A COMPONENT BASED MODEL DRIVEN SOFTWARE DEVELOPMENT FRAMEWORK FOR WEB-BASED APPLICATIONS

A MASTER'S THESIS

IN

SOFTWARE ENGINEERING

ATILIM UNIVERSITY

BY

AFRAH UMRAN ALRUBAEE

JANUARY 2017

A COMPONENT BASED MODEL DRIVEN SOFTWARE DEVELOPMENT FRAMEWORK FOR WEB-BASED APPLICATIONS

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF

ATILIM UNIVERSITY

BY

AFRAH UMRAN ALRUBAEE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF SOFTWARE ENGINEERING

JANUARY 2017

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof.Dr. İbrahim AKMAN

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof.Dr. Ali YAZICI

Head of Department

This is to certify that we have read the thesis "A Component Based Model Driven Software Development Framework For Web-Based Applications" submitted by "Afrah Umran Alrubaee" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst.Prof.Dr. Deniz ÇETİNKAYA

Supervisor

Examining Committee Members

Asst.Prof.Dr. Deniz ÇETİNKAYA _____

Asst.Prof.Dr. Çiğdem TURHAN _____

Assoc.Prof.Dr. Sevil ŞEN _____

Date: 25 January 2017

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name Last, name : Alrubaee, Afrah

Signature:

ABSTRACT

A COMPONENT BASED MODEL DRIVEN SOFTWARE DEVELOPMENT FRAMEWORK FOR WEB-BASED APPLICATIONS

Alrubaee, Afrah

M.S., Software Engineering Department

Supervisor: Asst.Prof.Dr. Deniz Çetinkaya

January 2017, 72 pages

Developing a high quality, cost effective, reliable and on time software systems is a challenging task due to the large size and complexity of these systems. Traditional developing approaches that are concerned with constructing software systems from scratch cannot be suitable for dealing with this challenge. For that reason several approaches have been introduced to increase the productivity of the development. Two of these approaches are component based software engineering and model driven software development.

Component based software engineering (CBSE) has been introduced as a solution for software reusability problem by using reusable software components to build new software system. Model driven development (MDD) is another approach in software development that was introduced to increase productivity and decrease the cost and effort. MDD aims to face the challenges of software development process through representing the essential aspects of the required system as models and generating the final source code from these models.

The aim of this work is proposing a software development framework that combines MDD and CBSE approaches for merging the advantages and features of these approaches to facilitate software development. The framework is used successfully to develop an e-learning system as a case study. The framework was evaluated by making a comparison between existing component based software development process models and our framework.

Keywords: Component based, Model driven development, Web application.

ÖZ

WEB TABANLI UYGULAMALAR İÇİN BİLEŞEN TABANLI VE MODEL GÜDÜMLÜ BİR YAZILIM GELİŞTİRME ÇERÇEVESİ

Alrubaee, Afrah

Yüksek Lisans, Yazılım Mühendisliği Bölümü

Tez Yöneticisi:Yrd.Doç.Dr. Deniz Çetinkaya

2017, 72 sayfa

Yüksek kaliteli, uygun maliyetli, güvenilir ve zamanında tamamlanmış yazılım sistemlerini geliştirmek, bu sistemlerin büyüklüğü ve karmaşıklığı nedeniyle oldukça zor bir iştir. Geleneksel sıfırdan yazılım geliştirme yaklaşımlarıbu zorlukla baş edebilmek için uygun olmayabilir. Bu nedenle yazılım geliştirme verimliliğini artırmak için farklı yaklaşımlar önerilmiştir. Bu yaklaşımlardan ikisi bileşen tabanlı yazılım mühendisliği ve model güdümlü yazılım geliştirme yaklaşımıdır.

Bileşen tabanlı yazılım mühendisliği (CBSE), bir yazılımı geliştirirken yeniden kullanılabilir yazılım bileşenleri kullanarak,yeniden kullanılabilirlik problemine çözüm olarak önerilmiştir. Model güdümlü yazılım geliştirme (MDD), verimliliği artırmak, maliyeti ve harcanan eforu düşürmek için ortaya çıkmış başka bir yaklaşımdır. MDD, istenen sistemin temel özelliklerini modeller olarak temsil ederek ve bu modellerden nihai kaynak kodu üreterek yazılım geliştirme sürecinin zorluklarıyla yüzleşmeyi amaçlamaktadır.

Bu çalışmanın amacı, yazılım geliştirmeyi kolaylaştırmak için MDD ve CBSE yaklaşımlarının avantajlarını ve özelliklerini birleştirerek yeni bir yazılım geliştirme yöntemive çerçevesi önermektedir. Önerilen yöntem, bir e-öğrenme sistemi geliştirmek için başarıyla bir vaka çalışmasında kullanılmıştır. Önerilen çerçeve, mevcut bileşen tabanlı yazılım geliştirme süreç modelleri ile karşılaştırılarak değerlendirilmiştir.

Anahtar kelimeler: Bileşen tabanlı, Model güdümlü geliştirme, Web uygulaması.

To My Family

ACKNOWLEDGMENTS

I express sincere appreciation to my supervisor Asst.Prof.Dr. Deniz Çetinkaya for her guidance and insight throughout the research. To my family, I offer sincere thanks for their continuous support during this period.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CBSE	Component Based Software Engineering
MDD	Model Driven Development
CBD	Component Based Development

CHAPTER 1

INTRODUCTION

Developing a software system is not an easy task because there are many challenges during the development process in different phases. Examples of these challenges are the problem where the owner does not provide a clear explanation of the problem or the requirements, the changes in user requirements, system design, implementation or deployment environment. Developing a large software system requires a big development team. Team members have to interact with each other in order to find a solution to the problem and implement that solution. Interacting and working together introduce new challenges to software development. All these challenges increase the cost, effort and time needed to deliver the software system to the customer.

With the increase in size and complexity of software systems, the challenges become more critical. In order to overcome them and develop high quality, cost effective and reliable systems, some development approaches have been proposed. Two of these approaches are component based software engineering (CBSE) and model driven development (MDD). Those are two different approaches but both of them attempt to produce software system with high quality, low cost and on time with a focus on reusability. CBSE depends on combining pre-built software components to develop new software systems. In MDD, software engineers try to achieve these goals by representing the system as models. These models are generated and transformed to the final software system source code (semi) automatically.

The aim of this work is proposing a software development framework that combines MDD and CBSE approaches for merging the advantages and features of these approaches to facilitate software development. The framework applies MDD

principles in CBSE lifecycle to generate models for the components. These models are transformed to intermediate models until it reaches to the final source code.

The outline of this thesis is as follows: Chapter 1 provides introduction to the study. Chapter 2 presents background information about CBSE approach, MDD approach and WEB applications development. The related work about using CBSE or MDD to develop WEB applications and information about merging CBSE with MDD is found in chapter 3. Chapter 4 proposes a component based model driven software development framework that merges the two approaches. The lifecycle and workflow of the framework are presented as well. Chapter 5 represents a case study to use the proposed framework. Chapter 6 contains evaluation of the study and conclusions. The remainder sections from this chapter provide an introduction about the two approaches.

1.1. Component Based Software Engineering

CBSE is a software development approach that is proposes constructing software systems from existing software components instead of developing systems from scratch and reinventing the wheel [4]. The goal of CBSE is reducing cost and producing high quality and more reliable systems. Jerry Gao and H. Tsao summarize the differences between component based software development and traditional software development as follows: [1]

1- While in traditional software, reusability is difficult and it is possible for individual classes only, In CBS, reusability is easier and it can be applied for component or component architecture.
2- CBS can be upgraded easily while upgrading traditional software may require modifying the whole system.

3- CBSE has activities like component selection, customization and composition, which do not exist in traditional software engineering (SE) process.

4- If the source code of component is not available, this may be more expensive when the component needs to be maintained.

5- Interoperability is a major issue in CBS because different components are combined together.

In the next chapter, we will review the component based approach, define the software component and state the differences with object oriented approach. The advantages of component based approaches will be discussed as well. After that, we will show some of CBD models and some component software standards.

Using CBSE approach to develop a new system is justifiable if the cost of reusing pre-build components is less than the cost of developing new components. To determine that, software reuse assessment should be conducted before starting a new software system development.

1.2. Model Driven Development

MDD is a software development approach which aims to face the challenges of software development process through representing the essential aspects of the required system as models and generating the final source code from these models (semi)automatically. That will reduce the development effort, minimize the cost and improve the quality of the resulting software system. A model is an abstraction of a software system, which shows the main properties of the system and excludes unnecessary details, which leads to reducing the complexity of the system, enhancing the communication among development team members, better planning and automating generation of the source code. In MDD, different models are generated and each of these models represents the system in different level of abstractions from different perspectives. This means that, what is an important aspect at a point in time of software development life cycle cannot be important at another point. Model can be textual, graphical or a combination of both.

The process of representing a system as a set of models to abstract the essential aspects is called modeling and the means of modeling process are the modeling languages. Any modeling language should consist of three elements that are the concrete syntax, abstract syntax and semantic. The concrete syntax determines how the modeling elements are illustrated when the language is used. The abstract syntax

contains a set of modeling concepts and rules that specify well-formed expressions of these concepts. The abstract syntax defines how the models elements look to the compiler. The last element is the semantics which is an explanation to the abstract syntax [5].

The core activity of MDD is the model transformation. Model transformation is the process of transforming a model to another form. The resulting form of model transformation can be a new model with different level of abstraction so we can call the transformation process as **model to model transformation**, or the result can be source code in which the model transformation is called **code generation**. The model transformations in MDD are formal and take place according to the rules of a modeling transformation language where the main objective is to automate model transformations [14].

In the next chapter, more background information is given about MDD.

1.3. Web Applications

Tim Berners-Lee created the first web site to enable researchers to share their documents and information at the European Laboratory for Particle Physics (CERN) in 1991[29]. Since that time, the number of web sites increased from one to millions of web sites.

Web application is an evolution of Web sites. It is a Web site that allows the user to affect the data in the server [29].Web applications were built usually according to the programmer's intuition, using informal approaches. Nevertheless, today the web applications became very complex. For that reason, software engineering approaches should be followed to build reliable web applications [26].In this thesis, Web application is built using CBS and MDD approaches.

CHAPTER 2

BACKGROUND

This thesis proposes a framework to merge component based approach with model driven approach to develop Web applications. Component based software approach, model driven approach and Web application development will be discussed in this chapter to provide the required background information to achieve the aim of this thesis.

2.1. Component Based Approach

As mentioned before, large size and complex software systems are challenges to develop high quality, cost effective and reliable systems. Component based approach has been introduced as a solution for software system development problems by using reusable software components to build new software system. When traditional approaches are used, any modification in one module of system may affect the entire system. By using CBSE, the modification will be easier because it impacts the components that are related with modification request [15].

CBSE depends on the availability of software components. It needs software components to be available in well-organized component repositories, which facilitate searching and finding the required components. Therefore, CBSE includes two development processes. The first one is component development that deals with developing reusable software components and storing them in repositories in such a way that makes them searchable and reachable. The second development in CBSE is software system development, which is concerned with constructing the system from

existing pre-developed components. In the following sections, some topics related to CBSE will be explained.

2.1.1. Software Component

Szyperski defines the software component, as "*a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third-parties*"[2].

A software element, which is defined as "*a sequence of abstract program statements that describe computations to be performed by a machine*." By Councill & Heineman [3], can be considered as a software component if it is possible to deploy it independently and we can compose it with other software elements without modifying it regardless of the programming language [3].

A component should have the following characteristics

1. **Independent**: It means that the component has the ability to be deployed independently without the need of other components. If the component needs external services, that should be explained in interface specification explicitly [16].

2. **Composable**: It means that a component can interact with other components. That interaction has to happen through defined interfaces [16]. Councill & Heineman defined a component interface as "an abstraction of the behavior of a component that consists of a subset of the interactions of that component together with a set of constraints describing when they may occur."[3]

3. **Deployable**: A component should have the ability to operate on some platform as standalone and does not need to be compiled before the deployment [16]. A deployed component is ready to provide services.[66]

4. **Standardized**: A component should be developed according to a standardized

component model. Therefore, the interfaces, meta-data, composition, deployment and documentation of the component should conform to that model [65].

5. **Documented:** A component should have a good and complete documentation to facilitate potential reuse by giving a full description about component problem, syntax and interfaces to help the future users to understand it and know if it conforms to their requirements [16].

Component based software development can be considered as a step of object oriented development toward reusability, flexibility and adaptability but component differs from object class in important characteristics. The first difference between the software component and object class is that, a component has interfaces that enable it to interact with other components. Object class does not have interfaces [4][16].

The second difference is that, a component can be considered as a system or subsystem; it can be deployed and installed directly. A class should be compiled into an application program. Besides, a component should follow the standards of some component model while object class can be implemented in anyway. [4][65]

2.1.2. Advantages of Component Based Software Engineering

Component based development has many advantages. Some of these advantages are:

1- **Reusability**: components are independent entities that can be plugged into another application to achieve software reuse [1].

2- **Reduce time, effort and cost**: using already existing components to build systems will reduce cost and time. Wayne C. Lim made a comparison between two systems, one of them was developed using reusable components and the other without reusable components. The time needed to develop the first system was 42% less than the second system [63].

3- **Facilitate upgrading**: In component based software system, it is easy to replace one component by a new component without changing the final

system functions or modifying other components in the system [6].

4- **Improve the reliability**: CBSD depends on constructing software system from pre-tested reliable components, which helps to build more reliable systems [1].

5- **Improve the quality of software system**: The newly developed software elements need a lot of testing and maintenance. However, even with this testing and maintenance, many bugs may appear when we integrate these new elements into the application. So, the quality of the system will be highly affected. Henry and Faller showed that the software system quality improved by 35% when reusable components are used [7].

2.1.3. Component Software Standards

The main objective of CBSE is to develop reliable and cost effective software system by composing existing reusable components. To enable composition between independently developed components, we need standards to explain composition and interaction mechanisms. The standards should define standards for component implementation, naming, interoperability, customization, composition, evolution, and deployment [65]. The following are some standards that were developed to support CBSE:

1- **CORBA**: Common Object Request Broker Architecture (CORBA) was developed by the Object Management Group to be platform for developing component based software.

2- **COM**: Component Object Model (COM) was developed by Microsoft and it contains two elements, which are COM interfaces and mechanisms to register and pass messages between interfaces.

3- **JavaBeans**: Developed by Sun Microsystems. JavaBeans are portable, platform-independent components written in Java.

2.1.4. Additional Activities in CBSE Processes

When existing components are used in CBSE, the effort needed for designing and implementation is reduced. However, additional processes are added to development process. These processes are component assessment process and adaptation process. CBSE is beneficial if the effort needed for these processes is less than the reduced effort in design and implementation processes, otherwise it is better to develop the system from scratch or using another software engineering approach.

2.1.4.1. Component Assessment

Component assessment includes searching for candidate components that may satisfy desired functionalities, selecting the most suitable components from these candidates and verifying their functional properties [17]. The verification should be conducted for the component in isolation and in combination with other components.

One of the critical problems in CBSE is searching and finding components that might be suitable for our purpose. With the growth in number of repositories, the problem becomes worse and worse. For searching activities, there are some techniques for searching components. These techniques can be summarized as follows:

1. Keyword Based Technique: This technique depends on the keywords that user defines in a query to be searched in a repository. The limitation of the technique is that it needs an expert user to define proper keywords in order to achieve successful results; otherwise, the results may be irrelevant [18].

2. Signature Matching Technique: This technique tries to determine the components with signatures matching a query. Component signature describes its structure and type information [19].

3. Classification Based Search Technique: This technique imposes that the components are classified according to pre defined classification schema in the repository. The user should have knowledge about classification schema to be able to find the components, which he/she needs. There are several types of classifications like enumerated classification and faceted classification [18].

2.1.4.2. Adaptation

When a component is developed as a reusable component, there is a contradiction between extending its functionality and making it reusable because reusability requires keeping the functionality of the component simple and well defined. This contradiction lead to the component not fitting the requirements exactly and what is known as component mismatches.

Another reason for component mismatches is architectural mismatches, which are caused by different assumptions about behavior of other components. To solve component mismatch problem, CBSE needs a new process, which is called the adaptation. Beckeret al. (2006) defines software component adaptation as "the sequence of the steps required to bridge a component mismatch."[20]

There are several well-known adaptation techniques that are used to solve component mismatches. Some of them are given below:

1- Wrapper: This adaptation technique makes a new interface to component and encapsulates it. That is achieved through adding a glue code that work on adding a property or extending or restricting the original interface.[17]

2- Parameterized Interface: This interface adds the ability of changing component properties by specifying parameters of original interface. These parameters can be a number of input data, memory allocation or similar parameters.[17]

3- Adapter: This technique does not modify or hide the component properties like the previous techniques. It modifies component interface to conform to another component interface by adding a glue code which is called the adapter.[20]

2.1.5. The Component Based Development Process

Component based development process can be divided into two processes and these processes follow the phases of software life cycle, which are requirements analysis, design, implementation, testing and maintenance [8].

The two processes are:

1. The component based system development process. It focuses on selecting, evaluating and testing the appropriate components. There is not much effort required for designing and implementation.

2. The component development process. This process focuses on developing new components from scratch. Figure A.1 in Appendix A shows the overall process for developing component based software system.

There are many proposed CBSD process models like Somerville's model [9]. Somerville proposed a sequential model that contains six phases. These phases are, outlining the requirements phases that includes outlining the user requirements briefly. The second phase is trying to find components that coincide with the requirements. The next step is modifying the requirements to comply with components, this phase is negotiate requirements, after that the system will be designed, then searching for components again and at last integrating the component to construct the final system.

In this model, Somerville supposes that all found components do not need any adaptation to satisfy the requirements, he depends on modifying the requirements to conform to components [9] (appendix A figure A.2). There are other CBSD process models that have component adaption phase to modify the component to conform to the requirements.

In the following, some of CBSD process models are discussed.

2.1.5.1. V Development Model

This model is proposed by Crnkovic [10] who adapted the V model in traditional software, He separated the life cycle to two processes: system development and component development each of them follows the V model but he added new phases to system process model which are select, adapt and test the components[10] (appendix A figure A.3)

2.1.5.2. Y Development Model

Capretz proposed the Y mode [11]. This model considers instability and change so it

allows iteration in all phases. It is uses domain engineering in component development process then the component is archived. After that, components me selected from archive according to a framework. Y model applies V model in system life cycle but in the component development life cycle it does not apply V model [11](appendix A figure A.4).

2.1.5.3. W Development Model

In this model life cycle consists two V models, one for component development and the other for system development. Component selection, adaptation, and deployment phases represent the link between the two V models. The purpose of using V model with component life cycle is to enable component verification and validation before store in a repository. [12]. (appendix A figure A.5)

2.1.5.4. X Development Model

Gill and Tomar proposed this model. It includes four subcycles, one for component based software development and three for component development. They consider three cases for component based development, which are development for reuse, development with modification and development without modification.

Two main components of X Model are Testable Component Repository (TCR) and Reusable Component Repository (RCR) which is used to store the reusable and testable components[13] (appendix A figure A.6)

2.1.6. A Comparison among CBSD Models

The following table (Table 1) summarizes a comparison among CBSD Models which are described above besides our proposed model. It compares among them regarding separation between component development and system development, adaptation, using domain engineering and verification and validation.

Table1. Comparison among CBSD Models

	Model V	Model Y	Model W	Model X
Does the model separate between component development and system development activities explicitly?	Yes	Yes	Yes	Yes
Does the model Include an adaptation phase to modify the components?	Yes	Yes	Yes	Yes
Is the component verified and validated before it is stored in repository?	Yes	No	Yes	Yes
Does the model use domain engineering?	No	Yes	Yes	Yes
Does the model embed another software approach to increase the productivity?	No	No	No	No
Does the framework merge the different approaches to reduce the cost?	No	No	No	No
Does the framework consider the potential reuse of components before build in them as reusable?	No	No	No	No

2.2. Model Driven Development

Model driven development is another approach in software development that was introduced to increase productivity and decrease the cost and effort. As mentioned before, models are the essential aspects of MDD, which are abstractions of the system. These models are transformed semi(automatically) to generate the source code.

Using models in software development is not a new thing and it became more popular after the Unified Modeling Language (UML). Usually software engineers use models during different software development process phases to facilitate explanation and understanding but using models in development process does not mean that is model driven development. The models in this case are just documentation. Abstraction and automation are the important concepts. The system

is defined as a model on a higher level of abstraction then this model is transformed into intermediate models until it reaches to the final source code (semi)automatically. Models, metamodels, modeling, modeling language and model transformation are the main concepts of MDD. These concepts are discussed in the following sections besides the advantages of MDD and life cycle.

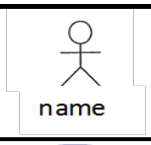
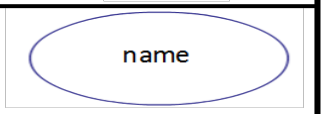
2.2.1. Modeling and Modeling language

Models are created for many purposes. They enhance the understanding of the problem, boundaries, design etc. of a system. Another purpose of models is reducing the complexity of the system by excluding unimportant aspects and focus on important ones. The process of creating models is modeling "*Modeling is the process of gaining a deeper understanding of a system through imitation. Models imitate the system and reflect properties of the system*." This is Lee is definition for modeling [21].

To represent a system as models we need a formal means to construct these models, called the modeling language. A modeling language is an artificial language that uses diagrams, rules, symbols to express the system [5]. Any modeling language consists of three parts: concrete syntax which is responsible for the look of the modeling elements for the user, abstract syntax which contains the textual expressions of the language and semantics which explains the abstract syntax. The model is defined using the modeling language and the relation between them is called *conforms to* [31].

The following figures show a modeling language for the use case diagram example. The use case diagram should have a number of actors, use cases and the connection between an actor and a use case. Each actor must have ID and name. The use case has ID and name too. The connection must have the actor ID and the use case ID which connects them. Figure2.1 (a) shows the concrete syntax, (b) shows the abstract syntax of the language and (c) shows use case diagram example by using our modeling language.

a) Concrete syntax

actor	
usecase	
conn	

b) Abstract syntax

<diagram> : : = <actor><usecase><conn>

<actor> : := "actor[<actorID>", "<name>"]"

<usecase> : := "usecase[" <usecaseID>", "<name>"]"

<conn> : := "conn[" <actorID> ", "< usecaseID>"]"

<actorID> : := <string>

<usecaseID> : := <string>

<name> : := <string>

c) Use case diagram using our modeling language example

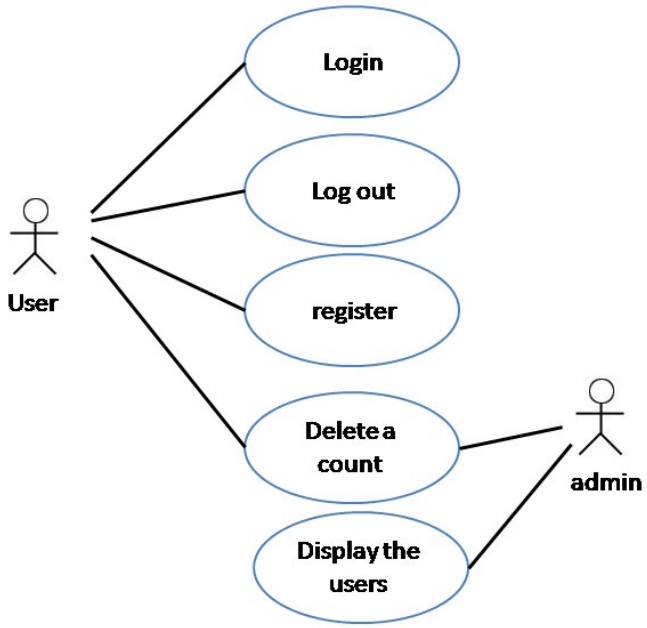


Figure 2.1. Use case diagram modeling language example

2.2.2. Metamodeling and Metamodeling languages

A metamodel is a model which is created to describe a modeling language. The process of creating this model is metamodeling [22]. In MDD, a metamodel is used to explain the abstract syntax of a modeling language. Metamodel is useful for software development for many reasons such as[42]:

1. Metamodel enables to define and handle models correctly.
2. It will be possible to use automated algorithms to check if the model matchs the rules and constraints.
3. Metamodel helps automating model transformation.
4. Manage changes in model implemention.

A metamodeling language is a modeling language and it, just like other modeling languages, contains abstract syntax and concrete syntax. The relation between

metamodel and metamodeling language is conforms to [31]. A metamodel itself can be described by another metamodel. In this case the new metamodel is called meta-metamodel. There are several well known metamodeling languages like MetaGME, Meta-Object Facility (MOF) and Eclipse Ecore meta-metamodel [5].Figure2.2 represents a metamodel for our use case modeling language example.

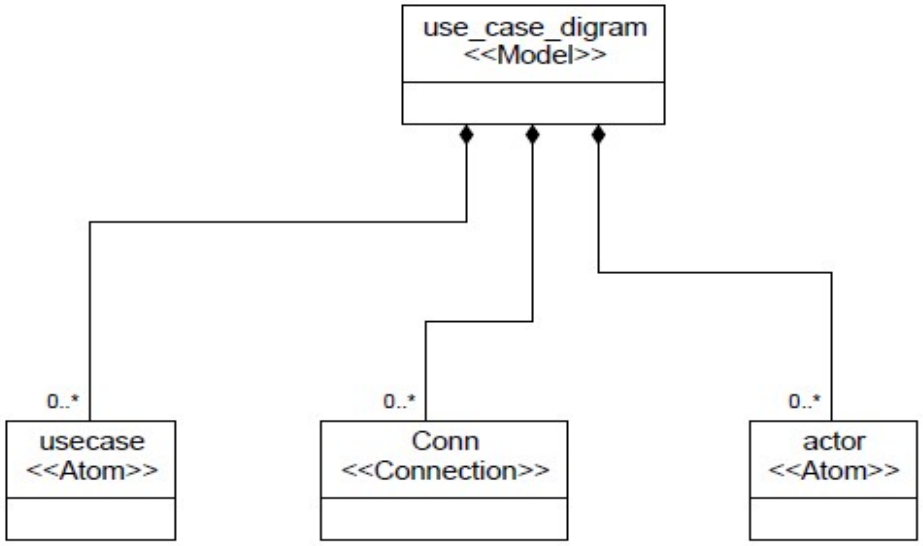


Figure 2.2. Metamodel for our use case modeling language example.

2.2.3. Model Types in MDD

During the MDD process, different models are generated. Each one of these models represents the system from different perspectives focusing on specific concerns to keep the model as simple as possible. Depending on the perspectives and their concerns, the models can be classified into three types: [40][41][5]

- A computation independent model (CIM): This model represents the system from a conceptual perspective related with requirements, environment and domain of the system.

- A platform independent model (PIM): This model is a formal model that describes the functionality and processing of the system regardless the platform. A platform means a set of technologies and it could be generic like object- oriented or specific like Java platform. PIM represents the system from specification and design perspective and it is constructed according to a system specification language like Petri Nets. Each CIM can be transformed to more than one PIM. Figure2.3 shows the model types in MDD.

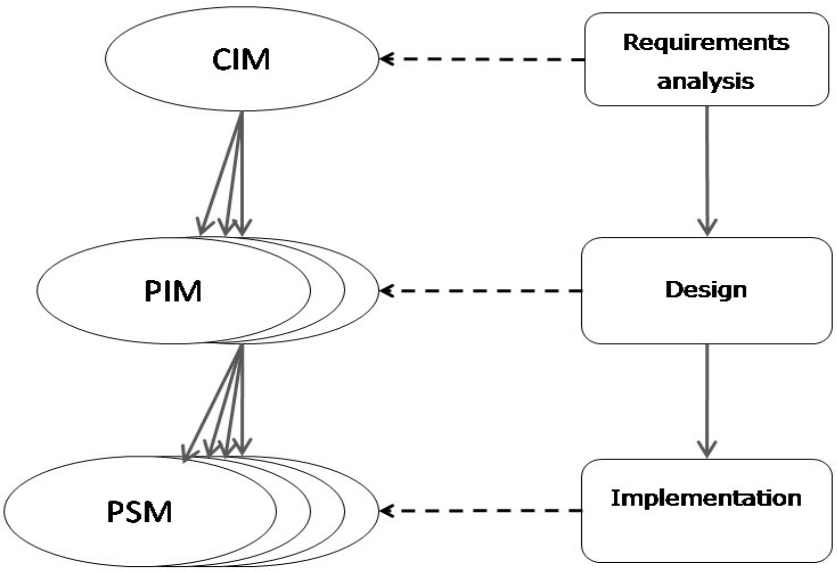


Figure 2.3. Models types in MDD

- A platform specific model (PSM): After the functionality of the system is defined in PIM, this model has to be converted to platform specific model (PSM) which describes the system design for one or more specific platforms. PSM is the final model and it will be converted to a source code to get the desired system.

2.2.4. Model Transformation

Model transformations are the key aspects to successful MDD. Model transformation is the process of converting a model with higher level of abstraction, combined with other information to another model until it reaches to the final executable source code of a system. A model transformation has input which is a source model and converts it to a target model or to a source code. Both source and target models should conform to their metamodels and they can have the same metamodel or different metamodels [23].Transformation applies a well defined formal rules to convert a model to another model or source code. These rules have to be written using a transformation language. There are several transformation language such as ATLAS transformation language (ATL) and GReAT (Graph Rewriting and Transformation language) [5]. Figure 2.4 shows the model transformation process.

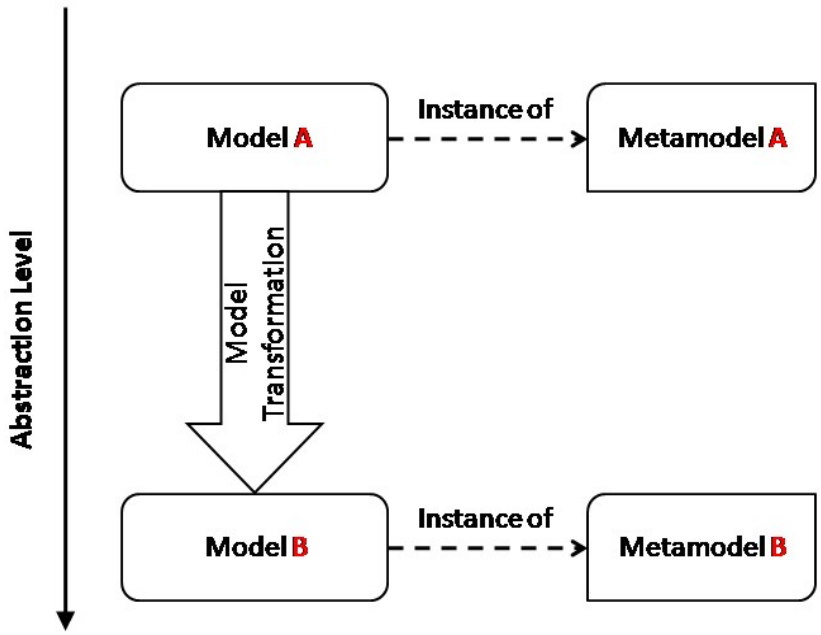


Figure 2.4. Model transformation

2.2.5. Advantages of MMD

MDD has several advantages such as: [41]

1- **Productivity**: MDD increases the productivity through the reuse and automation which reduce developing time because the executable code can be generated from models using few transformation steps.

2- **Enhance software quality**: using formally-defined modeling languages and model transformations improve the quality of resulted system and reduce the developers error such as typing errors in source code[46][64].

3- **Improve maintainability**: Maintenance can be performed on the models instead of the code which needs more effort and time. The other thing, which improves maintainability, is the separation of concerns.

4- **Reusability**: The models can be reused to regenerate source code automatically for different context. Modeling languages, transformations and architectures are reusable artifacts [64].

5- **Improved manageability of complexity**: Using abstraction at different levels to represent the system leads to improved manageability of complexity.

6- **Portability**: The same models can be used to generate code for different platforms using different transformations.

7- **Communication**: Facilitate the communication between the developers and other team members because the model is easier to understand.

2.3. Web Application development

A web application is a software system which can be accessed using the web browser on internet, intranet or extranet. Because of the huge increase in the number of internet users and the advance in technology, web applications become more important and are used in education, trade, health and many other fields. Web application enable users to access the desired information at anytime from anywhere.

Using web applications can also save money and time and facilitate the interactivity among users.

At first, the web applications were built depending on intuition, art, urgency and informality. These approaches lead to easy work environment, rapid decision making and aesthetic form and function but these approaches lead to problems most of the time [26] especially with the increment in complexity of web application development. These reasons impose adopting software engineering approaches to develop a reliable, high quality, cost effective and on time web application.

2.3.1. Web Development Life Cycle

Web application life cycle follows the phases of the general phases of software development, which are requirements analysis, design, implementation, testing and maintenance. Some methodologies are proposed to develop web application grounded on the software system life cycle phases [26].Figure 2.5 presents the phases of Web application life cycle.

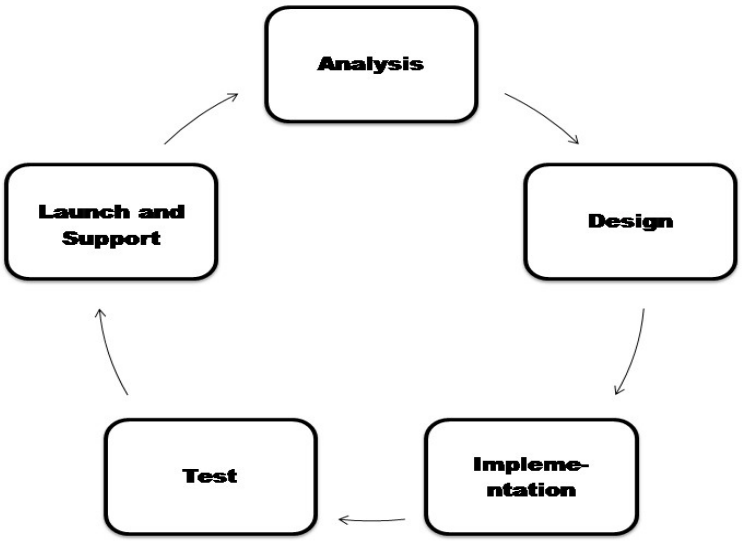


Figure 2.5. Web Development Lifecycle

Web applications have several differences over the other software systems such as [25]: High reliability, require continuous maintenance and short development life cycle.

CHAPTER 3

RELATED WORK

3.1. Component Based Approach with Web Applications

Today, Web applications are used in many fields and the importance of these applications is increased every day. With the increase in usage and importance of the web application, their complexity is increasing too. The traditional software development approach cannot manage this complexity for this reason; several new software approaches are submitted as solution. Many studies introduced component based approach as a solution to manage the complexity in web applications.

Min, Zhen and Hai-qiang [47] presented a framework (CBTOWADM) to simplify developing and testing a web application by dividing the system into subsystems which were called components. A component could be a web page, database, web service and so on.

Li and Chusho [48] introduced a framework and a tool for building a small or medium sized web application quickly that can be used by end user. Components represented backend functionalities such as searching in database.

Rana, Morshed and Synnes [49] proposed a component based approach to build social application and they called it social component framework for SatinII App development environment. The environment enables end users to create their application without knowledge of programming. Social components implement the main functionalities of social media such as social data visualization, social data collection, text communication and so on.

Zhang, Yang and Liu [50] also presented a framework for developing a small and

median size web application by low IT expertise. The framework had a repository of components which can be used to build a web application. These components can be customized by user. They also presented a platform to develop component based web application.

In [51], finance and crediting web applications were developed using component framework which allows the component's creation and integration. The authors compared between two projects one of them was developed using the framework and the other one was not. They found that using the component framework reduces development time, human effort and finance resources.

A component based architecture called BiTutor was introduced in [52]. BiTutor was developed using XML and Java. It is an educational open system constructed by composing a set of models related with the different aspects of the system.

3.2. Model Driven Approach and Web Applications

Model driven development is one of approaches which is submitted as a solution to web applications complexity in many studies. Fraternali and Paolini [53] described a model to develop web applications based on model driven approach and they presented a tool environment to create web application based upon model driven development. This tool called Autoweb System which is used to generate web application from the conceptual design.

The WebSA development process was proposed in [54] which is based on the MDD process. The output of each phase was models that had different abstraction levels of the system and transforming one model into another takes place using model transformations. The process divided the system specification into two viewpoints, functional and architectural and then integrated them.

Because web user interfaces are frequently renewing, Jinkui Hou [55] introduced a user interface modeling approach which depends on model driven development. The approach also achieved the reuse of interface design patterns.

Hou in [56] presented another MDD approach todevelopment of Web application

system. At the begining of the development process, platform independent models were described and then these models are transformed according to model transformations rules and metamodels.

In [60] Object-Oriented Web Solutions (OOWS) which is one of Web engineering methods combined with MDD to develop a web application. OOWS is a method that uses conceptual modelling to facilitate the development process of web applications.

3.3. Component based software development with Model Driven Development

In many studies, component based development was combined with model driven development to introduce methodologies to develop distributed applications, embedded systems, Web applications and other types of systems. A model driven development for component based adaptive distributed applications was presented in [57]. The authors separated the functional aspect and the communication aspect and then integrated composite model.

Agustinand Barco [62] proposed a model-driven development framework to develop web application by composing web components and generated web applications by transforming UML model. The authors in [61] introduced a model driven framework to develop a component based adaptive web presentation. The components in their work were the elements, which construct a web presentation such as media types or documents.

In [59] an approach to develop component based systems using model-driven development was introduced for developing embedded systems. Besides Model-to-model transformations of MDD, model to metamodel transformations (M2MM) was presented where the metamodel in a level could be generated from the model at the previous level.

Two refinement relations on components were proposed in [58] for component based model driven development to address refinement relations among models. These refinements are trace-based refinement and state-based refinement relation.

CHAPTER 4

FRAMEWORK

This chapter proposes a component based model driven software development framework. The framework represents an integrated approach to build software systems using component based development and model driven development .The main purpose of combining the two approaches is to take advantages of them (MDD and CBD) so the resulting system will have higher quality and it will be more reliable and maintainable with less cost and effort. In this chapter, we present the proposed framework with its process model and workflow.

4.1. The Process Model

The process model of our proposed framework embeds the phases of model driven development in component based software lifecycle. This model separates into two main activities: component development and software system development. In this section, the phases of the model will be explained in detail.

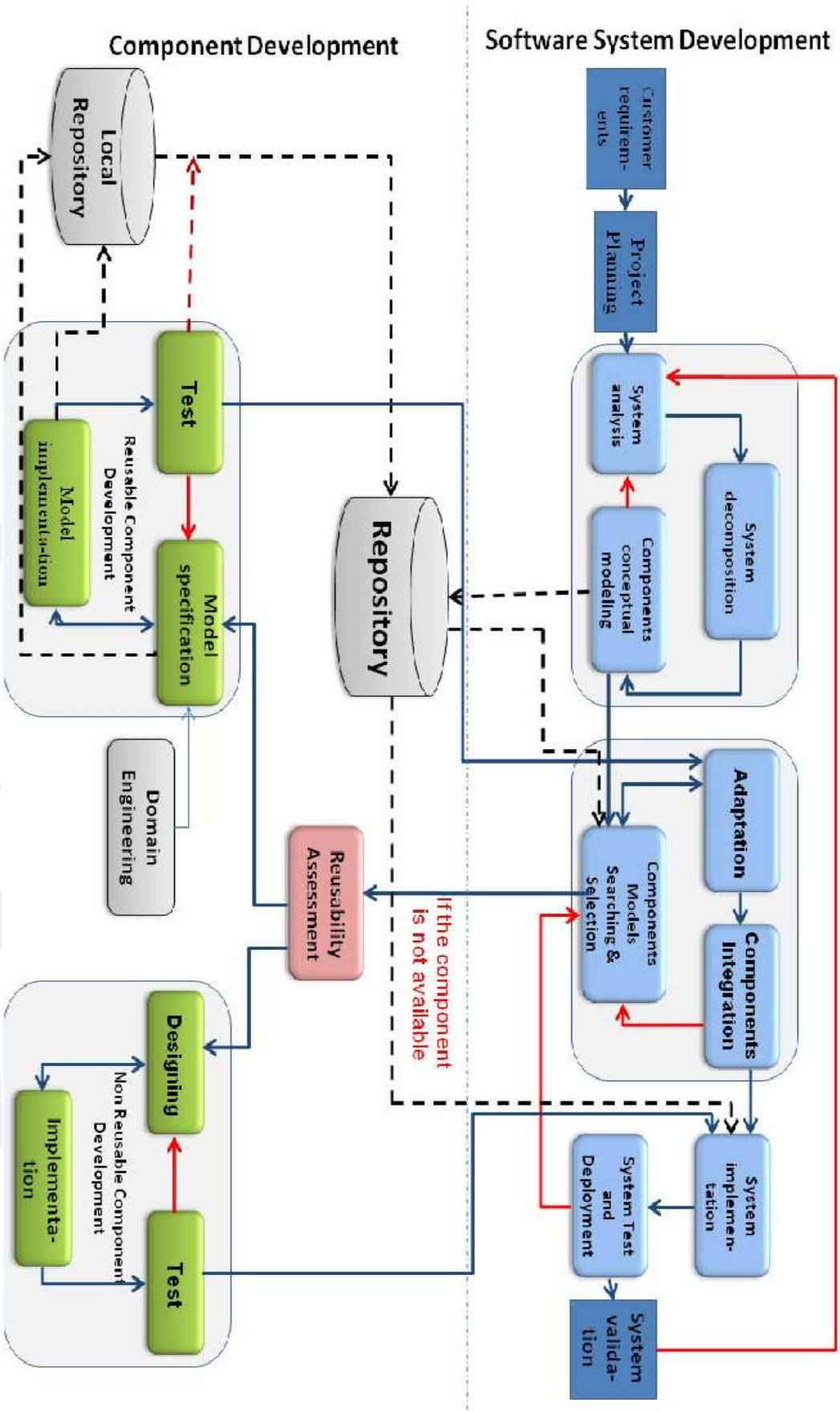


Figure 4.1. The proposed process model.

4.1.1. Software System Development

4.1.1.1. Project planning

Before starting system development, the system problem, purpose and the boundaries should be defined and the requirements of the system should be collected from the problem owner. The main purpose of this step is to understand the main characteristics of the application and prepare a project plan. The developers and the customer have to agree on this project plan to start system development.

4.1.1.2. System analysis

After the project plan is agreed, the developers should construct deep understandings of functional and non-functional requirements of the system and provide requirements document.

4.1.1.3. System decomposition

Component based approach focuses on dividing the system into logical or functional components which are connected to each other by well-defined interfaces. In our framework, after the system is analyzed and the essential characteristics are well defined, the next step in lifecycle is system decomposition in which general knowledge about the problem and the domain is improved, the system is decoupled and separation of concerns is made. The problem will be decomposed into sub problems, each one associated with one or more component. The outcome of this step is the decomposition model that describes the systems conceptual design and shows the components of the system and the relations among them. We can use a graphical notation for describing this model, which uses UML syntax to be able to use a UML tool.

4.1.1.4. Components conceptual modeling

When the components of the system are set, a high-level abstraction should be made for each component in the system in order to prepare a conceptual model for each component (CCM). This conceptual model will be prepared according to the

requirements, system boundaries and the conceptual modeling language. These models are non-executable models which will be translated to executable models.

4.1.1.5. Components Models Searching and Selection

This step starts with component model searching. The input will be the conceptual models of components. Identifying and selecting a subset of components models each of them satisfies the requirements and can construct the target system represent a difficult problem [1]. In order to solve this problem, many approaches have been suggested as using ideas of search based engineering [2].

4.1.1.6. Adaptation

In case the desired component models are found and selected from the repository, the next step will be the adaption. Adaptation means attempt to match the selected components with the requirements and modify its interfaces to be able to integrate with other components to construct the target system. Adapting software components that have different interaction behavior is a hard problem in component based software engineering [35]. There were some approaches proposed to solve this problem such as finite state grammars technique that is proposed by Yellin [22]. Other studies used algebras feature to handle adaptation problem [36][37]. Anyway, the components adaptation cost and effort must not be more than the cost and effort of building new component. If the adaptation activity fails, the component should be discarded and we should go back to selection step.

4.1.1.7. Components Integration

After all needed component models become available by selecting from repository or by building them from scratch, a composite model containing all selected CPIM of components with their relations should be constructed. This model will represent PIM for the system.

4.1.1.8. System Implementation

At the implementation phase, any missing parts needed to complete the system

should be implemented and the CPSM of selected CPIM will be put in the PSM of system. The CPSM models of component may have links to their source code at the implementation; these models are replaced by source code. Another possibility is the models have transformations rules and their source code is generated using them.

4.1.2.1 System Test and Deployment

The system should be tested before it reaches the user to ensure that the functions and graphics of the system meet the requirements and specification. After the system is tested, it can be released to the customer.

4.1.2. Component Development

4.1.2.2 Reusability Assessment

Before starting building a new component we should decide whether we want to create this component as a reusable or non-reusable component because developing reusable component is more costly. Because of that we need to estimate the number of potential reuse of this component to be able to estimate the reuse effectiveness which Sametinger defined as" the ratio of reuse benefits to reuse costs. It can be measured by the ratio of the difference between what the development of a new component would have cost to what it costs to reuse the component times the number of its reuse to the investments costs to acquire or develop the reusable component".[27]

4.1.2.3 Reusable Component Development

- **Model specification**

If the required component was not found in repository, it should be built. When we decide to build a new component we need past experience related with the domain of the component to have a better understanding and to make the component more reusable. This experience can be gained from domain engineering which "deals with collecting, organizing and storing past experience in building system or part of system in particular domain"[3]. Using this knowledge and the conceptual model, a component platform independent model can be introduced (CPIM) by transforming

the CCM into a formal model according to a system specification language such as Petri Nets. The specification model should describe the functionality, behavior and potential interaction of the component mathematically to be realized in the implementation model.

- **Model implementation**

At this stage, specification model is developed into the implementation model. The implementation tools, techniques and the platform are specified so the implementation model will be component platform specific model (CPSM). An essential concern of the model implementation is how develop a reusable component which can be adaptable in another system. The source code of component will be generated from CPSM.

For each CCM, It possible to define more than one CPIM and similarly for CPIM, more than one CPSM can be defined for each CPIM with the links between them or instead of defining CPSMs, we can define CPIM to CPSM transformations and during system implementation the CPSM will be generated by these transformations.

- **Test**

Component should be tested before deployment. Test stage is used for verifying and validating the component models and component source code. A test plan and test cases should be defined and run to get the results. The results will be analyzed to fix the bugs and defects. If the component passes the test successfully, then it will be integrated with the software system and stored in repository.

4.1.2.4 Non-Reusable Component Development

The phases of developing a non-reusable component are same as the phases of developing traditional software development because the number of potential reuse of this component is low as assessed in reusability assessment phase which will reduce the system cost. It possible to develop non-reusable component using the same phases of reusable component development but with one difference, which is that when we are implementing a non-reusable component, reusability is not one of

development concerns so the component has less cost and less effort. The resulting component will be integrated with the system and will not be stored in the repository.

4.2. The Outputs of the Life Cycle Phases

The outputs of the first phase in the life cycle include a requirements document that contains information about the desired application such as the problem, the customer's expectations and the boundaries of the system. This information represents a high level description of the system and does not contain any details about architecture or implementation. The other output of this phase is the project plan. A project plan is defined as " a formal, approved document used to guide both project execution and project control" by PMBOK GUIDE [39]. A project plan includes scope, cost, and schedule of the project as well as assumptions, constrains and decisions.

In system decomposition, the output should be a decomposition model that describes conceptual design and divides the system to components. A number of conceptual models of components that are suggested by decomposition model will be the outputs of components conceptual modeling phase. If the searching process succeeds during components searching and selecting in finding components in the repository that match the required conceptual models, the output of the phase will be the selected components otherwise the process will move to component development activities. These selected components are adapted in the adaptation phase. The adapted components represent the output of this phase. The output of the implementation phase is the source code of the system. The tested system is the output of the final phase. Table 2. represents the outputs of all phases of the process model.

Table 2. The life cycle phases outputs

The phase	The output
Project planning	Project plan
System analysis	Requirements document
System decomposition	Decomposition model
Conceptual modeling	Conceptual models of components
Components searching and selecting	Selected PIM of components
Adaptation	Adapted components
Components Integration	Integrated system model
System model implementation	PSM of system and Source code
System test and deployment	Final implemented system

The outputs of reusable components development are component platform independent model (CPIM) in model specification phase and component platform specific model (CPSM) and source code in model implementation phase. These models are stored in the local repository. After the test phase, these models are stored in the general repository. Table 3 shows the outputs of each stage in reusable components development.

Table 3. Reusable components development outputs

The phase	The output	After the phase	After Test
Model specification	Component platform independent model (CPIM).	Written to local repository	Written to general repository
Model implementation	Component platform specific model (CPSM) and source code.	Written to local repository	Written to general repository

4.3. Framework Workflow

The following figure shows the workflow for the proposed framework.

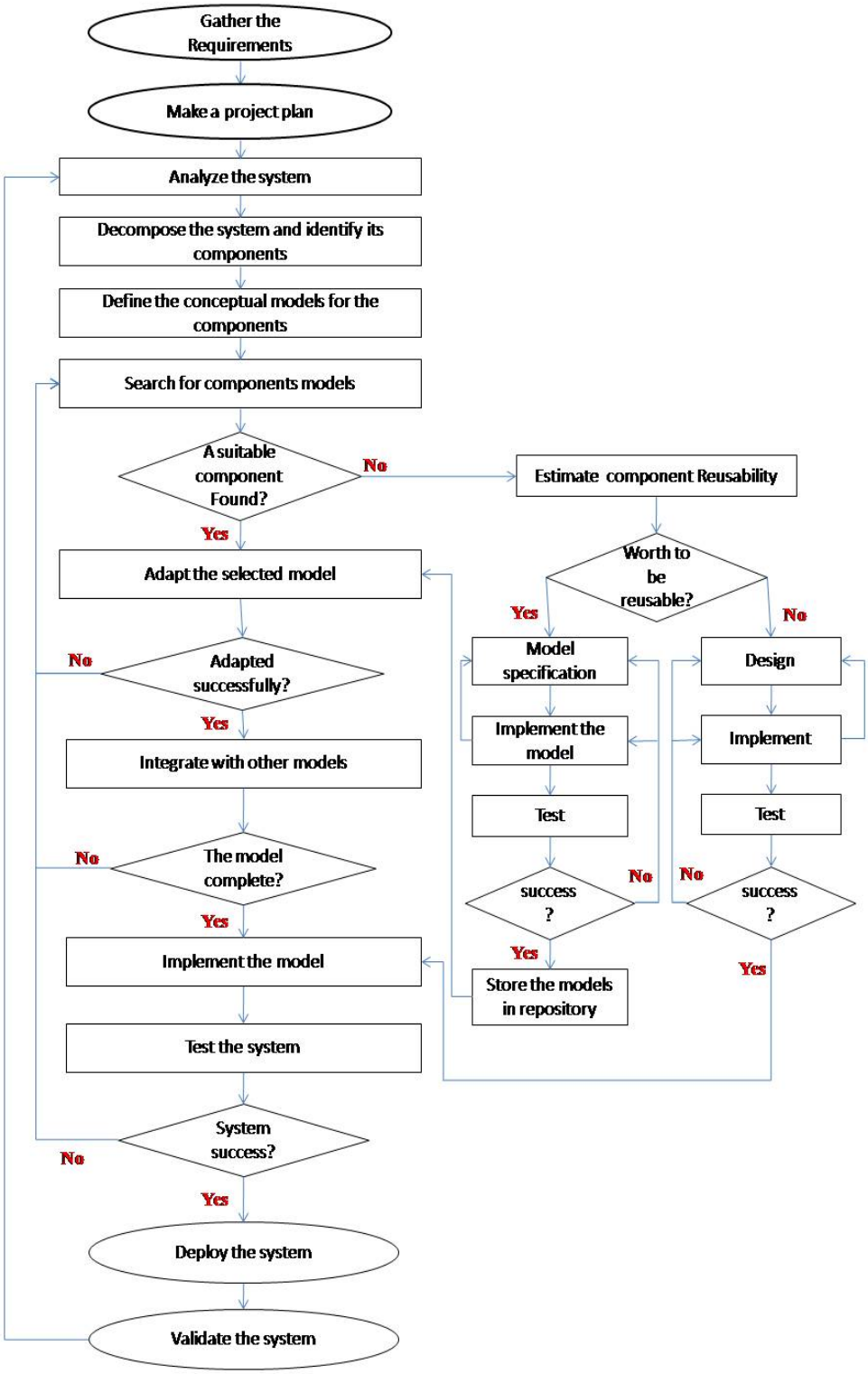


Figure 4.2. A workflow for the proposed framework

4.4. Metamodeling

Metamodeling is a modeling process to generate a model which defines the abstract syntax of a modeling language, which includes concepts, relationships and well-formedness rules [5][28]. The output of this process is a metamodel and the language to be used is a metamodeling language. A metamodel can be also used to explain the structure of another model without any explanation about the model's content.

In our proposed framework, there are a number of metamodels generated within the development life cycle. These metamodels are:

- Component conceptual metamodel(CC metamodel) : This metamodel is developed during the system development life cycle and contains a high level description of component, inputs, outputs, attributes and methods should be described in it besides all type of relations that can connect the component with other components regardless of the technical details.

- Component platform independent metamodel (CPI metamodel). These models represent the grammar of the component specification and depending on these metamodel PIMs, models will be generated.

- Component platform specific metamodel (CPS metamodel): The entities and relations in this metamodel correspond to the entities and relations in CPI metamodel and there may be additional entities or additional feature to original entities added after the domain knowledge is investigated. CPS metamodel describes these entities and relations for a specific platform implementation [30].

The generated PIMs will have more potential reuse and more impact if domain knowledge is added to conceptual model. However, that will make the model transformers more complex, time consuming and the probability of error will be high. Metamodels for source and destination models will help to make model transformers faster and easier. Figure4.3 shows the models and metamodels in the proposed reusable component development.

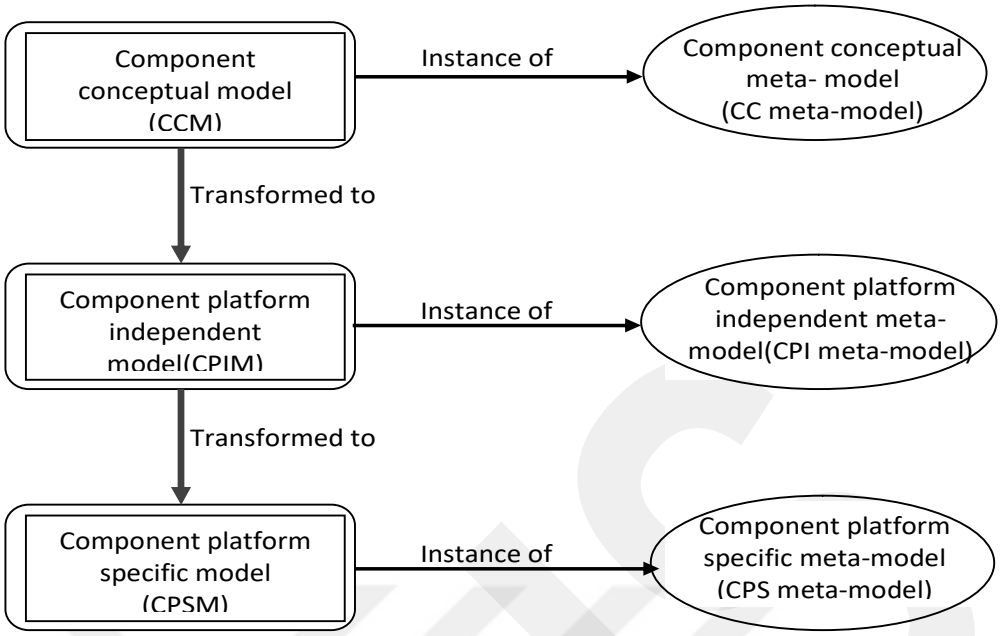


Figure 4.3. Metamodels and models in reusable component development

4.5. The Repository

One of the essential infrastructures needed for successful component based development approach is a good structured repository. The reusable components should be stored in such a way so they will be easily found and retrieved in the future. If the repository has a poor structure and the components in it have insufficient index, the performance of retrieval process may not be acceptable even if a good retrieval algorithm is used [32].

There are many methods to index the components in repository. These methods can be manual or automatic. Enumerated classification is one of the manual methods in which categories are organized in a hierarchical structure to form a tree. The components represent the leaves of this tree according to their categories. Hierarchical classification is easy to use but at the same time, it needs precise domain analysis to classify the component correctly, besides the user should be familiar with

the structure and the contents to be able to find the desired components easily. Finally, it is inflexible to change the index of the component.

Free text indexing is an automatic indexing method which creates index using the words in document and puts all the words, except high frequency words like “the" and "is", in term lists. When the user searches for a specific document, he should write keywords to be searched in the index term lists. This method is suitable for indexing the documents that contain natural language because it relies on these language rules but not for indexing software artifacts which use rules of grammar and many of these software artifacts do not have documentation [34].

In addition to the good structure, the repository needs retrieval interface that support finding the components effectively. This interface should deal with issues like ill-defined queries that are submitted by inexpert users or the users who do not know what exactly they are looking for, especially some studies show that novice developers depend on reusable components more than expert developer[33]. To overcome the retrieval issues, some tools have been proposed such as CodeFinder [34] and CodeBroker [35] which are trying to extract the information from ill-defined queries and find associated information to construct refinement queries.

When a repository is used to store models, it should have the capability to manage versions which means the user can revert to previous versions of models and can get the next versions. There are some studies suggest cloud repository to make search and retrieval process faster and easier. [38]

CHAPTER 5

CASE STUDY

5.1. Scope of the Project

This software system will be an Online Quiz System. The system will be designed to help learners to get a free quiz to assess themselves. It will be an online learning system that contains questions and answers database. The questions cover different fields and answers will have different types. It will support more than one language. The target users of the system are learners and specialists from different fields that means not all of them can use computers efficiently so the interfaces must be friendly and easy to use. Anyone can use the system to get a quiz and any one can register to add a question. It is free and public so the user does not have to be a member in a specific institute to be able to use it.

5.2. System Development

5.2.1. System analysis

In this system, there are three main actors who are Admin, quiz maker and learner. The quiz maker is the person who adds the questions and their answers to the database. The learner is the person who takes the quiz. The Admin is the person who checks the questions and the answers to ensure that these questions are proper academic questions.

5.2.1.1. Admin Functions

This part is supposed to contain admin functions which control the database .
Functions in this part are:

1. **Check question**: With this function, the admin can view the questions and check them.
2. **Delete question**: With this function, the admin can delete any question if it looks unsuitable or it is not academic question.
3. **Delete account**: With this one, the Admin can delete any account that adds unsuitable questions or is a fake account.
4. **Search question**: **Search question:** Admin can search questions.

5.2.1.2. Question Maker Functions

This part is related with making questions. The user should create an account to be able to add a question and he/she has to sign in his/her account every time he wants to add a new question or edit a question that he added before.

This part contains functions:

1. **Create account**: If the user wants to be able to add questions, he should create an account first. In this account, he will give information about his/her academic position or his/her previous experiments and an email address.
2. **Add question*:*** This function is responsible for adding the questions, with this one; the registered user can add questions.
3. **Delete question:** By using this one, the registered user can delete questions that he added.
4. **Search question:** The registered user can search the questions, which he added.

5.2.1.3. The Learner Functions

This part contains the functions, which are concerned with taking a quiz, and who are using this part the learners .They can take the quiz without registration.
These functions are:

1. **Taking a quiz.** The user who would like to take a quiz should use this function to determine his field and level.
2. **Checking answers checking*:*** This function is responsible for checking the answers and gives the final result. Figure 5.1 represents the use case diagram for online quiz system.

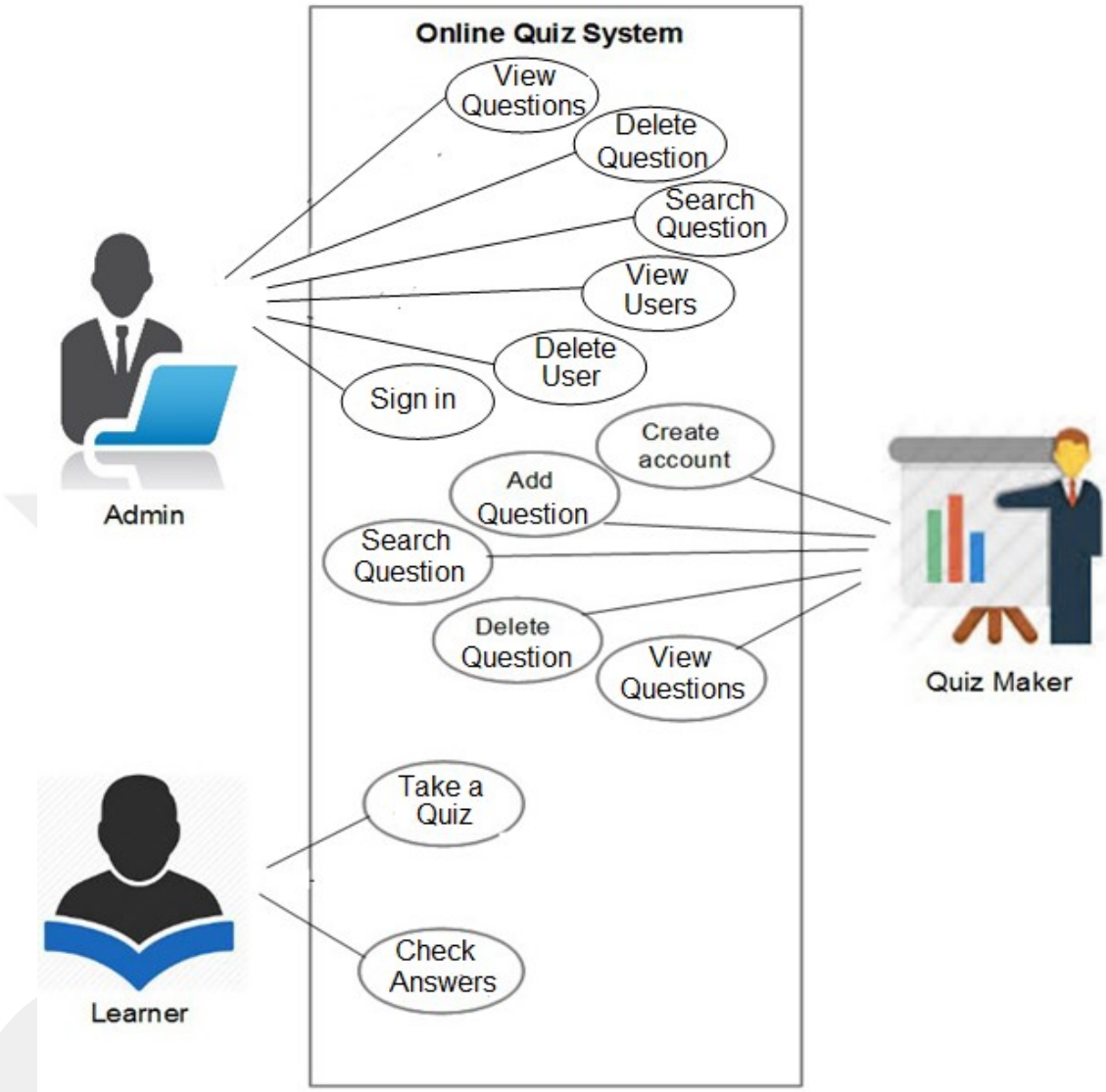


Figure 5.1.Use case diagram for online quiz system

5.2.2. System Decomposition

This system can be decomposed into five parts: authorizing, administrator, quiz maker, learner parts and database. Authorizing part contains sign in, sign up, logout, check. Administrator part contains view and delete users, view, delete and search questions and all these functions needed to connect to database. Quiz maker part contains add, view, delete and search questions. Quiz maker functions need to

connect to database as well. Learner part functions are for taking quiz and View result. According to that, the system can be expressed as shown in Figure 5.2

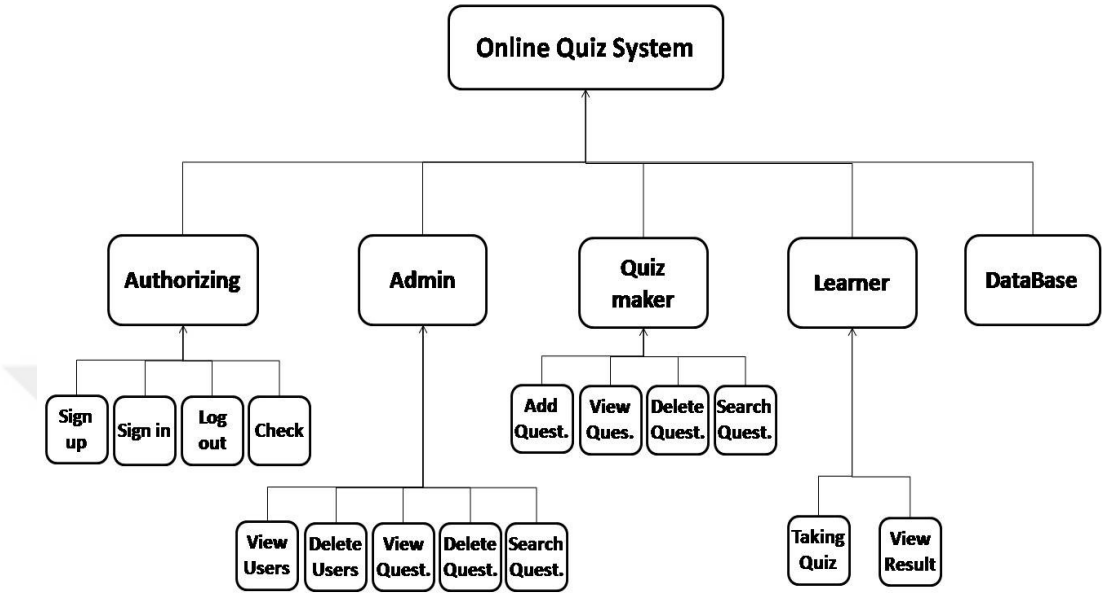


Figure 5.2.Online quiz system parts

There are some functions which are used in more than one part in the system like view, search and delete questions. View users and view questions can be considered as one function because both of them are about displaying records from a database. We can say the same thing for search and delete functions.

Moreover, delete function can be embedded in view function because usually the record is deleted after the user views it.

So the functions of the system can be compressed as following

- Authorizing: Sign in, Sign up, Logout, Check.

- Admin/Quiz maker: Search record, view record, delete record, and add record.

- Learner: Take a quiz and View result.

Figure 5.3 shows the functions in Online Quiz system.

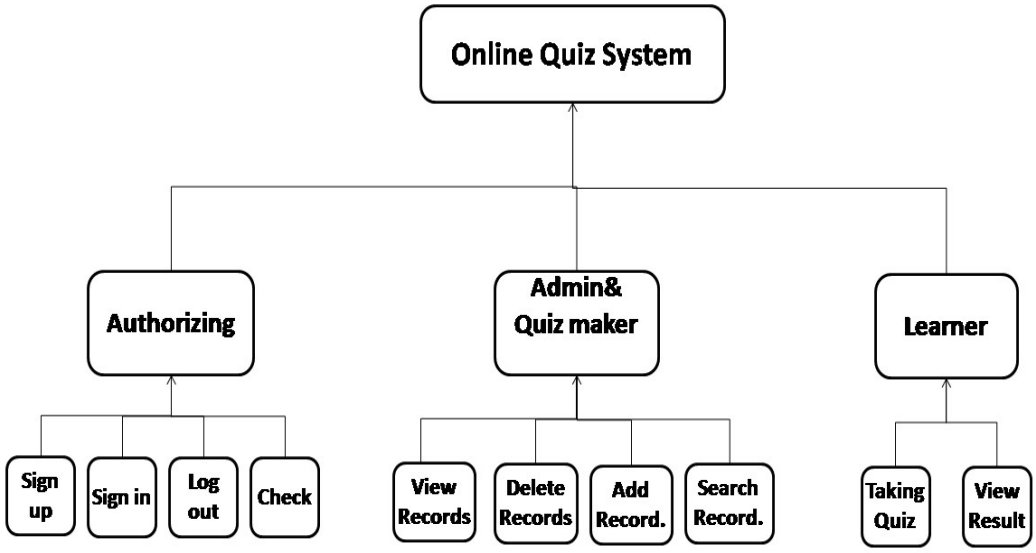
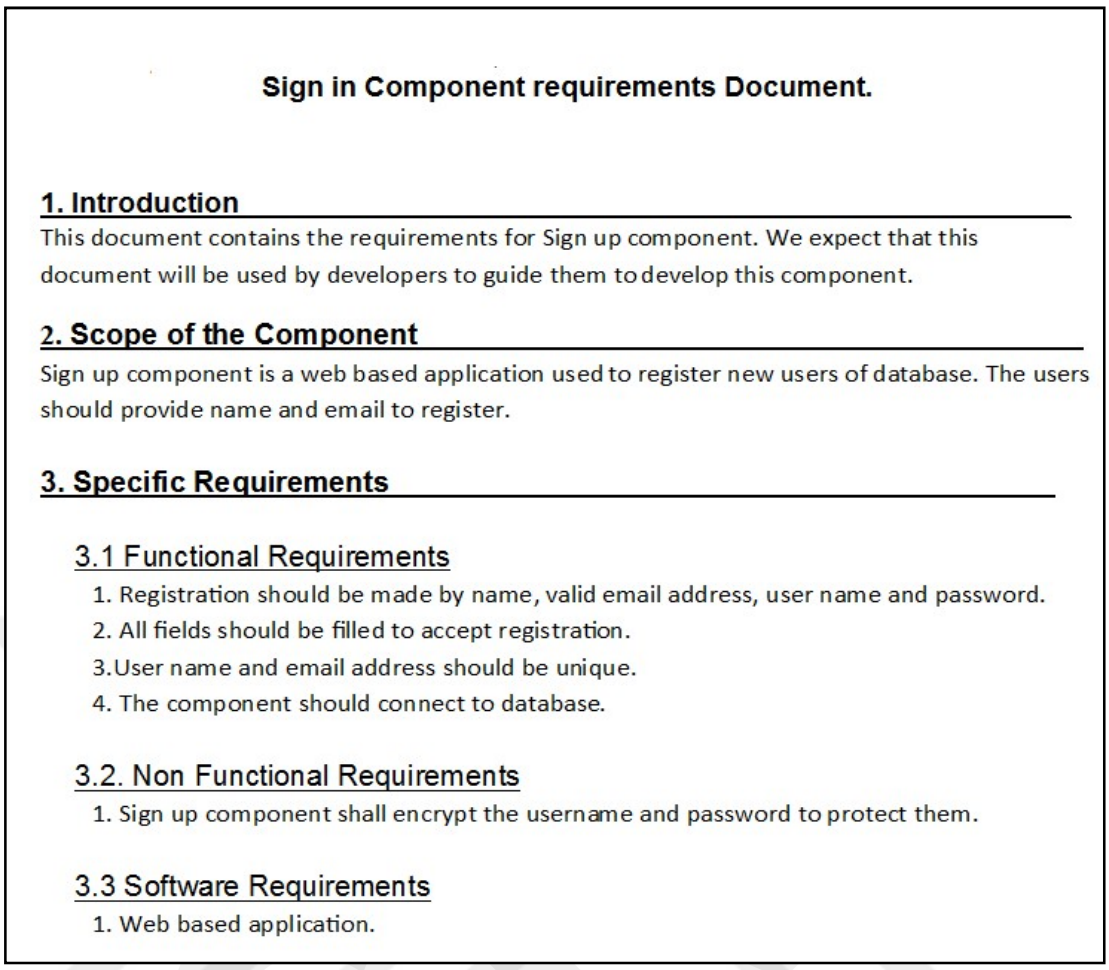


Figure 5.3.Functions in online quiz system

5.2.3. Components Conceptual Modeling

Conceptual model can be a graph, mathematical equation, text or other types. In this case study, component conceptual model is a requirement document for each component. Figure 5.4 shows the requirements document for sign up component as a sample of our conceptual models.



Sign in Component requirements Document.

1. Introduction

This document contains the requirements for Sign up component. We expect that this document will be used by developers to guide them to develop this component.

2. Scope of the Component

Sign up component is a web based application used to register new users of database. The users should provide name and email to register.

3. Specific Requirements

3.1 Functional Requirements

1. Registration should be made by name, valid email address, user name and password.
2. All fields should be filled to accept registration.
3. User name and email address should be unique.
4. The component should connect to database.

3.2. Non Functional Requirements

1. Sign up component shall encrypt the username and password to protect them.

3.3 Software Requirements

1. Web based application.

Figure 5.4. A conceptual model sample

5.2.4. Components Models Searching and Selecting.

Because this is the first, we do not have any stored components in the repository; therefore, all components will be constructed.

5.2.5. Reusability Assessment

Depending on the potential reuses and the difference in complexity between creating reusable component and non-reusable component, we can assess whether the desired component is worth to be a reusable component. For the components in Online Quiz System, sign in and sign up in authorizing part are assessed to be reusable components. In Admin/Quiz maker part, add record, view records and search record were assessed to reusable components because they can be used in any web

application that deals with a database. Other components like quiz maker page or submit for a quiz were assessed to be non-reusable components.

5.2.6. Components Development

5.2.6.1 Reusable Component Development

- **Model Specification**

  In this phase, the conceptual model of component transfers to component platform independent model. Each component may have inputs, outputs and parameters. The component can connect to other components or web pages through links, include or require connections. According to this description, we made a component metamodel in figure 5.5

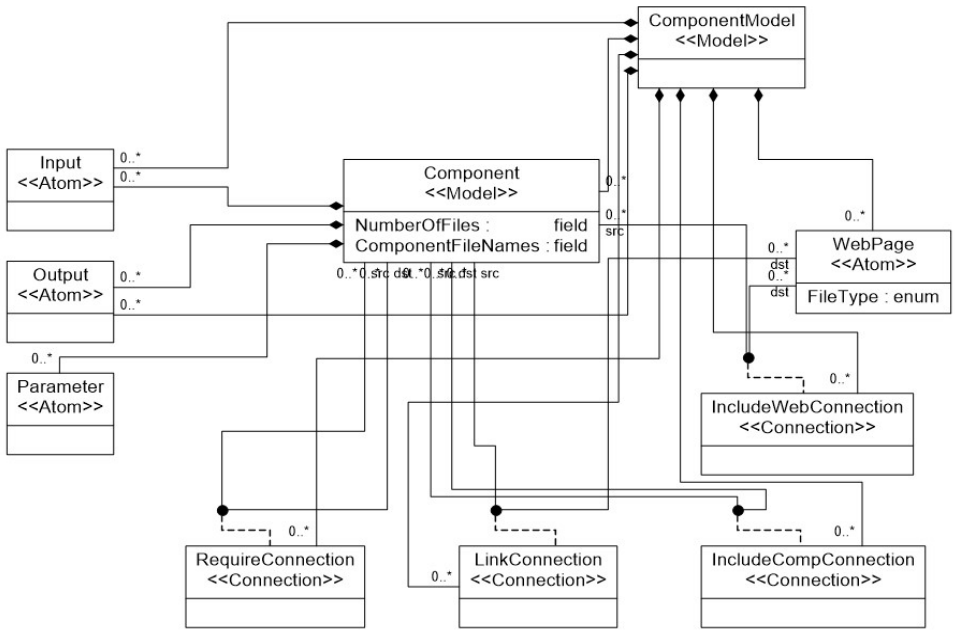


Figure 5.5. Component PIM metamodel

Depending on the CPIM metamodel in Figure 5.5 CPIM models for the reusable components were generated. The following figures show the CPIM for the components in Online quiz system.

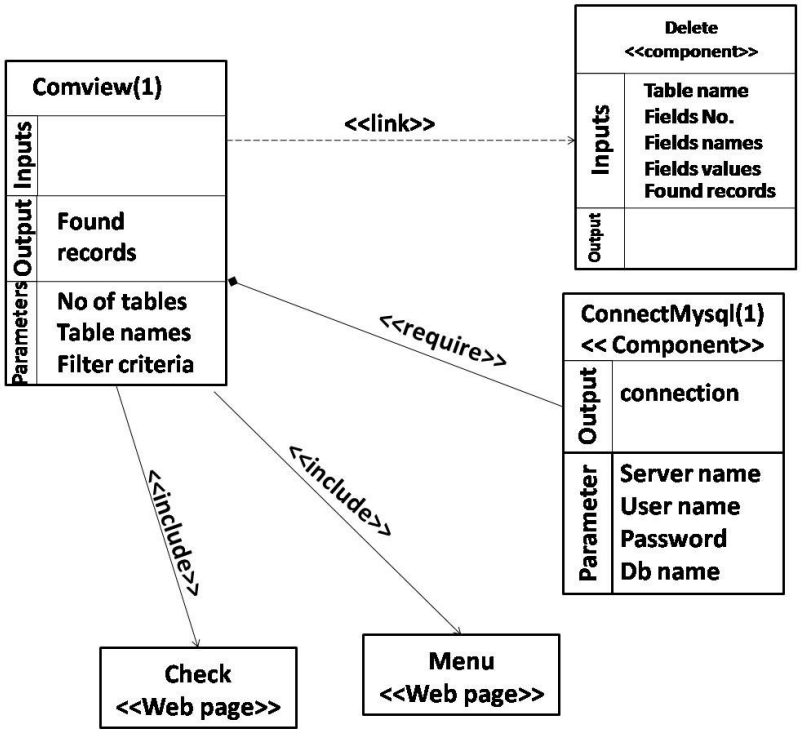


Figure 5.6. CPIM model for view component.

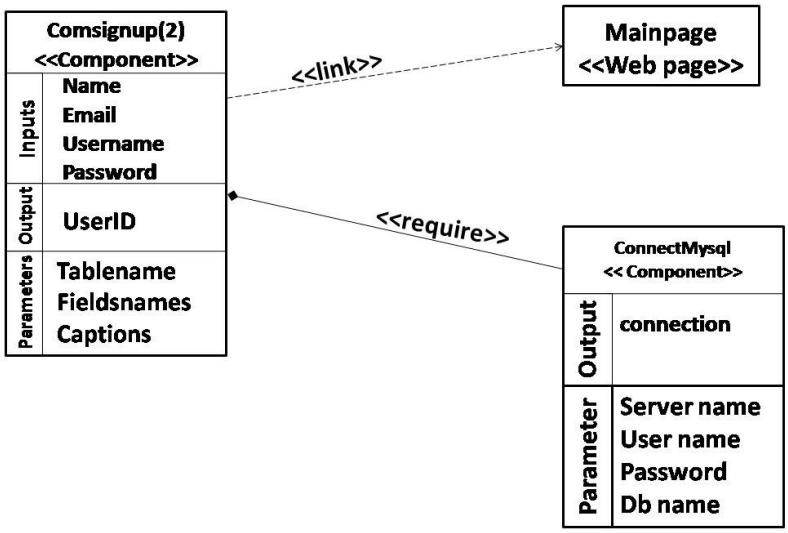


Figure 5.7. CPIM model for signup component.

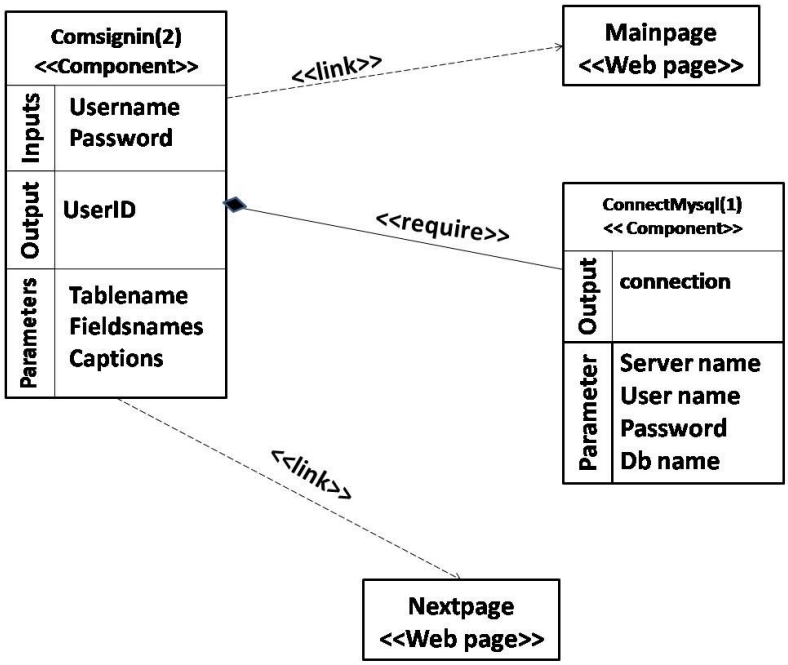


Figure 5.8. CPIM model for sign in component.

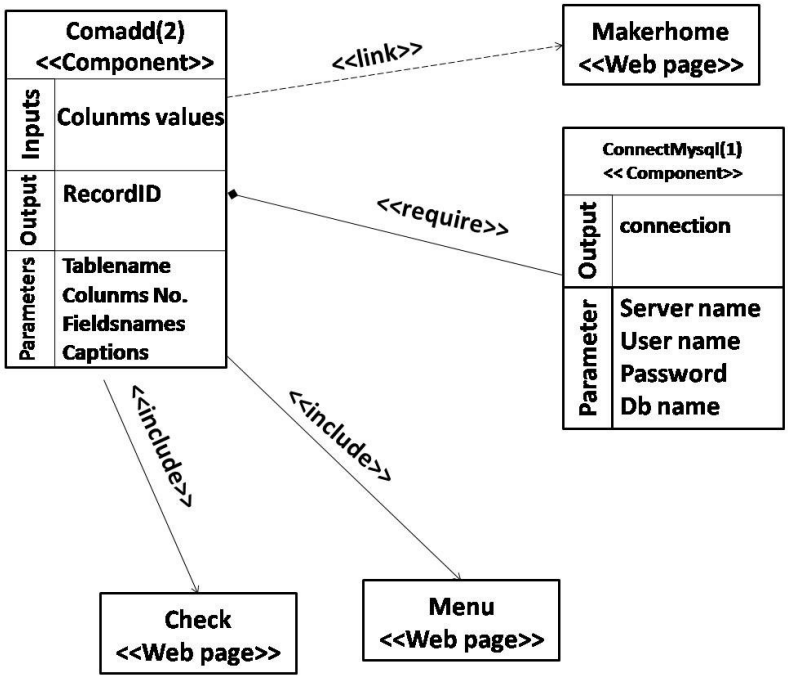


Figure 5.9. CPIM model For add component.

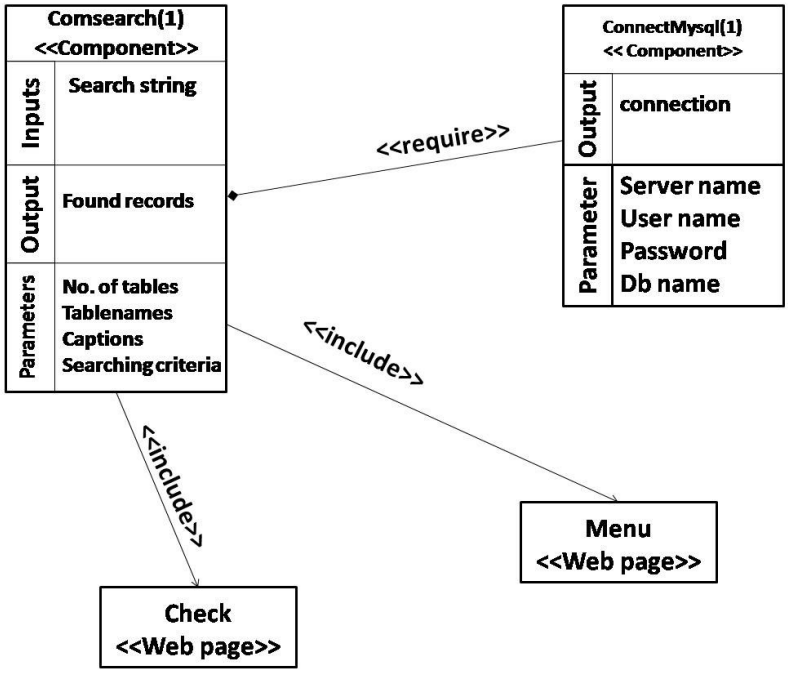


Figure 5.10. CPIM model search component.

- **Model Implementation**

  To implement the components, PHP and HTML were used because their performances are fast and they have big capabilities, moreover, PHP can interact with database languages like MySQL easily. In many cases PSM, is the code itself. Because there is no PHP specific model, in our case study the CPSMs are source code.

- **Test**

To test a component after implementation, we define the parameters required for it and run it using localhost. For example, for Add component we defined table name, column names and the captions we want to appear in the web page. After the test, we stored these components.

5.2.6.2 Non-Reusable Component Development

- **Design**

  Components like home page, admin page, quiz maker page and functions in learner part were assessed to be non-reusable components, therefore we followed the traditional approach. Figure5.11 represents the sequence diagram for learner part.

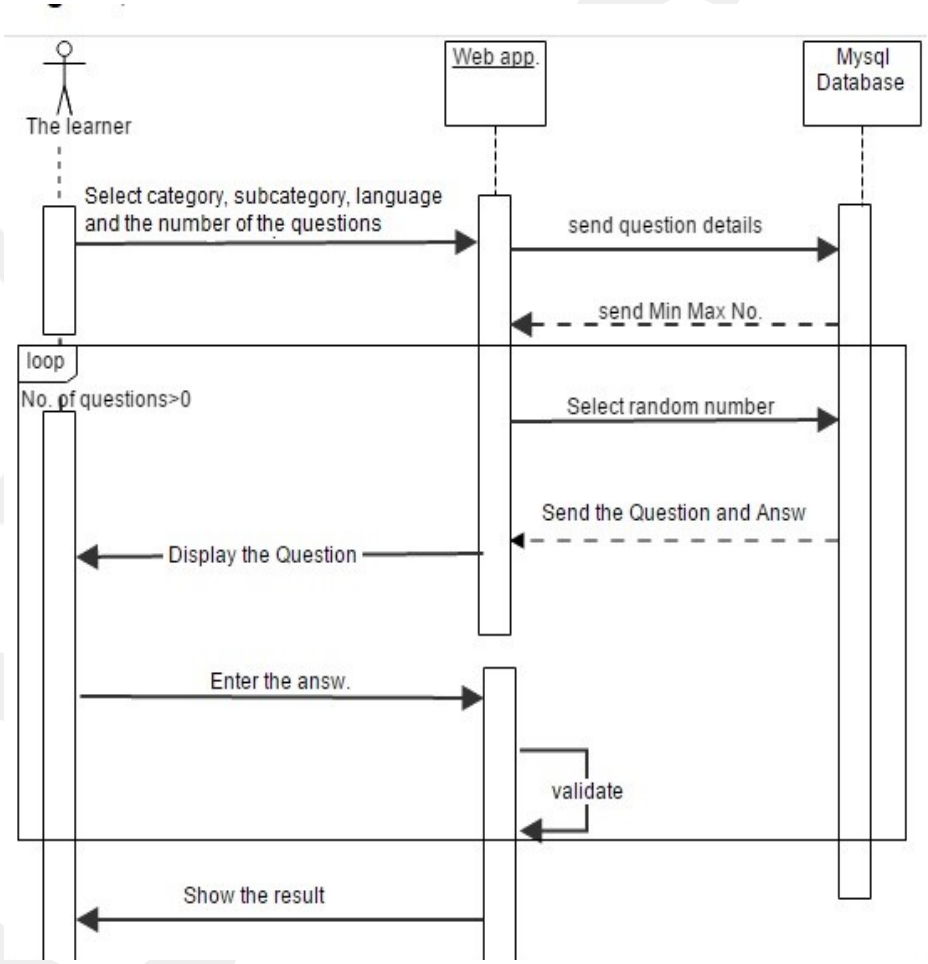


Figure 5.11. Sequence diagram for learner part.

The following figure represents the activity diagram for taking a quiz.

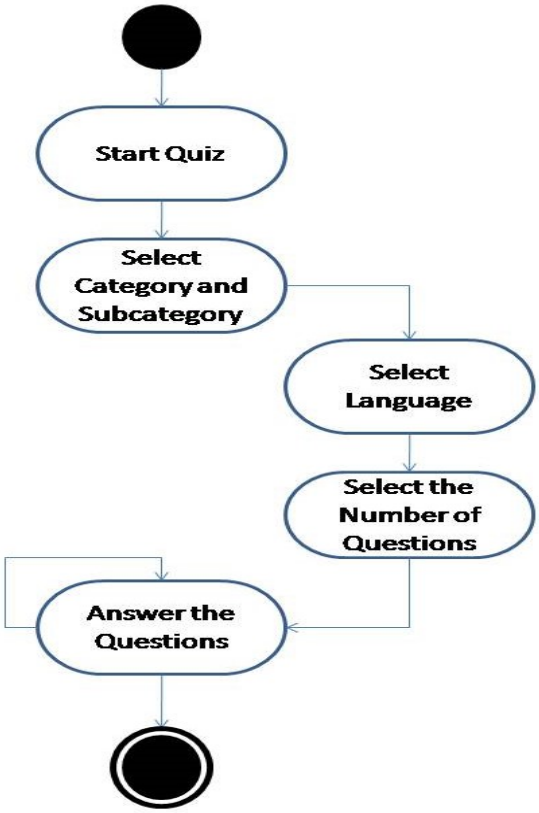


Figure 5.12. Taking quiz activity diagram

- **Implementation and Test**

  After a non-reusable component is designed, we implement it using PHP and HTML and run it using Apache server to test. Figure 5.13 is the screenshot for taking quiz page and figure 5.14 is the screenshot of quiz maker page. All of them were created with the traditional approach.

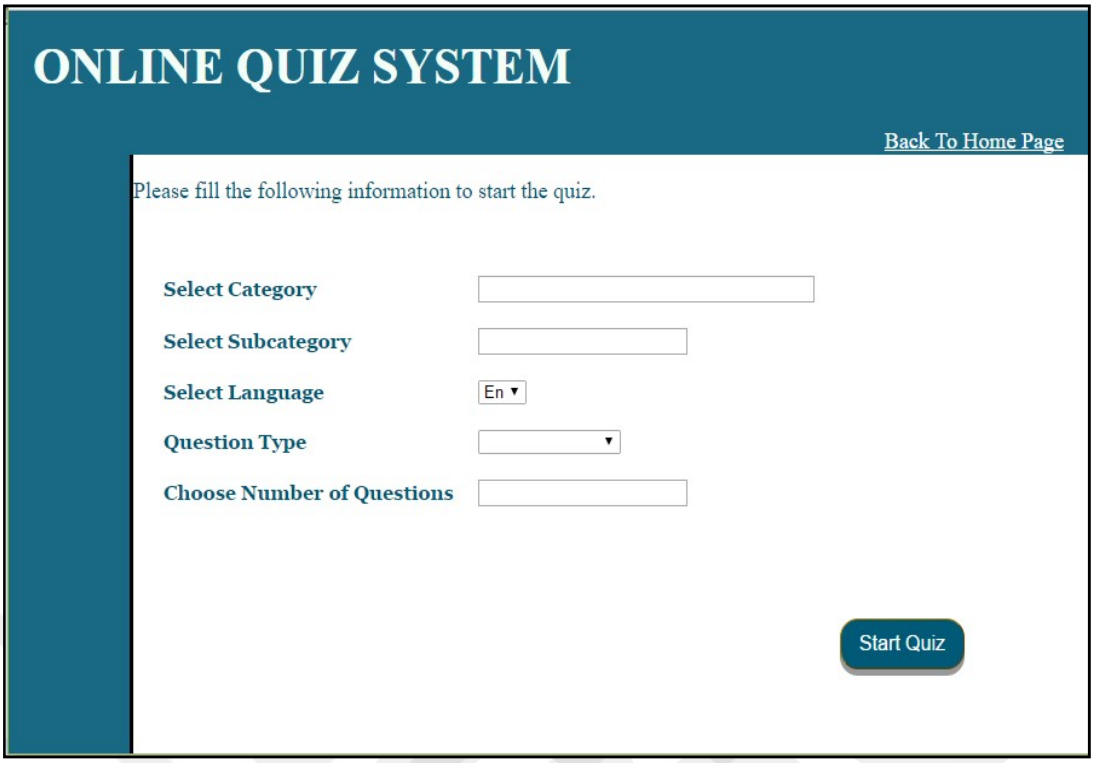


Figure 5.13. Screenshot for taking quiz page

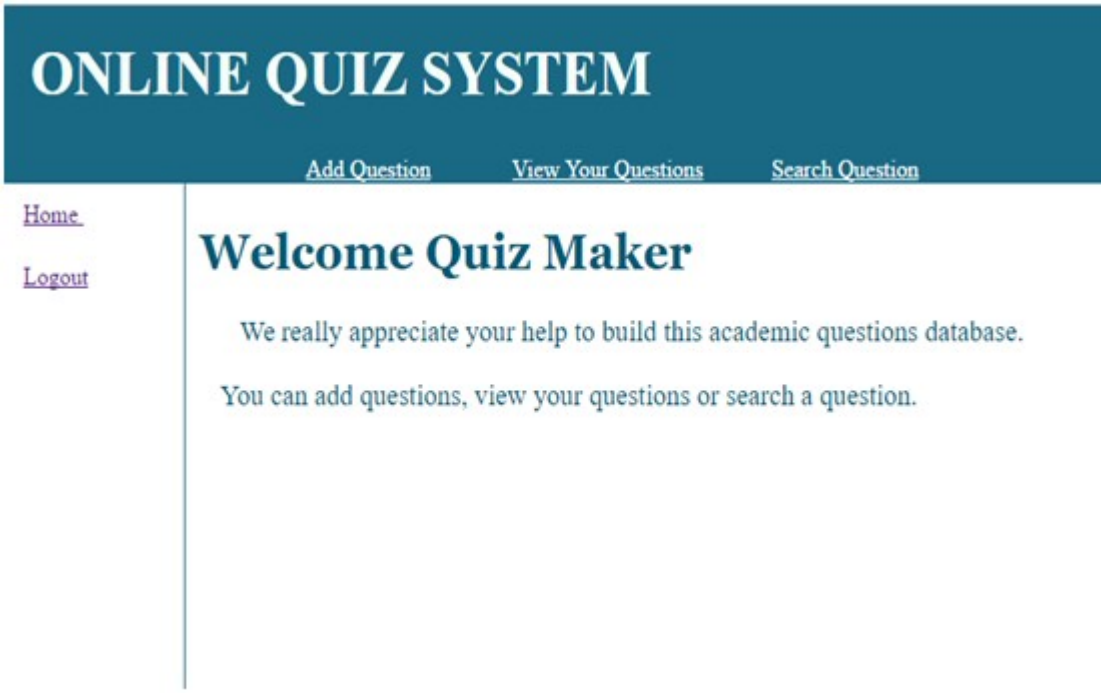


Figure 5.14. Screenshot of quiz maker page

5.2.7. Adaptation

After the components are constructed, we adapt the components by modifying the parameters file, which can be considered as component interface. The parameters can be the names of linked pages or components, number of inputs, table name, columns names or captions. We modify them to conform to the requirements. Some components are used more than once in system, such as view component, which was used in view questions function and view users function. Figure 5.15shows the parameters file for view users function while Figure 5.16 represents the parameters file for view questions function. Both of them are using the same component, which is view records, but by modifying parameters file, we can use it for different task.

```
<?php
require('connectmysql.php');
include('admin-menu.php');
require('check.php');

$Notb=1;              // There number of tables you want to display
$tbln=array();

$tbln[0]="usersq";    //the table name

$F="";  //If you have a filtering criteria write it here otherwise set $f to "" (empty);
?>
```

Figure 5.15. The parameters file for view users function

```
<?php
require('connectmysql.php');
include('menu.php');
include('check.php');
$Notb=3;                //Number of tables you want display them
$tbln=array();

$tbln[0]="textq1";       //tables names;
$tbln[1]="truefalsq1";
$tbln[2]="multiq1";
$F="Where makerid=".$_SESSION['mid'];  // If you have a filtering criteria write
?>                                     // it here otherwise set $f to "" (empty);
```

Figure 5.16. The parameters file for view questions function

5.2.8. Components Integration

All components models were integrated in one model to construct the integrated system model. Figure 5.17 represents the integrated system model.

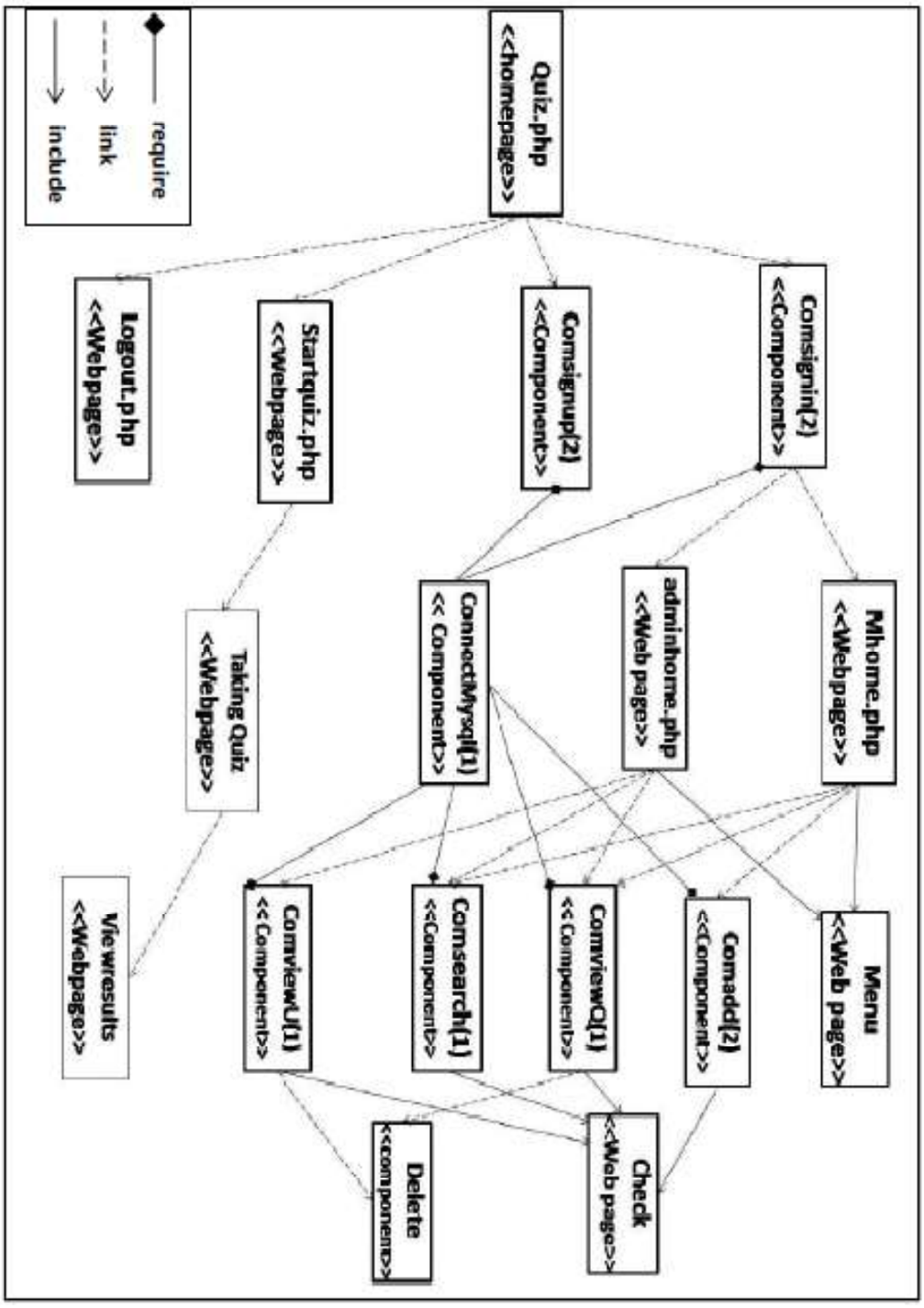


Figure 5.17. Integrated system model

5.2.9. System Implementation and Test

To implement the system, we put the code of reusable and non-reusable components to construct the final system and run it. Figure 5.18 shows home page screenshot.

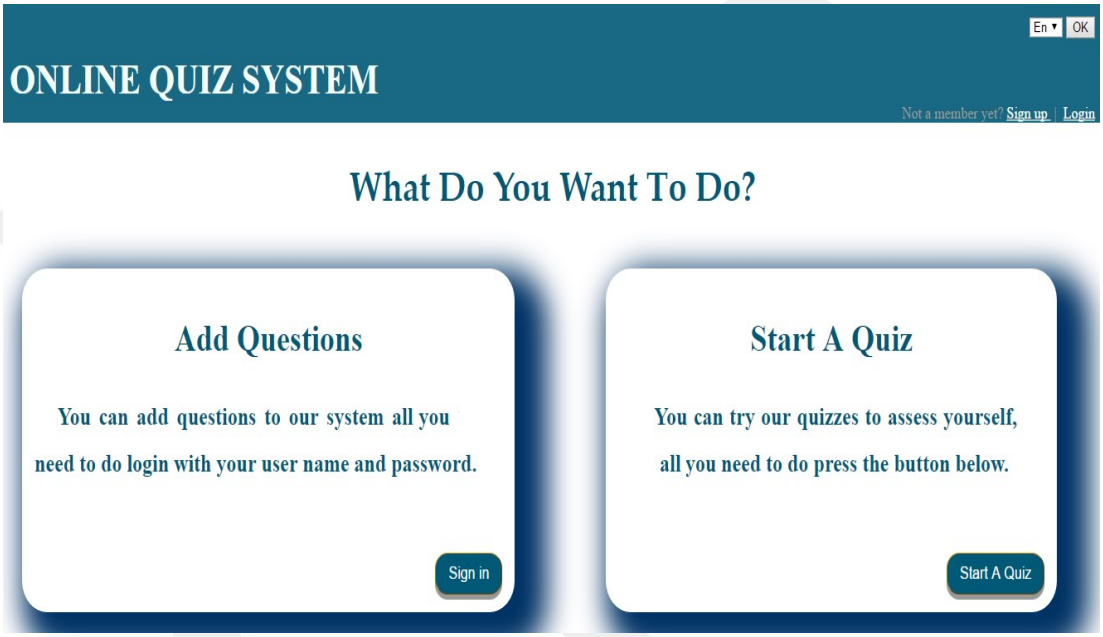


Figure 5.18. Screenshot for home page

CHAPTER 6

CONCLUSION

6.1. Conclusion

In this thesis, a component based model driven software development framework is proposed. At first, the main problems that are faced when developing software systems were discussed. The traditional approaches used for constructing software systems from scratch cannot deal with large size and complexity of software systems. The complexity and large size lead to challenges such as introducing high quality, cost effective and on time systems.

In order to overcome these challenges, some development approaches have been introduced in the earlier studies. A background about two of these approaches was presented. These two approaches are component based software engineering (CBSE) and model driven development (MDD). Both approaches aim to reduce cost and effort of developing software systems and improve quality, reliability and productivity. CBSE depends on constructing the system from reusable pre-built software components. MDD depends on representing the system as models.

This thesis presents a framework that combines CBSE and MDD as a solution to the problems of the development of software systems. Our proposed framework splits software lifecycle into two processes, system development process and component development. MDD is embedded in both developing processes.

Developing a reusable component is more costly than developing a non-reusable one. In order to reduce the cost of developing component based system, the framework merges component based approach and traditional approach. A component can be developed as a reusable component using component based approach or as a non-reusable component using traditional approach depending on reusability assessment phase result.

This framework was used to develop an online quiz system successfully. It is a web application which deals with mysql database. The database contains questions and answers to help learners to assess themselves. The system was decomposed into its essential parts. Sign up, sign in, add, view, search and delete functions were built as reusable components. Other parts in the system were built as non-reusable components. That was according to their potential reuse and reconstructing complexity.

6.2. Evaluation

We propose a framework for developing software systems. This framework attempts to increase the productivity by combining two software development approaches, which are component based approach and model driven approach. Developing a software component as reusable component costs more than developing it as non-reusable. If a component was developed to be reusable, and it is never used again that means we waste our resources. In our framework, we tried to overcome this problem by making reusability assessment before building the components. In that case, the resulted system can be more cost effective. The following table contains a comparison between component based models and our framework.

Table 4. Comparison among CBSD Models including our proposed model

	Model V	Model Y	Model W	Model X	Our proposed model
Does the model separate between component development and system development activities explicitly?	Yes	Yes	Yes	Yes	Yes
Does the model include an adaptation phase to modify the components?	Yes	Yes	Yes	Yes	Yes
Is the component verified and validated before it is stored in repository?	Yes	No	Yes	Yes	Yes
Does the model use domain engineering?	No	Yes	Yes	Yes	Yes

Table 4. (continued)

Does the model embed another software approach to increase the productivity?	No	No	No	No	Yes
Does the framework merge the different approaches to reduce the cost?	No	No	No	No	Yes
Does the framework consider the potential reuse of components before build them as reusable?	No	No	No	No	Yes

Table 4 shows that, the proposed framework, unlike the other models, embed MDD with CBSE to increase the productivity of the development process. Moreover, the framework reduces unnecessary cost of developing component based software system by merging CBSE with traditional approach. This unnecessary cost may come from developing a component as reusable component while it does not have potential reuses. The other models do not consider this issue while our framework manages it with reusability assessment phase which consider the potential reuse of components before build them as reusable.

We applied this framework to develop a web application. For future work, this framework can be applied to develop other types of software systems. Web application development can be split into two processes: graphical development and functional development to reduce the implementation. In the future, the framework can be more specific for web application by merging it with the two processes of web application development lifecycle.

REFERENCES

[1] GAO, J; TSAO, HJ; WU, Y. "Testing and Quality Assurance for Component-based Software". *Boston : Artech House, Inc*, 2003. ISBN: 9781580534802.

[2] SZYPERSKI, C." Component Software: Beyond Object-Oriented Programming. *Communications of the ACM*. 40, 10, Nov. 2, 1997. ISSN: 00010782.

[3] COUNCILL, B; HEINEMAN, G. "Definition of a software component and its elements". In *Component-based software engineering*, George T. Heineman and William T. Councill (Eds.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA 5-19. 2001. ISBN:0-201-70485-4.

[4] PATEL, A; et al. "A COMPARATIVE STUDY OF AGILE, COMPONENT-BASED, ASPECT-ORIENTED AND MASHUP SOFTWARE DEVELOPMENT METHODS", *Tehnicki vjesnik / Technical Gazette*. 19, 1, 175-189p, Jan. 2012.

[5] ÇETINKAYA, D. "Model Driven Development of Simulation Models". Delft University of Technology, Netherlands. Nov., 2013

[6] QURESHI, MJ; HAYAT, SA. "The artifacts of component-based development", *Science International-Lahore*. 19,3,187-192p, Feb. 11, 2012.

[7] HENRY, E; FALLER, B. "Large-Scale Industrial Reuse to Reduce Cost and Cycle Time". *IEEE Software*. 12, 5, 47-53p, Jan. 1, 1995.

[8] SENTILLES, S., " Towards Efficient Component-Based Software Development of Distributed Embedded Systems.", *Printed by Malardalen University, Vasteras, Sweden*, 2009 , ISSN 1651-9256.

[9] KAUR, K; SINGH, H. Candidate process models for component based software development. *Journal of Software Engineering*. 4, 1, 16-29p, Jan. 1, 2010.

[10] CRNKOVIC, II; CHAUDRON, MM; LARSSON, SS. "Component-based development process and component lifecycle". *Journal of Computing and Information Technology.* 2005. ISSN: 1330-1136.

[11] LUIZ F., C. “Y: A New Component-Based Software Life Cycle Model”. *Journal of Computer Science.* 2005. ISSN: 1549-3636.

[12] LAU, K; TAWEEL, F; TRAN, C. “The W model for component-based software development”. *Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011, 47-50p, Jan. 1, 2011*. ISSN: 9780769544885.

[13] TOMAR, P; GILL, N. “Verification & Validation of components with new X Component-Based Model”. *2010 2nd International Conference on Software Technology & Engineering (ICSTE)*. V2, Jan. 2, 2010. ISSN: 9781424486670.

[14] BROWN, A.; CONALLEN, J; TROPEANO, D. “Models, modeling, and model driven development,” in Model-Driven Software Development. Berlin: Springer, 2005, 1-17p.

[15] BAKSHI, A; SINGH, RUP. “Component Based Development in Software Engineering”. *International Journal of Recent Technology and Engineering (IJRTE).* March, 2013. ISSN: 2277-3878.

[16] SOMMERVILLE, I. 2010. “Software Engineering (9th ed.)”. *Addison-Wesley Publishing Company*, 452-475p.

[17] CRNKOVIC, I; CHAUDRON, M; LARSSON, S., “Component-based development process and component lifecycle,” in Proceedings of the International Conference on Software Engineering Advances. Washington, DC, USA: *IEEE Computer Society*, Jan. 1, 2006. ISSN: 0769527035..

[18] CHATTERJEE, R; RATHI, H. “A prolific approach towards automating component repository search”. *2014 Seventh International Conference on Contemporary Computing (IC3)*. *IEEE*, Jan. 2014. ISSN: 9781479951727.

[19] ZAREMSKI, A; WING, J. “Specification Matching of Software Components. *ACM Transactions on Software Engineering and Methodology*. 6, 4, 333-369, Oct. 1, 1997. ISSN: 1049331X.

[20] BECKER, S; et al. “Towards An Engineering Approach To Component Adaptation”. 3938 LNCS. *Springer Verlag*, Jan. 1, 2006. 193-215p . 3938 LNCS. (Lecture Notes in Computer Science). ISBN: 3540358005.

[21] LEE, E. “Heterogeneous Modeling”, In “System Design, Modeling, and Simulation using Ptolemy II”, *Ptolemy.org*, 2014.

[22] POMMIER-BUDINGER, V; BUDINGER, M. “Sizing optimization of piezoelectric smart structures with metamodeling techniques for dynamic applications”. *International Journal Of Applied Electromagnetics And*

Mechanics. 46, 1, 195-206p, 2014. ISSN: 13835416.

[23] SYRIANI, E; GRAY, J; VANGHELUWE, H. "Modeling A Model Transformation Language". *Springer Berlin Heidelberg*, 211-237 p, Jan. 1, 2013. ISBN: 9783642366543.

[25] FRENCH, A. "Web development life cycle: A New Methodology For Developing Web Applications". *Journal of Internet Banking and Commerce*. 16, 2, Aug. 1, 2011. ISSN: 12045357.

[26] UIKEY, N; SUMAN, U." A Lifecycle Model for Web-based Application Development: Incorporating Agile and Plan-driven Methodology". *International Journal of Computer Applications,*117, 19, May 2015. ISSN: 0975 – 8887.

[27] SAMETINGER, J., "Software Engineering with Reusable Components", *Springer Berlin Heidelberg.* 1997.

[28] EMERSON, M; NEEMA, S; SZTIPANOVITS, J., "Metamodeling Languages and Metaprogrammable Tools", In "Handbook of Real-Time and Embedded Systems, Ed. Lee, I; Leung, J; Son, S,. CRC Press, 2006.

[29] CONALLEN, J.2003. "Web Application Basics". In *Building Web Applications with UML, 2nd Edition*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 9-30p.

[30] KÖVI, A; VARRÓ, D." An eclipse-based framework for AIS service configurations". In Proceedings in the 4th International Service Availability Symposium, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics*)*, 110-126p, Jan. 1, 2007. ISSN: 3540727353.

[31] RODRIGUES DA SILVA, A. "Model-Driven Engineering: A Survey Supported By The Unified Conceptual Model". *Computer Languages, Systems and Structures*.43, 139–155p, Nov. 14, 2014. ISSN: 14778424.

[32] SHIVA, SG; SHALA, LA. "Software Reuse: Research and Practice". *Fourth International Conference on Information Technology (ITNG'07).IEEE*, 603-609 p, Jan. 2007. ISSN: 9780769527765.

[33] DESOUZA, K.; AWAZU Y; TIWANA ,A. "Dynamics Four For BRINGING USE BACK INTO Software Reuse". *Communications Of The Acm,* Vol. 49, No. 1, 97-100p, Jan. 2006.

[34] HENNINGER, S." An Evolutionary Approach To Constructing Effective Software Reuse Repositories". *ACM Transactions on Software Engineering and Methodology*. 6, 2, 111-140p, Apr. 1, 1997. ISSN: 1049331X.

[35] BROGI, E; CANAL, C; PIMENTEL, E. “Measuring component adaptation”.*2949 LNCS (Lecture Notes in Computer Science). Springer Berlin Heidelberg*, 71-86p, Feb.27, 2004. ISBN: 978-3-540-24634-3.

[36] CANAL, C; PIMENTEL, E; TROYA, JM. “Compatibility and inheritance in software architectures”. *Science of Computer Programming*. 41, 105-138p, Jan. 1, 2001. ISSN: 0167-6423.

[37] ALLEN, R; GARLAN, D. “A Formal Basis for Architectural Connection”. *ACM Transactions on Software Engineering and Methodology*. 6, 3, 213-249, July 1, 1997. ISSN: 1049331X.

[38] SINGH, S; SINGH, R. “Reusability Framework for Cloud Computing”. *International Journal Of Computational Engineering Research (ijceronline.com).* 2, 6,169-177p, Oct. 30, 2012. ISSN 2250-3005.

[39] “A Guide To The Project Management Body Of Knowledge” (PMBOK Guides), Project Management Institute, 2000.

[40] USLANDER, T., “Service-Oriented Design of Environmental Information Systems”. *KIT Scientific Publishing*, 49-53p, 2010.

[41] SANCHEZ,V; OLIVEIRA, R; FORTES, R., “RestML: Modeling RESTful Web Services”. In “*REST: Advanced Research Topics and Practical Applications”.,* Alarcón, R., Pautasso, C., Wilde(Eds.). New York : Springer, 2014. ISBN: 9781461492986.

[42] CHOURA, H; FEKI, J. “MDA Compliant Approach for Data Mart Schemas Generation. In Model and data engineering”, first international conference, MEDI 2011, Óbidos, Portugal*, ,* 2011. *Springer*, 262–269p, Sep.,30*,* 2011. ISBN: 3-642-24442-4.

[43] MELLOR, S.; et al. 2002. “Model-Driven Architecture”. In *Proceedings of the Workshops on Advances in Object-Oriented Information Systems* (OOIS '02), Jean-Michel Bruel and Zohra Bellahsene (Eds.). Springer-Verlag, 290-297p. September 02, 2002. ISBN:3-540-44088-7

[44] DANIEL, M; YELLIN; STROM, R. “Protocol Specifications And Component Adaptors”. *ACM Transactions on Programming Languages and Systems*. 19, 2, 292-333p., March 1997. ISSN:0164-0925.

[45] HARMAN, M; et al.. “Search--based approaches to the component selection and prioritization problem”. 8th annual conference on Genetic and evolutionary computation (GECCO '06). *ACM.* 1951-1952p. *July,12, 2006.* ISBN:1-59593-186-4

[46] PARVIAINEN, P; et al. “Model-Driven Development Processes and practices”. *VTT Technical Research Centre of Finland.* 38-61p*.,* Feb., 2009. ISSN: 1459-7683 .

[47] CAO, M; CAO, Z; & LI, H. “Support for development and test of web application: A tree-oriented model”. *Journal of Shanghai University (English Edition).*357-362p, May,15, 2011. ISSN:11741-011-0751-1.

[48] LI, J; CHUSHO, T. “ A web application framework for end-user-initiative development with a visual tool”. *Lecture Notes in Engineering and Computer Science*. 2195, International Multi Conference of Engineers and Computer Scientists, IMECS 2012, 816-822p, Jan. 1, 2012. ISSN: 9789881925114.

[49] RANA,J; MORSHED,S; SYNNES,K. “ End-user creation of social apps by utilizing web-based social components and visual app composition”. 22nd International Conference on World Wide Web *(*WWW '13 *Companion). ACM.* 1205-1214p, May 13, 2013 ISBN: 978-1-4503-2038-2

[50] XIAODONG, Z; et al. “Realization of a development platform for Web-based product customization systems”. *International Journal of Computer Integrated Manufacturing*. 20, 2/3, 254-264, Mar. 2007. ISSN: 0951192X.

[51] LILOV, V; ILIEVA, S. “Towards development and use of in-house component framework: results and expectations”. *31st EUROMICRO Conference on Software Engineering & Advanced Applications*. *IEEE.* Jan.,12, 2005. ISSN: 9780769524313.

[52] KACPRZYK, J; et al. “BiTutor-A Component Based Architecture for Developing Web Based Intelligent Tutoring System”. *Springer-Verlag Berlin Heidelberg*.266-271p, Jan. 2007. ISBN: 9783540725749.

[53] FRATERNALI, P; PAOLINI, P. “Model-Driven Development Of Web Applications: The Autoweb System”. *ACM Transactions on Information Systems*. 18, 4, 323-382, Oct. 1, 2000. ISSN: 10468188.

[54] MELIÁ, S; GÓMEZ, J. “Applying transformations to model driven development of web applications”. *3770 LNCS. (Lecture Notes in Computer Science). Elsevier B.V*. 63-73 p. Jan. 1, 2005. ISBN: 3540293957.

[55] HOU, J. “A Web UI Modeling Approach Supporting Model-Driven Software Development”. *Springer Verlag*, 265-274p, Jan. 1, 2014. ISBN: 9783642378287.

[56] HOU, J. “Model-Driven Approach For The Development Of Web Application System”. *Springer Verlag*, 257-264 p, Jan. 1, 2014 . ISBN:

9783642378287.

[57] PHUNG-KHAC, A; et al. “Model-Driven Development Of Component-Based Adaptive Distributed Applications”. *ACM*, 2186-2191p., March 20, 2008. ISBN: 978-1-59593-753-7.

[58] WANG, Z; Wang, H; Zhan, N. “Refinement of models of software components”. In Proceedings of the 2010 ACM Symposium on Applied Computing *(SAC '10). ACM, New York, NY, USA*, 2311-2318p., March,26, 2010. ISBN: 978-1-60558-639-7

[59] KAINZ, G; et al.” Model-To-Metamodel Transformation For The Development Of Component-Based Systems”. *6395 LNCS. Springer-Verlag Berlin Heidelberg.* 391-405 p, Jan. 1, 2010 . ISBN: 3642161286.

[60] FONS, J; et al. “Applying the OOWS Model-driven Approach for Developing Web Applications. The Internet Movie Database Case Study” In Web Engineering: Modeling and Implementing Web Applications (2008), pp. 65-108.

[61] FIALA, Z; et al. “Design and Implementation of Component-based Adaptive Web Presentations”. *ACM SA* . 1698-1704p, March,17, 2004. ISBN:1-58113-812-1

[62] HERRERO AGUSTIN, JL; DEL BARCO, PC.” A Model-Driven Approach To Develop High Performance Web Applications”. *The Journal of Systems & Software*. 86, 3013-3023p, *ScienceDirect.* Dec. 1, 2013. ISSN: 0164-1212.

[63] LIM, W.” Effects of Reuse on Quality Productivity and Economics”. *IEEE Software***,** 11, 5, 23-30p, Sep. 1994.

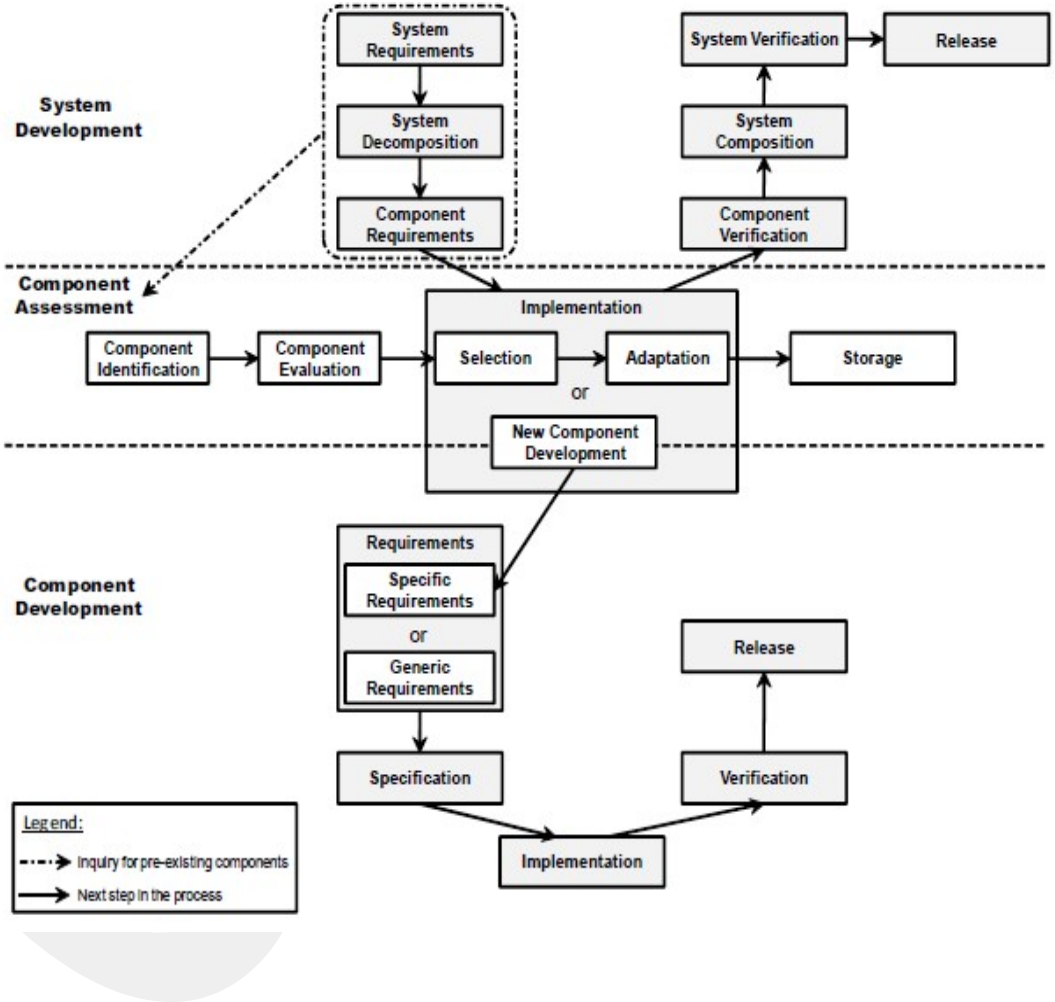
[64] MELLOR, S; CLARK, A; FUTAGAMI, T. “Model-Driven Development Guest Editors’ Introduction”. *IEEE Computer Society,* Oct., 2003. ISSN:0 7 4 0 - 7 4 5 9

[65] WEINREICH, R; SAMETINGER, J. 2001. “Component models and component services: concepts and principles”. In *Component-based software engineering*, George T. Heineman and William T. Councill (Eds.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 33-48p.

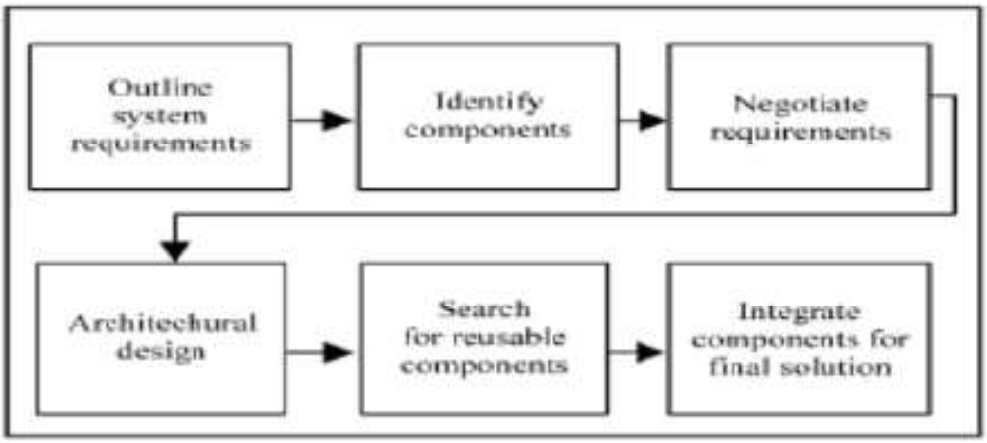
[66] CRNKOVIC, I; STAFFORD, J; SZYPERSKI, C. “Software Components Beyond Programming: From Routines To Services”. *IEEE Software*. 28, 3, 22-26, May 1, 2011. ISSN: 07407459.

APPENDIX A

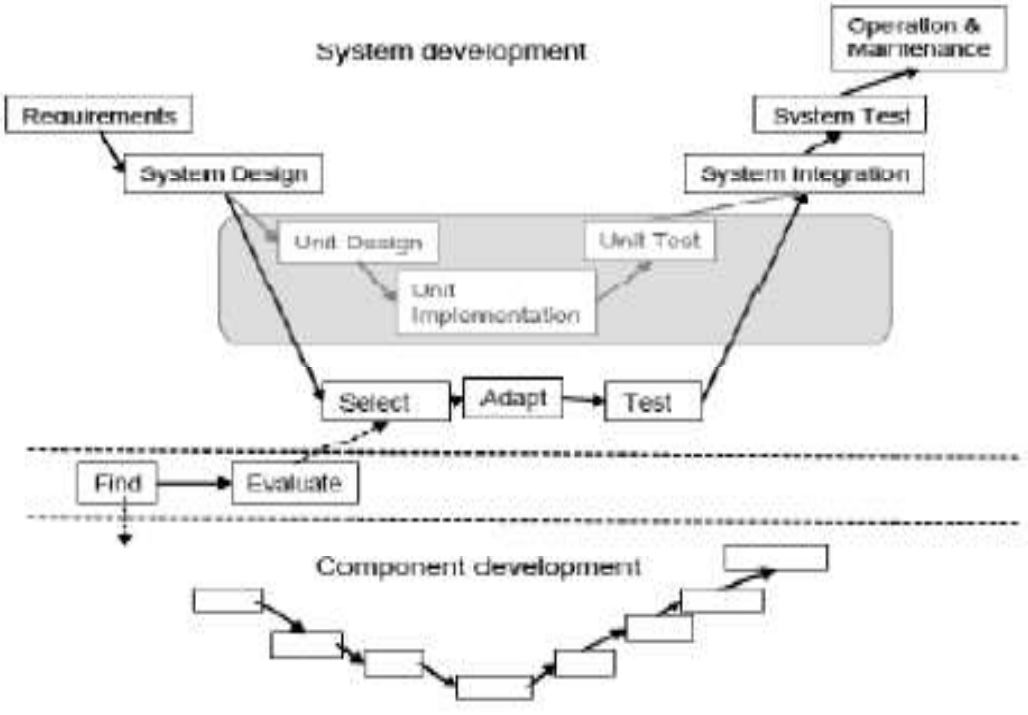
Software Process Models



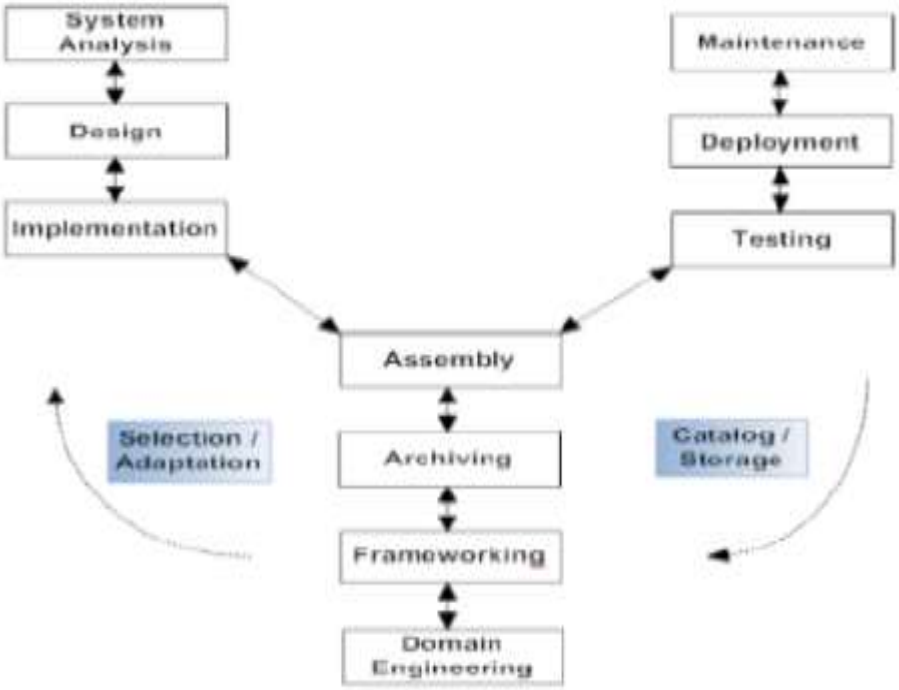
A.1. Component-based development process overview[8].



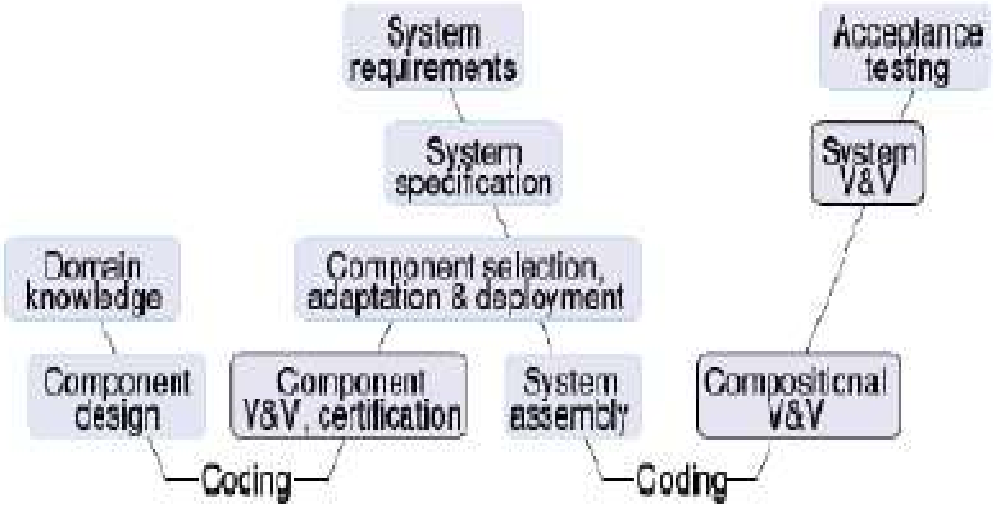
A.2. CBSD Process Model defined by Somerville [9]



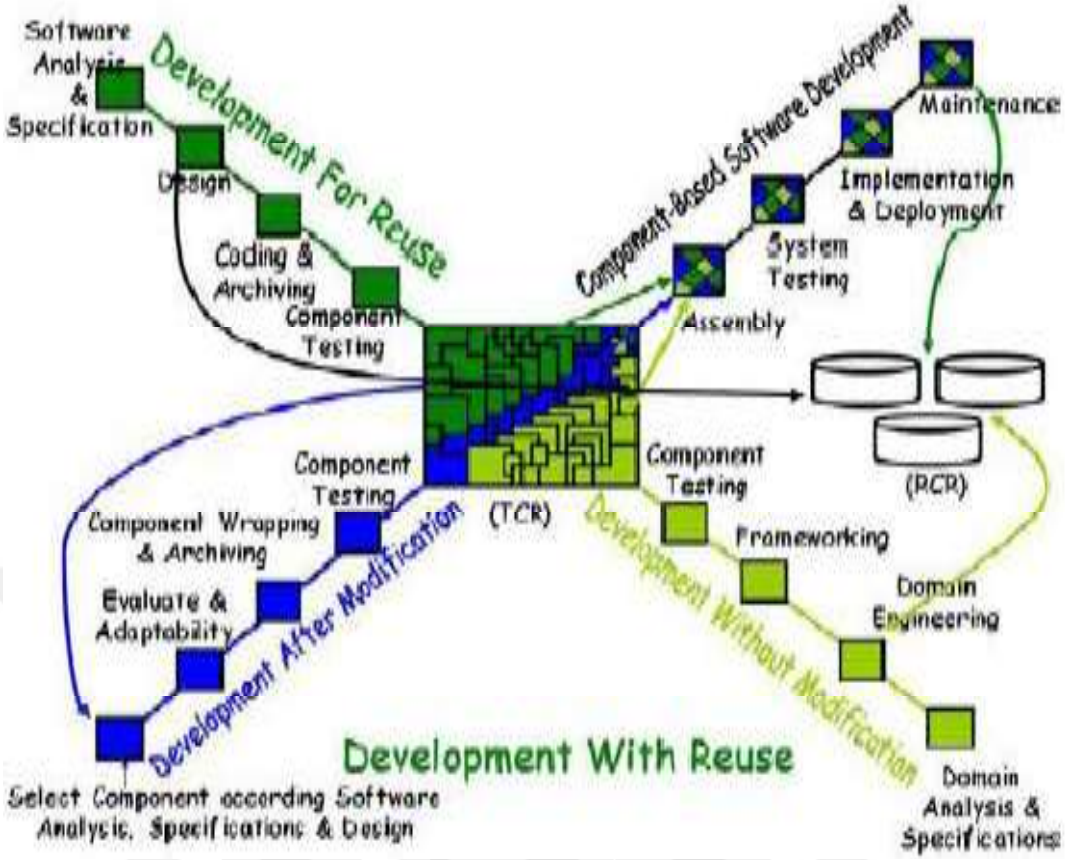
A.3. V Model[10]



A.4. Y Model [11]



A.5. W Model[12]


Software Analysis & Specification
Design
Coding & Archiving
Component Testing
Development For Reuse
Component-Based Software Development
Maintenance
Implementation & Deployment
System Testing
Assembly
(RCR)
Component Testing
(TCR)
Component Wrapping & Archiving
Evaluate & Adaptability
Development After Modification
Component Testing
Frameworking
Domain Engineering
Development Without Modification
Development With Reuse
Select Component according Software Analysis, Specifications & Design
Domain Analysis & Specifications

A.6. X Model [13]

APPENDIX B

Parts of Code

```
session_start();
$_SESSION['mid'];

$lang=$_POST['lang'];
if($lang=='ar'){
include('ar.php');}
else{
include('eng.php');
}

?>
<html>
    <head>
        <title>Quiz</title>
        <?php  echo$style; ?>
        <link rel="stylesheet" type="text/css" href="main.css"/>
        <script src="functions.js"></script>
    </head>

    <body>
        <header>


                <div   align="right" style="padding:0 0 0 0 ;color:#999;font-size:16">
                    <form action="quiz.php" method="POST">
                    <select name="lang">
                    <option selected value="eng">En</option>
                    <option value="ar">Ar</option>
                    </select>
                     <input type="submit" name="submit" value="OK"/>
                      </form>
                </div>
                <div>
                    <h1 style="width:100%;padding:0 0 0 0;">ONLINE QUIZ SYSTEM</h1>

                </div>
                <div   align="right" style="padding:0 0 0 0 ;color:#999;font-size:16">
                    Not a member yet? <a style="color:white" href="comsignup.php"/>Sign up </a>
                    <span> | </span>
                    <?php if(!isset($_SESSION['username'])){ echo'
                    <a style="color:white" href="comsignin.php"/>Login</a>';}
                    else{echo'<a style="color:white" href="logout.php"/>Logout</a>';}?>   </p>
                </div>
        </header>

        <div id="center">
            <center><strong><?php echo$h1; ?></strong></center>
        </div>
        <br>
        <div>
            <table width="100%" style="color:rgba(1,89,120,1)">


                <tr width='100%' class="tr11">
                   <td><div  class='box2'><form action=<?php  if(!isset($_SESSION['mid'])){echo "comsignin.php";} else
                   echo"next.php"; ?> method="POST"/><h2 style="font-size:150%;color:rgba(1,89,120,1);" ><center><b><?php
                   echo$add; ?></b></center></h2> <?php echo$section; ?><br><br>
                   <p align='right'><button class='button' type="submit"/><?php echo$add3; ?></button></p></form></div></td>

                   <td></td>
                   <td><div  width='100%' class='box2'><form action="quizstart.php" method="POST"/><h2
                   style="font-size:150%;color:rgba(1,89,120,1);" ><center><b><?php echo$add2; ?></b></center></h2><?php
                   echo$section2; ?><br><br><p align='right'><button class="button" type="submit"/><?php echo$add2;
                   ?></button></p></form></div></td>
                </tr>
                <tr>

                </tr>
            </table>
        </div>

    </body>
</html>
```

Figure B.1. Home page code

```
<!DOCTYPE html>
<?php
  session_start();
            $_SESSION['mesg'];
            $flag1=1;
  $base=basename($_SERVER['PHP_SELF'],".php");
   $paramarg = $_GET['argument1'];
  $varfile = $base.$paramarg.'var.php';
  require($varfile);
?>
<html>
    <body>
     <form action="<?php echo$base."1.php?argument1=".$paramarg;?>" method="POST"/>
                        <input type='hidden' name='base' value='<?php echo$base; ?>' >
       <table>
          <tr>
          <?php
              for($i=0;$i<$colno;$i++) {
                    echo "<tr><td>".$colc[$i]."</td>
                   <td><".$type[$i]. " list='".$colc[$i]."' id='col".$i."' name='col".$i."'/>";
                                                    include($colc[$i].'.php');
                                                            "</td></tr>"; }
            ?>
           </tr>
           <tr></tr>
        <tr>    <td></td><td><input type="submit" style="float:right" width="30px" value="Back"
name="Back" id="Back"> </td><td>
          <input type="submit" width="30px" value=" Add" name="adds"/></td>
                                  </tr>
                                  <tr><td></td><td></td> <td><?php echo $_SESSION['mesg'];
           ?></td></tr>
                                 </table>
                                 <?php $_SESSION['mesg']='';?>
                      </form>
                    </body>
</html>
```

Figure B.2. Part of code from Add component

```php
<?php
    $base=basename($_SERVER['PHP_SELF'],".php");
    $base1=basename($_SERVER['PHP_SELF']);
    require($base.'var.php');


    echo'
        <html>
            <div style="overflow-x:auto;">
                <form action="'.$base1.'" method="post">
                    <table width="80%"><tr>
                        <td><input size="60%" type="search" name="question" placeholder="
                                '.$searchc.'"/>  <input type="submit" value="search"
                        name="search"/></td>';
    $searchq=$_POST['question'];
    $x=0;

   if(!empty($searchq))
   {
       for($i=0;$i<$Notb;$i++)
       {


           $sql="SELECT* from ".$tbln[$i]." Where ".$col[$i]." LIKE '%".$searchq."%'".$F;
           $result=mysqli_query($conn,$sql);

           if ($result->num_rows != 0)
           {
               $x++;
               $fieldcount=mysqli_num_fields($result);
               echo"<table id='quiz' width='80%'><tr>";
               while ($fieldinfo=mysqli_fetch_field($result))
               {
                   echo "<th>".$fieldinfo->name."</th>";
               }
                   echo"</tr>";
                   while($row=mysqli_fetch_row($result))
                   {
                       echo"<tr>";
                       for($y=0;$y<$fieldcount;$y++)
                       {
                           echo'<td>'.$row[$y].'</td>';
                       }
                       echo"<tr>";
                   }
                       echo'</table><br><br>';
           }

           else $x=$x-1;
       }



       echo'</div></html>';

   }

   if($x<0){ echo"There is no search results";}

?>
```

Figure B.3. Part of code from search component

```php
<?php

$base=basename($_SERVER['PHP_SELF'],".php");
$base1=basename($_SERVER['PHP_SELF']);
require($base.'var.php');
$delA=array();

 echo'
    <html>
      <div style="overflow-x:auto;">';

        for($i=0;$i<$Notb;$i++)
        { $R=0;


            $sql="SELECT* from ".$tbln[$i]." ".$F;
            $result=mysqli_query($conn,$sql);
              $h=$result->num_rows;
            if ($result->num_rows != 0)
            {
                $x++;
                $fieldcount=mysqli_num_fields($result);

                echo"<table id='quiz' ><tr>";
                while ($fieldinfo=mysqli_fetch_field($result))
                {   $delA[$R]=$fieldinfo->name;

                 echo "<th>".$fieldinfo->name."</th>";
                 $tab=$fieldinfo->table;
                 $R++;
                }
                    echo"<th></th></tr>";$k=1;
                    while($row=mysqli_fetch_row($result))
                    {
                        echo"<tr>";
                        echo"<form action='del.php' method='POST'>";
                        for($y=0;$y<$fieldcount;$y++)
                        {echo"<input   type='hidden' value='".$row[$y]."' name='c".$y."' >";
                         echo"<input   type='hidden' value='".$delA[$y]."' name='h".$y."' >";
                         echo'<td>'.$row[$y].'</td>';
                        }
                        echo"<td class='del'>
                        <input  type='hidden' name='Fno' value='".$fieldcount."' >
                        <input  type='hidden' name='tab' value='".$tab."' >
                        <input  type='hidden' name='base' value='".$base1."' >
                        <input type='image' src='del2.png' alt='Submit' name='del' value='".$k."'></td>
                        </form></tr>";
                        $k++;
                    }
                        echo'</table><br><br>';
            }

            else $x=$x-1;
        }
        mysqli_free_result($result);
        echo'</div></html>';
?>
```

Figure B.4. Part of code from view component

```
<?php
session_start();
$_SESSION['mesg1'];
$base=basename($_SERVER['PHP_SELF'],".php");
$base1=basename($_SERVER['PHP_SELF']);
require($base.'var.php');
?>
<!doctype html>
<html>
    <head>
        <title>Quiz</title>
        <link rel="stylesheet" type="text/css" href="main.css"/>
        <script src="functions.js"></script>
    </head>

    <body>
        <header>
                <div>
                    <h1 style="width:100%"><?php echo $pagecaption ; ?></h1>
                </div>
       </header>

       <table width="100%"  >
       <tr>
       <td width="30%">
       </td>
       <td>
        <div id="box" >
       <form action="<?php echo$base."1.php";?>" method="POST"/>
       <input type='hidden'  name='base' value='<?php echo$base; ?>' >
       <input type='hidden'  name='base1' value='<?php echo$base1; ?>' >
          <table style="padding:25px 20px">
           <tr>
           </tr>
           <tr>
           </tr>
           <tr>
           <td><?php echo$c3; ?></td>
           <td><input style="font-size:16px;border-radius: 10px; "  type="text" id="username" name="username"
                    placeholder="Choose a user name" size="35"/></td>
           </tr>
           <tr>
           <td><label><?php echo$c4; ?></label></td>
           <td><input style="font-size:16px;border-radius: 10px;"  type="password" id="pass" name="pass"
                    placeholder="Choose a password" size="35"/></td>
           </tr>
           <tr>
           </tr>
           <br>
           <tr>
           <td><br><br><td>
           <td><input style="font-size:16px;border-radius: 10px; "  type="submit" id="save" name="save" value="Log in"/></td>
           </tr>
           </table>
       </form>
       </div>
       </td>
       <td>
       </td>
       <tr>
            <td width="30%"></td>
            <td width="30%"><table style="padding:0 0 0 0; border-collapse: collapse" width="100%" ><tr>
            <td width="50%" style="text-align:left" ><a href=<?php echo$mainpage; ?>><strong>Back to Main Page
                </strong></a></td><td width="50%" style="text-align:right"><a href="comsignup.php"><strong>
                Sign Up</strong></a></td></tr></table></td>
       <tr><td width="30%"></td><td width="30%" style="text-align:left"><?php echo $_SESSION['mesg1'];?></td><td></td></tr>
       </table>


   </body>
</html>
```

Figure B.5. Part of code from sign in component

```
<?php>
session_start();
     $_SESSION['mesg2'];
     include('quiz2.php');?>

          <p>Please fill the following information to start the quiz.</p>
        </div>
        <table id="section">
        <form action="test.php" method="POST"/>
            <div>
                    <tr>
                        <td>Select Category</td>
                        <td> <input size="35" list="Category" name="Category" /></td>
                            <datalist id="Category">
                                    <?php      include('Category.php'); ?>

                                </datalist></tr>
                    <tr>
                        <td>Select Subcategory</td>
                        <td> <input list="Subcategory" name="subcategory" /></td>
                        <datalist id="Subcategory">
                                        <?php  include('Subcategory.php'); ?>

                                      </datalist>
                    </tr>

                    <tr>
                        <td>Select Language</td>
                        <td><select id="lang" name="lang">
                           <?php  include('Language.php'); ?>
                           </td>
                    </tr>
                    <tr>
                        <td>Question Type</td>
                        <td><select id="qtype" name="qtype" onchange="myFunction()">
                            <option></option>
                            <option>Multiple choices</option>
                            <option>True and False</option>
                            <option>Text</option>

                            </select></td>
                    </tr>
                    <tr>
                        <td>Choose Number of Questions</td>
                        <td><input type="text" id="qn" name="qn"/></td>
                   </tr>
                   <tr>
                   <td><br></td>
                   </tr>   <tr></tr>
                    <tr>
                    <td>
                    </td>
                    <br><td></td>
                    <td align="right">
                    <br><button class='button' type="submit" id="start" name="start"
                         />Start Quiz</button>
                    </td>
                   </tr>
            </form>
            <tr><td><p style="color:red"><?php>
$s=$_SESSION['mesg2'];
echo$s;
$_SESSION['mesg2']='';
?></p>  </td></tr>
</table>
```

Figure B.6. Part of code from start quiz