc/168304435/Software-Re-Engineering1> [Accessed 26 April 2014].

Rozanski, N. & Woods, E., 2005. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. 2nd ed. s.l.:Addison Wesley.

Rozanski, N. & Woods, E., 2011. Applying viewpoints and views to software architecture. [Online] Available at: http://www.viewpointsandperspectives.info/vpandp/wpcontent/themes/secondedition/doc/VPandV_WhitePaper.pdf [Accessed 14 June 2015].

Rozanski, N. & Woods, E., 2011. Viewpoints and Concerns. [Online] Available at: <http://www.viewpoints-andperspectives.info/home/viewpoints> [Accessed 10 November 2015].

Rozanski, N. & Woods, E., n.d. Viewpoints and Perspectives Reference Card. [Online] Available at: <http://www.viewpoints-and-perspectives.info/home/viewpoints/functional-viewpoint/> [Accessed 10 November 2015].

Saeidi, A., 2013. Migrating a large scale legacy application to SOA: Challenges and lessons learned. In Reverse Engineering (WCRE), 2013 20th Working Conference, pp. 425-432.

Shahin, M., Liang, P. & Khayyambashi, M., 2009. A Survey of Architectural Design Decision Models and Tools. Technical Report SBU-RUG-2009-SL-01. [Online] Available at: <http://www.cs.rug.nl/search/uploads/Publications/shahin2009sad.pdf> [Accessed 8 July 2014].

Shahin, M., Liang, P. & Khayyambashi, M., 2009. Architectural Design Decision: Existing models and tools. In Proceedings of Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference 2009, pp. 293-296.

Somasegar, D. H. S. & Guthrie, S., 2009. MICROSOFT APPLICATION ARCHITECTURE GUIDE (Patterns & Practices Developer Center), s.l.: s.n.

Starke, G. & Hruschka, P., 2017. Arc24 Template for documentation of software and system architecture. [Online] Available at: <http://www.arc24.de>

Stringfellow, C., Amory, C. D. & Potnur, D., 2006. Comparison of software architecture reverse engineering methods. In Proceedings of Information and Software Technology, July, 7(48), pp. 484-497.

Woods, E., 2004. Experiences Using Viewpoints for Information Systems Architecture: An Industrial Experience Report. European Workshop on Software Architecture, May, pp. 182-193.

Zalazar, A., Gonnet, S. & Leone, H., 2015. Migration of Legacy Systems to Cloud Computing. Electronic Journal of SADIO (EJS), Issue 14, pp. 42-55.