SUDAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF GRADUATE STUDIES

A CONTAINER- BASED ARCHITECTURE FOR THE DESIGN OF PORTABLE CLOUD APPLICATIONS

المعمارية المبنية على الحاوية لتصميم التطبيقات السحابية المتنقلة

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

By

Amar Ibrahim Elhaj Sharaf Eldein

Supervisor

Professor
Dr. Hany Hussein Ammar

March 2019

الآيـــــــــة

قال تعالى:

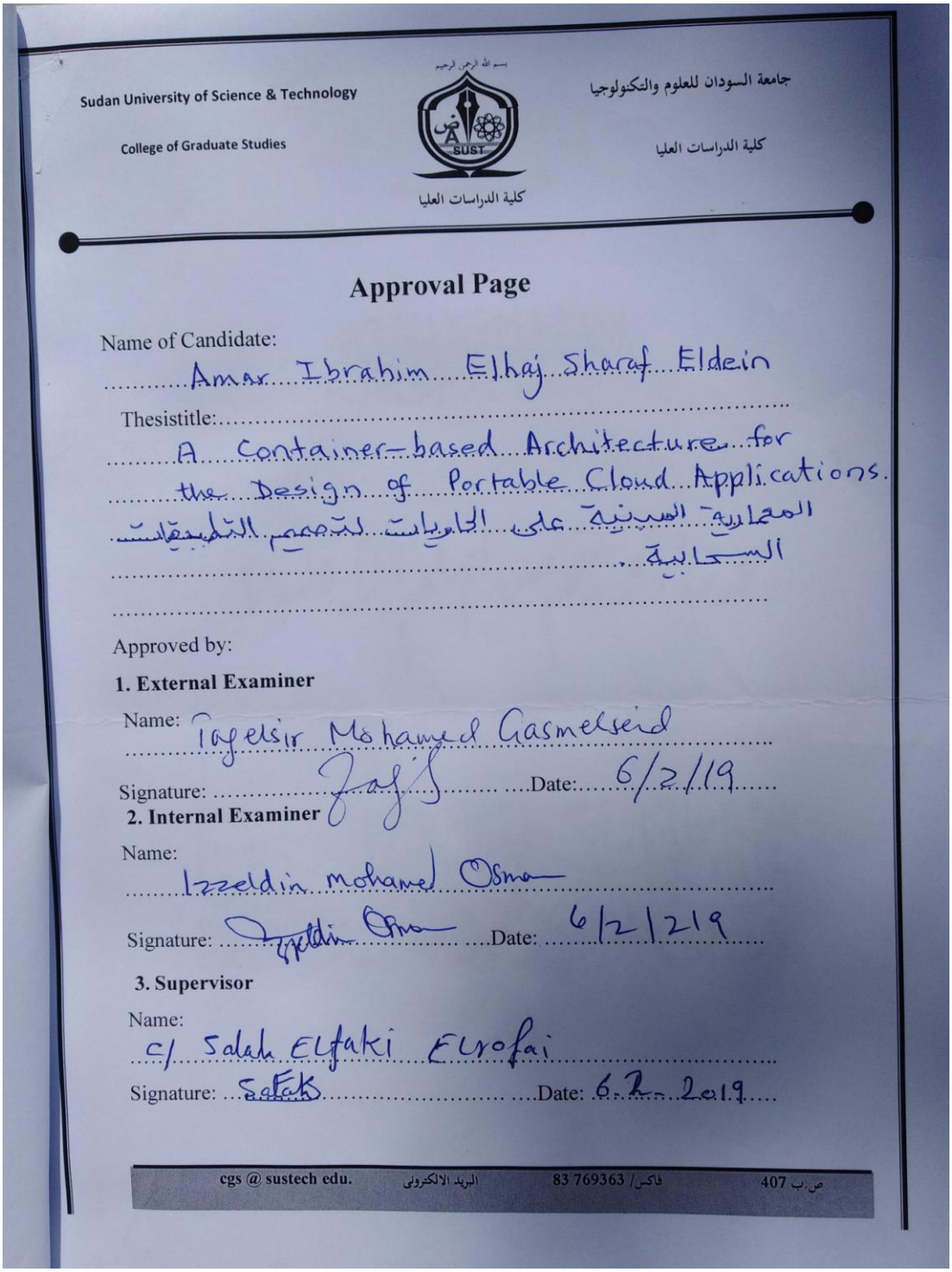
سورة الإسراء - الآية (85).

قال تعالى:

صدق الله العظيم

سورة المجادلة - الآية (11).

Examiners Approval



Sudan University of Science & Technology

College of Graduate Studies

بسم الله الرحمن الرحيم

جامعة السودان للعلوم والتكنولوجيا

كلية الدراسات العليا

كلية الدراسات العليا

Approval Page

Name of Candidate:

Amar Ibrahim Elhaj Sharaf Eldein

Thesistitle:

A Container-based Architecture for the Design of Portable Cloud Applications.

المعمارية المبنية على الحاويات لتصميم التطبيقات السحابية

Approved by:

1. External Examiner

Name: Tagelsir Mohamed Gasmelseid

Signature: ..... Date: 6/2/19

2. Internal Examiner

Name:

Izzeldin mohamed Osman

Signature: ..... Date: 6/2/219

3. Supervisor

Name:

c/ Salah Elfaki Elrofai

Signature: ..... Date: 6.2.2019

cgs @ sustech edu. البريد الالكتروني 83 769363 /فاكس ص.ب 407

Declaration of the statues of Thesis
(By Student)

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

Candidate's name: Amar Ibrahim Elhaj Sharaf Eldein

Candidate's signature: _____

Date:_____

Declaration of the statues of Thesis
(By Main Supervisor)

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

A Container-Based Architecture for the Design of Portable Cloud Applications

Supervisor's name: Prof. Hany Ammar

Supervisor's signature:

Date:

Assigning the copy-right to CGS

I, the signing hereunder, declare that I'm the sole author of the Ph.D. dissertation entitled:

A Container-Based Architecture for the Design of Portable Cloud Applications

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

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

Candidate's name: Amar Ibrahim El Haj Sharaf Eldein

Candidate's signature:

Date:

DEDICATION

I would like to dedicate my work to

My loving parents, for their prayers,

My brother Mohammed,

My sister Rihab,

Whom have given me all the support for my success.

My teachers,

My colleagues, and

All who assist, support, and encouragement me.

ACKNOWLEDEMENT

First Alhamdulillah. I will thank Almighty GOD, who kindly helped me to complete my thesis. Carrying out a research work and writing the PhD thesis may become a lifelong experiential journey that reacts and face the challenges. It is a journey through uncharted routes to a destination, which is not clear and by no means visible when you set off. You have a direction, but you are not aware of where exactly you will end up and which path you need to follow to reach your goal. There are moments you feel that you found your way to your destination, followed by the ones that you lose it again. Nevertheless, when you reach the end of your journey, there is only one feeling left, that is fulfillment, you feel that you have fulfilled the promise to yourself.

Nevertheless, it would be unfair if I didn't mention that throughout my studies there were several people who were always standing next to me and supporting me with their own unique way. However, the gratitude I feel for those people cannot fit within the limits of this section.

Therefore, I would express my deep gratitude personally and with a special way to each one of those persons.

I would like to gratefully and sincerely thank my supervisor, **Professor Dr. Hany Husain Ammar**, for his priceless guidance, continuous support, inspiration, motivation, encouragement, and immense knowledge. Throughout my studies **Prof. Hany** taught me everything I know about conducting academic research, and provided all the needed to support my research.

Many thanks to all **members of the college of Computer Science and Information technology** of the **Sudan University of Science and Technology** for hospitality, generosity, and helps.

I would like to extend my thank to **the Qatar Research Fund** (a member of Qatar Foundation) NPRP under grant # [7-662-2-247] for funding my research paper's publication.

Last but not least, I would like to express my deepest gratitude to all my friends for their continuous encouragement.

AMAR IBRAHIM

ABSTRACT

Cloud computing has recently emerged as a new technology paradigm for hosting and delivering Information Technology (IT) services to users over the Internet. Cloud Computing provide optimal and efficient computing through collaboration, agility, availability and scalability.

Cloud platforms provide essential benefits for organizations such as greater elasticity. These platforms eliminate the requirement for users to plan ahead for resource provisioning, faster delivery times. They provide faster delivery times to through the use of cloud-based data centers and services. However, cloud computing brings challenges alongside its benefits. First, developers often produce cloud applications for a specific cloud platform, and does not support portability to Multiclouds environments. The evolving this application to cloud services end up with major complexity issues.

To contribute to solving these problems, we proposed a model driven design approach, relying on development for portability-based design of cloud application architecture. This architecture can be easily deployed on Multicloud platforms, avoiding related portability or cloud service provider vendor lock-in problems.

In this dissertation, we present a framework for cloud application architecture development which encompasses strategic planning, architecture design, deployment, and evaluation phases. This thesis, within focuses on the architecture design phase and propose new scenarios based iterative process for architecture design. We introduce the concept of a scenario container at the architecture design level and develop the architecture iteratively as an integrated set of containers. In each iteration, we define the container of application components for one application scenario.

The proposed development process guides the architect in designing containerized portable cloud application architectures. We finally propose new container design patterns that integrate with selected cloud application patterns to support the development of scenario-based containerized architecture.

We validate the design approach by applying it to two different case studies: a Student Academic Result Record system, and Hajj and Umrah mobile healthcare

system. In the second case study, we compare our new architecture development approach with the (TOGAF) enterprise architecture framework.

We verify the deployment of portable architectures on multicloud platforms using the CloudMIG simulation tool, for migrating e-commerce online pet store case study to cloud environments. We present simulation results for cloud deployment options (CDO) based on a medium response time (MRT) of the case study, considering multiple cloud platform deployment selection.

المستخلص

ظهرت الحوسبة السحابية حديثاً كنموذج تكنولوجي جديد لإستضافة وتقديم خدمات لتطوير تكنولوجيا المعلومات (IT) للمستخدمين عبر الإنترنت. تعمل الحوسبة السحابية على تقديم الحوسبة الفعالة المثلى من خلال التعاون, توفير الموارد, قابلية التوسع والمرونة لقابلية التطويربواسطة إستخدام التطبيقات والأنظمة والخدمات السحابية.

تقدم المنصات السحابية فوائد أساسية للمؤسسات مثل توفير متطلبات المستخدمين للخطط المستقبلية لتزويد الموارد, السرعة وتوفر الوقت من خلال استخدام مراكز وخدمات البيانات المعتمدة على الحوسبة السحابية. ومع ذلك ، فإن الحوسبة السحابية تواجه تحديات إلى جانب منافعها. أول هذه التحديات متمثله في إنتاج التطبيقات من قبل المطورين لمنصة سحابة معينة ، ولا تدعم قابلية التنقل إلى بيئات أخرى (Multicloud) , فتضمين هذه التطبيقات إلى الخدمات السحابية يؤدي الى مشاكل كبرى في التعقيدات.

للمساهمة في حل هذه المشاكل ، اقترحنا نهج نموذج التصميم المقاد ، بالاعتماد على تطوير تصميم معمارية الحوسبة السحابية المعتمدة على قابلية التنقل. يمكن نشر هذه المعمارية بسهولة على منصات متعددة Multicloud ، وتجنب مشاكل قابلية النقل ذات الصلة لمزود الخدمة السحابية.

في هذه الرسالة ، قدمنا إطارًا لتطوير بنية تطبيقات السحابة التي تشمل على مراحل التخطيط الإستراتيجي ، تصميم المعمارية ، النشر ، و التقييم. هذه الأطروحة ، تركز على مرحلة تصميم المعمارية عن طريق إقتراح سيناريوهات جديدة تقوم على العملية التكرارية لتصميم هندسة المعمارية للتطبيقات. قدمنا مفهوم سيناريو الحاوية على مستوى تصميم وتطوير البنية بشكل متكرر كمجموعة متكاملة من الحاويات. في كل عملية تكرار ، نحدد حاوية مكونات التطبيق لسيناريو واحد للتطبيق. ترشد عملية التطوير المقترحة معماري النظم في تصميم بنية تطبيقات السحابة المحمولة على الحاويات. أخيراً قمنا بإقتراح أنماط تصميم حاويات جديدة تتكامل مع أنماط تطبيقات الحوسبة السحابية المختارة لدعم تطوير بنية الحاوية القائمة على السيناريو.

في هذا البحث، قمنا بالتحقق من صحة نهج التصميم من خلال تطبيقه على دراستي حالتين مختلفتين: نظام سجل النتائج الأكاديمية للطلاب ونظام الرعاية الصحية المتنقلة للحج والعمرة. في حالة الدراسة الثانية ، قمنا بمقارنة نهج تطوير البناء الجديد مع إطار عمل المؤسسات (TOGAF).

تم التحقق من النشر للمعمارية المتنقلة بإستخدام أداة محاكاة CloudMIG ، التي تدعم تهجير الأنظمة إلى البيئات السحابية عبر دراسة حالة متجر التسوق الإلكتروني. تم تقديم نتائج تقييم خيارات النشر السحابية (CDO) لمتوسط زمن الإستجابة (MRT) لدراسة الحالة بالإعتماد على إختيار العديد من المنصات السحابية للنشر.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADM	Architecture Development Method
API	Application Programming Interface
ARTIST	Advanced software-based service provisioning and migration of legacy Software
CAAP	Cloud Application Architectural Pattern
CCRA	Cloud Computing Characteristics and Reference Architecture
CDO	Cloud Deployment Options
CEC	Cloud Environment Constraints
EA	Enterprise Architecture
EHR	Electronic Health Record
ERP	Enterprise Resource Planning
HUMS	Hajj and Umrah Mobile Healthcare System
IaaS	Infrastructure as a Service
IT	Information Technology
MCC	Mobile Cloud Computing
MDA	Model Driven Architecture
MID	Mobile Internet Devices
MRT	Median Response Time
MVC	Model Viewer Controller
NIST	National Institute of Standards and Technology
OMG	Object Management Group
PaaS	Platform as a Service
PIM	Platform Independent Model
PSM	Platform Specific Model
QoS	Quality of Service
REMICS	Reuse and Migration of legacy applications to Interoperable Cloud Services
SA	Software Architecture
SaaS	Software as a Service
SDO	Standards Development Organization
SLA	Service Level Agreement
SMEs	Small and Medium Enterprises
TOGAF	The Open Group Application Framework
UML	Unified Modeling Language

LIST OF APPENDICES

CHAPTER I

INTRODUCTION

1.1 Introduction

Cloud Computing has recently emerged as a paradigm for managing and delivering reliable services over the internet and promises to rapidly changing the landscape of information technology, ultimately turning utility computing into a reality, and enable users be able to access applications and data from a Cloud anywhere in the world (Shawish & Salama 2014).

As a result, most of critical applications migrate to cloud computing. While Cloud computing has been gaining more popularity, but poses a threat from both business and technical that affect the cloud success. On the technical side, one of the main reasons for success or failure is the architectural design of the system. In addition, customers want to avoid risk and increase flexibility through the movement of applications and data among cloud providers (Dimitrov 2015).

Software architecture aimed to support system design, architectural patterns, and process changes as an important concept applied to cloud computing. There is a need for a software architecture design and development approach to mitigate the undesirable effects of technology change in clouds for the application components to provide the best software and hardware configuration to ensure the architecture design for cloud application is portable to reduce the effects of building and easily deploying cloud applications between platforms with minimal requirements modifications, without compromising the efficiency, and utilization of the whole system (Dimitrov 2015).

Cloud applications using software architecture through architectural cloud patterns as one of the significant solution, that capture practical knowledge and give a general solution on how to cope with software architectural problems, based on using model driven design approach for development and design of cloud application architecture as an active area of research, since design architecture approach ensuing software solutions that are more robust, flexible and agile for evolving applications(Sharma & Sood 2011).

1.2 Problem Statement

Cloud providers produce cloud applications for their a specific cloud platform, that lock-in enterprise customer and does not support portability across multiple cloud environments (Ferry et al. 2013; Beslic et al. 2013).

This problem has negatively impacted for cloud provider and enterprises (victims of the problem). The core solution is applying a new set of processes for designing portable cloud application architecture for deployment across multicloud platforms.

More challenges for cloud application portability are represented in:-

(1) Cloud providers (e.g. Rack space) promoting options to facilitate portability, while other not care to accommodating it.

(2) The lack of standardization in cloud computing (Rimal et al. 2011), makes the task of portability of applications between clouds challenging.

(3) Limited Support for the Migration plan of Enterprise Software Systems to the Cloud (Gholami et al. 2016).

(4) The portability of applications across various clouds, impose a level of a complexity, that need additional efforts for developing and to move an application to/ from one cloud service to another.

- Platform as a Service (PaaS): lack of a consistent platform definition among PaaS Providers: Some PaaS providers pose greater risks than others, required major changes in software and caused delays and productivity losses problems, developers have to address them to create portable applications.

  If providers make different selections, then applications use different features can't be ported, and if provider customized the PaaS features to move the application back to the same platform on-premises will be difficult.
- Infrastructure as a Service (IaaS): lack of alternative providers of a platform: developers providing all of the software needed for their applications to work with IaaS, since platform moves with the application from one IaaS provider to another and can be ported back to a virtual machine on premises.

(5) Enterprises software system with limited plan to face new technologies and requirements, such as migrate to the Cloud.

1.3 Research Questions

To conduct the design review objectively, The research aims to answer the following research questions (RQs) regarding to portability for cloud application architecture bellows:

(1) RQ1: What are the framework phases to develop a portable cloud application architecture?

(2) RQ2. What are new design patterns to support the development of cloud application architecture?

(3) RQ3. How to improve the deployment of portable architecture using a simulation tool?.

1.4 Research Hypotheses

The hypotheses of this research are formulated as follows:

- The nature of Cloud is dynamic, so number, size, and complexity of applications need to communicate via a number of domains.
- The process of software architecture design insufficient understood, needs to use driven techniques with highlighting its importance.
- There is no standard for cloud patterns, we used different concepts in the cloud patterns solution.
- The proposed methodology is expected to give promising results as it will be applied in different applications and domains such as web-based application and Mobile application.

1.5 Research Objectives

The general goal is mainly to focus on using existing development processes to propose a framework for Cloud Application Architecture development, that ensure the architecture design for cloud application is portable, easily without affecting the efficiency and utilization of the whole system. In addition, the specific objectives are outlined as follows:-

(1) To look at the current status of a large needs of portable cloud applications which use of the architectural design. This objective has been achieved through: defining the portability of cloud applications, defining portability scenarios in

cloud computing, and determine the suitable development approaches for cloud application architecture design.

(2) To propose and develop a portability-based methodology, that can used to enhance previous works on cloud application architecture design.

(3) To evaluate the proposed development framework for portable cloud application architecture.

1.6 Research Scope

We are concerned with the development of SaaS application design phase, the research study does not include the second part of operation phase. Our study assumes that cloud environment is an interface to existing case studies architectures.

The proposed approach focuses on iterative development process to guide in designing scenarios-based container architecture of portable cloud applications; We used annotated UML diagrams to understand the application architecture at early design stages; We proposed containers design patterns developed within identified cloud application architecture patterns, that can be used to be easily deployed in multicloud environments.

1.7 Research methodology

Cloud applications are complex and composed of multiple services. In addition, there is no standard approach for a portable cloud application design that can be used to face development challenges. We involve strategic design for the process to develop our architectural design.

Firstly, we surveyed the existing related works in this area; as well, we analyzed the issues and challenges that have been emerging in various works.

Based on the results of a survey, we set up our framework for portable cloud application architecture development and its phases, using a model driven approach.The model driven design (MDD) process is defined outlining the different phases with a brief description. Each phase contains a set of activities. The development process used to describe the phases that support the research method as follows:

(1) Strategic plan: describe system behavior and requirements. Requirements as a reference for a process to design and deploy portable cloud applications on different cloud environment.

(2) Develop Architecture Design: define architecture design to be easily understood and implement changes during run time.

(3) Deployment architecture design: verify portability deployment options using simulation tools for multiclouds.

(4) Portability evaluation: validate proposed methodology using case studies. Generally two common domains academia and healthcare.

1.8 Research Contributions

This thesis, focuses within the architectural design phase. The contributions of this thesis can be summarized as follows:-

(1) We make a comprehensive survey to highlight future research problems in this area.

(2) Propose a framework for cloud application development to support cloud application architecture portability.

(3) Propose scenarios-based container architecture design approach, based on an iterative process for development.

(4) Propose a new containers design pattern, integrate with identified cloud application architecture patterns, that can be used to support cloud application architecture development, to be easily deployed on different cloud environments.

(5) Validate a methodology, using a scenario-based for case studies:-

- Migrate Students Academic Result Records Web base Service to the cloud.
- Hajj and Umrah Mobile Healthcare System.

(6) Present an integrated enterprise architecture views model based on (TOGAF) development method, and compared with our proposed method for case study (Hajj and Umrah Mobile Healthcare System).

(7) Verify deployment using a simulation tool for case study (Migrating Online Pet Store Application to IaaS cloud).

1.9 Thesis Organization

The rest of the thesis is organized in six chapters as follows:

Chapter II presents a theoretical background information for studies. Explains the following concepts: Cloud Computing, Cloud application design patterns, cloud application architecture development methods, covers the concepts of cloud application portability. In addition, a review for some related works of cloud application architecture development.

Chapter III introduces and explains the framework of portable cloud application development phases, focuses in architecture design phase. In addition, describes our proposed scenarios-based container architecture design approach, and illustrate new design patterns investigated for portable cloud application architecture design support.

Chapter IV introduces a case study of web base service Migrating Students Academic results record to the cloud, we model a system into annotated UML sequence diagrams, then development processes used to develop multiple scenarios of case study, using scenarios-based container architecture design for a portable cloud application.

Chapter V introduces a case study of Hajj and Umrah Mobile Healthcare System, we describe application architecture, focus on using scenario based for retrieving patient's Electronic Health Records, then we developed mobile application using Enterprise architecture using TOGAF development approach. Then we validate the application using independent scenario-based container design architecture. We compare our new architecture development approach with the TOGAF approach.

Chapter VI illustrates the verification of methodology using the CloudMiG Xpress simulation tool with case study migrating Online Pet Store application to the cloud and describes the deployment approach, and present the simulation results and discussion about these results.

At the end, we presented the conclusion and the future work based on the methodology validation and verification results mentioned the open issues for researchers. In addition to references and appendices.

CHAPTER II

BACKGROUND AND RELATEDWORK

2.1 Introduction

This chapter aims to give an overview of cloud computing. It also includes the general concepts of cloud application architecture development. The chapter describes the preliminary concepts and presents current approaches for design cloud application architecture. Next, it presents the related works of cloud application architecture development. A summary of related work is shown in this chapter after the detailed analysis, based on the driven architecture as new software development adopted for cloud application architecture design.

2.2 Cloud Computing Architecture

Cloud computing is the new perspective that changes the way we perceive Information Technology and its applications. Therefore, this chapter presents its main benefits, main roles in different sectors and how all this can come together under the perspective. This dissertation considers the technology needed to bring the idea of software architecture solutions to be developed in a manner that is independent of technology change, and to design and develop portable cloud application architecture. The design of application architecture can be deployed on heterogeneous cloud platforms, giving it the ability to scale easily (Ardagna 2015). This is different than other forms of cloud computing which may give the user software applications for them to use.

2.2.1 Cloud Computing Definitions

There is no unique definition for cloud computing. While there may be confusion about the right definition of cloud computing, it should be underlined for the gullible readers that cloud computing is not a technology revolution, but rather a process and business revolution on how we use these technologies that enable cloud computing as it exists today.

There are different cloud computing(CC) definitions in use, the state-of-the-art definition from the National Institute of Standards and Technology (NIST) has been adopted in this thesis.

Definition 1: According to NIST, cloud computing is " *a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"*. (Mell & Grance 2011; Ghanam et al. 2012; Ardagna et al. 2012). This cloud model is composed of five essential characteristics, three service models, and four deployment models.

Definition 2: *"A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service level agreements established through negotiation between the service provider and consumers.*". Using virtualization techniques, these virtualized resources, such as hardware, platforms, or services, are dynamically allocated to scale on demand according to customers' needs. If a cloud service provider (CSP) fails to offer the demand, the CSP may outsource to other cloud service providers (Cheng & Lai 2012).

Definition 3: According to the IEEE computer society cloud computing is: "*A paradigm in which information is permanently stored in servers on internet and cached temporarily on clients that include desktops, entertainment centers, table computers, notebooks, wall computers handhelds, etc."* so cloud computing provide every facility as a service, provides infrastructure as a service, software as a service and platform as a service (Ghanam et al. 2012)**.** In other form is a source for the dynamic provisioning of computing services, supported by data centers containing a group of networked Virtual Machines (Cheng & Lai 2012).

Definition 4: Sosinsky (2011) defined it as *"Cloud computing(CC) refers to applications and services that run on a distributed network using virtualized resources and accessed by common Internet protocols and networking standards. It is distinguished by the notion that resources are virtual and limitless and that details of the physical systems on which software runs are abstracted from the user".*

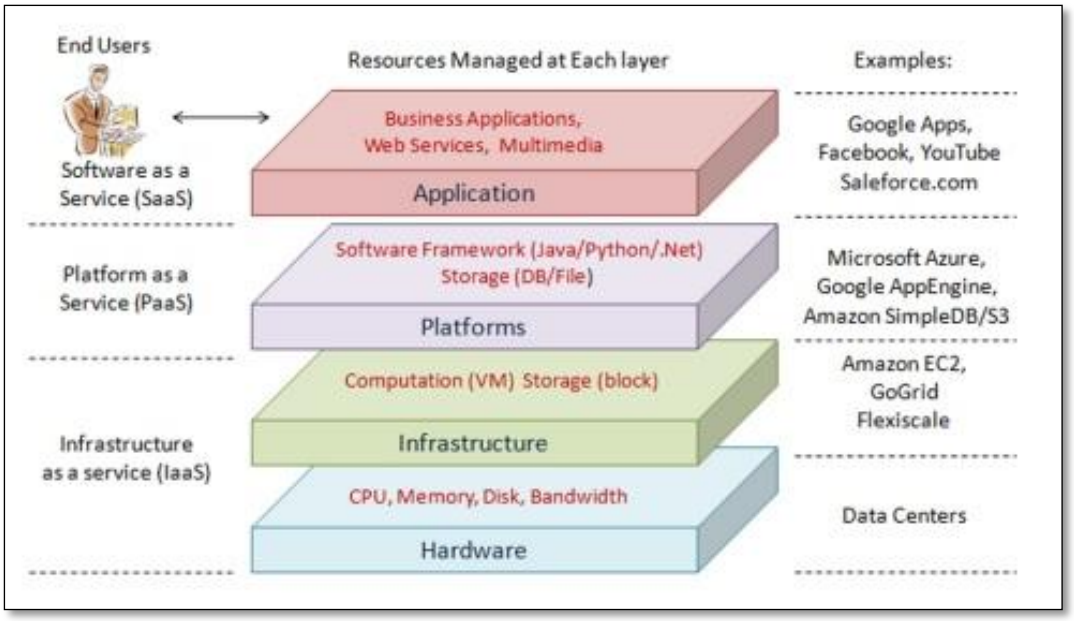


Figure 2.1 Cloud computing architecture (Zhang et al. 2010).

2.2.2 Cloud Computing Service Models

The NIST categories cloud computing into three service models, as shown in Figure 2.1, and described below

Infrastructure as a Service (IaaS) model provides infrastructure components to clients ranging from CPU power to storage are exposed as a resource over the Internet. Clients dynamically align their infrastructure per their needs, while resources are provided on demand. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage deployed applications, and limited control of select networking components (e.g., host firewalls).

Platform as a Service (PaaS) model delivers a pre-built application platform to the client; consists of application development platforms, remotely accessible through the web and able to connect to locally execute frameworks and IDEs, allowing fast development and deployment of applications. The consumer has control over the deployed applications and possibly application hosting environment.

Software as a Service (SaaS) provides software solutions, allows providers to expose stand-alone applications, running on a distributed cloud infrastructure completely hidden from customers, as resources through the Internet. The consumer

mange for limited user-specific application configuration settings (Zhang et al. 2010).

2.2.3 Cloud Deployment Models

A deployment models defines the purpose of the cloud and the nature of how the cloud is located. Acording to the NIST definition, four deployment models are defined , and described below (Mell & Grance 2011; Ghanam et al. 2012; Ardagna et al. 2012; Zhang et al. 2010).

A. Public Cloud

A cloud in which service providers offer their resources as services to the general public. Known as external cloud and describes the conventional meaning of cloud computing: scalable, dynamically provisioned, often virtualized resources available over the Internet from an off-site third- party provider, owned by an organization selling cloud services.

B. Private Cloud

Referred as corporate internal Cloud. Used to denote a proprietary computing architecture, providing hosting services for private networks. Designed for exclusive use by a single organization. A private cloud may be built and managed by the organization or by external providers. A private cloud offers the highest degree of control over performance, reliability and security.

C. Community Cloud

The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g., mission, security requirements, policy, and compliance considerations).

D. Hybrid Cloud

Compose of two or more clouds mentioned above (private, community, or public) that remain unique entities but are bound together. A hybrid cloud environment combining resources from both internal and external providers to become the most popular choice for enterprises. In a hybrid cloud, part of the service infrastructure runs in private clouds while the remaining part runs in public clouds. Hybrid clouds offer more flexibility than both public and private clouds. Specifically, they provide tighter control and security over application data compared to public clouds, while still facilitating on-demand service expansion and contraction (Zhang et al. 2010).

Table 2.1 Public Versus Private and Hybrid Cloud Computing Services (Goyal 2014).

	Public cloud	Private cloud	Hybrid Cloud
Characteristics	Available to users from a third-party provider, made available via the Internet and may be free or inexpensive to use, such as Amazon Web Services(AWS), providing services across open, public networks.	Provide many benefits same as "public", but managed within the organization. Are unburdened by network and availability issues, or by the potential security offer, associated with public clouds.	Benefits for risks, ICT functions (e.ge, email, document production, or business application runtime), handled in lower-cost public clouds. Functions such as data storage or mission-critical business applications may be kept in-house.
Benefits	Provide best practices, key benefit: greater elasticity and lowest cost.	Private clouds can offer the provider and user greater control, security, and resilience than public clouds.	Hybrid clouds offer greater architectural flexibility. Key benefit: offers greater business choice and avoids all-or-nothing approach.
Risks	Greater risks in terms of security, resiliency, transparency, and performance.	Potentially less risk security, resiliency, infrastructure, and support processes will not differ from the current environment.	Risks and costs fall between public and private models.

Table 2.2 A comparison: Private cloud vs Public cloud vs Hybrid cloud.

Deployment models	Public cloud	Private cloud	Hybrid Cloud
Cloud environment	Multi-Tenancy-Shared environment.	Single tenancy-only for single use of an organization.	Both single tenancy and multi – tenancy. Data stored delivers a multi - tenant environment, data from multiple organizations is stored in a shared environment, whereas when data is stored in a private cloud, it is kept private for the use of a single organization.
Data center location	Anywhere – where the cloud service provider's services are located.	Inside the organization's network.	Inside the organization's network for private cloud services as well as wherever service provider's services are there for public cloud services.
Resource sharing	Server hardware, network and storage are shared by multiple users in the cloud.	No sharing of resources. Hardware, storage and network are dedicated to the use of a single client.	Very secure; integration options, add an additional layer of security.
Cloud storage	Delivers storage as a service on a pay per use. Best for backups for disaster recovery. E.g. One Drive.	Delivers internal cloud storage that runs on a dedicated infrastructure in a data center.	Manages streamlined *storage* that uses both local and off-site resources and serves as a gateway between on premise and public cloud storage.
Scalability	Instant and unlimited.	Sacrifices scalability, but provides greater control and security.	On demand unlimited resources.
Pricing structure	Charged on the usage basis.	Comparatively expensive.	High but delivers competitive advantage.
Cloud Security	Good, but depends on the security measures of the service provider.	Most secure.	Secure.
Performance	Low to medium.	Very High.	Very High.

2.2.4 Cloud Computing Characteristics

The essential cloud characteristics highlight as follows:-

- **On-demand Self Service:** a consumer can provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.
- **Broad Network Access:** capabilities are reachable over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- **Resource Pooling:** provider's computing resources are pooled to serve multiple consumers using a multi-tenant model by assigning the physical and virtual resources according to consumer demand. There is a notice of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources, but may be able to specify location at a higher level of abstraction.
- **Rapid Elasticity:** capabilities can be rapidly and flexibly provided, in some cases automatically, to quickly scale out, and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be limitless and can be purchased in any quantity at any time.
- **Measured Service:** cloud systems automatically control and optimize resource used by supplying a metering capability at some level of abstraction appropriate to the type of service. Resource usage can be monitored, controlled, and reported for both the provider and consumer of the utilized service by providing transparency (Mell & Grance 2011).

2.2.5 Cloud Computing Challenges & issues

Cloud computing has been widely adopted, benefits such as agility, elasticity, availability, and cost efficiency requires software engineered for cloud platforms. However, due to the concept's recent development, is still at an early stage. Many open existing issues surround the use of software services from the cloud (Grundy 2012) have not been fully addressed, while new challenges keep emerging from applications (Zhang et al. 2010).

In this section below, we summarize some of the research challenging issues in cloud computing of meeting the requirements for cloud computing architecture , also the challenges of allowing applications and development platforms to take advantage. Some research issues are given below:-

- **Lack of Service Level Agreements (SLA's):** that allow several instances of one application to be replicated on multiple servers if need arises; A big challenge for the Cloud customers is to evaluate SLAs of Cloud vendors (Kuyoro et al. 2011). Currently, SLA prevents wide adoption of cloud computing. Cloud computing infrastructure services such as EC2 are not yet able to sign the SLA needed by companies that want to use cloud computing for serious business deployment (Tsai et al. 2010; Jamshidi et al. 2015). Moreover, business is dynamic. Static SLA is not able to adapt to the changes in business needs as cloud computing promises.
- **Cloud Data Management & Security:** cloud data can be very large, unstructured or semi-structured, and typically append-only with rare updates. Data must rely on the infrastructure provider to achieve full data security.
- **Migration of virtual Machines:** various programs may run on one machine using virtualization or many machines may run one program. Migration lacks the agility to respond to workload changes. Moreover, the in memory state should be transferred consistently and efficiently, with integrated consideration of resources for applications and physical servers (Kuyoro et al. 2011).
- **Lack of Multi-tenancy supports:** Multi-tenancy can support multiple client tenants simultaneously to achieve the goal of cost effectiveness. Currently, one has three types of multitenancy enablement approaches: virtualization, mediation and sharing (Jamshidi et al. 2015). There are multiple types of cloud applications that users can access through the Internet, that have increased security requirements based on the type of data being stored on the software vendor's infrastructure. These application requests require multi-tenancy for many reasons, the most important is cost. Multiple customers accessing the same services, may affect response times and performance for other customers.
- **Novel cloud application architectures:** most of the clouds are implemented in large data centers and operated in a centralized fashion. Although this design achieves economy-of-scale and high manageability, it also comes with its

limitations such high energy expense and high initial investment for constructing data centers.

- **Quality Assurance attributes:** are met for cloud services, such as performance, scalability, security, and availability; Cloud performance can vary at any point in time. Elasticity may not ramp at desired speeds. Given the criticality of many business applications, analytical techniques are needed to predict QoS and to reason on software system properties at design-time, but also run-time mechanisms and policies able to provide end-to-end quality(Ardagna et al. 2012);
- **Risk Management:** there are several concerns such as payment models, security, legal and contractual, quality and integration with the enterprises architecture and culture. Thus, proper tools to support such choice could be beneficial and limit serious financial consequences for a Small and Medium Enterprise (SME). However, while risk management presents only tools and decision support methods exists to support selecting and binding to a specific target Cloud, or taking a decision to move from Cloud to Cloud in case requirements or services change. Binding to a specific target Cloud can decrease project failure risks, and not supported neither on the design nor in the run-time level (Ardagna et al. 2012);
- **Platform Management:** platform challenges in delivering middleware capabilities for building, deploying, integrating and managing applications in a multi-tenant, elastic and scalable environments. One of the most important parts of cloud platforms provide various kind of platform for developers to write applications that run in the cloud, or use services provided from the cloud, or both. This new way of supporting applications has great potential, such as development for what that application needs already exists (Kuyoro et al. 2011). In addition important issue is migrating from one vendor to another based on changing needs, i.e., the problem of vendor lock-in (Ghanam et al. 2012).
- **Open standards and Interoperability:** most research on issues and challenges with cloud computing recognize interoperability as a major adoption barrier because of the risk of a vendor lock-in. Amongst the many problems being discussed are: the lack of standard interfaces and open APIs, and the lack of open standards for VM formats and service deployment interfaces. These issues result in integration difficulties between services obtained from different cloud

providers as well as between cloud resources and internal legacy systems (Ghanam et al. 2012).

- **Application Portability:** for application portability, the biggest challenges are to build applications for PaaS platforms. For IaaS cloud services, there are in practice a number of standards that enable portability of applications, while PaaS platforms can vary widely between different providers (Council 2014). A major issue is the current difference between the vendor approaches, and lack of portability and interoperability between cloud platforms at different service levels, affecting the cloud in several ways, for developers application increases application design and operational requirements need to react in real time throughout the application lifecycle and insure quality need (Di Martino, Cretella, et al. 2015).

2.2.6 Cloud Applications

Applications today are often composite, multi tier applications, consisting of application components such as user interfaces (UIs), services, workflows and databases as well as middleware components such as application servers, workflow engines and database management systems. When moving such a composite application into the cloud, decisions must be made about putting which tier and even which component of such an application to which cloud drivers for these decisions include functional properties of a cloud such as the possibility to run a specific required middleware and nonfunctional properties such as data privacy, cost and offered quality of service by a specific cloud provider. Effectively, moving an application to the cloud is a rearrangement of the application's deployment topology in which component dependencies are captured.

Cloud computing supports some application features better than others. To determine Whether application will port successfully, should perform a functionality mapping exercise. The Process involves determining the critical application features and then matching them to the cloud provider offering to see if those features can be supported. The criticality of applications needed to predict Quality of Services and to reason on software system properties at design time, also run time mechanisms and policies able to provide end to end quality.

The need for developers to be able to design their software systems for multiple Clouds and for operators to be able to deploy and redeploy these systems on various Clouds based on the convenience (Ardagna et al. 2012).

2.3 Cloud Application Architecture Development

The development of cloud computing technology is an important issue need more attention. Software Architecture (SA) and models are helpful for describes systems structure and its major components (Naumann et al. 2015). Software architecture and design reflect multiple contributions.

There is a need for developing system architecture and application development environments that can simplify and benefits from cloud adoption to improve the tasks, change in clouds for the application provisioned to compute the best software and hardware configuration to ensure the quality of services (QoS), without compromising the efficiency and utilization of whole system.

2.3.1 Cloud Computing Reference Architecture

While there is confusion about the right definition of cloud computing, Cloud Computing Reference Architecture (CCRA) provides a technical blueprint for a system with a well-defined scope, the requirements it satisfies, and the architectural decisions, also ensures consistency and quality across the development (Liu et al. 2011). The CCRA is a natural extension to the NIST cloud computing definition, is a powerful tool used for discussing the requirements, structures, and operations of cloud computing. It defines a set of actors, activities, and functions that can be used in the process of developing cloud computing architectures, as well as provide guidelines for creating a cloud environment, as shown in Figure 2.2.

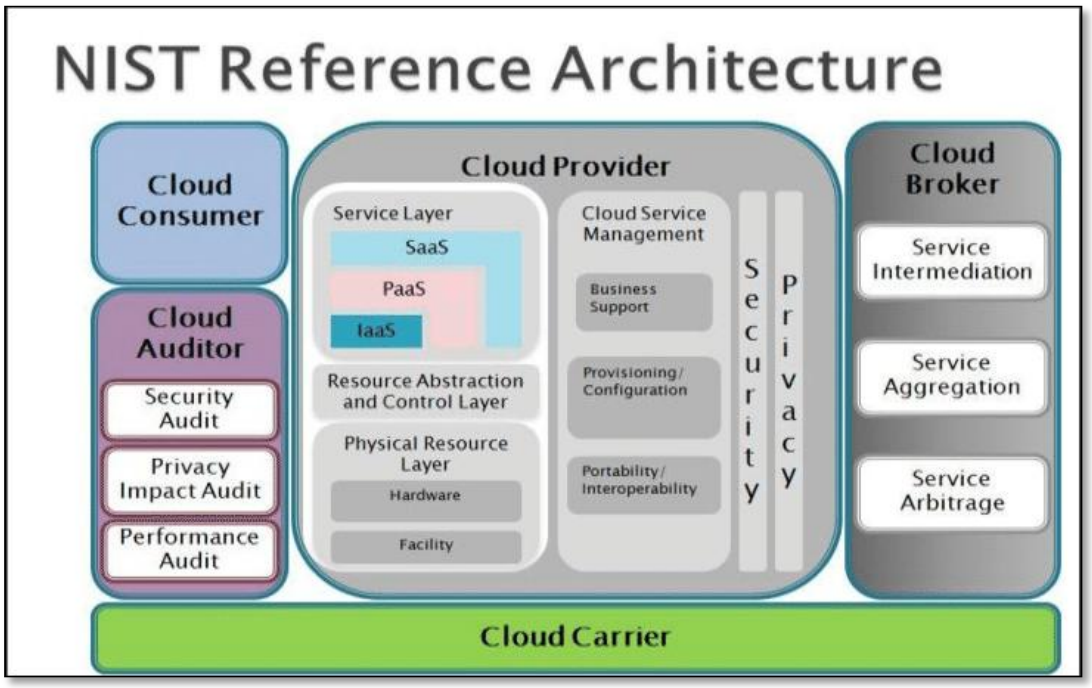


Figure 2.2 The NIST Cloud Reference Architecture (Liu et al. 2011).

The NIST cloud computing reference architecture defines five major actors: *cloud consumer*, *cloud provider*, *cloud carrier*, *cloud auditor* and *cloud broker*. Each actor is an entity (a person or an organization) that participates in a transaction or process and/or performs tasks in cloud computing (Liu et al. 2011), as shown in Table 2.3, breifly list the actors in the NIST reference architecture.

Table 2.3 Actors in Cloud computing.

Actor	Definition
Cloud Consumer	A person or organization that maintains a business relationship with, and uses service from, Cloud Providers.
Cloud Provider	A person, organization, or entity responsible for making a service available to interested parties.
Cloud Auditor	A party that can conduct an independent assessment of cloud services, information system operations, performance and security of the cloud implementation.
Cloud Broker	An entity that manages the use, performance and delivery of cloud services, and negotiates relationships between Cloud Providers and Cloud Consumers.
Cloud Carrier	An intermediary that provides connectivity and transport of cloud services from Cloud Providers to Cloud Consumers.

2.3.2 Cloud Service Management

Cloud Service Management includes all of the service-related functions that are necessary for the management and operation of those services required by or proposed to cloud consumers (Liu et al. 2011).

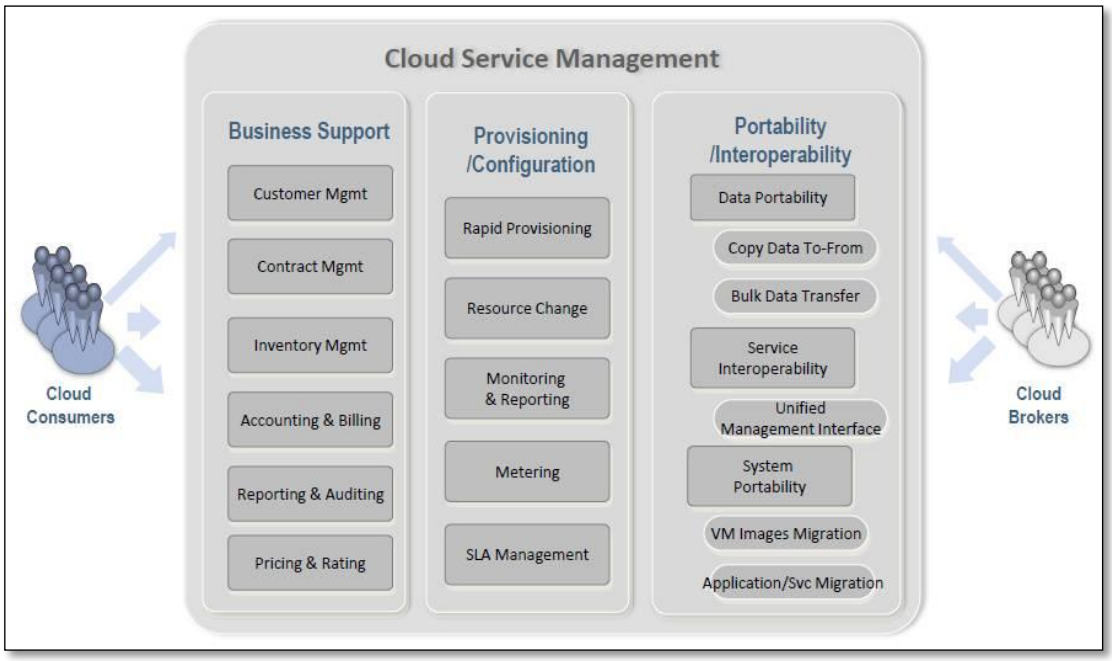


Figure 2.3 Cloud Service Management (Liu et al. 2011).

As shown in Figure 2.3, cloud service management can be described from the perspective of business support, provisioning and configuration, and from the perspective of portability and interoperability requirements. However, the CCRA focuses on the requirements of what cloud service providers, not does not represent the system architecture of a specific system; instead, it is a tool for discussing, and developing the system-specific architecture (Hogan et al. 2011). Moreover, the role of a "cloud service developer" has not been included in NIST CCRA.

2.3.3 Software Architecture for cloud applications Development

The development of a software system involves a large number of design decisions that eventually lead to an executable specification of its behavior, typically in the form of source code. For a long time, it has been realized that, next to behavior, it pays off to be also concerned with a software system's structure and organization for reasons of dependability, understandability, and maintainability.

Therefore, for large systems, these design decisions not only consider the *behavior*, but also the *structures* of the software system. The key principles on which the design of software architectures is based are separation of concerns [Dijkstra, 1974] and abstraction. For complexity of software systems, multiple levels of abstraction are necessary to ensure designs remain comprehensible (Graaf 2007). This gives rise to several types of design. Two levels of design are detailed design involve the decisions related, and higher level of abstraction, design is called software architecture design, which is the primary key of this thesis.

2.3.3.1. Software Architecture Definitions

It is difficult to capture the notion of software architecture in a single definition. The software architecture is described in terms of components and connectors, which can be deployed to distribute configurations (Shin & Gomaa 2007). A software architecture is a description of the subsystems and components of a software system and the relationships between them. A software architecture is subsystems and components are typically specified in different views to show the relevant functional and non-functional properties of a software system. The software system is an artifact. It is the result of the software design activity.

A more recent definition of software architecture can be found in IEEE- 1471 [2000]: *The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.*

An alternative definition that is frequently used is given (Bass et al. 2003): *The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them* (Hutchinson 2011).

The software architecture developed several methods and techniques to support the architectural design process. One of the key differentiating aspects of the design methods developed by the SA researchers and practitioners is that they elevate ASRs from being almost totally ignored to being an important consideration during SA design. Each of architecture-centric design methods has its strengths and weaknesses. One way of leveraging their strengths and overcoming weak points is to select different approaches and techniques from different methods and apply them based on contextual requirements.

2.3.3.2. Software Architecture Goals

The primary goal of the architecture is to identify requirements that affect the structure of the application. A well-laid architecture reduces the business risks associated with building a technical solution and builds a bridge between business and technical requirements (Witt et al. 1993). Some of the other goals are as follows:

- Expose the structure of the system, but hide its implementation details.
- Realize all the use-cases and scenarios.
- Try to address the requirements of various stakeholders.
- Handle both functional and quality requirements.
- Reduce the goal of ownership and improve the organization's market position.
- Improve quality and functionality offered by the system.
- Improve external confidence in either the organization or system.

2.3.3.3. Software Architecture Limitations

Software architecture is still has the following limitations:-

- Lack of tools and standardized ways to represent architecture.
- Lack of analysis methods to predict whether architecture will result in an implementation that meets the requirements.
- Lack of awareness of the importance of architectural design to software development.
- Lack of understanding of the role of software architect and poor communication among stakeholders.
- Lack of understanding of the design process, design experience and evaluation of the design (Witt et al. 1993).

2.3.3.4. Software Architecture Design

The architecture design (AD) is typically the most difficult task an architect undertakes, can be called the 'solution phase' of the life cycle because it defines the software in terms of the major software components and interfaces (Gorton 2006). The 'Architectural Design' must cover all the requirements in the software requirements design (SRD). The goal of software architecture design is to define the constraints for dependent design and implementation activities that result in the development of a system that fulfills its functional and other quality goals.

The design stage has two steps, which are iterative in nature. The first involves choosing an overall strategy for the architecture, based around proven architecture patterns. The second involves specifying the individual components that make up the application, showing how they fit into the overall framework and allocating them responsibilities. The output is a set of architectural views that capture the architectural design, and a design document that explains the design (Gorton 2006).

2.3.4 Model-based Approaches for Cloud Application Development

As the demand on software system increases, new software methodologies and techniques are growing to build reliable software, reduce the development effort, and produce high quality software. Reusing software becomes the main concern of modern software design methods. Different strategies have been proposed to utilize the notion of software reuse during different software engineering stages, but our interest is on the most common and influential two approaches, models and patterns.

Models are used to predict system properties, changes in some parts that will affect the rest of a system, and communicate key system characteristics to various stakeholders. The models developed as a paradigm or blueprint prior to implementing the physical system, or they may be derived from an existing system or a system in development, as an aid to understanding its behavior (Karakostas 2008), e.g. Model driven architecture (MDA); and cloud design patterns. The main benefits of MDA from a cloud prespective are the facilitation of portability, interoperability, and reusability of parts of the system that can be easily moved from one platform to another, as well as the maintenance of the system through human-readable and reusable specifications at various levels of abstraction. In the context of cloud computing, model-driven development can be helpful in allowing developers to design a software system in a cloud and to be supported by model transformation techniques in the process of instantiating the system into specific and multiple clouds (Di Martino, Cretella, et al. 2015). The two different techniques are introduced and explained in more details in the following section.

2.3.4.1. Model Driven Architecture (MDA)

Traditional software design and development processes create applications for deployment to a specific technology platform, MDA is a model-based approach to development of software systems, that introduces higher levels of abstraction, enabling organizations to create models that are independent of any particular technology platform.

Model Driven Architecture (MDA) is a new paradigm of software development aimed at raising the abstraction and re-use levels. MDA is adopted as standard by The Object Management Group (OMG) as an attempt to develop applications in establishing domain without entirely writing new code (Gavras et al. 2004). Model driven architecture (MDA) is a recent re-using approach in the software development technique.

MDA Definitions

There are a number of definitions used to describe the use of MDA in software development, are:-

Definition 1: The MDA is an Initiative proposed by the Object Management Group (OMG), is an open, vendor neutral, approach to software development which is characterized by the use of models as the primary artifacts of understanding, design, construction, deployment, operation, maintenance and modification of a system. It reflects separation of concerns by separating business functionality from the implementation technology (Wojcik et al. 2006).

Definition 2: Model Driven Architecture (MDA) *is a software development approach where the models are used as prime artifacts throughout the process of software development*. These models are defined at different levels of abstraction to represent various aspects of the system (Kleppe et al. 2003). Besides, they are formal in nature and can be machine processed. The transformation of models from one level of abstraction to another, or the transformation of models to executable code is performed by using (semi) automated transformation tools (Wojcik et al. 2006; Sharma & Sood 2011).

Definition 3: Is the Attribute Driven Design (ADD) method *is an approach to defining a software architecture in which the design process is based on the software quality attribute requirements,* follows a recursive process that decomposes a system

or system element by applying architectural tactics and patterns that satisfy its driving requirements (Wojcik et al. 2006).

MDA Models

A model is useful if it helps to gain a better understanding of the system. In an engineering context, a model is useful if it helps deciding the appropriate actions that need to be taken to reach and maintain the system's goal.

Models of software requirements, structure and behavior at different levels of abstraction help all stakeholders deciding how this goal should be accomplished and maintained transformations (Mens & Van Gorp 2006).

Models are specified at different levels of abstraction, tools are used for transformations between the levels (model to model and model to code).

MDA defines software application at three different abstraction levels:-

1. **Computation Independent Model (CIM):** specify the **c**omputation independent view of the system, capture system requirements in a vocabulary familiar to domain practitioners. Software independent business domain model that bridges the gap between business experts and system experts.
2. **Platform Independent Model (PIM):** specify the system at the next lower level of abstraction; capture a platform independent view, focusing on the operation of the system while hiding specific details. Specifies the functionality of the system independent of the technology that would be used for its implementation.
3. **Platform Specific Model (PSM):** defined at the next lower level of abstraction, focusing on the details the use of a particular platform, providing a platform specific view of the system. Specifies the system in terms of implementation constructs that are specific to the implementation technology.

A single PIM can be transformed into one or more PSMs (Sharma & Sood 2011), as shown in figure 2.4. Each PSM is specific to the technology platform on which the system would finally be implemented (Kleppe et al. 2003)**.**

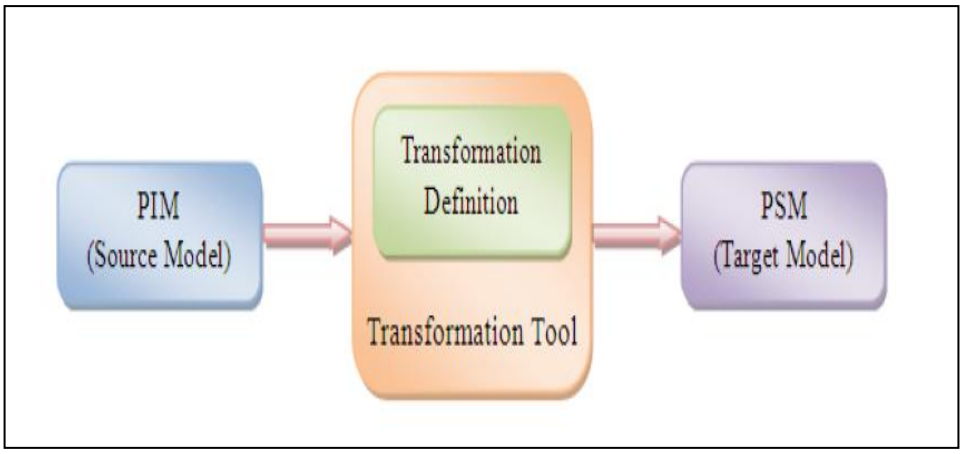


Figure 2.4 PIM to PSM models (Sharma & Sood 2011).

The key to the success of MDA lies in automated (model-code) or semi-automated models (model-to-model). Automatiomated model genertae architecture views from source code (Haitzer & Zdun 2014), while semi-automated approach demand identified tool for software development process to reduce the efforts that manual approach (Pilar et al. 2014). The transformation tool executes a transformation definition that is specified for the purpose of transforming higher-level, platform independent business models into lower level platform specific models and finally into executable code. A transformation definition is a set of transformation rules that together describe how a model in the source language can be transformed into a model in the target language (Sharma & Sood 2011). The important concerns the source and target artifacts of model transformation, if artifacts are programs (i.e., source code, byte code or machine code), we use the term program transformation, and if software artifacts are models, we use the term model transformation (Mens & Van Gorp 2006).

MDA for Clouds

MDA architecture used for Design, provisioning, execution, or migration to the Cloud, is a model-based approach for the development of software systems that aims at separating the platform-independent design of a software application from its implementation on a given platform. From the cloud perspective, the main feature and benefits of MDA are the enablement of portability, interoperability, and reusability of (parts of) the system, as well as its easy maintenance, through human-readable and reusable specifications at various levels of abstraction (Di Martino, Cretella, et al. 2015). Model-driven development in the context of cloud computing,

allows developers and enterprise architects to design software systems in a cloud-agnostic manner. Particularly relevant in designing and managing applications across multiple clouds, as well as migrating them from one cloud to another. Combining MDA in the cloud computing domain is currently the focus of several research groups and projects, such as MODAClouds (Ardagna et al. 2012), ARTIST (Troya Castilla et al. 2015) and REMICS.

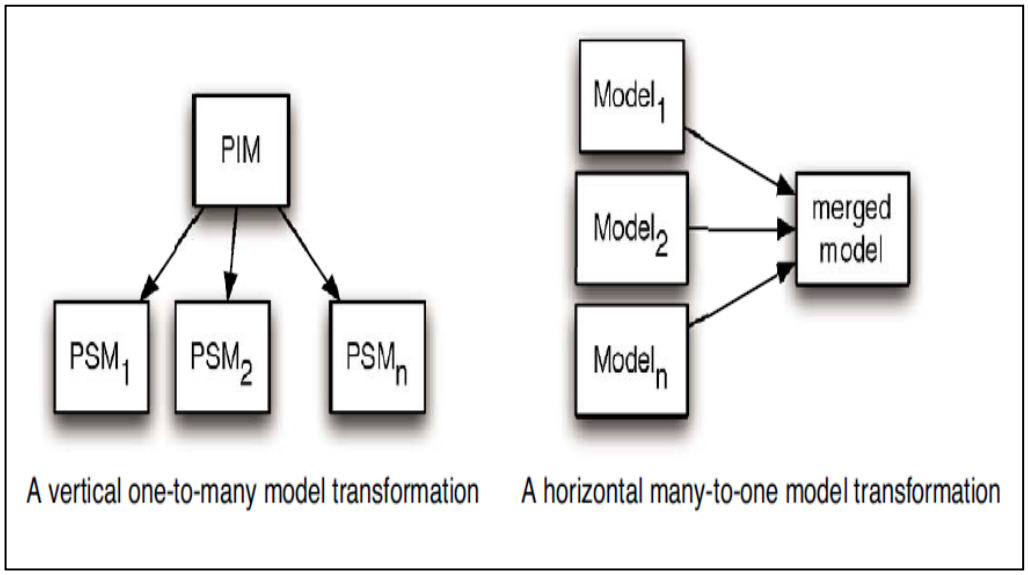


Figure 2.5 An example of the transformation that takes single(PIM) for cloud software application mapped to several (PSM) based on defining rules (Sharma & Sood 2011).

While traditional software design and development processes create applications for deployment to a specific technology platform, MDA introduces higher levels of abstraction, enabling organizations to create models that are independent of any particular technology platform. The strength of MDA lies in the fact that it is based on widely used industry standards for visualizing, storing and exchanging software designs and models.

2.3.4.2. Cloud Design Patterns

Cloud computing patterns are logical descriptions of the physical and virtual assets that comprise a cloud computing solution (Opara-Martins 2017), capturing best practices on how system applications should be designed. The cloud patterns benefits development of cloud application architecture (Adewojo et al. 2015),

describe how different application components can be integrated to provide architecture design solution (Yacoub & Ammar 2001), that can be deploying on clouds.

Cloud patterns arise from the need to provide both general and specific solutions to recurring problems in the definition of the architectures for cloud applications. While classical design patterns deal with problems related to different aspects of software development, cloud patterns mainly focus on the architecture of the cloud application. Despite the poor flexibility of any vendor-specific patterns, cloud patterns still represent a valuable means to enhance portability and interoperability between cloud platforms. Patterns can be used to describe and model existing cloud applications in a very easily understandable manner, tracing back cloud implementations to a set of well-known and stable solutions. In this way, it becomes easier to understand the exact functionalities and responsibilities of a specific cloud application component, which can later be substituted with a compliant one having the sam+e or similar characteristics. This approach can be exploited also in the case of porting non-cloud applications (i.e. Traditional enterprise applications), describable through classic design patterns, to a cloud environment, provided a mapping between design and cloud patterns' participants exists (Di Martino, Cretella, et al. 2015).

2.4 Related work

Modeling of cloud-based applications is a new research topic, we present our method based on Model Driven Architecture (MDA) as the most important paradigms, because the MDA approach becomes an evident choice for ensuring software solutions developing cloud applications that are robust and flexible. The main objective of this section, is to present a survey analysis on crosscutting concerns within the MDA. We analyze the current published research on MDA with respect to cloud application development. We first classify the research work accordingly to literature selection, Data filtering and analysis. We discuss some open issues and challenges that need further research in developing cloud applications based on MDA, such as cloud application portability.

2.4.1 Literature Selection

In this section we survey related work and determined different ways to compare achievements in development methodologies for cloud applications. We present a taxonomy of eight categories, according to more cloud uses, as shown in the Table 2.4.

Table 2.4 Cloud uses support by MDA taxonomy.

#	Category	Purpose	Papers No
1	Interoperability	• The ability of computer systems to access, and exchange resources with one or more other performers and to use resources to accomplish its performed activities according to expected criteria.	3
2	Deployment	• Help developers to be able design their software systems for multiple clouds and for operators to deploy and re-deploy these systems on various clouds.	3
3	Development	• Development of software to mitigate unfavorable effects of technology changes.	4
4	Evolution	• Used as a process of developing software as a service initially on the basis of some requirements, also to model requirements from iterations to be evolving. • Used to improve dynamic cloud service in a heuristic manner with healthiness validated.	3
5	Quality attributes	• Multi tenancy helps to determine the number of resource provisioning to meet Service Level Objectives.	3
		• To run and manage multi-cloud systems, allows cloud solution that optimizes the performance, availability and cost of the applications.	1
		• Achieves Reliability and scalability	1
		• Achieves Resource Scalability &Provisioning	2
6	SOA	• Deliver services to other users or other services; created to satisfy business goals, using web services to handle communications.	3
7	Migration	• Provide a holistic view to inform decisions when migrating to clouds. • Benefits organizations to select efficient transition architectures to increase productivity and reduce complexity.	3
8	Evaluation	• To analyze the impact of cloud adoption to identify potential risks and verify that the quality requirements have been addressed in the design, also to determine the robustness of systems.	1

Recently, many researchers presented in different studies approaches to use MDA to develop cloud applications.

Hugo & Manu Sood (2010) explore the interaction between service oriented engineering and model driven engineering. The presented modeling as a service (MaaS) allows the deployment and execution for services on the cloud (Bruneliere et al. 2010), however, model driven used as part of service oriented architecture (SOA) (Jamshidi et al. 2015), but there is no general agreement on the right set of models, languages, model transformations and software processes in the model driven development of SaaS systems. Furthermore, legacy system needs to evolve and be adapted to be executed as a service.

Frey and Hasselbring (2010) presents a framework to facilitate the migration of legacy software to the cloud. The steps begin from existing legacy systems, extract the actual architecture, then use a Meta-model to generate the target model to system migration (Frey & Hasselbring 2010a). A reference model starts from the cloud platform to extract elements and vocabulary to create the cloud Meta-model. However, The model needs to refine the syntax and create a platform independent modeling language for cloud applications.

Mohammad Hamdaqa et al (2011) presents a model driven approach for building cloud application solutions. The proposed approach presents a Reference model (Meta model) that facilitates cloud application development from the design to implementation without depending on specific PaaS or IaaS components. This approach can be used by developers to better understand cloud applications independent of any specific cloud development environment. Moreover, the approach can improve developers to select a cloud vendor before porting the legacy application to the cloud.

Ritu & Manu Sood (2011) implement the MDA approach for the development of Online Hotel Reservation System (OHRS) that runs as services in the cloud. An approach based on levels of abstraction, enabling to create models that are independent of any particular technology platform(Sharma & Sood 2011). However, the requirement model for Cloud Independent Model (CIM) level proposed to produce the required systems, but in actual practice the requirements model needs to be refined into a computational model to generate the process. However, the platform independent model (PIM) of the OHRS describes the attributes and operations in a mode that is entirely independent of any programming

language or operating system in which the system would finally be implemented. However, the service useful for building utilized by scale enterprise, but do not develop the software for the purpose.

FrancescoMoscatoe.tal (2012) their method uses Model driven Engineering and Model Transformation Techniques to analyze services, focused on using ontology to build modeling profiles that help to analyze complexity of systems, by developing open source platform that enables applications to negotiate cloud services requested by users via interface and targeted platform for developing multi cloud applications (Moscato et al. 2012). One of the main goals is to enable access of heterogeneous cloud resources and to avoid locked-in problems. However, using an ontology achieves interoperability, but requires addition efforts for other quality services.

Lin and Chang (2012) their approach goals is to achieve architecture design for evaluating the system performance and determine its robustness based on measuring the system reliability of a cloud computing system (CCS) (Lin & Chang 2012).

Ardagna et al. (2012) implemented model driven approach for the design and execution of applications on multiple Clouds (MODACLOUDS) that aims to support system developers and operators in exploiting multiple Clouds and in migrating systems from cloud to cloud as needed (Ardagna et al. 2012). They presented framework used for developing and deploying applications in multi Clouds. In addition, to enable risk analysis for the selection of Cloud providers, and for the evaluation of the cloud impact on internal business processes. Furthermore, the work offers a runtime environment for observing the system under execution and for enabling a feedback loop with the design environment that allows system developers to affect performance change and to redeploy applications on different Clouds. However, there are many challenges such as vendor lock-in (Antoniades et al. 2015) , applications portability, and cloud data migration are not addressed. In addition, Risk Management uses primitive tools. However, need to offer Cloud providers for auto scaling mechanism for interoperability and federation between Clouds.

Nicolas Ferry et al. (2013) proposed Cloud Modeling language (CloudML) explains that model driven techniques and methods facilitating the specification of

provisioning and deployment concerns of multi cloud systems, this will enable the continues evolution of system between design time and run time activities (Ferry et al. 2013), argue model driven is suitable for developing complex systems.

There are several projects that aim at addressing challenges by providing solutions for provisioning, deployment, monitoring and adoption of cloud systems such as Modeling QoS constraints. In the addition time consuming services are identified as a challenge for adaptive systems. Moreover, there is a lack of a systematic engineering process and tools supported by reusable architectural artifacts (Zhang & Zhang 2009).

2.4.2 Summary of Related work

Table 2.5 Summary of using MDA for cloud application development.

#	Author/s	Approach	Advantages	Limitations	Domain
1	**Hugo et al.** **2010**	**Modeling as a Service (MaaS)** to provide modeling and model driven engineering services from the cloud.	• Using Model Driven Engineering (MDE) for the development of SaaS applications. • Using SaaS to deploy modeling services in the cloud.	• No general agreement on the right set of models, languages, model transformations and software processes in the model driven development of SaaS systems.	SaaS application (SOA)
2	**Ritu Sharma & Manu Sood** **2011**	The **MDA approach** an asset facilitates creation of good designs that easily cope with multiple implementation Technologies and drive process of software development.	• MDA approach need in the development of cloud, SaaS in order to minimize the time, cost and efforts in application development and to enhance the Return on development.	• Need an approach for ensuring interoperability among the models of cloud software services.	Web base application (SaaS app)

#	Author/s	Approach	Advantages	Limitations	Domain
3	**Danilo Ardagna et al.** **2012**	**MODACLOUDS, model Driven Approach** for the design and execution of applications on multiple Clouds. Allows early definition and assessment of quality at design time.	• Supporting system developers and operators in exploiting multiple Clouds. • Migrating applications from Cloud to Cloud (performance). • Inform evolution process to design time.	• Vendor lock in on cloud customer to decide on adoption model. • Risk management used primitive tools. • Quality need mechanisms to be able to deploy and redeploy systems.	Business Application
4	**Mohammad Hamdaqa et al.** **2011**	Defining **A Reference Model (Meta model)** for developing cloud application environment.	• The present Meta model shows main cloud vocabulary, design elements, configuration rules and semantic interpretation. • Facilitates cloud application development from the design to implementation.	• The lack of standardization and terminologies challenges portability and migration between different cloud environments.	Cloud Application
5	**Francesco Moscato et al** **2012**	**MOSAIC Ontology methodology and Framework,** aims at creating, promoting, open source (API) and platform for developing multi cloud oriented applications. **Frameworks** enhance modeling profile for verification QoS of cloud services.	• Simple access to heterogeneous resources. • Design interface for users and implemented existing services. • Enable intelligent service discovery. • (QoS) given for users to avoid locked-in and for providers to build on demand services.	• Do not provide approaches to model and verify dependability during all phases of the life cycle. • Difficult to achieve interoperability.	Multi Agent System
6	**Nicolas Ferry et al 2013**	**Cloud Modeling language (CloudML)** aims at facilitating the provisioning, deployment, monitoring and adaptation of multi cloud systems.	• Enables the evolution of system between design time and run time activities. • Enables developers to work at a higher	• Model under development and many challenges identified, such as. • Time consuming development activities for adaptation.	Multi Cloud Systems

			level of abstraction of cloud concerns rather than implementation details.	• Techniques and methods to prevent failure. • Data movements from region to another without legal consequences.	

2.4.3 Result Analysis

We provide a survey analysis on MDA with respect to cloud application development (Ibrahim & Hany 2015). We outline literature selection based on cloud taxonomy issues as shown in Table 2.4. Furthermore, we compare different MDA used for cloud application development as shown in Table 2.5, and discuss some issues and challenges that need further research in developing cloud applications based on MDA as follows:

- There is no generic cloud software architecture for designing and building cloud applications (Hamdaqa et al. 2011), applications with cloud need new technologies (Gohad et al. 2012) to differentiate the cloud development paradigm from the existing ones.
- Developers argue for the need for model driven approaches and supporting tools to facilitate the specification (Ferry et al. 2013) of provisioning, deployment, monitoring and adaptation concerns at design time and at run time. The need to bind configuration management in order to minimize the shortcomings (Gitzel et al. 2007).
- Need to build cloud applications that offer cloud providers for auto scaling mechanism for interoperability between clouds (Ardagna et al. 2012).
- The lack of standardization and common terminologies that challenges portability, also need to migrate application's components between cloud providers as needed (Ardagna et al. 2012).
- Lack of interoperability and portability (Gonidis et al. 2012), the difference between the individual vendor approaches, cause Vendor Lock-in problem. The risk of lock-in is a major concern for cloud customers. Cloud providers, in fact, offer proprietary solutions that force cloud customers to decide, at the early stages of software development the design and deployment models to

adopt (e.g., public vs. hybrid Clouds) as well as the technology stack (e.g., Amazon Simple DB vs. Google Big table). Thus, the portability of applications and data between clouds be addressed (Anderson & Rainie 2010).

- The Quality of applications poses a need for developers to be able to design their applications and for operators to operate, monitor and assure performance change of cloud applications to be able to deploy and redeploy on multi Cloud environments.

The above described effort in the area of applications on clouds, the current literature, indicates that the topic still requires research on new programming abstractions, developing and presents best solutions.

2.5 Summery

In this chapter, the theoretical background for Cloud Computing has been presented. Firstly, the chapter explored Cloud computing architecture. These stages are: cloud definitions, cloud services, deployment models, cloud characteristics, cloud challenges and issues, and cloud application development. Many efforts have been reported for solving the problem of developing cloud application architecture. Different state-of-the-art methods apply different software approaches, techniques and patterns to design and develop cloud application architecture.

Surveyed research work on using model driven approaches for cloud application development are explained, based on model driven architecture (MDA). We covered the major methods, and summarized their features. We also discussed several open research issues, that provide a better understanding the principles and challenges of developing applications in the clouds, that help developers, architects and researchers support, evaluate and predict different methods and techniques.

MDA approaches that promise to reduce the overhead of developing, configuring, deploying and maintaining cloud applications. Moreover, help of architecture that can easily improve the quality of the system thus can develop a product with quality attribute such as portability, interoperability, and reusability of parts of the system that can be easily moved from one platform to another, as well as the maintenance of the system through human-readable and reusable specifications at various levels of abstraction.

CHAPTER III

METHODOLOGY FOR PORTABILITY-BASED DEVELOPMENT OF CLOUD APPLICATION ARCHITECTURE DESIGN

3.1 Introduction

This chapter describes the methodology applied in this research. As Figure 3.1 shows, this research is accomplished through a number of steps. It begins with a study on related works in the literature. Then, problem statement and research objectives are explained. In the next step, cloud application requirements are needed to support the development of portable cloud application architecture. The details of the proposed framework phases are presented. It includes the architecture design phase of the proposed framework. Finally, the evaluation methods of the portability framework are presented.

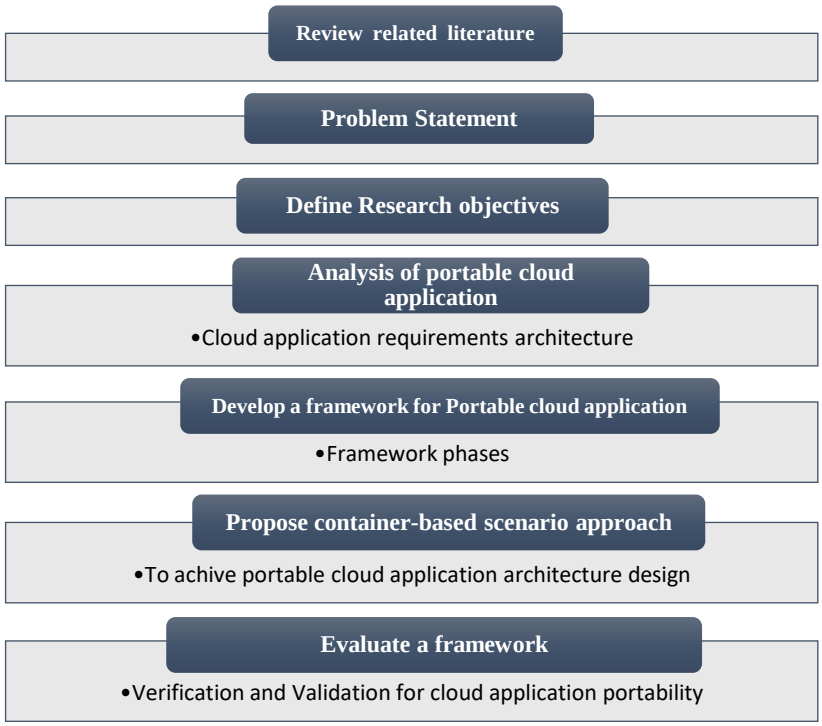


Figure 3.1. Research Methodology.

3.2 Cloud Application Portability Analysis

Chapter 2 presents a background study and extensive literature review about the overview of cloud computing, cloud application, and models for cloud application. In order to achieve the first objective, we need to provide cloud application portability analysis, to help in developing our proposed framework.

Portability defines the ease of ability to which application components are moved and reused elsewhere, regardless of provider, platform, operating system, infrastructure, location, storage, data format, or API's. Portability is based on a set of attributes that design on the ability of software to be transferred from one environment to another (Kolb & Wirtz 2017).

3.2.1 Cloud Portability Definitions

As mentioned before, portability is about the ability to move an entity from one system to another (Council 2014). Cloud portability is a concept that also concerns both the cloud customer and provider to avoid lock-in of cloud offerings (Hill & Humphrey 2010).

Definition 1: Cloud portability is defined as the ability to migrate a cloud-deployed asset to a different provider, and it is a direct benefit of overcoming vendor lock-in (Opara-Martins et al. 2014).

Definition 2: Cloud Application Portability refers to the ability to move any component of any of the three cloud service models across cloud platforms (Gonidis et al. 2012)**.**

Definition 3: Cloud Applications' Portability is a desirable feature for Cloud Developers, prevent an effective utilization of multiple cloud providers' services and offers among cloud platforms, at different service levels (Di Martino, Esposito, et al. 2015).

Cloud Application Portability enables the reuse and migration of entire applications, or of some of their components, across cloud PaaS services or even from on premise environments to the cloud, to minimize efforts when migrating an application, data or service from one cloud to another, which reduce the need for re-designing and rewriting parts of the application.

3.2.2 Cloud Application Portability Adoption

Enterprises and cloud service providers need to adopt cloud portability, because cloud application designs increasingly span multiple cloud providers' platforms, need to ensure designing new applications to run effectively in such "Multiclouds" application architectures while avoiding related portability or vendor lock-in problems (Petcu & Vasilakos 2014). There are many reasons, that let application portability in clouds is an important concern.

- **Economic reasons:** is necessary for several reasons.
  - The customers of the cloud would gain protection over their investments in their application development. Due to the heterogeneity of the cloud, the migration of a software application, along with its data, from one cloud to another, often requires the rewriting of large parts of both the services and the application, all in order to comply with the new environment to which the application is migrating.
  - The cloud service providers interested in enhancing the opportunity for application portability in order to promote their own attractiveness within a highly dynamic and demanding market.
  - Allows the development of third party organizations which would be able to mediate between the cloud customer and multiple cloud providers. Enable deployments depending on the customer's requirements, and all kinds of necessary adaptations would be provided by the third party.
- **Technical reasons:** cloud portability is of immense importance in order to exploit the cloud elasticity to a maximum extent, as well as to ensure the continuity in a given application along with its functionality of a service. Throughout a lifetime of an application, a point might come at which external resources are required from a public cloud, and so it needs to be redeployed from a private cloud. This kind of a porting process is triggered by the cloud customer. Furthermore, in order to achieve continuity, precaution measures such as regular backups must be made, not only of data, but of application instances as well. This implies that an application must be able to use various cloud services which in turn will aid the replication of the application as well as its data. This kind of a porting process is most commonly in accordance with the provider's agreements, and is usually considered in the design phase.

- **Legal reasons:** the laws within a country may change rather abruptly, triggering the need for software to be deployed from one cloud system to another. A cloud service provider may also provide faulty service which in turn would call for immediate redeployment of applications on another cloud platform. In this sense, efficient portability mechanisms are essential for fast recovery. Situations like the aforementioned one are also attempting to be avoided by creating contracts which guarantee a sufficient contingency plans if the cloud provider.

3.2.3 Cloud Application Portability Scenarios

Cloud application portability issues differ according to the target service level (Di Martino, Cretella, et al. 2015), we have identified five major portability scenarios for cloud application, discussed in different scenarios as shown in Figure 3.2.

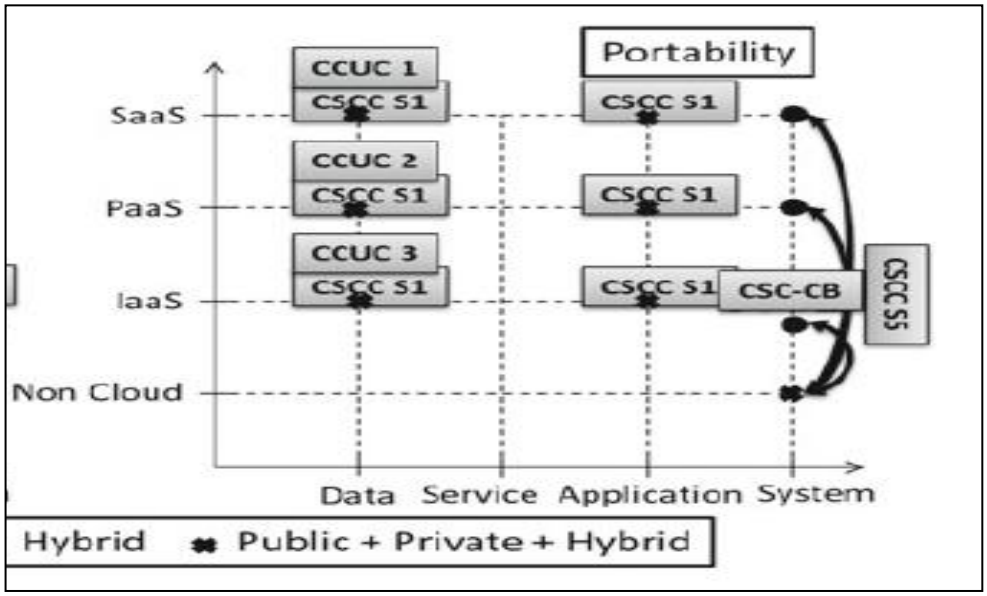


Figure 3.2. Portability scenario classification (Di Martino, Cretella, et al. 2015).

1) CSCC S1: Customer Switches Providers for a Cloud Service

Scenario addresses the case of customers currently using a cloud service provided by provider A, wish to switch to an equivalent service from provider B. They are able to transfer their data or applications within multiple cloud environments with reasonable expense and minimum disturbance (Rezaei et al. 2014). Interoperability and portability are illustrated in Figure 3.2 below.

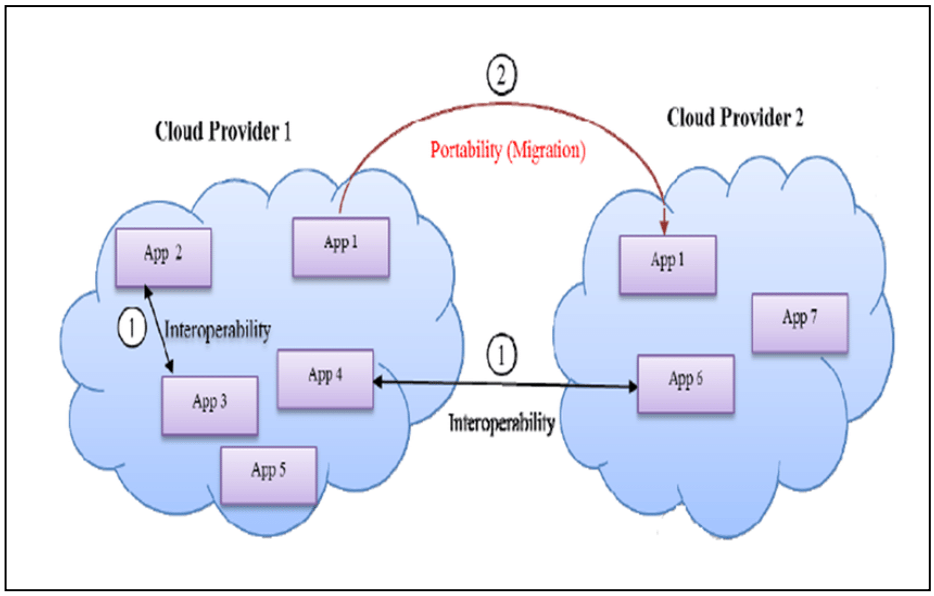


Figure 3.3. Portability of an application between cloud providers (Rezaei et al. 2014). Interoperability refers to the ability of two or more systems or applications to exchange information, whereas portability refers to ability to move an entity from one system to another so that it is usable on the target system.

This scenario touches many of the issues associated with portability, as explained below:-

A. CCUC 1: Cloud Customer Changing SaaS Vendors

   The data handled by one vendor's software should be importable by the second vendor's software, which means that both applications need to support common formats. Standard APIs for different application types will also be required.

B. CCUC 2: Cloud Customer Changes cloud middleware vendors.

   Existing data, queries, message queues, and applications must be exportable from one vendor and importable by the other. The requirement to achieve this porting is a common API for cloud middleware. Cloud database vendors have enforced certain restrictions to make their products more elastic and to limit the possibility of queries against large data sets taking significant resources to process.

C. CCUC 3: cloud customer changing VM Hosts.

   Cloud customer wants to take virtual machines, built on one cloud vendor's system, and run them in another cloud vendor's system. The main requirement of this operation is a common format for virtual machines.

2) CSC-CB: Cloud Bursting

   Focusing on interoperability issues at the IaaS level, describes a scenario where multiple cloud platforms need to work together. Similar to CCUC3.

3) CSCC 5: Migration of Customer Capabilities into Cloud Services.

   This scenario addresses the case of a customer, currently running an application or service on-premise, who wants to move that capability to a public cloud environment (Di Martino, Cretella, et al. 2015).

3.2.4 Cloud Application Portability PaaS Issues

For application portability, the biggest challenges are for applications built for PaaS platforms (Council 2014). We focus on cloud applications to be ported across different platforms, exposed by supporting platform, enabling the application to use services, also providing access to the capabilities that support the application. PaaS portability issues are caused by:-

[1] Lack of a shared platform defined among PaaS providers; each provider chooses the operating system and middleware elements, it will support, if we make different selections then applications using those features cannot be ported.

[2] Lack of alternative providers of a platform; a provider may make a cloud version of a server platform available. Providers have been adding services to their platform, as there are no standards using the added services could lock applications to a cloud provider.

[3] Portability between the development and operational environments (DevOps).

3.2.5 Platform as a Service (PaaS) Portability Solutions

PaaS generally provide mechanisms for deploying applications, designing applications for the cloud, pushing applications to their deployment environment, using services, migrating databases, mapping custom domains, IDE plugins, or a build integration tool (Pahl 2015). The Following are existing solution approaches used to achieve PaaS portability:-

[1]. Adopt container technology, such as Docker, for application hosting (Pahl et al. 2017).

[2]. Involved the definition of a common set of standards that need to be adopted by platform vendors in order to provide uniform services (Gonidis et al. 2012), in addition hrough protocols, used APIs or through abstraction layers which

decouple application development from specific target Clouds (Petcu & Vasilakos 2014).

[3]. Use a cloud technology that deploys PaaS on IaaS or bare-metal servers. Cloud Foundry and Open Shift are examples of these tools, but newer products, such as Morpheus, are gaining attention.

[4]. Service Orchestration: involves microservices architecture to developing a single applicationas a suit of small services. Organizations could take application components that need specialized middleware and make them into services that are called on demand by the rest of the application (Pahl 2015).

3.3 Cloud Application Architecture Design Patterns

Cloud application architecture patterns describe how applications have to be designed to benefit from a cloud environment, also described how applications themselves can be offered as configurable cloud services (Zhao et al. 2012). In addition pattern approach provide solutions to cloud application challenges and requirements (Erl et al. 2015), guide application developers during the design and implementation of applications components that use cloud offerings and are deployed to different cloud types. Benefit cloud application development, reduce development time, deploy multiple applications, test, configure and integrated solutions (Brandle et al. 2014).

3.3.1 Fundamental Cloud Application Architecture Design patterns

The following section addresses design patterns taxonomy for cloud application architecture (Ibrahim & Eldein 2016) as in figure3.3. Based on (Fehling et al. 2014) cloud design patterns based taxonomy as follows phases, which describe abstract solutions to problems in how cloud application can build on top of an elastic platform. We cover the fundamental architectural styles that architects and developers have to be aware of when building a cloud native application categorize to:-

A. **Loose Coupling Architecture:** communication separates application functionality from concerns of communication partners regarding their location, implement the platform, the time of communication, and the used data format.

By using Broker solution that can communicate components and multiple integrated applications to decouple from each other's.

B. **Distributed Application Architecture:** describe how the application's functionality may decompose to be distributed among several resources. Cloud application solutions rely on dividing provided functionality among multiple application components that can be scaled out independently, redundant resources ensure that the unavailability of one resource does not affect the application as a whole.

C. **Cloud Application component patterns:** patterns of this category refine how the functionality of a cloud application can be implemented in separate components. Applications components are developed specifically for cloud offering and requirements because they are not explicitly specified (Fehling et al. 2012). Cloud component patterns characterized by three central patterns.

- **User interface components:** provide application functionality to users.
- **Processing components:** handle computational tasks.
- **Data access components:** handle data stored in storage offerings. They can deal with storage offerings at different cloud providers with different consistency levels. Data access components can further be adjusted to inherently support eventual consistency by abstracting data to hide that there may be data inconsistencies.

D. **Multi-tenancy patterns:** Multi Tenancy *"refers to the capability to host a single instance of a software solution that serves multiple clients.*". Describe how cloud applications and individual components can be shared by multiple customers on different levels of the application stack. Can support multiple client tenants simultaneously, that achieves the goal of cost effectiveness.

- **The shared component**: provides functionality to different tenants without maintaining a notion of tenants itself.
- **The tenant-isolated component**: does the same, but ensures that tenants do not influence each other while they access shared functionality.
- **The dedicated component pattern**: enables some functionality to be provided exclusively to tenants without sharing it with others.

E. Cloud Integration patterns: describe special application components to enable the communication across cloud boundaries, as applications are often not

standalone and must be integrated with other cloud applications and non-cloud applications (Fehling et al. 2014).

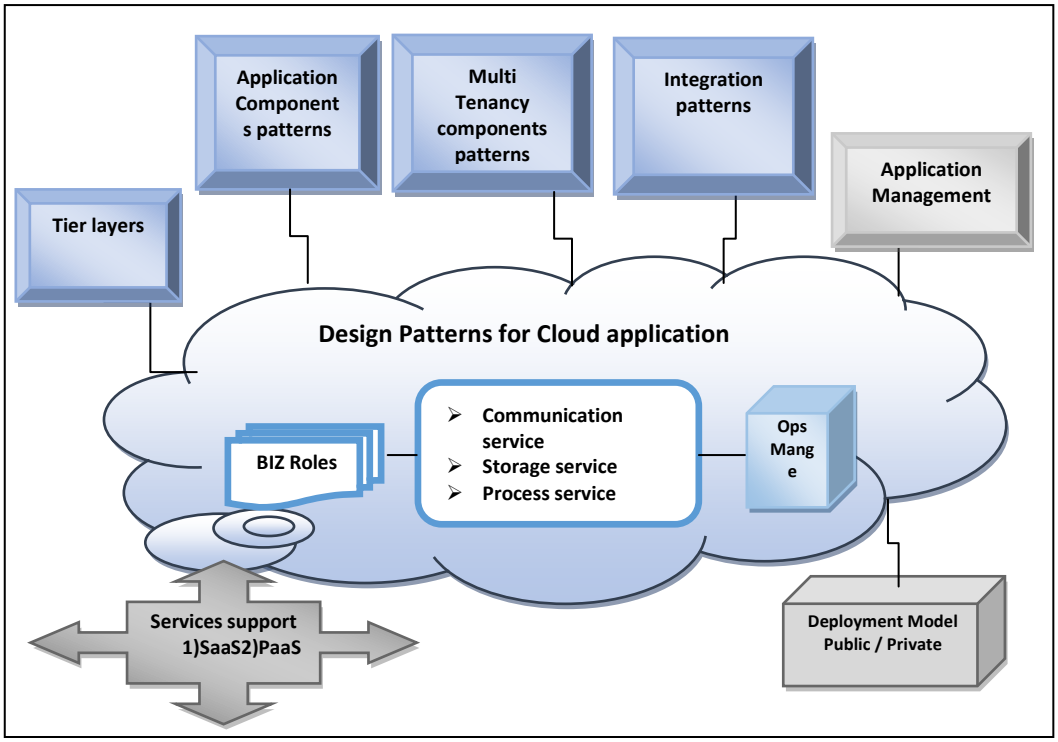


Figure 3.4 Design patterns for cloud applications architecture (Ibrahim & Hany 2015) overview.

3.3.2 Cloud Application Patterns Portability Support

Architectural styles and patterns applied to a cloud application, to describe the structure of the solutions provided to play a basic role in designing and developing different components of cloud application for different platforms to deploy and redeploy on the various Clouds (Di Martino, Esposito, et al. 2015)**.** Cloud design patterns focusing on a general and reusable solution can be applied to any target environment for cloud computing (Fehling et al. 2011), understanding the changes to apply, and to enhanced an application architecture (Jiang & Mu 2011). Architecture include quality attributes such as “efficiency”, "portability”, “usability”, “maintainability”, etc. We give some overview of many cloud patterns, that support cloud application portability as shown in Table 3.1.

Table 3.1 A summary of Cloud application architecture patterns QoS Support.

Cloud Application Architecture Patterns	Category	Pattern Name	Scalability	Elasticity	Accessibility	Integrity	Performance	Flexibility	Security	Maintainability	Reusability	Portability	Interoperability
	Fundamental Architectures	Loose Coupling	++	++	-	+	++	++	++	++	++	++	++
		Distributed Application	++	++	+	++	+	+	+	+	++	++	+
	Cloud Application Components	Stateful Component	++	+	+	+	+	+	++	++	+	+	+
		Stateless Component	+	++	+	+	++	++	+	+	++	++	++
		User Interface Component	++	++	++	+	++	++	++	++	++	++	+
		Processing Component	++	++	+	+	++	++	++	++	++	++	++
		Data Access Component	++	++	++	++	+	++	++		++	++	+
	Multi-Tenancy	Shared Component	++		++	++	-	++	+	+	++	++	++
		Isolated Component	++	++	+	+	++	++	++		+	+	-
		Dedicated Component	+	++	+	+	+	++	++		+	+	+
	Cloud Integration	Restricted Data Access	-	++	++	++	-	+	++	-	-	-	-
		Message Mover	-	-	++	++	-	+		+	+	++	++
		Application Component Proxy	++	++	++	++	+	+	+	+	+	++	+
		Compliant Data Replication	++	+	++	++	++	+	+	+	++	+	++
		Integration Provider	++	+	++	++	+	+	+	++	+	+	++

Legend

(++): Strong support and achieve QoS; (+): Partially support and achieve QoS; (-): Not achieving QoS.

Since cloud application is in terms of components, it results in high maintainability and portability. Also, we can easily attach a new component on demand, it will be easy to scale up an application (Sharma et al. 2015).

We give some overview of many cloud patterns, that support cloud application portability as below:-

a. **Pattern coupling**: effects on maintainability, factorability, and reusability when patterns are coupled in various ways. Patterns are loosely tied together with few connections, making it easier to separate the patterns. They should promote quicker, simpler future design changes (McNatt & Bieman 2001).

b. **Distributed application:** divided functionality among multiple application components that can be scaled out independently, and support portability (Fehling et al. 2014).

c. **Stateless component:** multiple components can share a common external state, make application most tolerant to component failures, support performance by reducing requests to the central data store and component instances keep replicas of central store state information, this can strongly support for application portability to multiple cloud environments. In stateful application component

synchronize their internal state (instances replicate among all instances), this strong support for scalability, security, maintainability and partially support for portability and reusability.

d. **Multitenancy patterns**: is one of software architectural styles and patterns consists of a single instance of an application with multiple tenants, describe how application components comprising a SaaS application can be shared between different customers (Sharma et al. 2015).
e. **A restricted data access component**: extends the functionality of the data access component to incorporate data obfuscation if sensitive data may not be retrieved completely from a less secure environment. Alternatively, data may be replicated between environments

3.4 A Framework for Development Cloud Application Architecture

Developing cloud application raises portability challenges, requires a methodical approaches that define and support application portability to the cloud. In order to establish portable architecture design to cover portability requirements, there is a need for a development portability framework for cloud applications.

The framework for portable cloud application architecture development, aims at supporting system developers in designing portable architecture for cloud applications which can easily migrate (part of)/ system and deploying in multiclouds platforms as needed. A framework encompasses four phases, strategy and plan phase, architecture design phase, deployment phase, and evaluation phase.

For the architecture design phase, since there is no standard approach for portable cloud application architecture design that can be used, the architecture design approach of portable cloud application is required (as in Figure 3.9). The architecture design approach based on proposed containers patterns and cloud application patterns for development. Methodology achieves the following goals:-

- **Design quality:** to achieve Portability by applying capability for cloud application component, that minimizes efforts for developers, in addition to reduce implementation time.
- **Run-time quality:** to achieve agilely for system operation by communicating and exchange information to be reused for development (DevOps).

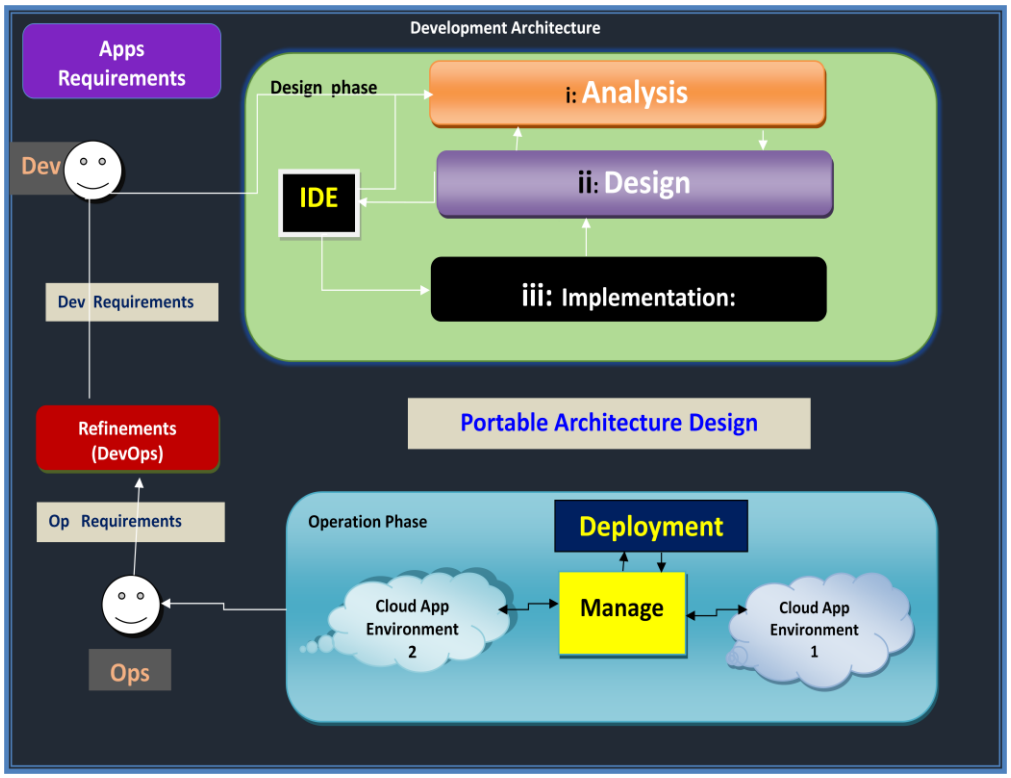


Figure 3.5 A Framework for cloud application architecture development overview.

3.4.1 The Proposed framework Phases

The initial methodology steps involve proposing a development framework to achieve the second research objectives. The proposed framework divided into following phases, as shown in Figure 3.6.

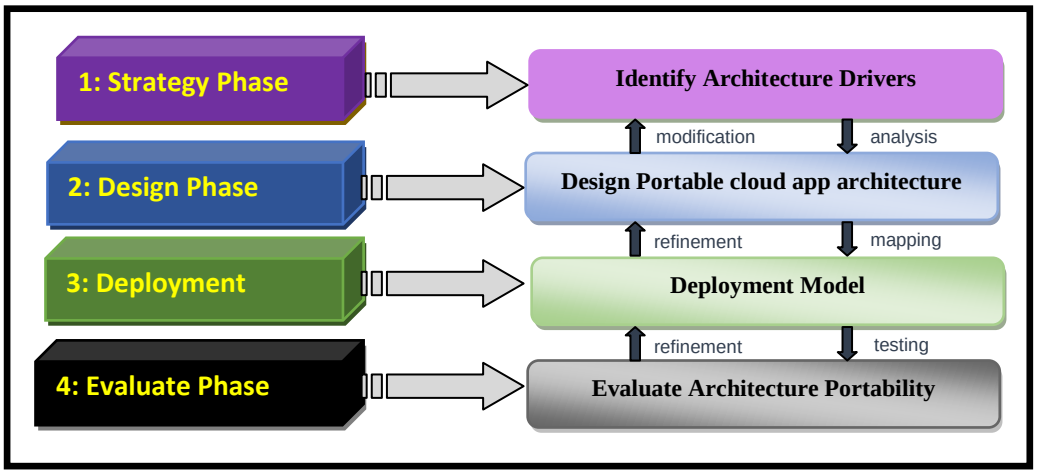


Figure 3.6 The Proposed framework phases for developing a portable Cloud Application Architecture.

Phase 1: Portability Strategy Plan

A process need to identify and specify architecture drivers; Identify and select cloud portability scenario; Describe the behavior and requirements of a system; Provide a generic description of the application.

Phase 2: Design portable application architecture

Need to adopt an appropriate architecture design for specific application. Identifying, using cloud design patterns that specifically apply to provide an application architecture design to deploy on the clouds.

Phase 3: Portable cloud application Deployment

The solution is to develop a portable deployment model to be ported and deploy to cloud platform, also ported from one cloud platform to another. Development based on Cloud patterns with a driven design method for mapping between models.

Phase 4: Evaluation architecture portability

To achieve the third objective (evaluate the proposed portability architecture), the evaluation is to ensure portable cloud application requirements are met, using validation and verification.

During the architecture process, the aim of the validation is to increase the confidence that the architectural design support portability for deployment. Scenarios-based used as evaluation criteria to support architects and developers during the evaluation process (Ionita et al. 2002). Scenarios are a technique developed through manual evaluation and testing. Scenarios are related to architectural concerns such as quality attributes, and they aim to highlight the consequences of the architectural decisions that are encapsulated in the design (Gorton 2006). Thus, we evaluate the applicability of our approach in practice by uses of case studies.

Validating an architecture design poses some tough challenges. The tools for architecture for a new application, implement and test data required from developed step to deploy target application on different cloud platforms. We use CloudMiG simulation tool to evaluate deployment options using a case study.

3.5 Container-based Architecture Design of Portable Cloud Applications

We are focusing on phase 2: design portable cloud application architecture. This is because there is no standard process approach that can used to face challenges like tightly coupling and poorly designed interfaces.

3.5.1 Container benefits

Containers are used for deployment, but a more concern is needed for designing portable cloud applications to build better and faster deployment containers on multicloud PaaS. Container benefits represent in:-

1) Ability to achieve portability with containers (Pahl et al. 2017) .
2) Reduced complexity through container abstractions; allow applications to be localized within the container, and then ported to other public and private cloud providers that support the container standard (Linthicum 2016), such as Docker.

3.5.2 Container for portable cloud application

Container is a new technology for portable cloud platform, allow the sharing of the underlying platform and infrastructure in a secure way. It holds packaged self-contained, ready-to-deploy parts of applications, to run the applications (Pahl et al. 2017). Container is a standard way to package an application and all its dependencies that can be moved between environments and run without changes. Containers work by isolating the differences between applications inside the container, eliminating the need for instruction level emulation (Dua et al. 2014), so that everything outside the container can be standardized.

3.5.3 Comparison of a container Related work

Containers are used for deployment, but a more concerns is needed for designing portable cloud applications to build better and faster deployment containers on different PaaS. Several works and experiences uses containers, summarized in the Table 3.2.

Table 3.2: A summary of related work for container used.

Author/s	Context	Problem	Proposed solution
Pahl, Claus, and Brian Lee. 2015	Edge cloud architecture, Various devices such as IOT, is the context of edge cloud computing.	Virtual devices to host application and platform services, and the logistics still required to manage architecture setting.	▪ Edge cloud environment, application and service orchestration can help to manage and orchestrate applications through containers as an application packaging mechanism. ▪ Facilitate applications through distributed multi-cloud platforms build from a range of networked nodes.
Kang, e.tal 2016	Identifies the key challenges of containerizing infrastructure services.	It is difficult to use containers to manage the cloud infrastructure, without sacrificing many container offers.	▪ Redesign Open Stack deployment architecture to enable dynamic service registration and discovery. ▪ Explore different ways to manage service state in containers, and enable containers to access the host kernel and devices.
Burns, e.tal 2016	Container architectures leading to design patterns for container-based distributed systems	Distributed system development, need for enabling a standardization and regularization.	▪ Identified three types of patterns: single-container patterns for system management, single-node patterns of closely-cooperating containers, and multi-node patterns for distributed algorithms,
Kozhirbayev, e.tal. 2017	Container based technologies such as Docker allow hosting of micro-services on Cloud infrastructures	Cloud computing has overheads and can constrain the scalability and flexibility, especially when diverse users with different needs wish to use the Cloud resources.	▪ Compare and contrast a range of existing container-based technologies such as Docker for the Cloud and evaluate pros and cons and overall performances.
Jain, Rakesh, et al. 2018	Migrate a composite application to a container-based environment	How software components to be composite applied?	▪ Creating a plurality of containers, and communications channels between the plurality of containers, for the software components of the composite application based on the containerization plan.

In comparison to the related work for containers (Pahl & Lee 2015; Kang et al. 2016; Burns & Oppenheimer 2016; Kozhirbayev & Sinnott 2017), all Containers technology solutions discussed above focus on deployment at run time, such as

Docker and Kubernetes for application hosting on the cloud, while the (Jain et al. 2018) is relevant to our work, but concern in provisioning to run the migrate application on the specific cloud environment. However, to the best of our knowledge, there is no related work that suggests the use of application containers at the architecture design phase.

3.6 Container-based Architecture Design Solutions

The proposed approach depends mainly on the architecture design phase of a framework for developing portable cloud application architecture, as in figure 3.6. Need to adopt and develop appropriate architectural design solutions depend on the scenarios-based container design approach, that group application behavior to several application scenarios in a container, each scenario is a solution using container patterns to describe cloud application architecture design. Benefits outcome is to reduce the complexity, minimize dependence, avoid vendor lock-in, and exploits the functionality benefits that can be achieved through multi cloud architectures. Our solutions represent in:-

1) Propose Development process to guide in designing containerized portable cloud applications, for easier and better deployment on multiple cloud platforms.
2) Propose and uses new container design patterns to describe architecture design.
3) Present a development patterns approach for designing Scenarios-based container, that group application behavior to several application scenarios in a container, each scenario container is a solution that the system can be used, reducing the complexity.

3.6.1 Development process for Container-based Design

Foe cloud application design, loosely couple identified as design criteria with application components, that individual application components can use different cloud offerings requirements. The loose coupling depend on (interdependence, differentiation, and integration), provide a basis for developing the definition (Pinelle & Gutwin 2005).

Loose coupling is a design strategy which allows us to reduce the interdependencies between components of a system with the goal of reducing the risk that changes in one component will require changes in any other component. It's all about thinking a problem in generic manner and which intended to increase the flexibility of a system, make it more maintainable, and makes the entire framework more stable.

Loose Coupling provides many advantages, such as:-

- Components in a loosely coupled system can be replaced with alternative implementations that provide the same services, to the same platform, language, operating system, or build environment.
- In the design of organizational processes, a modular approach with loosely coupled components produce flexibility, helps to create the ability to reconfigure the system by enabling new functional process variations.
- Adaptability: future functional requirements to take advantage of the existing loosely coupled module than build a new, redundant module.

Process approach based on cloud design pattern development to describe architecture components for cloud application to be configured, mapping between patterns, identifying and selecting required patterns to decrease the level of development (Cretella & Di Martino 2012) to gain container design for portable cloud application. The design process is composed of iterative refinement development process steps as shown in Figure 3.7.

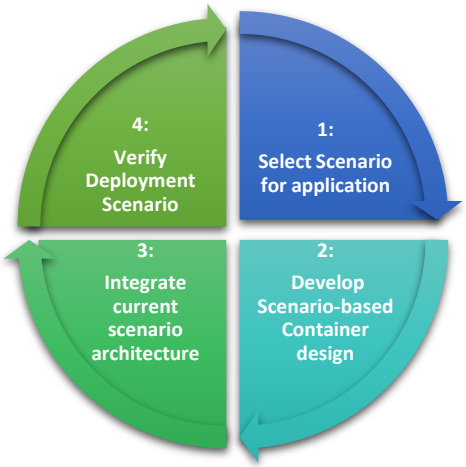


Figure 3.7 The iterative development process for multiple scenarios.

[1]. Select Scenario for application: based on application analysis, metric of selecting scenario with a high frequency of usage.

[2]. Develop scenario-base container design: based on proposed container patterns, the development process for independent scenario steps as shown in Figure 3.8.

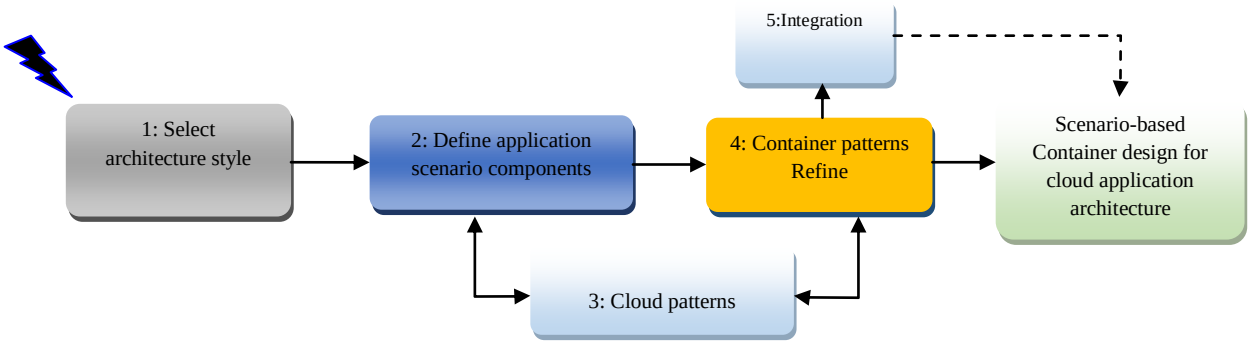


Figure 3.8 The development process for independent scenario-based container design.

[3]. Integrate current scenario architecture: integrate step 2, in addition to multiple develop scenarios with their dependencies into a portable cloud application container design, as shown in Figure 4 for deployment.

[4]. Verify deployment Scenario: approach based on using approaches and tools to check deployment options for running cloud applications on multiple clouds.

3.6.2 Development of Independent Scenario-based Container Design

The steps of the development independent scenario-based design are:-

Step 1: Select architecture style for application

Architecture styles characterize families of architectures sharing common characteristics specification, benefits in:-

- Identify Application type.
- Identify relevant architecture design for application support, based on annotated UML diagrams to describe software system.

PaaS provide set of capabilities, using PaaS portability as evaluation criteria private/ public to specific applications.

Step 2: Define Application components for the scenario

To gain a flexible design need to benefit from loose coupling to allow reuse and improve portability. Container: focuses on design application components that represent well-defined communication interfaces containing methods, events, and properties to be tightly coupled for deployment.

Step 3: Cloud pattern selection

Select adaptive cloud application patterns for scenario, to provide optimal solutions, and to meet the scenario changes during development.

Pattern oriented analysis approach based on requirements analysis is used (Amar Ibrahim 2016), consists of major steps. Requirements analysis, is important activities (Lin et al. 2016), identifying app components and relevant cloud patterns. Adapted cloud patterns include refinement to trace candidate selected cloud patterns to a lower level of abstraction to give scalable design.

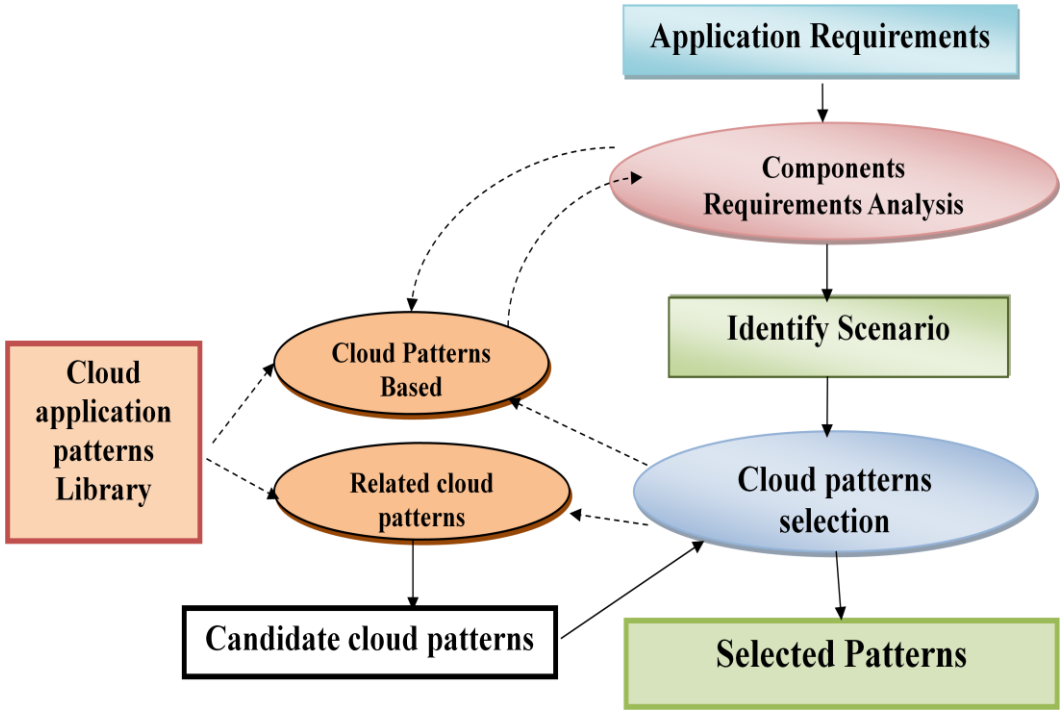


Figure 3.9 Pattern Oriented Analysis approach overview (Amar Ibrahim 2016).

This section discussed patterns oriented analysis approach (Amar Ibrahim 2016) steps as follow:-

1) **Application Requirements:** Application requirements are a detailed description of application services can be required from a customer, system and software for a design or implementation.
2) **Components Requirements Analysis:** Analyze the existing application requirements to identify problems to be solved and to determine conceptual components.

   Process: finding components to functionally identify problem and generate use case diagrams. Identify application component (logical level), if the application is large analyzed into subsystems and subsystem analyzed in more details. Need to iterate with other analytical activities:-

   - **Acquaintance activity:** to identify sets of cloud design patterns will be used in application design (from pattern library) patterns database activity.
   - **Retrieval**: retrieve from patterns repository related patterns, process focus on how to select given patterns, matching if related patterns should be selected, to set as candidate design patterns.
3) **Identify Scenario:** Set of conceptual components for selected system use cases.
4) **Cloud pattern selection:** Select a set of cloud design patterns that will be used in the cloud architecture design of each component, based on studying the relationship between patterns.

Overall output selection of set of cloud design patterns will be used in design for the various parts of the system.

Step 4: Container patterns Refinement

New candidate design patterns for cloud application architecture generate for a solution put in a template-based container design patterns deal with how container be designed to benefit a cloud application.

[5]. Integrate current scenario architecture: integrate step 2, in addition to multiple develop scenarios with their dependencies into a portable cloud application container design, as shown in Figure 4 for deployment.

[6]. Verify deployment Scenario: approach based on using approaches and tools to check deployment options for running cloud applications on multiple clouds.

3.6.3 Multiple Scenarios Integration

Move application to another cloud provider, require to build an application in small containers, that make application to work better, easier on multiple platforms. We present development patterns for designing portable cloud applications in containers. The presented design approach, group cloud application behavior to several application scenarios of components in a container, as in Figure 3.10. The development processes based on driven design for reuse multiple components of the cloud application scenario to reduce the complexity. First cloud application components should be loosely coupled into scenarios of containers to reduce the dependencies among components. For each independent scenario-based container design there are fundamental container patterns, as proposed in section 3.6.4 to use with related cloud patterns, that help in developing container scenario functionalities. The developed containers scenarios integrate to be portable and easily allow multiple container scenarios/ independent container scenario for deployment of multiclouds platforms.

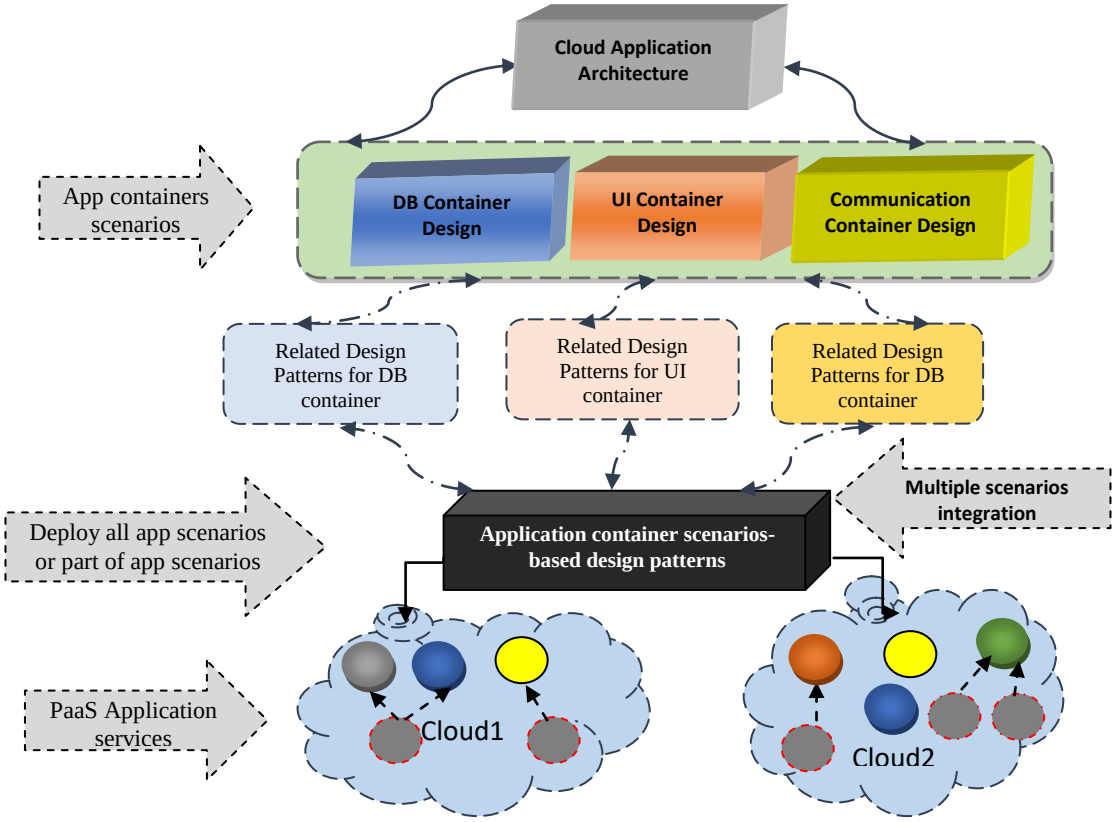


Figure 3.10 A Container based patterns for Portable cloud application design overview.

3.7 Proposed Container design patterns for Cloud Application Architecture

We investigate fundamental Container design patterns for Cloud application architecture, generate for a solution put in template based container design patterns deal with how container be designed to benefit a cloud application.

3.7.1 Interface container design pattern

Table 3.3 Cloud Application Interface container design pattern

Category	Fundamental Container design patterns for Cloud app architecture
Pattern Name	**2. Interface Container**
Intent	Scenario container dependencies to other containers, that should be explicit by defining interfaces to be used by different containers.
Design Problem	How can scenario containers be accessed decoupled from the other container scenario?.
Design Solution	Create segregate container interface for each scenario design and multiply inherit them into application container. Interfaces for each container serves as a bridge between the synchronous access of the specific scenario container and the asynchronous communication with other containers scenarios.
Related Patterns	1. Communication Container scenario based Patterns. 2. User Interface. 3. Data abstractor. 4. Adapter pattern.
Reference Diagram	Figure 3.10: Integration container interface pattern.

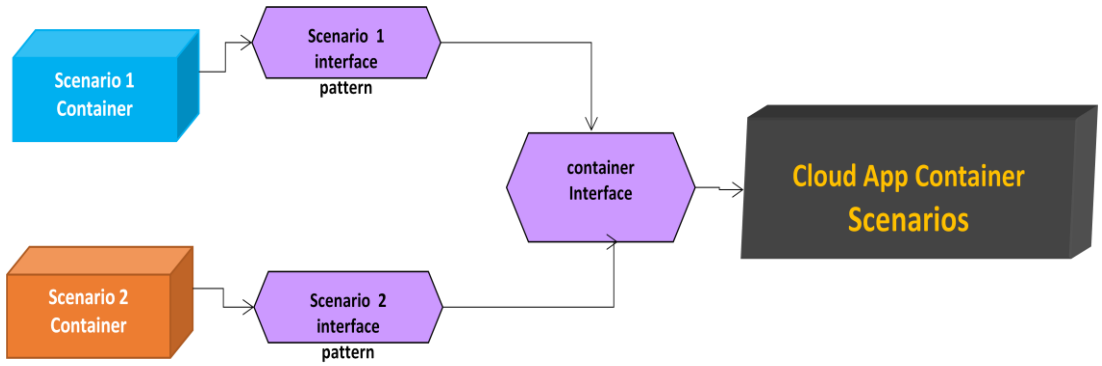


Figure 3.11 Integration container Interface pattern, for each container scenario work synchronously and for other scenarios works asynchronously.

3.7.2 Communication container pattern

Table 3.4 Communication container.

Category	Fundamental Container design patterns for Cloud app architecture
Pattern Name	**3. Communication Container**
Intent	Scenarios containers share application container interface, that want to collaborate for specific scenarios .
Design Problem	How can Container scenario communicate with another scenario container and isolate scenarios containers from complexity?
Design Solution	❑ Collaboration containers proxy scenario communication to and from the application container. Simplifies and standardizes the outside of the application container. ❑ Enables the heterogeneous of legacy and open-source applications to present a uniform interface without requiring modification of the original application container.
Related Patterns	1. Adapter pattern 2. Load Balancer 3. Interface container scenario design 4. Data access. 5. Chain container.
Reference Diagram	Figure 3.11: Scenario based for integrating specific container scenario into application container

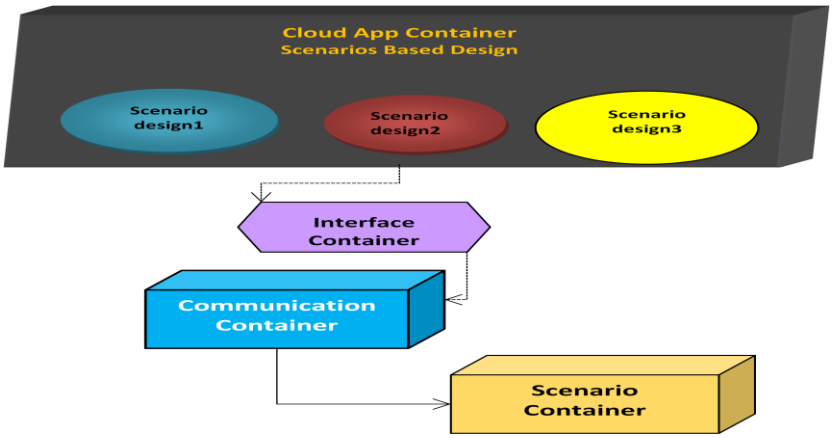


Figure 3.12 Communication container interaction with scenario container using interface container.

3.7.3 Database container design pattern

Table 3.5 Data Base container.

Category	Fundamental Container design patterns for Cloud app architecture
Pattern Name	**4. Data Base Container**
Intent	Functionality to store and access data elements is provided by special components that isolate complexity of data access, enable additional data consistency, and ensure adjustability of handling data elements to meet different customer requirements
Design Problem	How can the complexity of data storage due to access data consistency be hidden and isolated while ensuring data structure configure?
Design Solution	❑ Access to different data sources is integrated by a Data Access Component. ❑ Component coordinates all data manipulation. ❑ In case a storage offering replaced or the interface of a storage offering change, the Data Access Component is the only component that has to be adjusted, ensuring loose coupling between application and cloud offerings.
Related Patterns	1. Communication container pattern. 2. Interface Container pattern. 3. Stateless pattern. 4. Stateful pattern 5. Storage pattern

3.7.4 Application scenarios container design pattern

Scenarios derive the characteristics of architecture directly from the high-level requirements of the business. They are used to help identify and understand business needs, and thereby to derive the business requirements that architecture development has to address. Scenario describes business process, application, or set of applications that can be enabled by the architecture, and the people (called “actors”) who execute the scenario (Desfray & Raymond 2014).

Table 3.6 Cloud Application scenarios container design

Category	Fundamental Container design patterns for Cloud application architecture
Pattern Name	**1. Application Scenarios container**
Intent	Cloud application divides functionality requirements among multiple application components , using multiple scenarios for certain use case.
Design Problem	How can we design the application architecture to define an executable release that can be migrated easily in multiple cloud platforms?
Design Solution	Define the application architecture design such that we allocate tightly coupled functional scenarios to one container.
Related Patterns	1. Distributed application patterns. 2. Layer-based.
Reference Diagram	Figure 3.9: Scenario based for integrating specific container scenario into application container

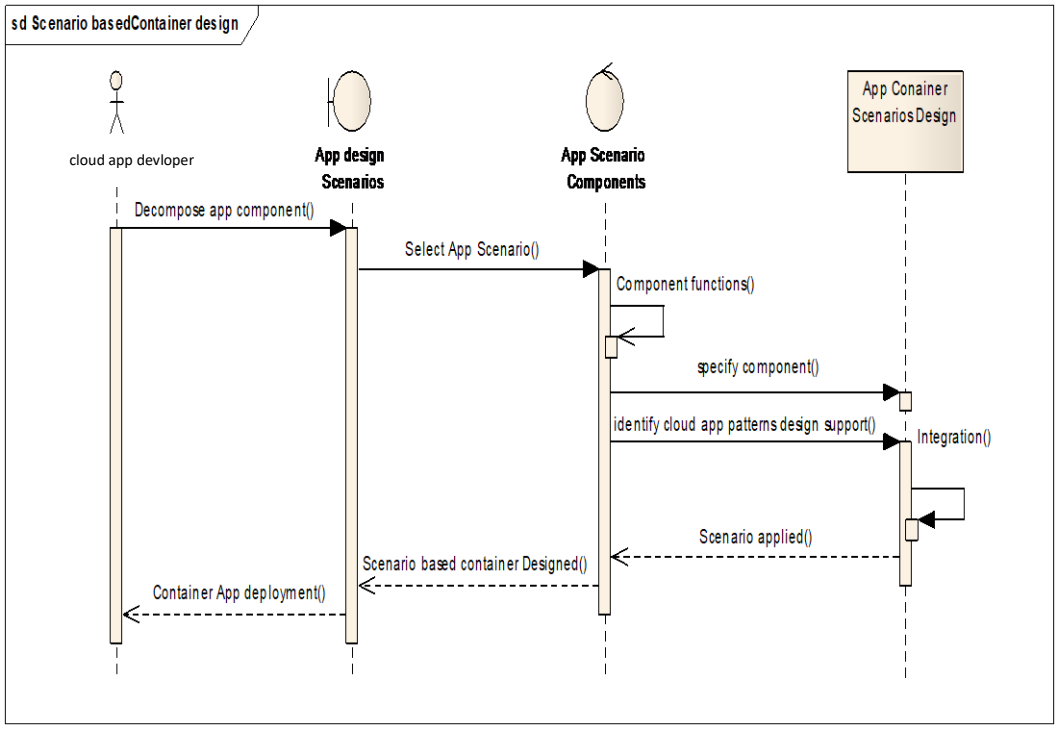


Figure 3.13. Scenario based for integrating specific container scenario into application container.

3.7.5 Chain container design pattern

Table 3.7 Chain container.

Category	Fundamental Container design patterns for Cloud app architecture
Pattern Name	**5. Chain Container**
Intent	Multiple scenarios containers, binding into a cloud application container design for cloud deployment demand in a specific order.
Design Problem	How Application container interacts with other containers and get deployed?
Design Solution	The operational scheduling logic is pre-defined in a chain configuration that enables containers to be bound with various scenarios supported, including the option of having pushed for deployment and suspend dependently of a scenario.
Related Patterns	1. Communication container pattern. 2. Interface Container pattern. 3. Scenario container
Reference Diagram	Figure 3.14: Scenario based for integrating specific container scenario into application container

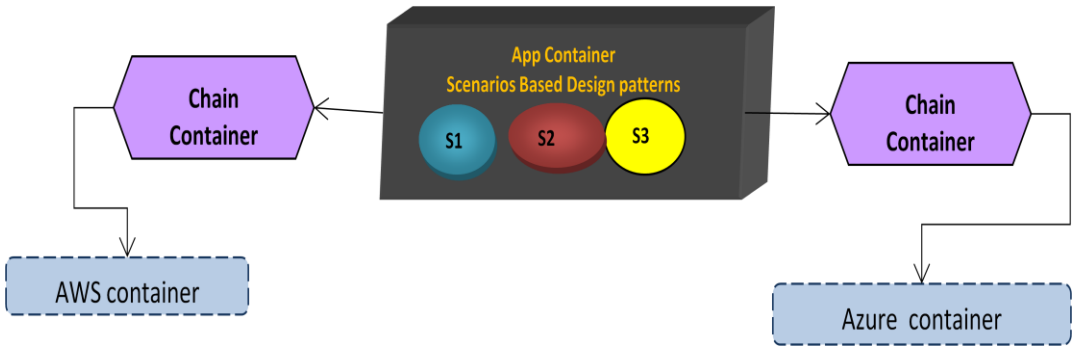


Figure 3.14 Chain container to get deployed.

3.8 Proposed Method Evaluation

Proposed method evaluation based on the following approaches:-

1) Validate a methodology using scenario- based container design discussed in chapter 4 and chapter 5 of case studies:-
   - ❑ Migrating Students Academic Result Records Web base Service to the cloud.
   - ❑ Hajj and Umrah Mobile Healthcare System.
2) Verification deployment using a simulation tool, as in chapter 6 of case study:-
   - ❑ Migrating Online Pet Store Application to IaaS cloud.

3.9 Summery

In this chapter, we discussed the methodology adopted for this research. A research methodology was proposed as an approach to achieve the research objectives. The proposed research methodology is explained according to the framework of portable cloud application architecture. The framework development phases have been explained in detail. For architecture design phase, the development process for the container design approach has been proposed for multiple scenario development. In addition, new container design patterns have been proposed for architecture development support. The next two chapters present the evaluation in architectural design of case studies for portable cloud applications.

CHAPTER IV

PORTABLE ARCHITECTURE DESIGN OF MIGRATING STUDENT ACADEMIC RESULT RECORDS TO THE CLOUD

4.1 Introduction

There is a little concern for portability to benefit migrating existing applications to cloud. Migrating application, data or services require efforts pose many challenges such as data handling, architecting of the web applications to move among different cloud platforms (Murugesan & Bojanova 2016). Layer architecture provides features for designing cloud applications; these features decompose the application into logical layers that benefits, enhance reusability, improve portability, scalability, and support application changes for migrating (Council 2016) and deployment in the cloud environments, act as loosely coupling solution to a gain a flexible portable architecture design.

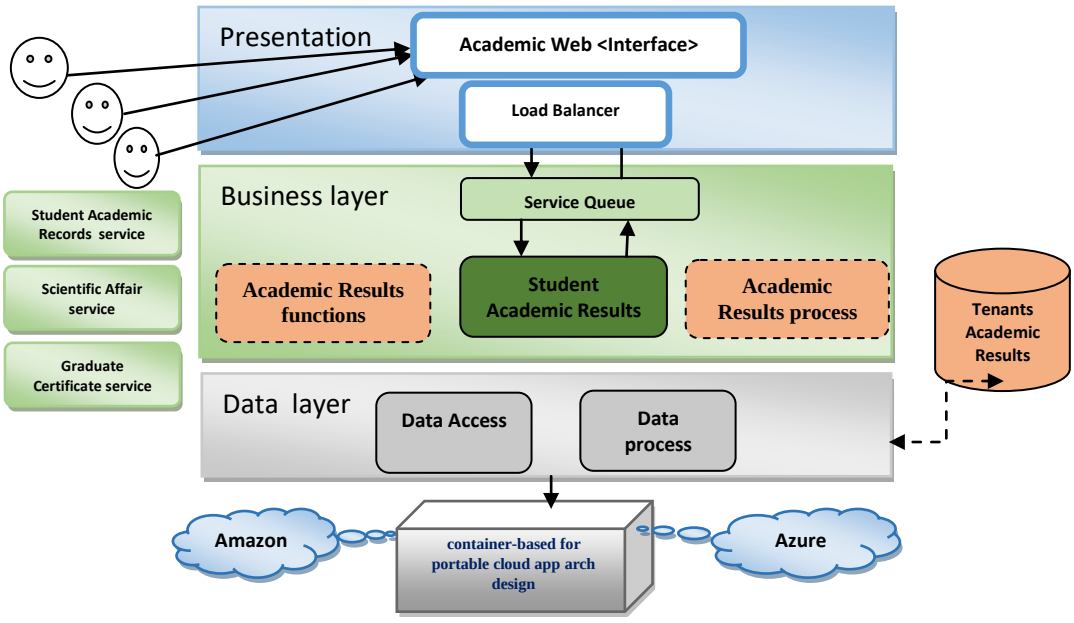


Figure 4.1 Layer architecture for student Academic Results Record Web service.

Presentation layer enables Academic officer and the system administrator to add and update Academic Results; Students of college can view their academic Result Records; Services layer to interact and select academic Examination Result service; Business layer consists of implementing an application component. For validation of methodological approaches, a case study had to be designed as scenarios-based container for migrating to the cloud.

4.2 SAAR Architecture

The case study is motivated by our experience of Students Academic Result Records System (SARR) for academic college with two different branches, need to migrate examination results into the cloud, because the application lack of resource sharing, and time consuming for producing reports. First, we reengineer application into a web-based, because web application use for developing, deploying, and maintaining. In addition Web applications is simple to use, scalable to service requests, and has built-in data store and a flexible interface (Soliman et al. 2013).

Our architecture is based on Cloud computing that provides storage and computing resources to to produce new Academic service for students. Our Web application is categorized into two main parts: a front-end and a back-end. The front-end serves as a Web interacting with students to review their academic results records. The back-end serves as computing services for storage services for the migrated data storing , based on public cloud deployment model.

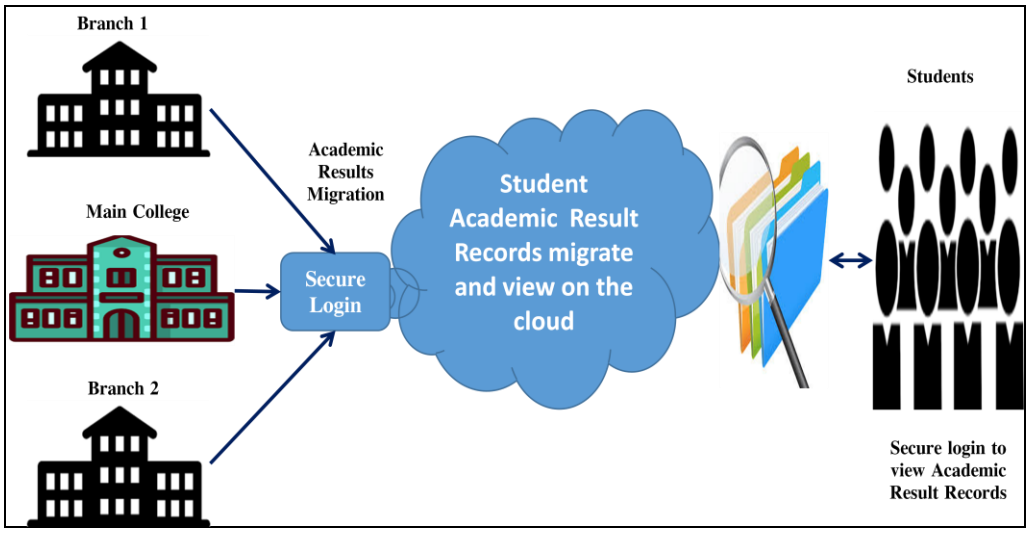


Figure 4.2 Overview of the Student Academic Results Records Web based cloud application.

To better understand the needs of migration to cloud for the case study, application architecture has been analyzed, defines detailed requirements mapped to UML scenarios, described a high level includes actors, use cases and sequence diagrams. Using UML notation as follows:-

(i) Use Case diagram: to describe the functional requirements of application architecture.

(ii) Sequence diagram: address the dynamic behavior of a system. Used to understand an interaction scenarios for the system.

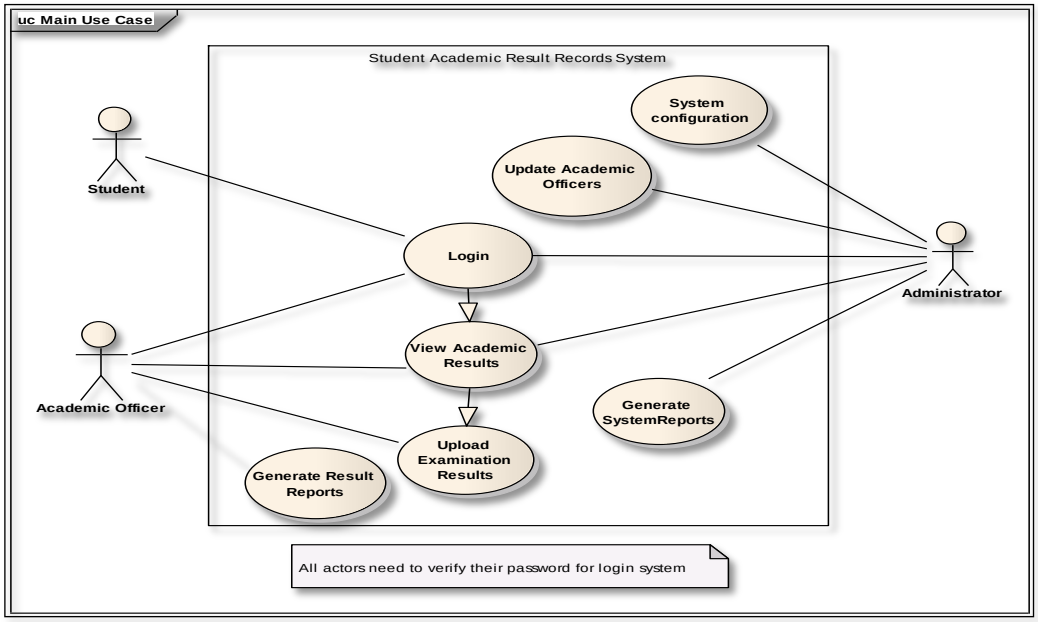


Figure 4.3 Use case diagram for the Academic Result system service.

The use case diagram illustrates the system objects, to determine the scenarios of system. The system actors represent in:

1) Academic Officer: login the system, manage every branch students account, create and update student result, search students' records, adds examination results for students, entering student examination marks, and generate results reports;
2) The administrator can view academic results, system maintenance, aggregating different branches students result records, producing different Results services in each college branch, and generate system reports; Students login the system to query and view their academic result details.

4.3 Multiple Scenarios-based Development

To better understand, and develop the case study migrating Student academic for examination results, the requirements for application described using UML notation. The presented case study developed according to the proposed iterative development process for multiple scenario steps as in Figure 3.7.

4.3.1 Scenario 1: Scenario-based on Student to view academic results

In regard of the academic result service, we considered that the number of students remains constant in order to view academic results from different branches. The requests are submitted to a web server, and the web server interacts with a cloud server that consists of Multitenancy shared database. Finally, students view the generated academic result records.

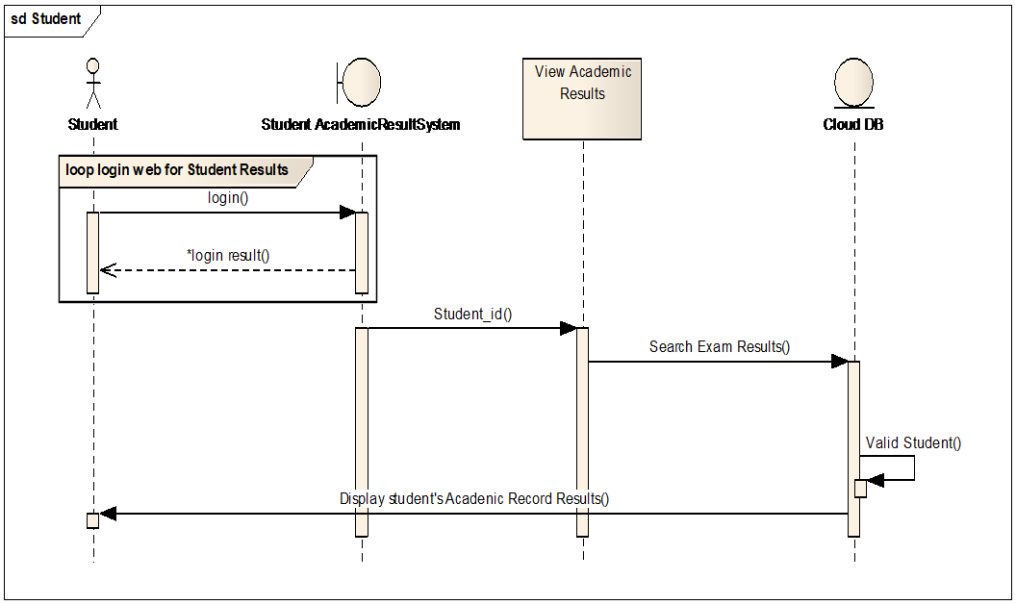


Figure 4.4 Scenario-based for view student Academic Results Records use case.

The diagram represents the dynamic view of the system architecture. In addition, it presents the searching scenario for a specific student. Based on student-ID, the query moves from the service web to cloud database.

[1]. Scenario1 development using cloud application patterns

Selected cloud patterns for application facilitate application developing, application decomposes into isolated layers of components. Each layer consists of application

components providing certain function. Patterns selection based on pattern analysis for cloud application requirements approach (Amar Ibrahim 2016).

Table 4.1 Summary for cloud application pattern selection.

#	Acquaintance	Retrieval	Solution
1	Fundamental Architectures	Loose Coupling	Makes the system flexible to run different components on different cloud.
2	Cloud Application Components	Stateful Component	A synchronized internal state; Replication Internal state.
3		Stateless Component	Storage Offerings (external); Increase Elasticity.
4		User Interface Component	Reduce Dependencies; e.g. Elastic Load Balancer: to spread traffic to web server auto scaling group.
5		Processing Component	Processing functions to meet different customers, Split into separate functional blocks (stateless).
6		Data Access Component	Integrity of data and coordinates. Allow multiple customers access the single instance of the DB.
7		Transaction-based Processor	Ensure messages receive are processed successfully and altered data successfully after processing. Deliver services to enhance portability.
8	Multi-Tenancy	Shared Component	To improve resource sharing and user's access to the system. Deploying components on selected clouds; Deploying solution mechanism over clouds; Optimize requirements for clouds.
9		Tenant-isolated Component	Ensure isolation between branches by controlling tenant access, processing performance used, and separation of stored data.
10		Dedicated Component	Provided exclusively for each tenant using the application.

[2]. **Scenario1 Container-based architecture design**

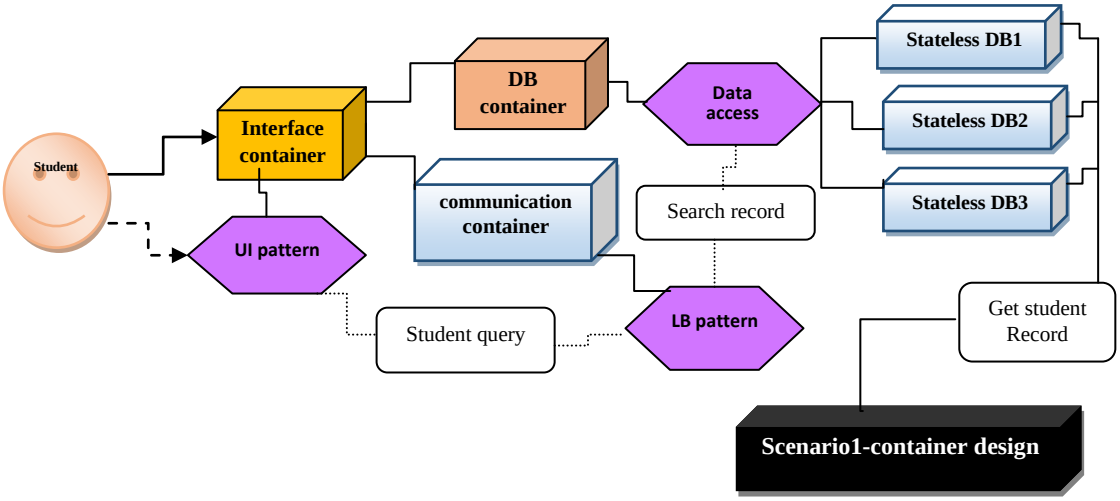


Figure 4.5 An annotated proposed scenario-based container design for web-based Student Results service on clouds.

The figure above present scenario based architecture design diagram. The diagram explains the use of container patterns with related cloud design patterns, for specific student to searching academic result records. Based on student-ID, the query moves from the web service to cloud database.

4.3.2 Scenario 2: Scenario-based on academic officer

An academic officer manages every branch of students account, upload exams results, updates and maintains student results. Also view student Academic result details.

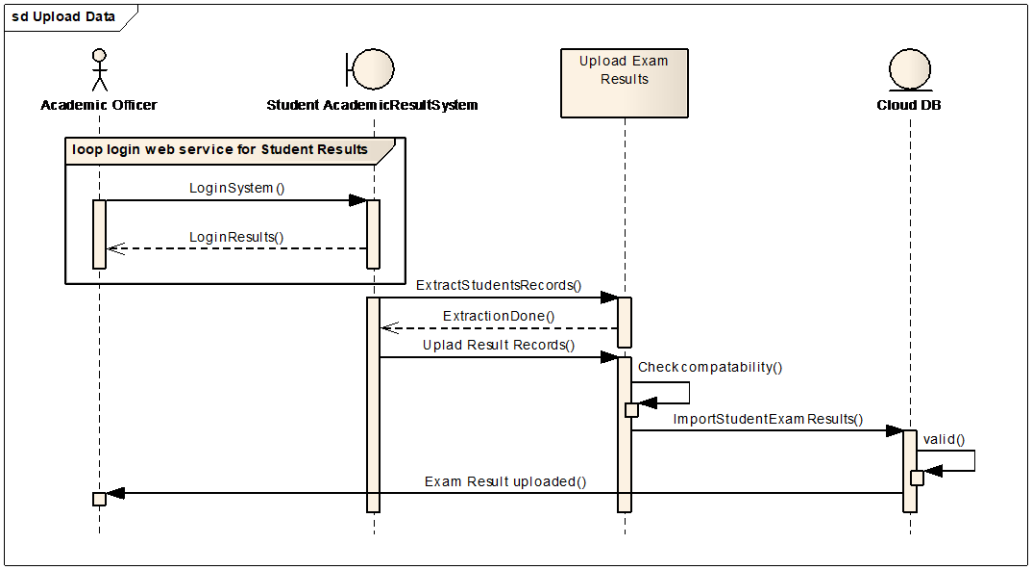


Figure 4.6 Scenario-based for uploading student Academic Results Records.

[1]. Scenario 2 Development using cloud application patterns

Based on cloud patterns selection (Amar Ibrahim 2016) and application component, each layer provides certain functions. Through user interface pattern: login to specific college branch via web base service; then check the status of academic officer requests, through elastic load balancer; The processing component pattern used to perform specific academic Result tasks, such as result entry, modify, upload exam data records, generate reports, query via the Service user interface.

[2]. Scenario2 Container-based architecture design

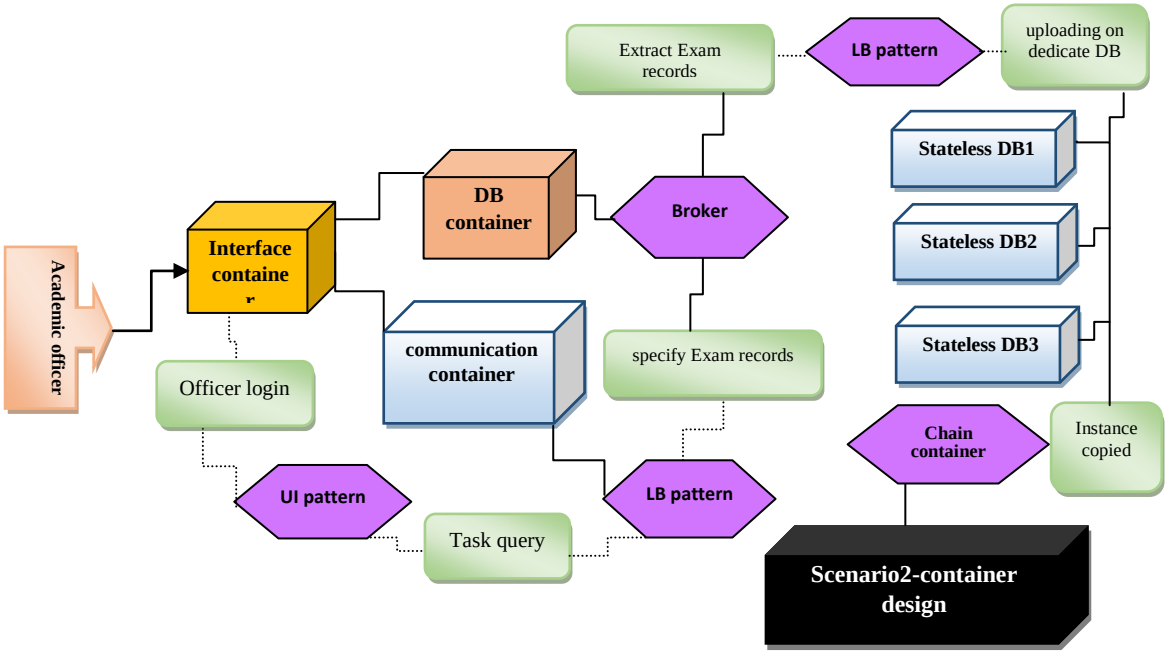


Figure 4.7 An annotated proposed container design for an Academic officer to upload Student Examinations Results on a cloud.

For an academic officer to upload students' exam records, we use a Broker to access different data sources (branch web server). Cloud's side based on stateful pattern: to create an instance of exam results; Multitenancy patterns used for shared different college branches database, that help in reducing complexity of separate web application into self contained services to interact with instances directly using queue pattern.

4.3.3 Container-based architecture design for multiple integration scenarios

Multiple scenarios of usage for a case study, integrate into a container for designing portable cloud application architecture.

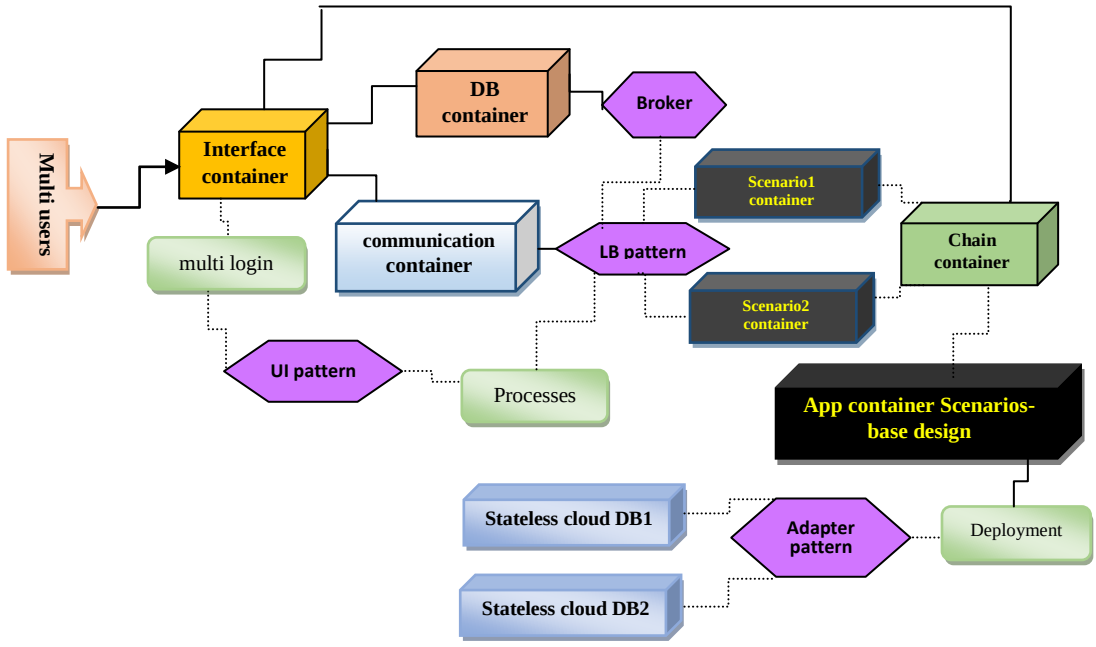


Figure 4.8 An annotated proposed container design for migrating Students Examination Results on clouds.

Figure 4.8, explain the coupling of multiple scenario dependencies' architecture design for case study. The diagram is driven design that integrates the scenarios-based development in a cloud application container design, to facilitate the portable architectural container designed of case study for deploying on multiple cloud platforms.

4.4 Summary

In this chapter, we presents Web-base service migrating Student Academic Examination Results Records on clouds (SAAR) case study. Case study used to validate our methodology for portability-based of cloud application architecture design.

We applied our scenario- based development approach on a case study of a web-based service for student academic result records service to validate our portable architecture design. First step, we studied the legacy application to migrate students' records in the cloud, because porting legacy applications, data or services, requiring a

significant effort to be invested in putting into in the cloud environment. In the second step, the system requirements described using UML diagrams.

According to the scenario-based development process, the case study is evaluated through iterative development steps for multiple system scenarios using cloud patterns. Multiple scenarios integrated into a container for designing portable architecture. The portable architecture design solution act as pluggable design, provides a flexible cloud application architecture that can be easily deployed to specific cloud / multiple clouds. Moreover, easing the development process, minimize migration efforts, and prevent lock-in within cloud application.

CHAPTER V

PORTABLE ARCHITECTURE DESIGN FOR HAJJ & UMRAH MOBILE HEALTH CARE SYSTEM (HUMS)

5.1 Introduction

Information and Communication Technology (ICT) and Internet of Things (IOT) technologies used to connect human life from different perspectives, such as smart connectivity, smart home connectivity and smart cities, are forecasted to grow at an astounding rate. In addition, these devices are expected to accelerate benefits for major social and environmental needs such as improved healthcare using smart mobile technology, that supports the backbone of enhancing quality of life (Kuo & Hsu 2017) .

The Smart Mobile device requires a great deal of effort to benefit from capabilities such as, Wi-Fi, cameras, storage, GPS and speed processors. As a result, developers build more complex mobile applications (Elgendy et al. 2014). We motivate to benefit from smart mobile capabilities (Ahmad et al. 2017) for patient healthcare in large crowd events.

5.1.1 Electronic Health Records (EHRs)

Electronic Health Record (EHR) is defined as *"a longitudinal electronic record of patient health information generated by one or more encounters in any care delivery setting*". Included patient demographics, progress notes, problems, medications, vital signs, past medical history, immunizations, laboratory data, and radiology reports. Another definition "*the set of components that forms the mechanism by which EHRs are created, used, stored, and retrieved*", allow sharing health information between different systems in different Care Delivery Organizations (Youssef 2014). The key benefit, as shown in Figure 5.1, with the access of medical information that will improve patient safety, enhanced accuracy health Information, decrease cost and medical errors, more thorough documentation and increased quality of care and better patient notification.

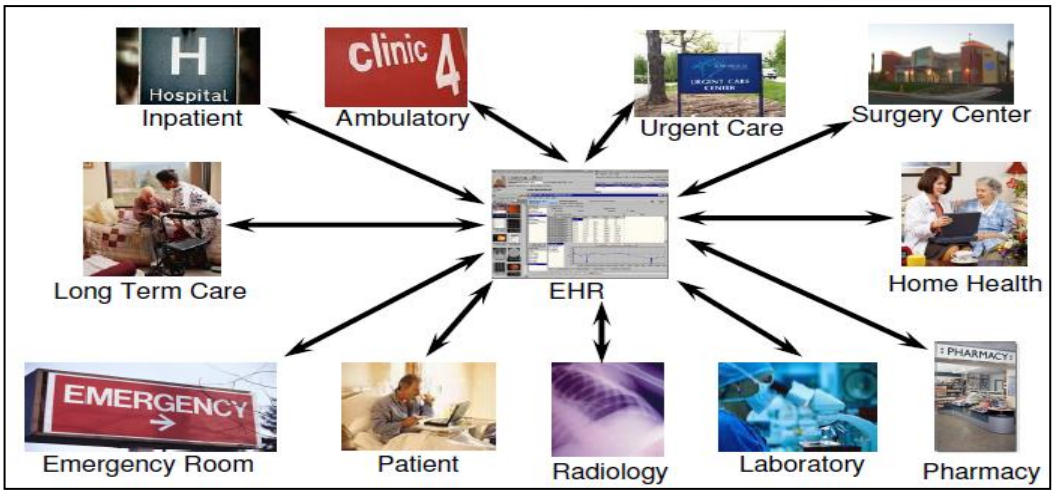


Figure 5.1 The importance of the electronic health record(EHR), with integrated information from multiple resources (Amar & Hany 2017).

5.1.2 Mobile Technology

Mobile Device is a generic term used to refer to a variety of devices that allow people to access data and information from where ever they are (Kottari et al. 2013). Mobile phones present applications that have been developed to provide various services and facilities to assist healthcare professionals such as: information and time management; health record maintenance and access; patient management and monitoring; clinical decision-making; and medical education and training. In addition mobile technology provides a number of important benefits to healthcare such as improve geographic coverage by providing patient care through information exchange and better connect to healthcare professionals anywhere and anytime (Ventola 2014), communication capabilities, information sharing facilitate faster diagnoses and treatment and reducing paper consumption for both hospitals and healthcare professionals.

Moreover, Smart Mobile technology enables users to access data and services for patients in the health domain. Mobile Health is an emerging field of medical technology, where mobile applications developed to provide many tasks and functions on mobile devices, such as assist the public health activities. Mobile health, defined as “mobile computing, medical sensor, and communication technologies for healthcare,” refers as a new approach to health care based on mobile communication devices such as cell phones and tablets to collect data which increases patients’ information, reduces medical centers efforts, and reduces costs (Duarte et al. 2015).

5.1.3 Mobile Cloud Healthcare

Mobile Cloud Computing (MCC) define as an emergent mobile cloud paradigm which leverage mobile computing, networking, and cloud computing to study mobile service models, develop mobile cloud infrastructures, platforms, and service applications for mobile clients (Huang 2011). Mobile Cloud Computing has many advantages such as improving data storage capacity and processing power (Lo'ai et al. 2016), improving reliability and availability, scaled to meet unpredictable user demands, allows portable communication, and integrated different services from different providers easily to meet the users' demands.

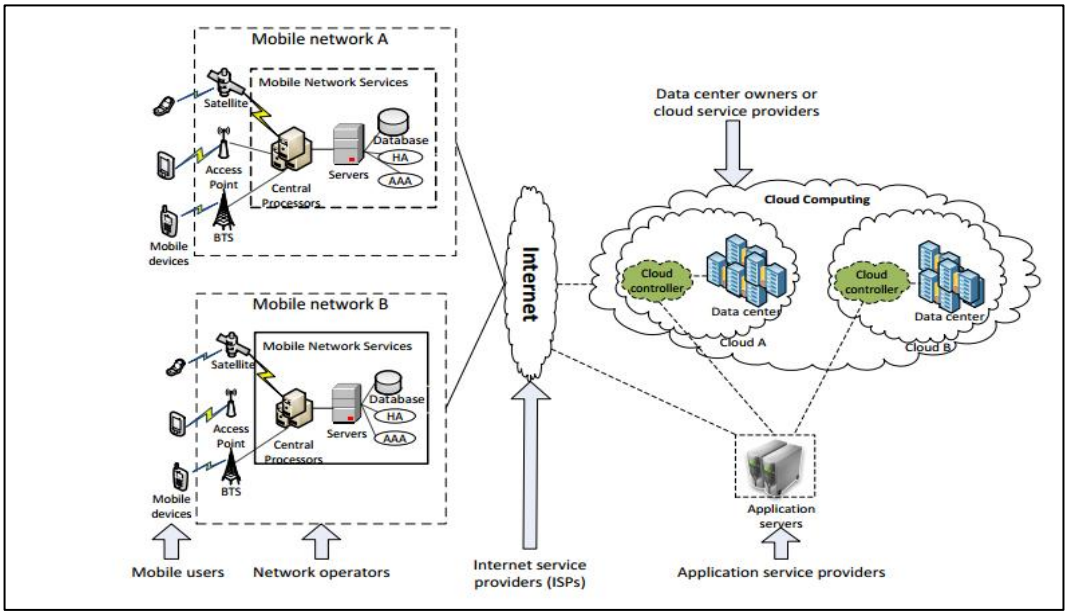


Figure 5.2. Mobile Cloud Computing Architecture (Lo'ai et al. 2016).

The traditional healthcare method is being replaced by smarter healthcare such as Mobile Healthcare (Lee et al. 2015). The combination of cloud computing and mobile networks bring benefits for mobile users, network operators and cloud providers.

The existing health information systems suffer from many challenges to developing such as standards for information sharing, high cost of creating independent systems, management problems, updating and maintenance issues (Setareh et al. 2014). Solutions depend on the adoption of cloud computing technology in healthcare to analyze and provide patients' information from multiple EHR repositories accurately, securely and fast, to exchange HER as shown in Figure 5.3.

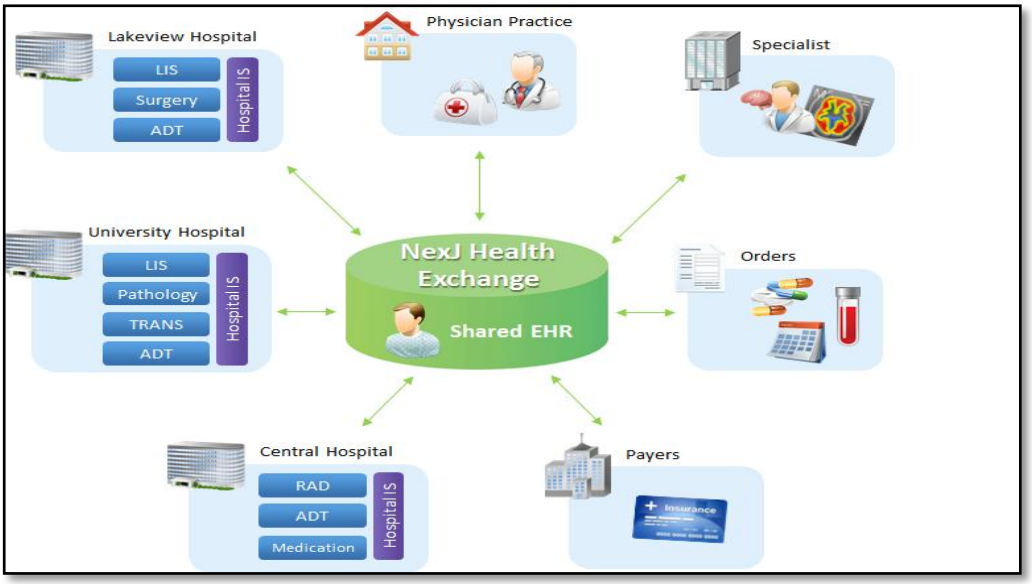


Figure 5.3. Cloud Health Exchange (Amar & Hany 2017).

5.2 Hajj and Umrah Mobile Healthcare System (HUMS) Architecture

The role of Information and Communication Technology (ICT) for large crowd events in smart cities will continue to grow with the growing service demands. An important service is Healthcare's use of smart mobile technologies for patients'. One of the important event is Hajj and Umrah for Muslims.

Hajj is a hard journey and requires great effort. The Hajj is an Islamic event once every year, while Umrah continues during a year. Muslims from all over the world arrive to Kingdom of Saudi Arabia (KSA) for the purpose of Hajj or Umrah or for seasonal work surrounding the holy cities and visiting the historical sites. The Ministry of Health established hospitals, health centers and employ qualified medical staff to provide all levels of healthcare to pilgrims such as emergencies for adults, women and children. To obtain an entry visa for Hajj and Umrah, it is necessary to get health requirements such as a vaccination certificate for Yellow fever, Meningococcal meningitis, Poliomyelitis, Seasonal influenza and Zika virus disease and Dengue (Al-Tawfiq & Memish 2014). In addition, health education is required to protect pilgrims against infectious and communicable diseases, such as chronic diseases, hygiene and general cleanliness, protection against food poisoning and heat exhaustion, managing with the crowds.

The following present the requirements with the help of a UML notation, to describe a high level system architecture includes main system stakeholders and use cases as follow:-

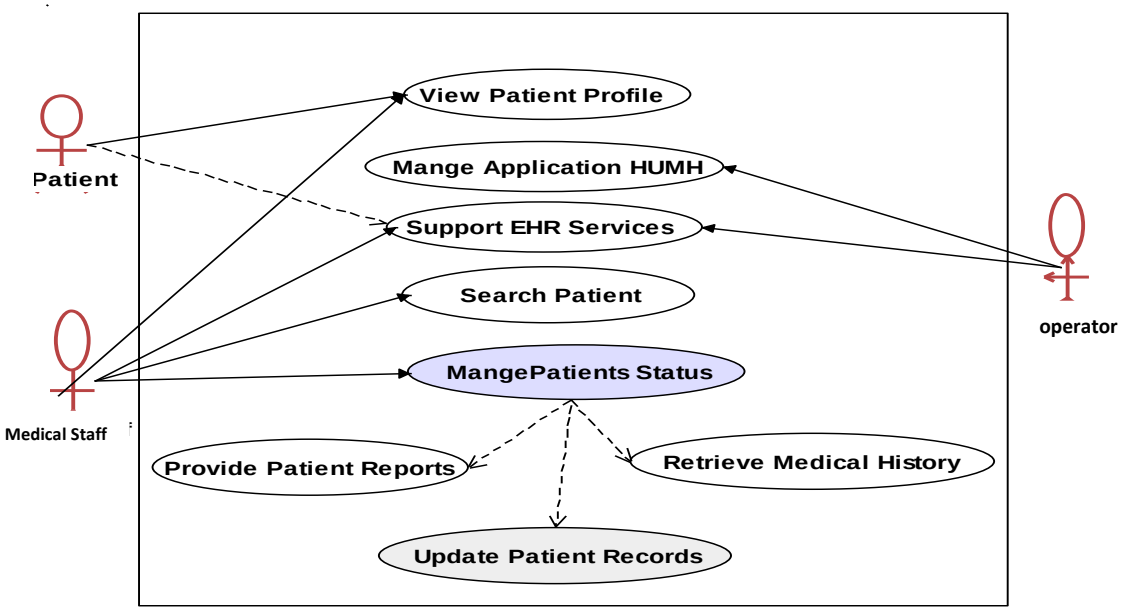


Figure 5.4 HUMH system use case diagram.

The different use case for actors' interaction with Hajj and Umrah Healthcare System (Amar & Hany 2017) are:-

- ❑ View patient profile: patient mange and edit his account login password, initial information includes (trip visa information, hosting agency, location address Hajji will be transported to, hosting agency and medical record information).
- ❑ Mange Application: used to maintain the records and provide off premises repository for health information.
- ❑ Support EHR Services: when a service used, the request firstly goes to a service gateway to meet the requirements and then sends the result to the user.
- ❑ Mange Patient Status: include all or part of retrieving medical history for what a patient is suffering from, provide patient reports for diagnostic test result and update patient statues, statues can be:-
    - Diagnose the problem to give the patient the proper treatment.
    - Approve: add or edit new patient status.
    - Review: for current patient status.

5.3 Enterprise Architecture (EA) Development Approach

Many enterprises have started to develop their architecture capabilities to utilizing IT resources and deliver business values, based on EA to achieve the vision and strategic goals by providing the enterprise views used to integrate technology and business.

There exists a gap between smart mobile technology and development needs for healthcare companies (Elgendy et al. 2014) that continue to possess increasing levels of complexity.

Our goal is to develop an architectural container-based design facilitates the process of getting healthcare services over cloud environment efficient and increase electronic collection of health information.

5.3.1 Enterprise Architecture (EA) Definition

Enterprise Architecture is a practical management approach, which offers improvement to an enterprise in many ways. An enterprise architecture is a conceptual blueprint that defines the structure and operation of an organization, determines how an organization can achieve effectively its current and future objectives (Pescosolido et al. 2016).

Enterprise architecture as a planning tool that can be used to design the new dimension of services which provides solutions to improve the productivity of enterprises. Without an enterprise architecture, the result could be a source of duplication, lack of integration, inefficient information exchange, or ineffective technology support (Ahsan et al. 2010).

5.3.2 TOGAF Development Approach

The Open Group Architecture Framework (TOGAF), describes required business and ICT architecture. In addition, Provides a step by step approach in building and implementing EA (Yuliana & Rahardjo 2016); Focuses on the process to develop and implement architectures.

TOGAF has long been recognized as a major reference in the field of enterprise architecture. It meets a real need: the need for a common framework that

will facilitate the capitalization and mutualization of architectural practices throughout a community (Desfray & Raymond 2014). More specifically,

- ❑ TOGAF is positioned as a generic method, which groups together a set of techniques focusing on the transformation of enterprise architecture.
- ❑ TOGAF can be applied to all types of architecture, including architecture based on enterprise resource planning systems.
- ❑ TOGAF provides a pragmatic view of enterprise architecture, while highlighting the central role of the organization.

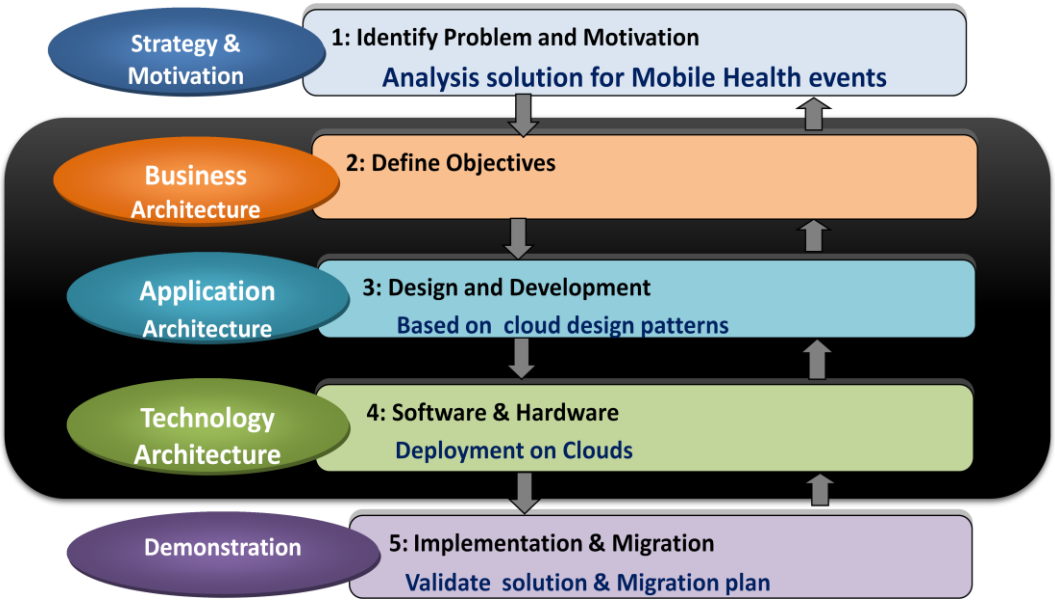


Figure 5.5 Enterprise Architecture development phases (Desfray & Raymond 2014).

Architecture Development Method (ADM) is the main entry point to the TOGAF. The aim of an ADM cycle is to successfully complete a transformation project, whose aim is to enable the enterprise to respond to a set of business goals. ADM presents the structure of the method with its phases and transitions (Desfray & Raymond 2014). The phases define the high-level work stages, which consume and provide products (deliverables). Each of the phases contributes to achieving determined strategic objectives. The development phases are:-

1) Business Architecture phase

   Based on the required processes to offer services for business applications, describing application components and interaction, logical data entities and relationships. Covers strategy, goals, business processes, functions, and organization.

2) Application Architecture phase

   Defines software components (applications and data) that support their interactions of business capabilities and functions. For Data architecture, dedicated to the organization and management of information.

3) Technology Architecture phase

   Describes the techniques and components deployed, as well as networks and the physical infrastructure upon which the applications and data sources run.

5.3.3 The ArchiMate Tool

The ArchiMate is a visual modeling standard for enterprise and solution architecture, published by The Open Group. An Open Group standard aligned with the TOGAF framework for enterprise architecture. The ArchiMate modeling language is dedicated to enterprise architecture modeling, provides a representation for models to support the complete architecture development cycle. Added concepts for modeling strategy, capability-based planning and related domains. Adapting existing standards, both in order to benefit from tools and to address a wide community of practitioners familiar with UML and BPMN (Desfray & Raymond 2014).

5.3.4 HUMS Architecture Design uses Enterprise Architecture Approach

We are focusing on application architecture phase. For application architecture phase, we are adapting TOGAF with architecture development method (ADM) to propose a high level HUMS architecture design view. Details architecture views of an ArchiMate model illustrated in Appendix B.

The development steps are:-

Step1: Mobile Application Architecture design

Need to create architecture that, minimizes costs and maintenance, requirements. Mobile application be structured as a multilayered application consisting of presentation, business, and data layers. Each layer is separation of concern with distinct features. Moreover, layer promotes usability , extendibility, maintainability and portability for mobile application (Meier et al. 2008).

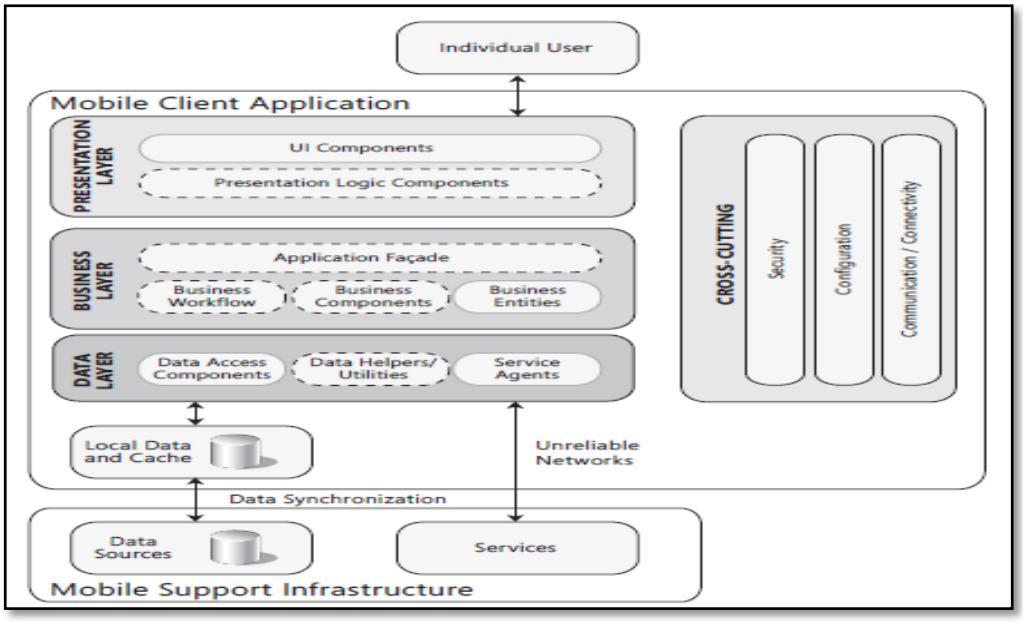


Figure 5.6 Layer Architecture for Mobile Application (Meier et al. 2008).

Step2: TOGAF Development

The TOGAF and Architecture Development Method (ADM) are used.

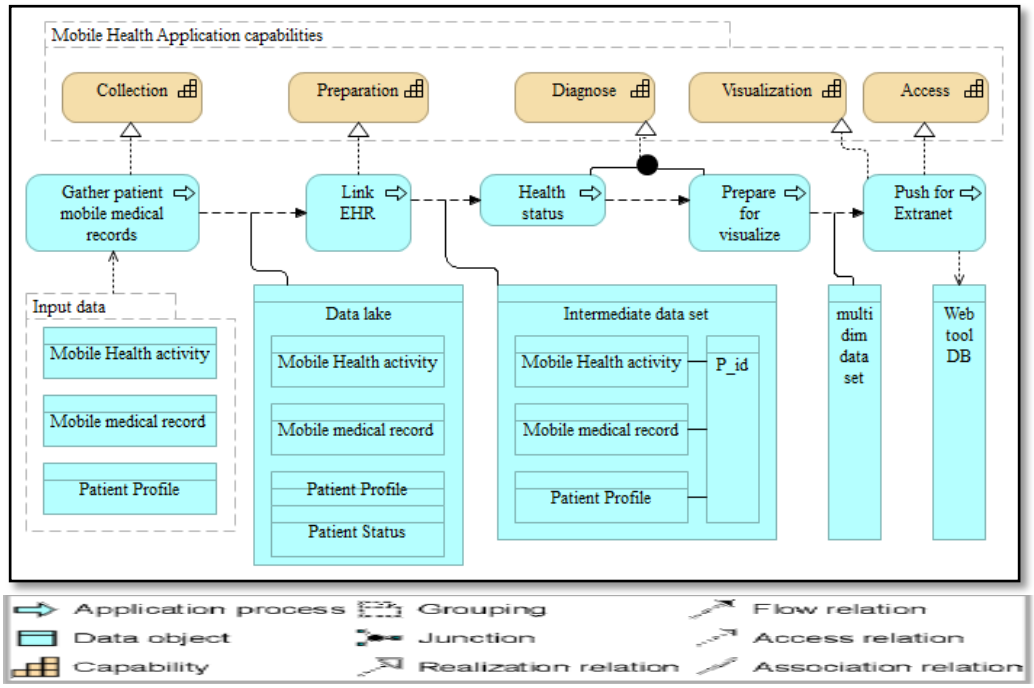


Figure 5.7 Enterprise Mobile healthcare application capabilities based design.

Based on the figure above, mobile healthcare application architecture consists of:-

- ❑ Application Capabilities: represent in:-
  - Collection: by extending access to information and processes to mobile, ensures that business continues to run efficiently.

- Preparation: prepares all data in the mobile featured service from an enterprise geographical database (EHR).
- Diagnostic Analytics: provide accurate analytics on mobile health app users, measure customer in app behavior events, can make data driven decisions to increase connection and monetization for Health app.
- Visualization: to give the best and accurate data, analyze multiple sources from anywhere with instance mobile and easily create integration.
- Access: describes a general Mobile Access (MA) capabilities solution to make work much easier, depending on implementing the solution to protect across either an un-trusted network or a network of a different classification level.

❑ Application Process: consist of:-

- Gather Data: Gather Patient Mobile Medical Record.
- Link Data: Link EHR.
- Health status: from the patient profile, determine health status scale, such as high risk, low risk, etc.
- Prepare for Visualization: describe any effort to help people understand the significance of data by placing it in a visual context.

❑ Data Objects: consist of the following:-

- Input data object: includes mobile health, activity, mobile medical record and patient profile. For enterprise goals, patient profile is a demographic data collected to build and generate a profile for the enterprise's patients to be used for medical purposes by specialists.
- Data lake object: using cloud data storage, to store an amount of online data, where data are remotely maintained, managed and backed up.
- Intermediate data set object: using patient identity for improving cloud efficiency.

5.4 Scenario-based Container Architecture Design Approach

We use our proposed method to evaluate a case study based on selecting application scenario for development.

5.4.1 HUMS Architecture

Hajj and Umrah Mobile Healthcare System (HUMS), provide services for online access to health records for specific patients, maintains the medical history records aim to assist medical staff to aid pilgrims by accessing their Electronic Health Records and gather data for patient's to provide accurate and quality services (Amar Ibrahim et al. 2017). Electronic Health Records are in the cloud database, shared from multiple healthcare organizations, allow patient to view their health record profile and for medical staff to view and mange patient healthcare status.

5.4.2 Independent Scenario-based Development

In this thesis, the metric of selecting application scenario depend on a high frequency of usage. The UML notation in Figure 5.8, for medical staff to retrieve patient medical records.

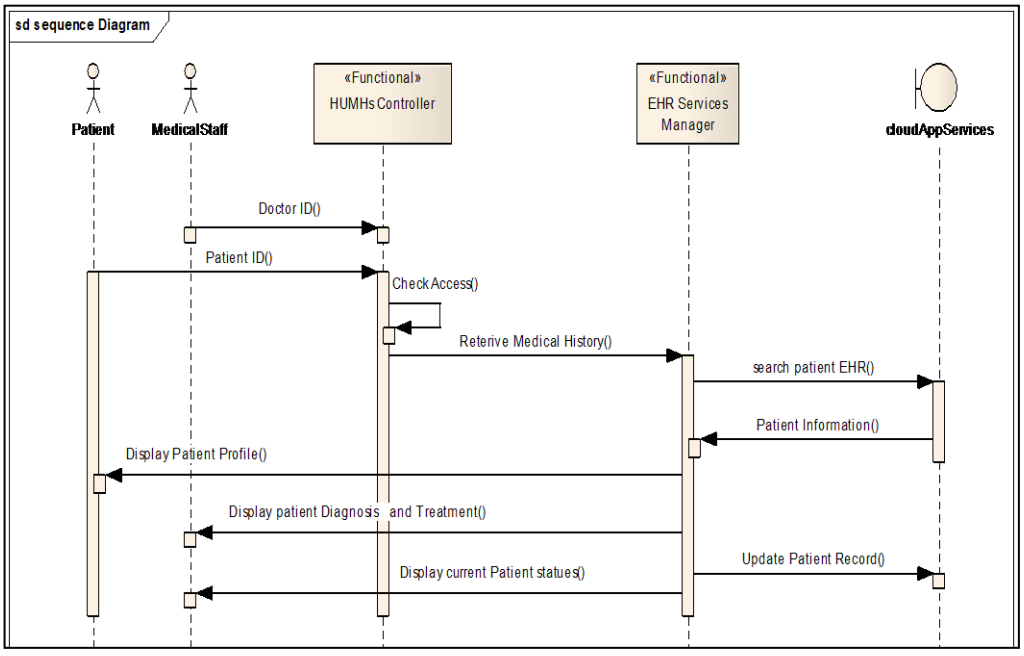


Figure 5.8 Scenario-based for the retrieve medical history use case.

The scenario represents the management of access to electronic health record, supports medical staff and professionals for better patient diagnosis, checking patient history records, and collaborating with other enterprises such as the patient's physician offices, Emergency Medical Services (EMS) and the World Health Organizations for data exchange over the cloud.

5.4.3 Scenario-based Development process

Scenario-based development process, based on Figure 3.8 for independent scenario-based container design is adopted. The steps as follows:-

Step 1: Select Architecture style:

Layer architecture is used as shown in Figure 5.6. And we used Hybrid cloud platform as portability criteria for deploying our application.

Step 2: Independent Scenario Development

Applying cloud patterns for scenario, based on (Amar Ibrahim 2016) to select cloud application patterns, that used with proposed container patterns for development.

Step 3: Container-based Architecture Design

Integrate step2 into a mobile cloud application container design.

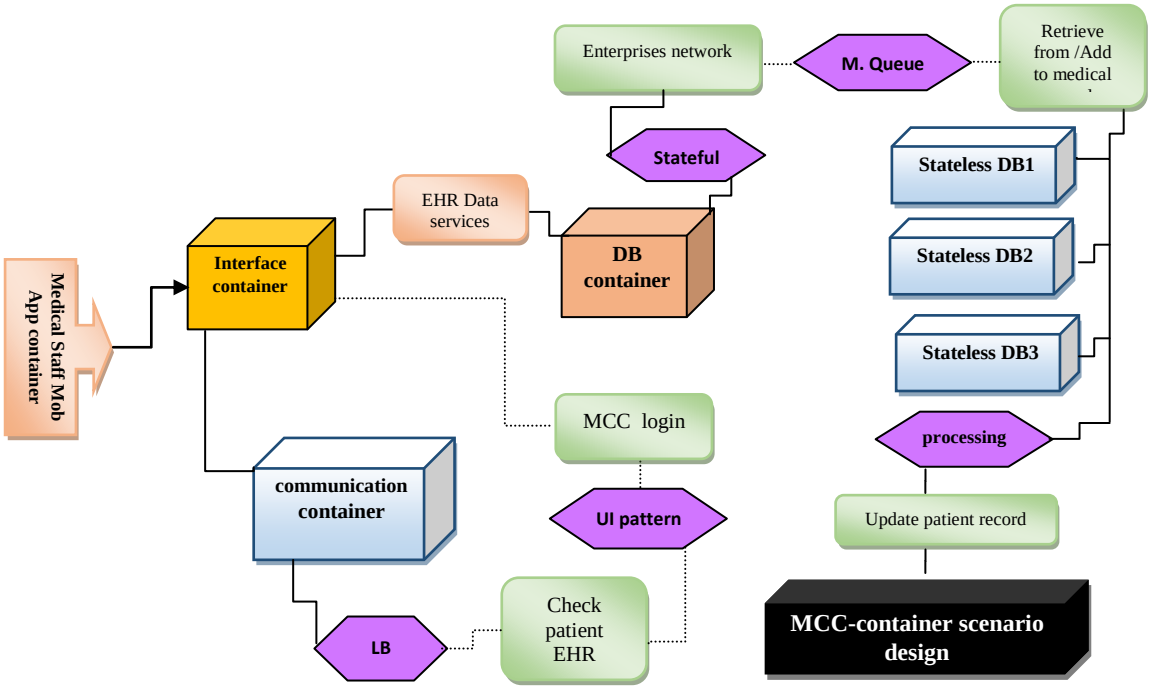


Figure 5.9 Annotated scenario-based container Architecure Design, for searching, retrieving / adding diagnose EHR.

Based on the high frequency of usage, scenario-based for retrieving medical history records has been selected. Medical staff for Hajj and Umrah push mobile application over a public network, based on communication container services to connect through a mobile network provider using interface container pattern as gateway, based on the load balancer to sending requests. To retrieve patient health information, we used database container for cloud data service providers, that store instances of EHR for enterprise health organizations with stateful pattern support. Then, based on authentication through data access for enterprise health network to check stateless enterprise database for retrieving patient information record scenario, in addition we also achieve patient records update by adding new diagnoses and treatment. The architecture provides flexible services that improve the patient's quality of life.

5.5 A comparison of two development approaches with different Design

An architecture design method provides support for a design process to meet the development goal. The proposed method is compared with Enterprise Architecture (EA) for development of portable cloud applications. We summarize the comparison as follows:-

- EA is a business oriented approach provide views, from different application's functionality to support business processes, while our container is a scenario-based approach for architecture evaluation.
- Both EA and Container-based approaches use model driven design (MDD) for a development based on capabilities.
- EA design is developed using TOGAF architecture, defined based on strategy to represent deployment and allow application components to communicate and exchange data, while our method based on iterative scenario generate based on requirements influence by certain stakeholder, that improve the interaction during system development process (Ionita et al. 2002).
- EA goals and capabilities take more time to facilitate architecture, while scenario is driven architecture to specific services in less time.
- Container-based design evolve container patterns with cloud application patterns techniques for development and deployment support, while EA

collaborate based on capability based planning for deployment target architecture.

In our experience, scenario-based assessment is particularly useful for development portable cloud application architecture design through develop scenarios.

5.6 Summery

This chapter presents a detailed explanation of the scenario-based container design that has been designed for Hajj and Umrah Mobile Healthcare System (HUMS) case study. The main objective is to validate our methodology container-based design for portable cloud application architecture.

Two papers have been published with support in the case study. The first one presents a Requirements model for HUMS event case study. The objective of the model is to improve health care procedures during Hajj and Umrah.

The second paper, used Enterprise Architecture to design standard architecture views for mobile healthcare generic events and integrate ICT services from different enterprises to retrieve health information, that improved accuracy and help to make healthcare duties more efficiently.

To validate HUMS case study, the development process for independent scenario-based container design approach has been designed for portable cloud application architecture design. In addition, another development approach for the case study has been designed, based on the TOGAF framework. The comparison for two development approach has been presented.

CHAPTER VI

MIGRATING ONLINE PET STORE APPLICATION TO CLOUD

6.1 Introduction

In the previous two chapters, the architectural design of portable cloud applications has been validated. This chapter elaborates the details of verifying deployment options, for case study migrating (an Online Pet store application) to the cloud. The systems were built on a cloud to benefit it's capabilities for enabling scalability and cost effectiveness, for migration to IaaS and PaaS environments.

There exist different cloud deployment options (CDOs) tools that can be used to verify cloud deployment options (CDOs), but appropriate support for comparing CDOs is missing. The CloudMIG simulation tool is used. CloudMIG from OMG's Architecture-Driven Modernization (ADM), that supports SaaS providers to migrate existing enterprise software systems to IaaS and PaaS-based cloud environments. The reason behind choosing a simulation tool, that it is much more powerful, clearer, and occupies with multiple cloud applications.

6.2 CloudMIG Xpress

CloudMIG Xpress is a Graphical User Interface (GUI) application that provides tool support for cloud migration approach. CloudMIG aims at supporting SaaS providers to semi-automatically migrate existing enterprise software systems for scalability and resource efficient PaaS and IaaS-based applications. A focus lies on the migration of client/ server enterprise systems as those often exhibits varying user demand. CloudMIG Xpress is developed to support (future) cloud users during the process of migrating software systems in a cloud environment and allows conformance checks between deployments and cloud offerings (Bergmayr et al. 2014).

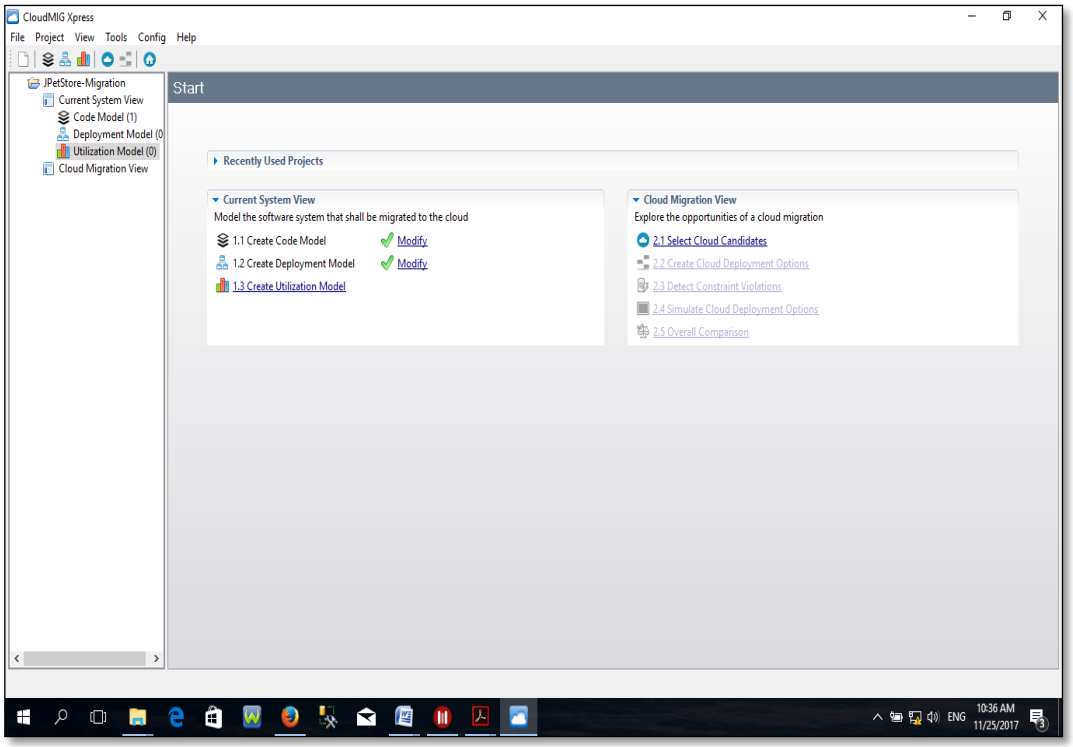


Figure 6.1 CloudMIG Xpress main screen.

The CloudMIG Xpress addresses those kinds of challenges and provides tool support for the comparison and planning phases to migrate software systems to PaaS or IaaS-based clouds.

6.2.1 CloudMIG Xpress Features

- Focuses on the technical challenges of a migration, also provide tool support for cloud migration approach.
- Bases on the Eclipse client platform and can be used with several databases.
- Extract code models from Java-based software
- Model the current system deployment
- Create workload profiles from real monitoring data
- Simulate various cloud deployment options
- Graph-based visualization of detecting cloud environment constraints (CECs).
- Compare the suitability of different cloud profiles (e.g., Costs and CECs).
- Estimate future costs, response times, and SLA violations

6.2.2 CloudMIG Xpress Activities

CloudMIG is composed of six main activities for migrating an enterprise system to a cloud environment. It provides model-driven generation of considerable parts of the system's target architecture (Frey & Hasselbring 2010b).

1) Extraction of a model which defines the architecture and an SMM model for relevant metrics.
2) Selection of a cloud provider, which is defined according to a Cloud Environment Metamodel.
3) Generation of the target architecture.
4) An Adaptation of the target architecture to accommodate user-specific requirements.
5) Evaluation of the target architecture using CloudSIM.
6) The transformation of the legacy system to match the target architecture.

6.3 The Pet Store Web-based Application Service

Pet Store is an open source web-based shop system for selling pets like birds and fishes. It is mainly written in Java and JSP and comprises only 24 Java classes and 1,432 lines of Java code. We selected pet store as a case study for evaluation.

6.3.1 Pet Store Overview:

Pet Store is an e-commerce application where customers can buy pet products in various categories online. The application has a Web site through which it presents an interface to customers.

The Pet Store is original Java based web application powered by Sun MicroSystems, was built to provide a working model of various components integrated together and also to demonstrate how different technologies could be used. The Java Pet Store application demonstrates certain models and design patterns (Nambiar 2005).

The Pet Store contains three sample applications:

- Java Pet Store: The main Blueprints application.
- Pet Store Administrator: The administrator module for the Java Pet Store.
- Blueprints Mailer: A mini-application that presents some of the Blueprints design guidelines in a smaller package.

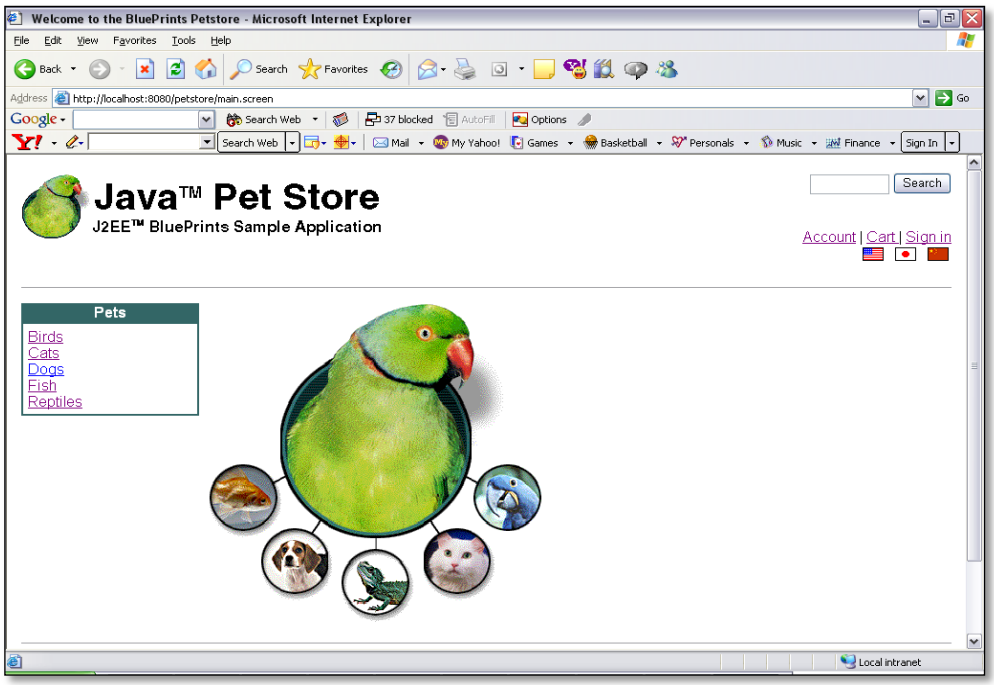


Figure 6.2 Home page for online Pet Store web application
The figure view how users interact with the application,
that allows customers to buy items online.

6.3.2 Requirement & Specifications

In the Pet Store other than the customer, there are other users. Each class of users has access to specific categories of functionality, and interact with it using specific user interface (Nambiar 2005). We describe the pet store architecture using UML notation, as in Figure 6.3.

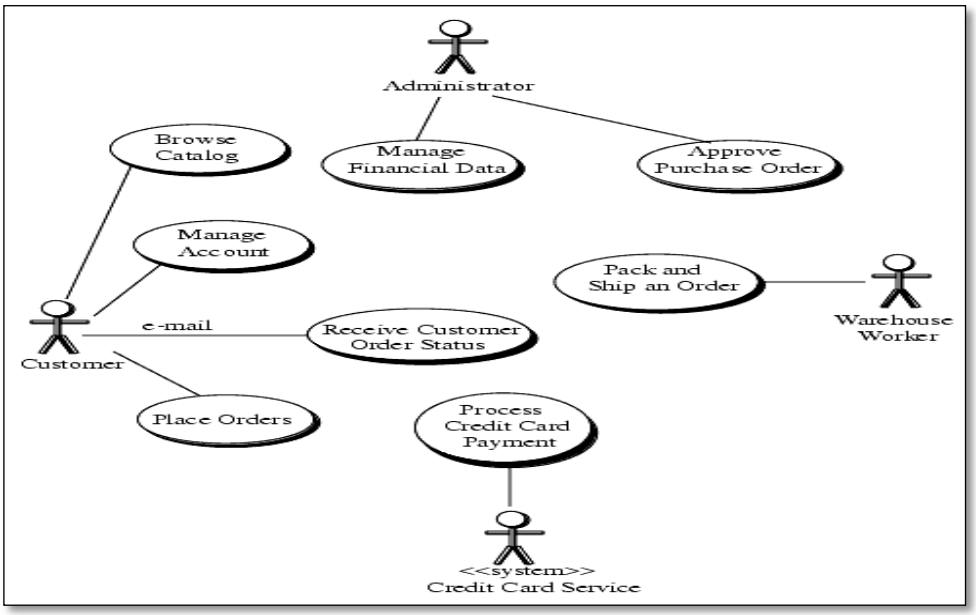


Figure 6.3 Use case diagram for Pet Store Application.

- Customer: A Customer will need some links on the page so as to get quick access to all tasks. It also requires a catalog and a search mechanism to get an organized view of items and providing a way to locate items. For the products all the detail showing their price, availability, picture should be available. A shopping cart to add and remove items, and a checkout bill showing the total order cost and billing information.
- Administrator: requires all the features of a customer, modifying options for the product and the inventory status, such as maintaining inventory is committed to the database and performing other managerial tasks, and associated businesses such as suppliers.
- Business: requirement and specifications are mostly related to security. Hence user authentication is important so that a user is identified to access a protected area. Some kind of user information, such as a credit card number, must be transmitted confidentially to the application and some kind of user administrations must be present for the growing number of users.

6.3.3 Use case for customer Interactions

A customer connects to the application's home page. The customer can browse through the catalog to see a list of all the products or search for products, the customer can sign into the application by providing an account identifier and a password. When the customer signs in, the application can recall information about the customer such as a preferred shipping address and billing information, buying preferences, and so on.

The customer then selects a particular product in the list resulting in the detailed information like image of the product along with pricing information of the product. When the customer decides to purchase using the shopping cart to order the items in the shopping cart at any time. When the customer confirms the order, the application begins to gather shipping and billing information for the order. First, it presents a form, where the customer can enter shipping information (Nambiar 2005). Finally the customer confirms the order and the application accepts the order for delivery. A receipt including a unique order number and other order details is presented to the customer. The application validates credit card and other information, updates its inventory database, and optionally sends a confirmation message via email.

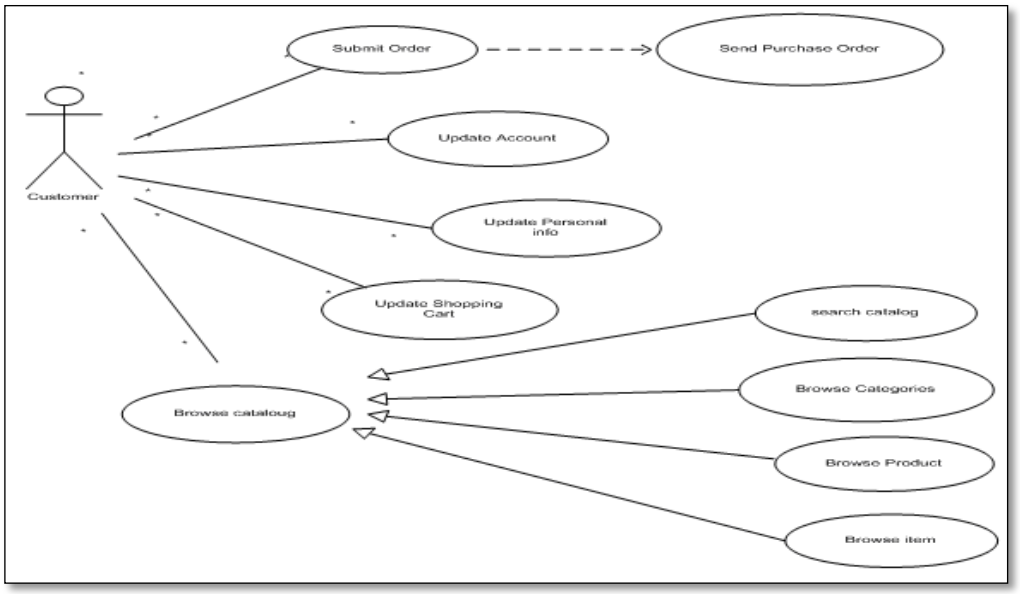


Figure 6.4 Use Case between the Customer and the Web Site.

Table 6.1 Summary for customer actions and description.

ACTOR	ACTION	DESCRIPTION
Customer	Browse catalog	Each category has several products associated with it
Customer	Browse Detail	Each product variant has detailed view that displays the product description, a product image, price, and the quantity in stock.
Customer	Browse Item	Each Item is viewed.
Customer	Browse Products	If we now select a product the application will display all variants of the product.
Customer	Update Cart	This allows the user to manipulate the shopping cart (add, remove, and update line items).
Customer	Update Personal Info	This allows user to update his personal information
Customer	Update Account	The checkout page displays the shopping cart.
Customer	Submit Order	The billing and the shipping addresses are displayed.
Customer	Purchase Order	The final step wherein the order is committed to the database.

6.3.4 Pet Store Architectural Design

The Pet Store application is based on the Model View Controller pattern, separates data presentation, data representation, and application behavior. The architecture consists of three components: the model, the view and the controller.

The model encapsulates the core data and business functionality of the application. The view encapsulates the task of displaying the information to the user i.e. data presentation. Each view has an associated controller, which encapsulates the interaction of the user with the system, and abstracts the system behavior by sending service requests to the model for some operation on the data. By separating business and control logic from data presentation, the architecture provides the flexibility to handle such application complexity. The architecture provides flexibility, reusability, testability, extensibility, and clear design roles for application components (Nambiar 2005).

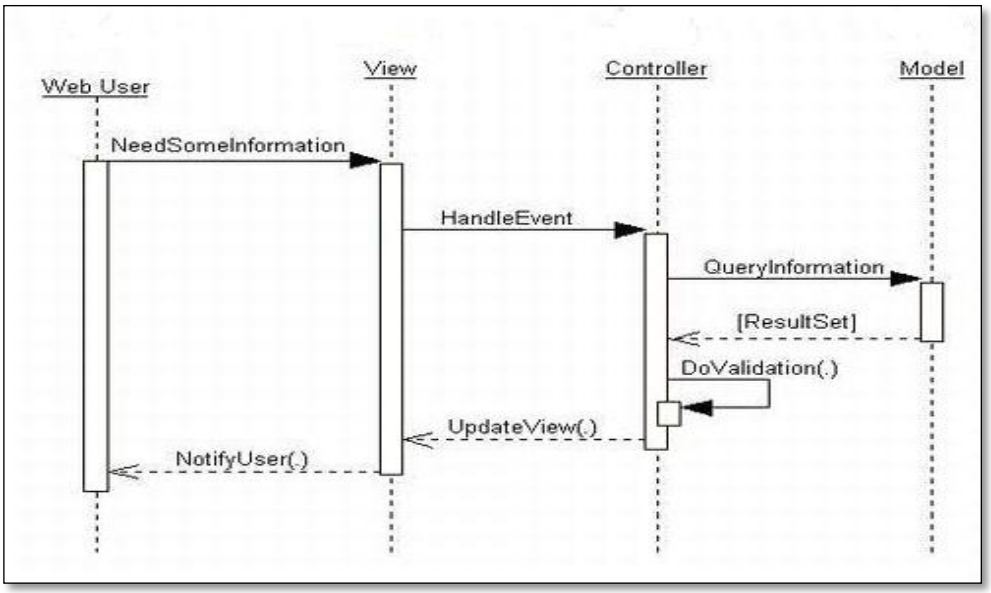


Figure 6.5 Sequence diagram for Pet store using MVC model, adpated from (Nambiar 2005).

6.4 Verification Method for migrating Pet Store to the cloud

The objective of this section, to answer the third research question, using and explores the CloudMIG Xpress simulation tool to verify cloud application portability on multiclouds. As mentioned in chapter III, a framework for cloud application development. For deployment phase the developed Architecture design needs to verify various deployment options to achieve portability on multiple cloud platforms.

In the following subsections, we will discuss the case study verification through the stage's steps, using various approaches for simulation to estimated attributes for deployment as follows:-

6.4.1 Select cloud Candidates

Portability achieved by selecting a cloud profiles for Public cloud environments Google App for Java, Microsoft Windows Azure's virtual machine, and Eucalyptus cloud.

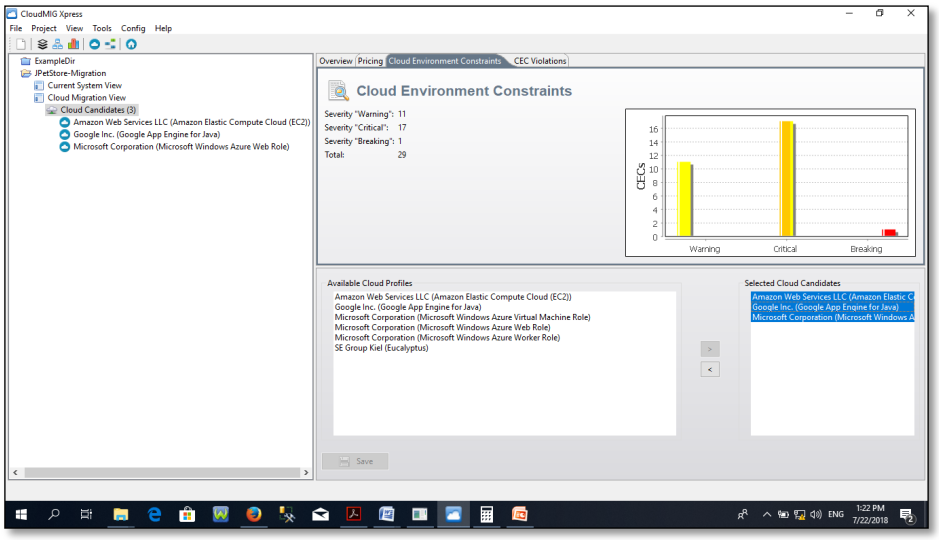


Figure 6.6 The selection of Candidate cloud environments.
Cloud profiles selection, used to evaluate the possibilities of a migration to a specific provider or many providers.

6.4.2 Create Utilization Model (workload profile)

Workload profiles specify a particular user demand. There are two approaches, workload synthetic and from monitoring data. We used the first approach. We used, workloads from synthetic Data approach to be correlated with the computational capacity of the machine that was used to run the software.

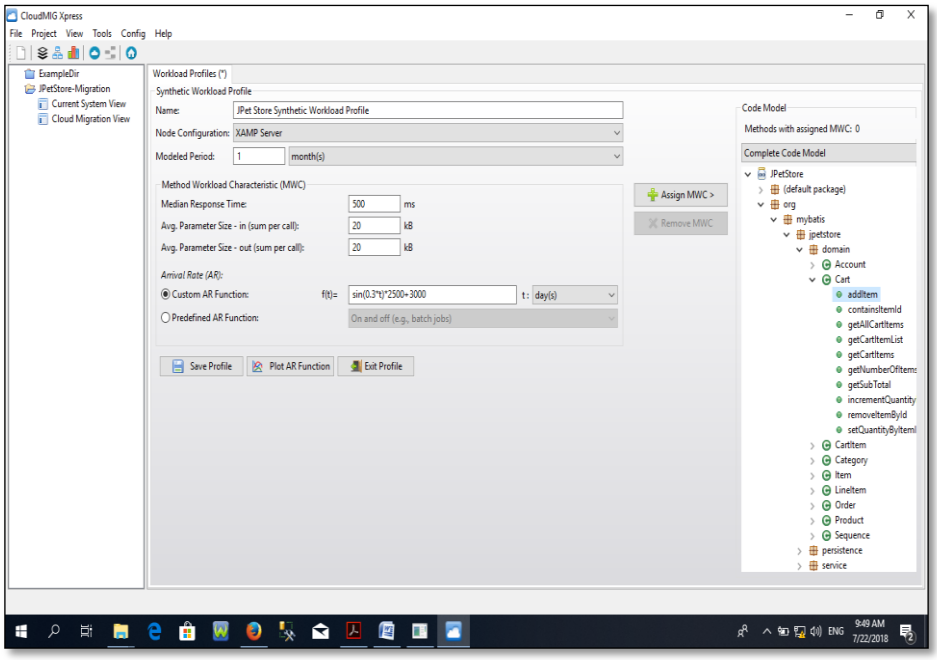


Figure 6.7 Workload profile using synthetic approach.

- Model period: time period spanning the modeled time framework and a unit of time.
- Method workload characteristic (MWC) : describes the characteristic values for modeling a user demand for one or more procedures, functions, or methods.
- Arrival Rate (AR) function: to specify the workload, get assigned to one or more methods and take a time designation as their argument. AR function has Custom and Predefined method.

6.4.3 Create Cloud Deployment Option (CDO)

Cloud deployment options (CDOs), comprises a combination of a specific cloud environment, deployment architecture, and runtime reconfiguration rules for dynamic resource scaling (Frey et al. 2013).

Methods to create CDOs are either manual or automatic that are comprised of the code elements of virtual machines included in that cloud environment, and reconfiguration rules that allow to specify starting and stopping criteria for virtual machines.

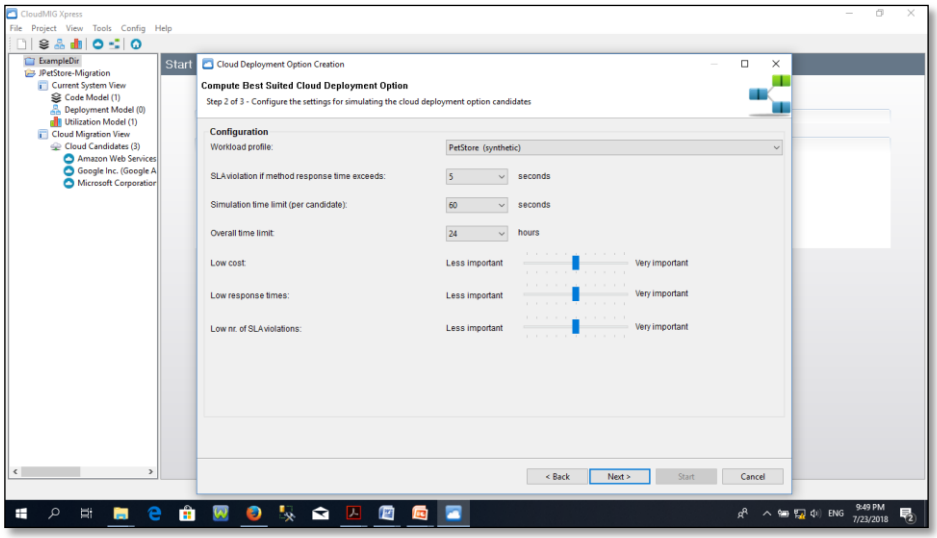


Figure 6.8 Automatic Optimized method used to create CDOs.

Our created workload profile (Pet Store (synthetic)) is used as an input to evaluate a CDO with selected mode configuration. CDO determines which cloud environment, cloud resource types, deployment architecture, and runtime reconfiguration rules for exploiting the cloud's elasticity should be used

6.4.4 CDO Simulation

The optimization process using Cloud Deployment Option Simulator (CDOSim), and method (Optimized) to evaluate CDOs, and computes potential costs, response times, and number of SLA violations. Automatic Optimized creating CDOs for IaaS-based cloud profiles that are selected as the input for the simulator.

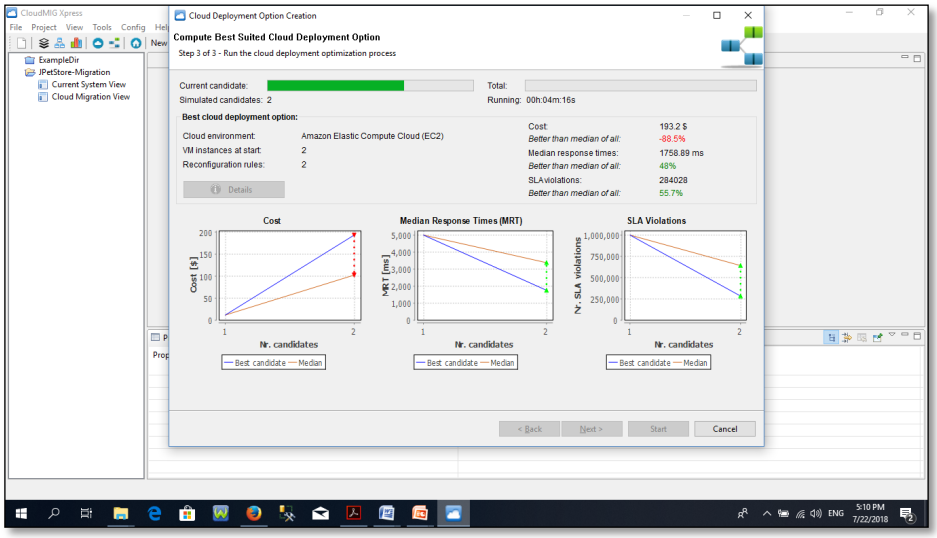
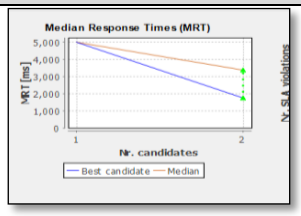
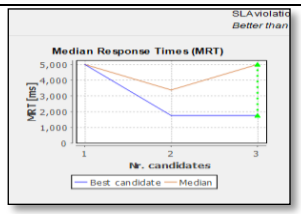
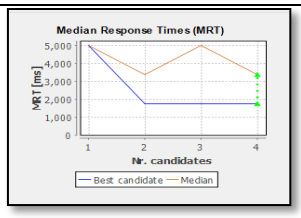
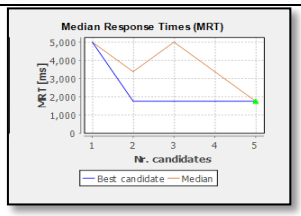
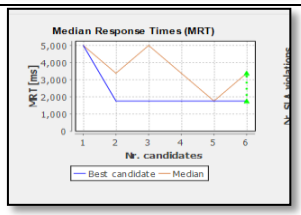


Figure 6.9 Different optimization process parameters simulated for case study data using many (cloud environment) candidates.

6.5 Results Discussion

This section shows the results of the experiments which were carried out based on CloudMIG Xpress methods that directly reflect the verification of cloud deployment options (CDO). The simulation verifying of a case study against different deployment locations and different attributes scale (Cost, Medium response time, and SLA violation). Based on the results, the evaluation criteria Medium Response Time (MRT) was used. Furthermore, the detailed analysis of results is also provided, as presented in Table 6.2.

Table 6.2 Median Response Time (MRT) Simulation Results for Pet Store data.

#	Simulation Result	Cloud Candidates	MRT (Ms)	Better than all
1		2	1758.89ms	48%
2		3	1758.89ms	64.8%
3		4	1753.46ms	48.1%
4		5	1753.46ms	0.3%
5		6	1753.46ms	48.1%

The results of the experiment verification are shown in Table 6.2 with different simulation for response time of the case study, and illustrates the using many cloud candidates. Even though these factors depend on network locality and traffic congestion, the main purpose is to show the difference in response time depending on different conditions. For our case response time includes the send a request from a client, the time to redirect the request by a load balancer (if there are several role instances), the time to process it by the application, and to get a response back from the server. Method cope with all types of cloud environments for deployment.

6.6 Summery

This chapter has presented the evaluation for case study migrating Online Pet Store application on the cloud using a simulation tool for verifying deployment options. Requirements and architectural design for case study have been introduced, to give a detailed understanding of the cloud application migration.

The CloudMIG Xpress tool has been implemented to verify and support migrating a case study to the cloud, because there is a lack of tool support to automate migration tasks. Cloud application portability achieved by selecting multiple candidate PaaS profiles, each profile includes the corresponding cloud resources, pricing model, and CEC definitions.

The CloudMIG tool has been simulated for cloud deployment options. The verification results show different deployment option parameters, we focus on identifying median response time (MRT), which compared with multiple candidate cloud profiles.

. There are some drawbacks for the CloudMIG Xpress simulation tool, that the current version not composed of multiple languages for systems integration. In addition, Workload profile for monitoring not support interleaved data from several nodes. Since the tool is available, but its source code has not been published. Moreover, tool lack of consideration for cloud database migration.

Overall, the CloudMig simulation tool is satisfactory and encouraging, moreover, provide important insights into future work to support different programming language to handle the migration of all types of application.

CONCLUSION AND FUTURE WORK

One of the important issues that need to be widely adopted against cloud computing is a portability affect the cloud applications. Cloud application portability is classified into two parts of vertical PaaS portability and horizontal PaaS portability, our concern is to architecting and developing cloud applications to be ported to Multiclouds platform.

An effective development framework for portable cloud applications has been proposed, and applied for this purpose. The proposed framework for cloud application architecture has been developed, composed of four phases: strategic planning, architecture design, deployment, and evaluation. To come up with this, a comparative survey involving several studied for cloud application architecture development has been conducted.

The research began with the review of the concepts of cloud computing, portability, and related works on cloud application portability. The architecture design phase chosen as a research area, to develop portable cloud application architecture design that can be used to achieve application portability on multiple cloud platforms.

To achieve the objective, scenarios-based container design methodology and new container pattern are performed.

In this research, efforts at designing portable cloud application architecture include: propose development process to guide in designing containerized portable cloud applications, propose and uses new container design patterns, and present scenarios-based container approach, that group application behavior to several application scenarios in a container to describe architecture design.

Accordingly, the evaluation of the proposed framework has been achieved through validate two case studies uses scenario-based. The first case study on a web-based for migrating Student Academic Records to the cloud has been developed, using multiple scenario development process, while the other case study Hajj and Umrah Mobile Healthcare System (HUMS) has been developed based on

independent scenario development process. The proposed container patterns and cloud application design patterns applied to support the development of portable cloud application architecture design. In addition, the second case study HUMS, has been developed using Enterprise Architecture , based on TOGAF approach.

Assessing the deployment of the proposed framework for cloud application portability has been verified. The verification of deployment has been approved for case study migrating Online Pet Store application on the cloud using the CloudMIG Xpress simulation tool, results verification for a Median Response Time (MRT) has been achieved.

Future Work

This section presents some recommendations for future work. While many issues related to this area of research remain to be explored, moreover, this thesis could be extended in several directions. These issues and directions can be addressed as follows:

- Develop Automate migration containers for applications to migrate from cloud to cloud as needed to support the application's requirements.
- Focus on maintainability during application development to facilitate easier portability between different architectural platforms (DevOps).
- The developmental design process used various methods and techniques, need more attention to be standard for easily developing.
- Enterprise Architecture (EA) development approach can be used in a wide range of health area, such as remote patient monitoring. We are planning to propose an adaptive model to adopt variations of different events**.**
- In case study student academic result records, college need to benefit from integrated student academic results, by reuse the existing application component to develop new services, that can assist academic officer to gain online student's transcript, and certificate services from the system. Need for adopting Software Product Line (SPL) Development.

REFERENCES

Adewojo, A.A. et al., 2015. Cloud deployment patterns: Migrating a database driven application to the cloud using design patterns. In *Proceedings of the World Congress on Engineering and Computer Science*.

Ahmad, A., Altamimi, A.B. & Alreshidi, A., 2017. TOWARDS ESTABLISHING ACatalogue OF PATTERNS FOR ARCHITECTING MOBILE CLOUD SOFTWARE. *Computer Science & Information Technology*, p.19.

Ahsan, K., Shah, H. & Kingston, P., 2010. Healthcare modelling through enterprise architecture: a hospital case. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. IEEE, pp. 460–465.

Al-Tawfiq, J.A. & Memish, Z.A., 2014. Mass gathering medicine: 2014 Hajj and Umra preparation as a leading example. *International Journal of Infectious Diseases*, 27, pp.26–31.

Anderson, J.Q. & Rainie, H., 2010. *The future of cloud computing*, Pew Internet & American Life Project Washington, DC.

Antoniades, D. et al., 2015. Enabling Cloud Application Portability. In *Utility and Cloud Computing (UCC), 2015 IEEE/ACM 8th International Conference on*. IEEE, pp. 354–360.

Ardagna, D., 2015. Cloud and multi-cloud computing: current challenges and future applications. In *Principles of Engineering Service-Oriented and Cloud Systems (PESOS), 2015 IEEE/ACM 7th International Workshop on*. IEEE, pp. 1–2.

Ardagna, D. et al., 2012. Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds. In *Proceedings of the 4th International Workshop on Modeling in Software Engineering*. IEEE Press, pp. 50–56.

Bass, L., Clements, P. & Kazman, R., 2003. *Software architecture in practice*, Addison-Wesley Professional.

Bergmayr, A. et al., 2014. Cloud Modeling Languages by Example. *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pp.137–146. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6978602 [Accessed May 26, 2015].

Brandle, C. et al., 2014. *Cloud computing patterns of expertise*, IBM Redbooks.

Bruneliere, H., Cabot, J. & Jouault, F., 2010. Combining model-driven engineering and cloud computing. In *Modeling, Design, and Analysis for the Service Cloud-MDA4ServiceCloud'10: Workshop's 4th edition (co-located with the 6th European Conference on Modelling Foundations and Applications-ECMFA 2010)*.

Burns, B. & Oppenheimer, D., 2016. Design Patterns for Container-based Distributed Systems. In *HotCloud*.

Cheng, F.-C. & Lai, W.-H., 2012. The impact of cloud computing technology on legal infrastructure within internet—focusing on the protection of information privacy. *Procedia Engineering*, 29, pp.241–251.

Council, C.S.C., 2014. Interoperability and portability for cloud computing: a guide.

Council, C.S.C., 2016. Migrating applications to public cloud services: roadmap for success.

Cretella, G. & Di Martino, B., 2012. Towards automatic analysis of cloud vendors APIs for supporting cloud application portability. In *Complex, intelligent and software intensive systems (CISIS), 2012 sixth international conference on*. IEEE, pp. 61–67.

Desfray, P. & Raymond, G., 2014. *Modeling enterprise architecture with TOGAF: A practical guide using UML and BPMN*, Morgan Kaufmann.

Dimitrov, D., 2015. Towards cloud application architectural patterns: transfer, evolution, innovation and oblivion.

Dua, R., Raja, A.R. & Kakadia, D., 2014. Virtualization vs containerization to support paas. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, pp. 610–614.

Duarte, J.M.G., Cerqueira, E. & Villas, L.A., 2015. Indoor patient monitoring through Wi-Fi and mobile computing. In *New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on*. IEEE, pp. 1–5.

Eldein, A.I.E.S., Ammar, H.H. & Dzielski, D.G., 2018. Enterprise architecture of mobile healthcare for large crowd events. *2017 6th International Conference on Information and Communication Technology and Accessbility, ICTA 2017*, 2017–Decem, pp.1–6.

Elgendy, M.A., Shawish, A. & Moussa, M.I., 2014. MCACC: New approach for augmenting the computing capabilities of mobile devices with Cloud Computing. In *Science and Information Conference (SAI), 2014*. IEEE, pp. 79–86.

Erl, T., Cope, R. & Naserpour, A., 2015. *Cloud computing design patterns*, Prentice Hall New York, NY.

Fehling, C. et al., 2011. An architectural pattern language of cloud-based applications. In *Proceedings of the 18th Conference on Pattern Languages of Programs*. ACM, p. 2.

Fehling, C. et al., 2014. *Cloud computing patterns: fundamentals to design, build, and manage cloud applications*, Springer.

Fehling, C. et al., 2012. Pattern-based development and management of cloud applications. *Future Internet*, 4(1), pp.110–141.

Ferry, N. et al., 2013. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*. IEEE, pp. 887–894.

Frey, S., Fittkau, F. & Hasselbring, W., 2013. Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In *Software Engineering (ICSE), 2013 35th International Conference on*. IEEE, pp. 512–521.

Frey, S. & Hasselbring, W., 2010a. Model-Based Migration of Legacy Software Systems into the Cloud: The CloudMIG Approach.

Frey, S. & Hasselbring, W., 2010b. Model-based migration of legacy software systems to scalable and resource-efficient cloud-based applications: The cloudmig approach.

Gavras, A. et al., 2004. Towards an MDA-based development methodology. In *European Workshop on Software Architecture*. Springer, pp. 230–240.

Ghanam, Y., Ferreira, J. & Maurer, F., 2012. Emerging issues & challenges in cloud computing—a hybrid approach. *Journal of software engineering and applications*, 5(11), p.923.

Gitzel, R., Korthaus, A. & Schader, M., 2007. Using established Web Engineering knowledge in model-driven approaches. *Science of Computer Programming*,

66(2), pp.105–124.
Gohad, A., Ponnalagu, K. & Narendra, N.C., 2012. Model driven provisioning in multi-tenant clouds. In *SRII Global Conference (SRII), 2012 Annual*. IEEE, pp. 11–20.
Gonidis, F., Paraskakis, I. & Kourtesis, D., 2012. Addressing the challenge of application portability in cloud platforms. In *7th South-East European Doctoral Student Conference*. pp. 565–576.
Gorton, I., 2006. *Essential software architecture*, Springer Science & Business Media.
Goyal, S., 2014. Public vs private vs hybrid vs community-cloud computing: a critical review. *International Journal of Computer Network and Information Security*, 6(3), p.20.
Graaf, B., 2007. *Model-driven evolution of software architectures*, IEEE.
Grundy, J., 2012. Software Engineering for the Cloud. In *IEEE Sof*. IEEE, pp. 26–30.
Haitzer, T. & Zdun, U., 2014. Semi-automated architectural abstraction specifications for supporting software evolution. *Science of Computer Programming*, 90, pp.135–160.
Hamdaqa, M., Livogiannis, T. & Tahvildari, L., 2011. A Reference Model for Developing Cloud Applications. In *CLOSER*. pp. 98–103.
Hill, Z. & Humphrey, M., 2010. CSAL: A cloud storage abstraction layer to enable portable cloud applications. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, pp. 504–511.
Hogan, M. et al., 2011. Nist cloud computing standards roadmap. *NIST Special Publication*, 35, pp.6–11.
Huang, D., 2011. Mobile cloud computing. *IEEE COMSOC Multimedia Communications Technical Committee (MMTC) E-Letter*, 6(10), pp.27–31.
Hutchinson, J.E., 2011. An empirical assessment of model driven development in industry.
Ibrahim, A. & Eldein, E.S., 2016. Pattern Oriented Analysis for Web Based Applications on Cloud. , 9(1), pp.1–12.
Ibrahim, A., Eldein, E.S. & Ammar, H.H., 2017. Requirements Model For Hajj and Umrah Mobile Healthcare System (HUMHS). , 5(1), pp.53–62.
Ibrahim, A. & Hany, A., 2015. Model-Driven Architecture for Cloud Applications Development , A survey. , 4(9), pp.698–705.
Ionita, M.T., Hammer, D.K. & Obbink, H., 2002. Scenario-based software architecture evaluation methods: An overview. In *Workshop on methods and techniques for software architecture review and assessment at the international conference on software engineering*. pp. 19–24.
Jain, R. et al., 2018. Container provisioning based on communications patterns between software components.
Jamshidi, P., Pahl, C. & Chinenyeze, S., 2015. Service-Oriented Computing - ICSOC 2014 Workshops. , 8954, pp.6–19. Available at: http://link.springer.com/10.1007/978-3-319-22885-3.
Jiang, S. & Mu, H., 2011. Design patterns in object oriented analysis and design. In *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*. IEEE, pp. 326–329.
Kang, H., Le, M. & Tao, S., 2016. Container and microservice driven design for cloud infrastructure devops. In *Cloud Engineering (IC2E), 2016 IEEE International Conference on*. IEEE, pp. 202–211.

Karakostas, B., 2008. *Engineering Service Oriented Systems: A Model Driven Approach: A Model Driven Approach*, IGI Global.

Kleppe, A.G. et al., 2003. *MDA explained: the model driven architecture: practice and promise*, Addison-Wesley Professional.

Kolb, S. & Wirtz, G., 2017. Data Governance and Semantic Recommendation Algorithms for Cloud Platform Selection. In *Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on*. IEEE, pp. 664–671.

Kottari, V. et al., 2013. A survey on mobile cloud computing: Concept, applications and challenges. *International Journal of Advances and Innovative Research*, 2(3), pp.487–492.

Kozhirbayev, Z. & Sinnott, R.O., 2017. A performance comparison of container-based technologies for the cloud. *Future Generation Computer Systems*, 68, pp.175–182.

Kuo, Y.-H. & Hsu, W.H., 2017. De-hashing: server-side context-aware feature reconstruction for mobile visual search. *IEEE Trans. Circuits Syst. Video Techn*, 27(1), pp.139–148.

Kuyoro, S.O., Ibikunle, F. & Awodele, O., 2011. Cloud computing security issues and challenges. *International Journal of Computer Networks (IJCN)*, 3(5), pp.247–255.

Lee, Y.S. et al., 2015. Hybrid cloud service based healthcare solutions. In *Advanced Information Networking and Applications Workshops (WAINA), 2015 IEEE 29th International Conference on*. IEEE, pp. 25–30.

Lin, J., Lin, L.C. & Huang, S., 2016. Migrating web applications to clouds with cloud-based MVC framework. In *Computer, Consumer and Control (IS3C), 2016 International Symposium on*. IEEE, pp. 1039–1042.

Lin, Y. & Chang, P., 2012. Evaluation of system reliability for a cloud computing system with imperfect nodes. *Systems Engineering*, 15(1), pp.83–94.

Linthicum, D.S., 2016. Moving to autonomous and self-migrating containers for cloud applications. *IEEE Cloud Computing*, 3(6), pp.6–9.

Liu, F. et al., 2011. NIST cloud computing reference architecture. *NIST special publication*, 500(2011), pp.1–28.

Lo'ai, A.T. et al., 2016. Mobile cloud computing model and big data analysis for healthcare applications. *IEEE Access*, 4, pp.6171–6180.

Di Martino, B., Cretella, G. & Esposito, A., 2015. *Cloud Portability and Interoperability: Issues and Current Trends*, Springer.

Di Martino, B., Esposito, A. & Cretella, G., 2015. Semantic representation of cloud patterns and services with automated reasoning to support cloud application portability. *IEEE Transactions on Cloud Computing*.

McNatt, W.B. & Bieman, J.M., 2001. Coupling of design patterns: Common practices and their benefits. In *Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International*. IEEE, pp. 574–579.

Meier, J.D. et al., 2008. Mobile application architecture guide. *Pattern & Practices, Microsoft*.

Mell, P. & Grance, T., 2011. The NIST definition of cloud computing.

Mens, T. & Van Gorp, P., 2006. A taxonomy of model transformation. *Electronic Notes in Theoretical Computer Science*, 152, pp.125–142.

Moscato, F., Di Martino, B. & Aversa, R., 2012. Enabling model driven engineering of cloud services by using mosaic ontology. *Scalable Computing: Practice and Experience*, 13(1), pp.29–44.

Murugesan, S. & Bojanova, I., 2016. *Encyclopedia of Cloud Computing*, John Wiley & Sons.

Nambiar, R., 2005. Java PetStore: A Case Study.

Naumann, S. et al., 2015. Sustainable software engineering: Process and quality models, life cycle, and social aspects. In *ICT Innovations for Sustainability*. Springer, pp. 191–205.

Opara-Martins, J., 2017. A decision framework to mitigate vendor lock-in risks in cloud (SaaS category) migration.

Opara-Martins, J., Sahandi, R. & Tian, F., 2014. Critical review of vendor lock-in and its impact on adoption of cloud computing. In *International Conference on Information Society (i-Society 2014)*. IEEE, pp. 92–97.

Pahl, C. et al., 2017. Cloud container technologies: a state-of-the-art review. *IEEE Transactions on Cloud Computing*.

Pahl, C., 2015. Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3), pp.24–31.

Pahl, C. & Lee, B., 2015. Containers and clusters for edge cloud architectures--A technology review. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*. IEEE, pp. 379–386.

Pescosolido, L. et al., 2016. An IoT-inspired cloud-based web service architecture for e-health applications. In *Smart Cities Conference (ISC2), 2016 IEEE International*. IEEE, pp. 1–4.

Petcu, D. & Vasilakos, A. V, 2014. Portability in clouds: approaches and research opportunities. *Scalable Computing: Practice and Experience*, 15(3), pp.251–270.

Pilar, M., Simmonds, J. & Astudillo, H., 2014. Semi-automated tool recommender for software development processes. *Electronic Notes in Theoretical Computer Science*, 302, pp.95–109.

Pinelle, D. & Gutwin, C., 2005. A groupware design framework for loosely coupled workgroups. In *ECSCW 2005*. Springer, pp. 65–82.

Rezaei, R. et al., 2014. A semantic interoperability framework for software as a service systems in cloud computing environments. *Expert Systems with Applications*, 41(13), pp.5751–5770.

Setareh, S. et al., 2014. A cloud-based model for hospital information systems integration. In *Telecommunications (IST), 2014 7th International Symposium on*. IEEE, pp. 695–700.

Sharma, A., Kumar, M. & Agarwal, S., 2015. A Complete Survey on Software Architectural Styles and Patterns. *Procedia Computer Science*, 70, pp.16–28.

Sharma, R. & Sood, M., 2011. Enhancing cloud SaaS development with model driven architecture. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 1(3), pp.89–102.

Shawish, A. & Salama, M., 2014. Cloud computing: paradigms and technologies. In *Inter-cooperative collective intelligence: Techniques and applications*. Springer, pp. 39–67.

Shin, M.E. & Gomaa, H., 2007. Software requirements and architecture modeling for evolving non-secure applications into secure applications. *Science of Computer Programming*, 66(1), pp.60–70.

Soliman, M. et al., 2013. Smart home: Integrating internet of things with web services and cloud computing. In *2013 IEEE 5th international conference on cloud computing technology and science*. IEEE, pp. 317–320.

Troya Castilla, J. et al., 2015. ARTIST: model-based stairway to the cloud. *PS-STAF*

2015: Projects Showcase, part of the Software Technologies: Applications and Foundations 2015 (2015), p 1-8, pp.1–8.

Tsai, W.-T., Sun, X. & Balasooriya, J., 2010. Service-oriented cloud computing architecture. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. IEEE, pp. 684–689.

Ventola, C.L., 2014. Mobile devices and apps for health care professionals: uses and benefits. *Pharmacy and Therapeutics*, 39(5), p.356.

Witt, B.I., Baker, F.T. & Merritt, E.W., 1993. *Software architecture and design: principles, models, and methods*, John Wiley & Sons, Inc.

Wojcik, R. et al., 2006. *Attribute-driven design (ADD), version 2.0*, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST.

Yacoub, S.M. & Ammar, H.H., 2001. UML support for designing software systems as a composition of design patterns. In *International Conference on the Unified Modeling Language*. Springer, pp. 149–165.

Youssef, A.E., 2014. A framework for secure healthcare systems based on big data analytics in mobile cloud computing environments. *Int J Ambient Syst Appl*, 2(2), pp.1–11.

Yuliana, R. & Rahardjo, B., 2016. Designing an agile enterprise architecture for mining company by using TOGAF framework. In *Cyber and IT Service Management, International Conference on*. IEEE, pp. 1–6.

Zhang, L.-J. & Zhang, J., 2009. Architecture-driven variation analysis for designing cloud applications. In *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*. IEEE, pp. 125–134.

Zhang, Q., Cheng, L. & Boutaba, R., 2010. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1), pp.7–18.

Zhao, L. et al., 2012. An architecture framework for application-managed scaling of cloud-hosted relational databases. In *Proceedings of the WICSA/ECSA 2012 Companion Volume*. ACM, pp. 21–28.

APPENDICES

APPENDIX A

Reengineer Student Academic Results System for Web-based Cloud Application Service

Legacy application face many challenges to be used and benefit from recent technology such as cloud technology. In chapter 3, Student academic result record case study is a desktop application need to benefit from cloud for application data to provide as a service benefit main college with two branches students to log in and querying their academic records.

For the realization of this proof of concept a specific multi-tier application service analyzed using UML for modeling requirements to understand the application architecture for evaluation methodology; Assistant migration tools for extracting and migrate application data with PHP- MySql Database have been used, and the Graphic user interface (GUI) had been designed using Artisteer and Dream weaver web Design tools. The GUIs are explained as follows:
Explains the license plate detection an extraction stage, and present the experiments results and discussion about these results.

Collecting a data set for (126) students, from the legacy application. The data set can be used in the field of Migration legacy application to a PHP MySql database.

A.1 Data mapping for student result records

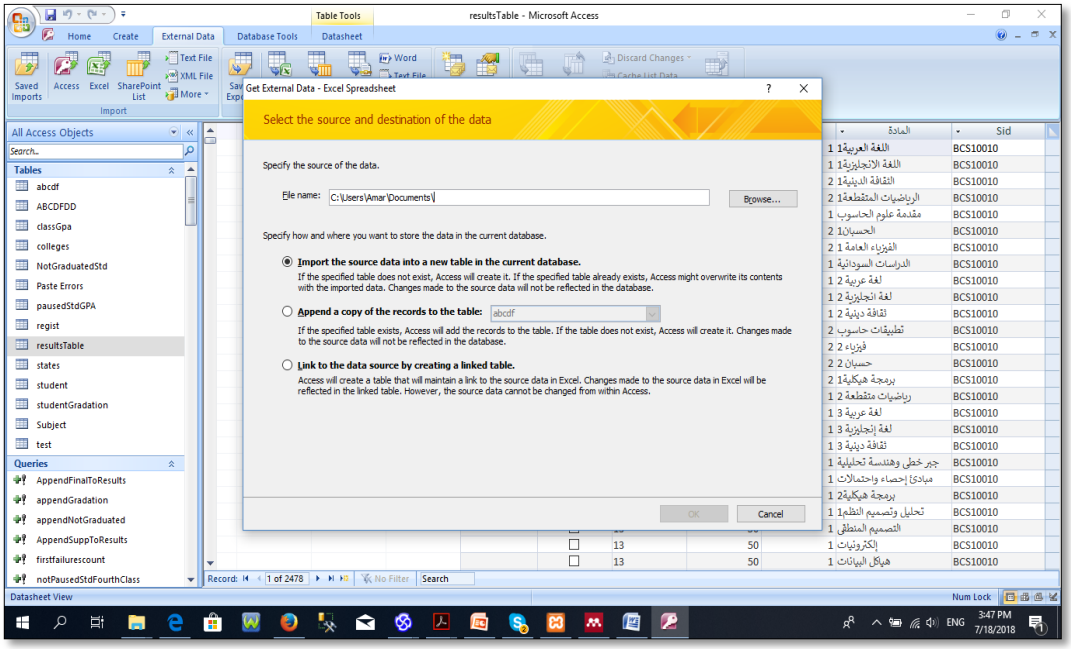


Figure A.1: Extracting Data from legacy system.

A.2 A.4 License Plate Characters Segmentation

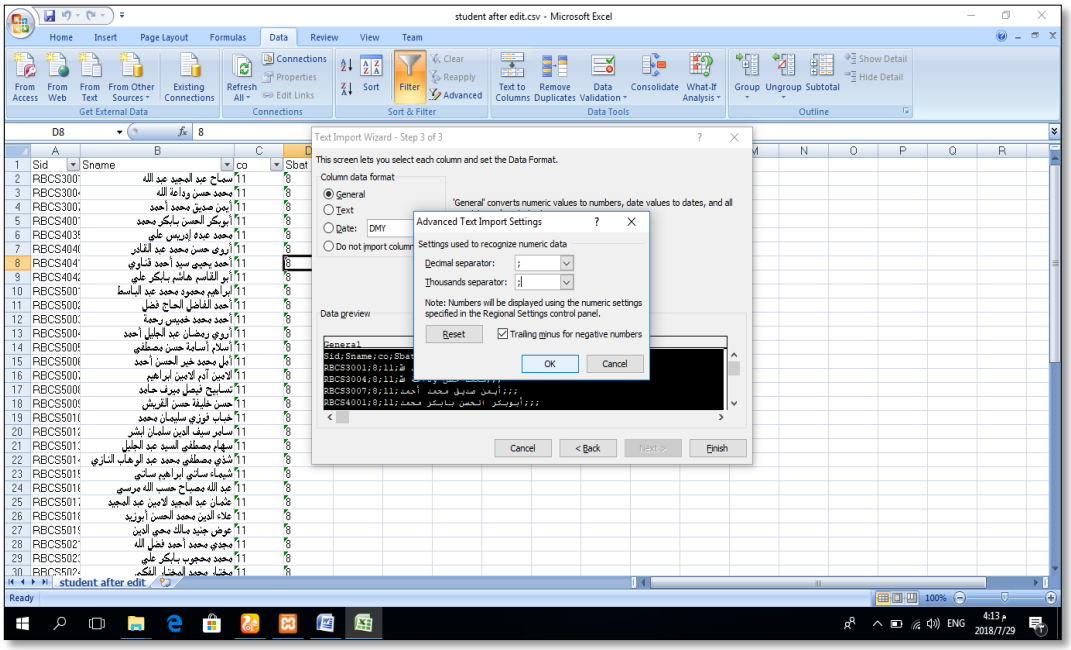


Figure A.2: Data Filtering.

A.3 The assistant migration tool

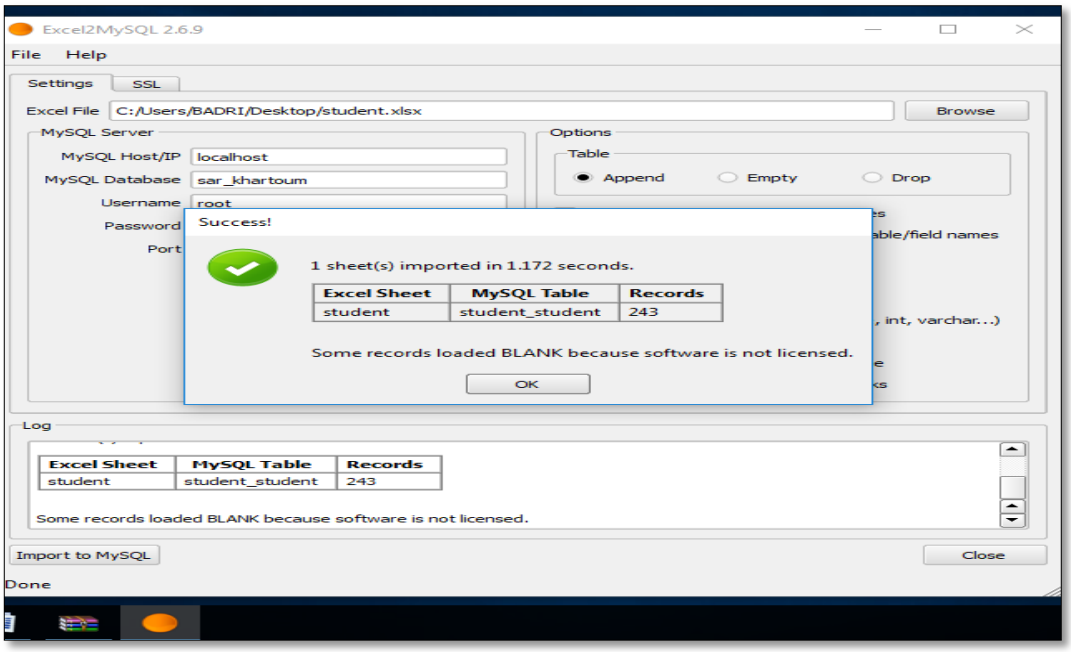


Figure A.3: Converted data mapping using the assistant migration tool.

A.4 Uploading Data

Figure A.4: Uploading data

A.5 Upload Data

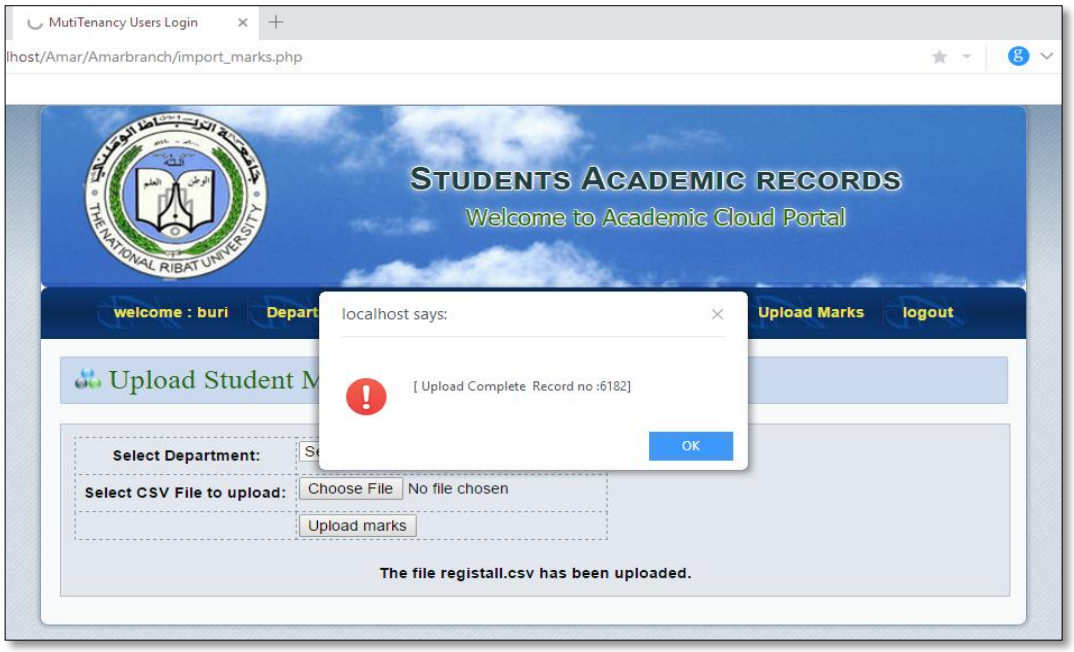


Figure A.5: Data has been uploaded successfully.

A.6 Target database

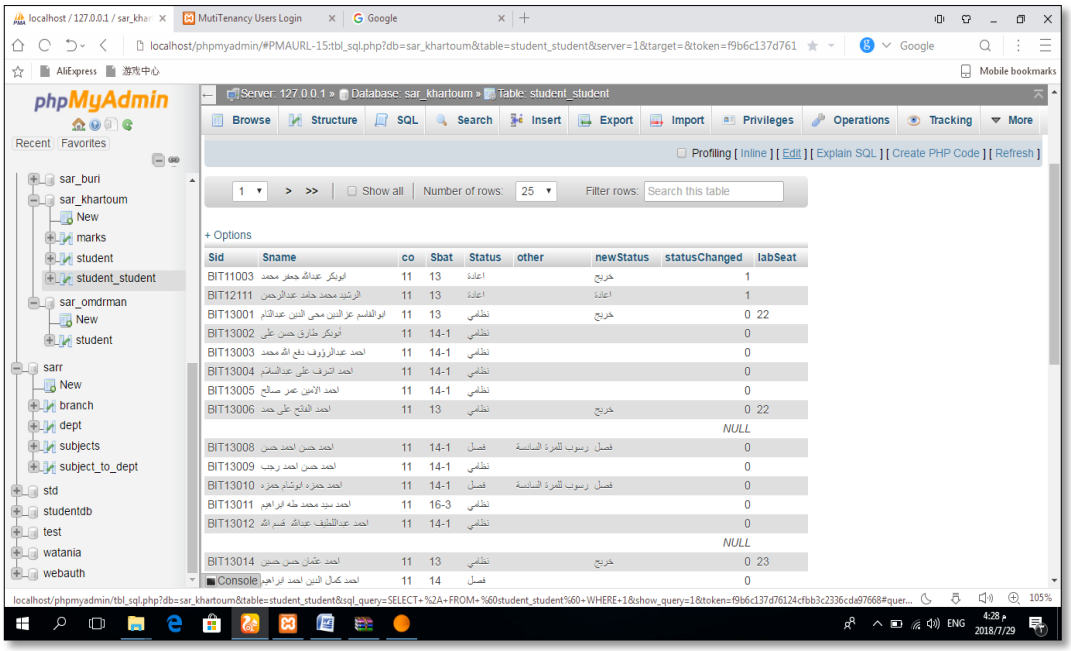


Figure A.6: Display loaded in MySql database (target).

A.7 Main Interface

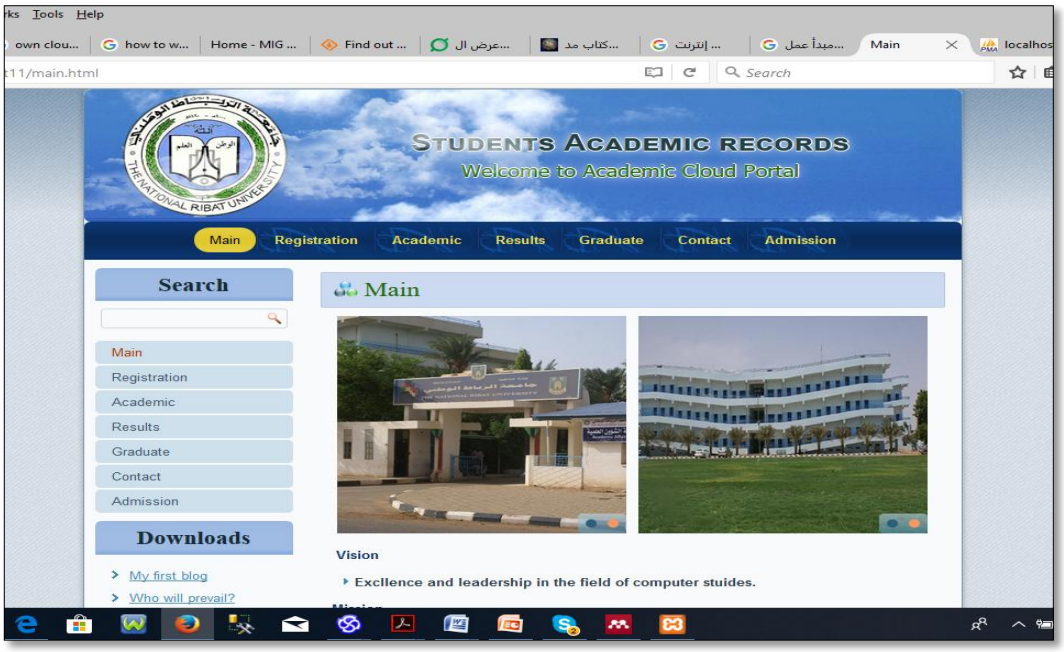


Figure A.7: Main GUI for Student Academic Result Records Web Portal.

A.8 Multi Users Interface

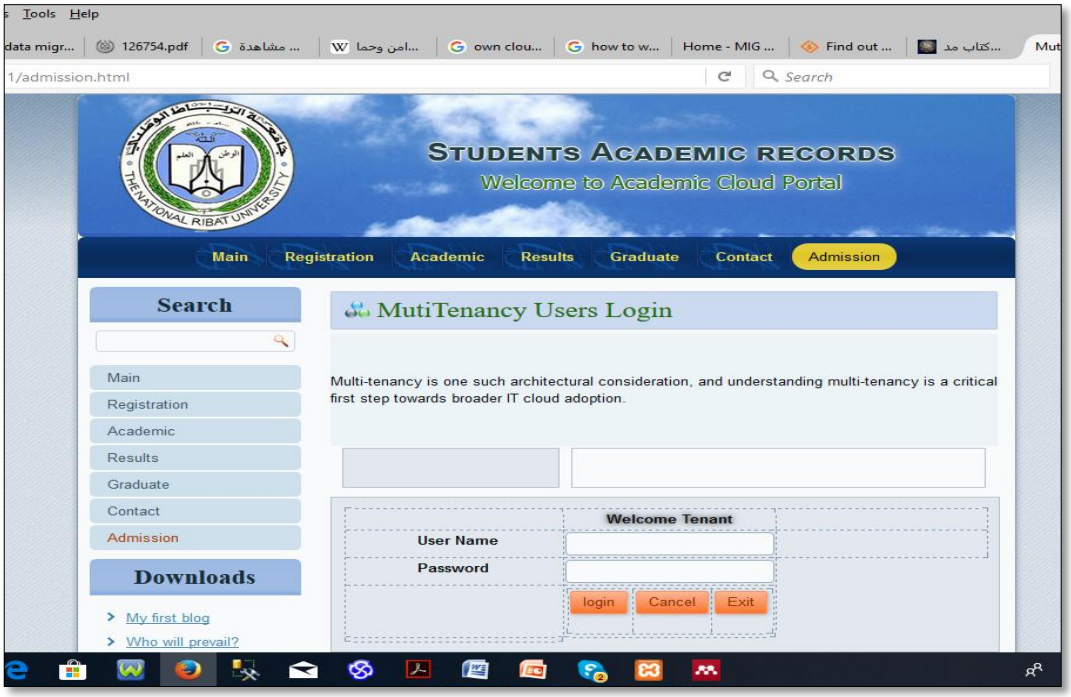


Figure A.8: Users login enable cloud web portal user to access functionality into custom Web sites.

A.9 Student Academic Examination Results Service view

Figure A.9: Secure student login to view the selected semester exams record.

A.10 Student Academic Examination Results Service view

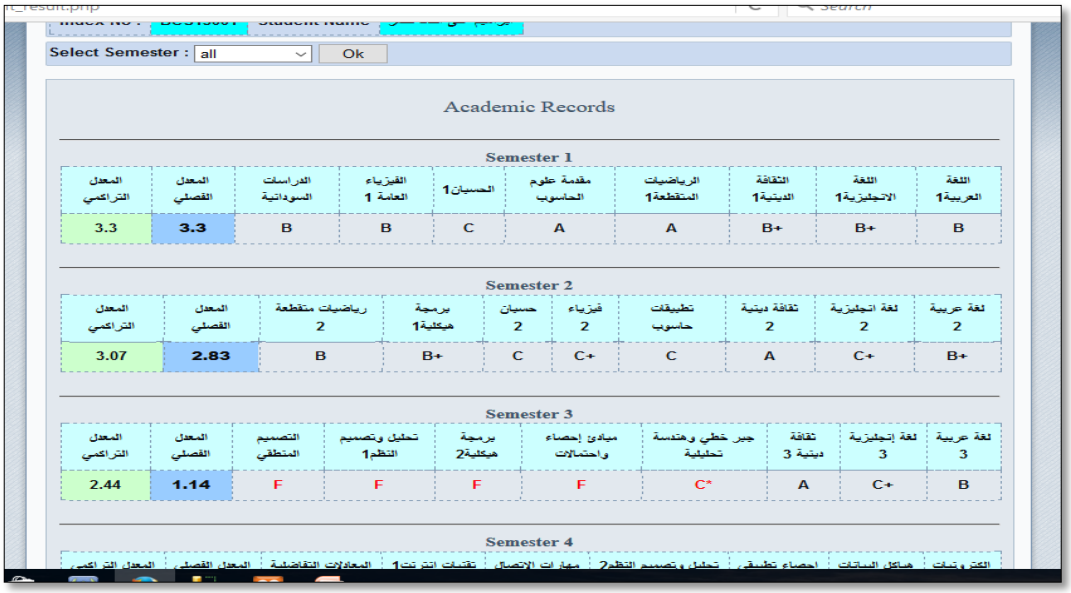


Figure A.10: Secure student login to view all semester records results.

APPENDIX B

A view's model for developing Hajj and Umrah Mobile Healthcare System

Enterprise Architecture is adopted for healthcare solutions to influence and guide changes in business procedure and information technology. Business models are used to represent the organization structure and services, business rules and processes.

The Overall objective is to: Present views of an ArchiMate model for developing Mobile healthcare for large crowd Events System (MHES) for case study Hajj and Umrah Mobile Healthcare System, using the TOGAF framework scenario to describe Enterprise Architecture. Secondary goals, find a suitable Architecture to help Healthcare services adjust to the Cloud environment; Describe scenario and its applicability in the new business environment; Facilitate communication between healthcare providers.

B.1 Enterprise Architecture development phases

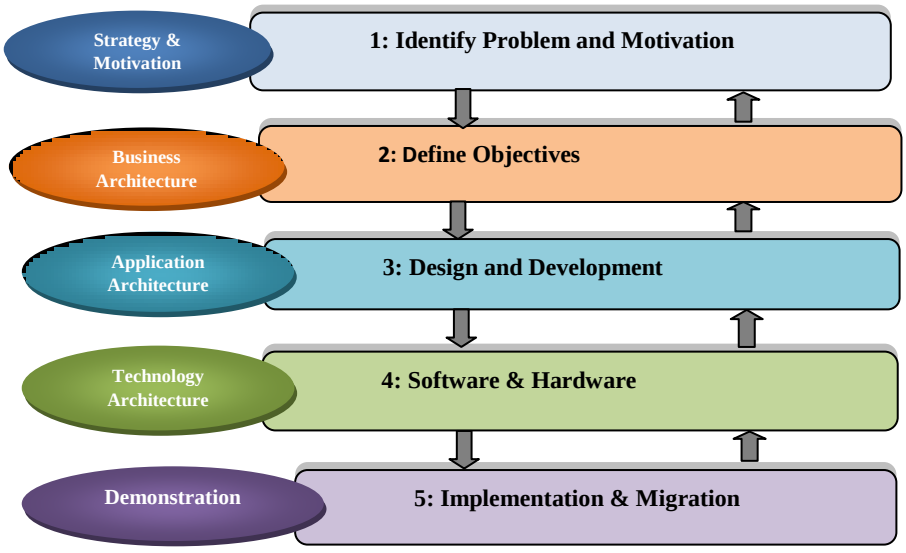


Figure B.1: Secure student login to view all semester records results (Eldein et al. 2018).

Enterprise Architecture development phases are represented as follows:-

B.2 Strategy and Motivation Phase

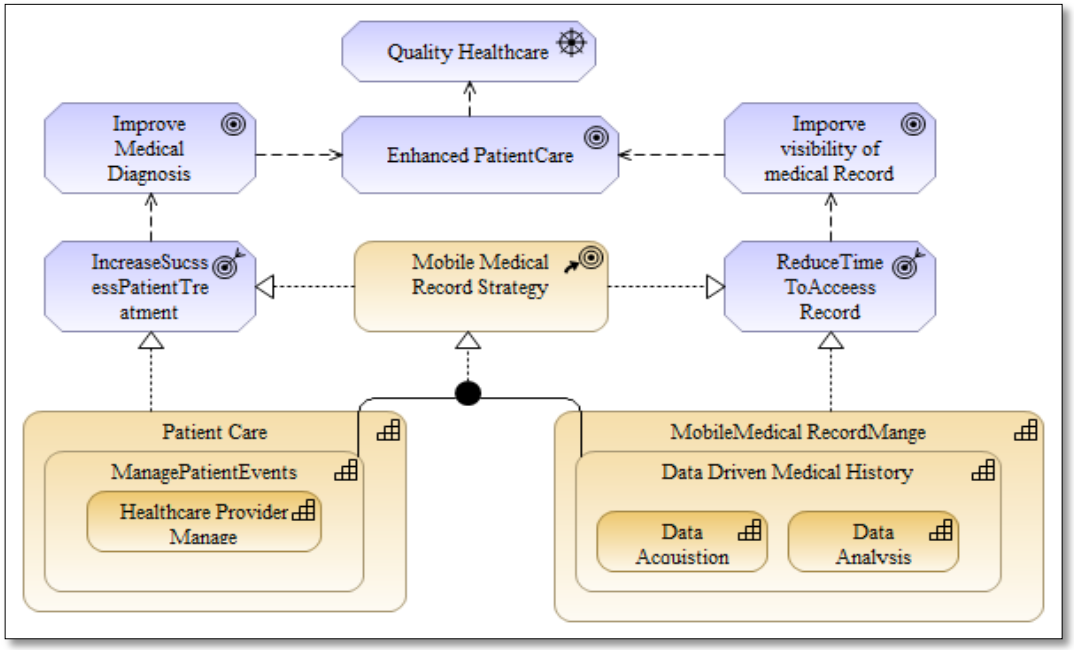


Figure B.2:Capture business requirements and a set of capabilities
To support and use of mobile solutions.

B.3 Strategy and Motivation Phase

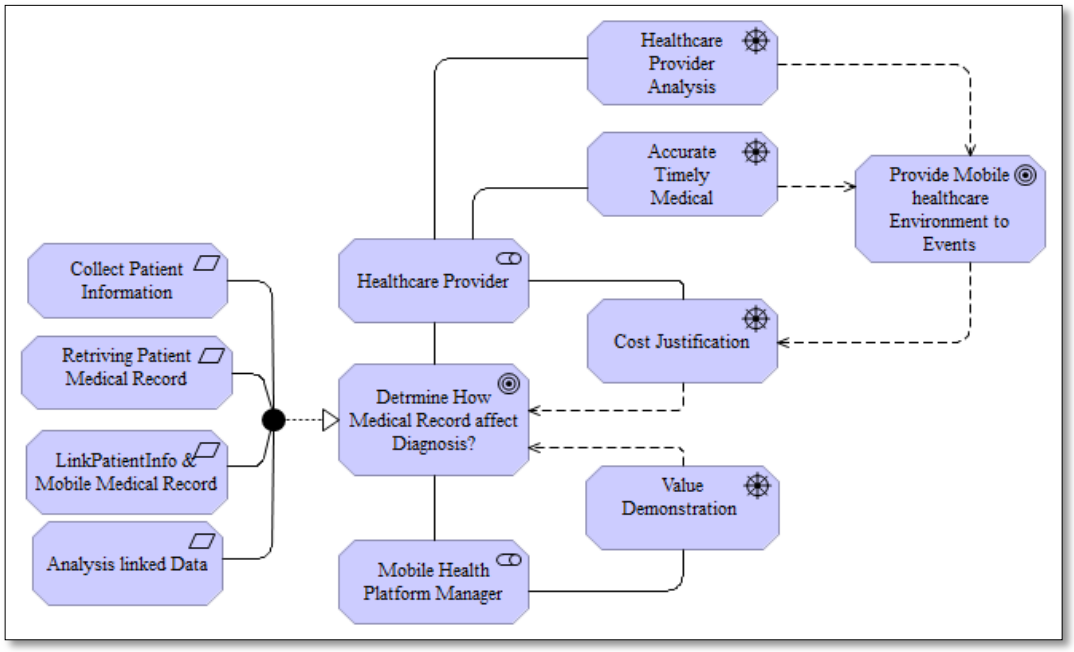


Figure B.3: Strategy analysis goals for healthcare Events

B.4 Phase 2: Business Architecture Stakeholders

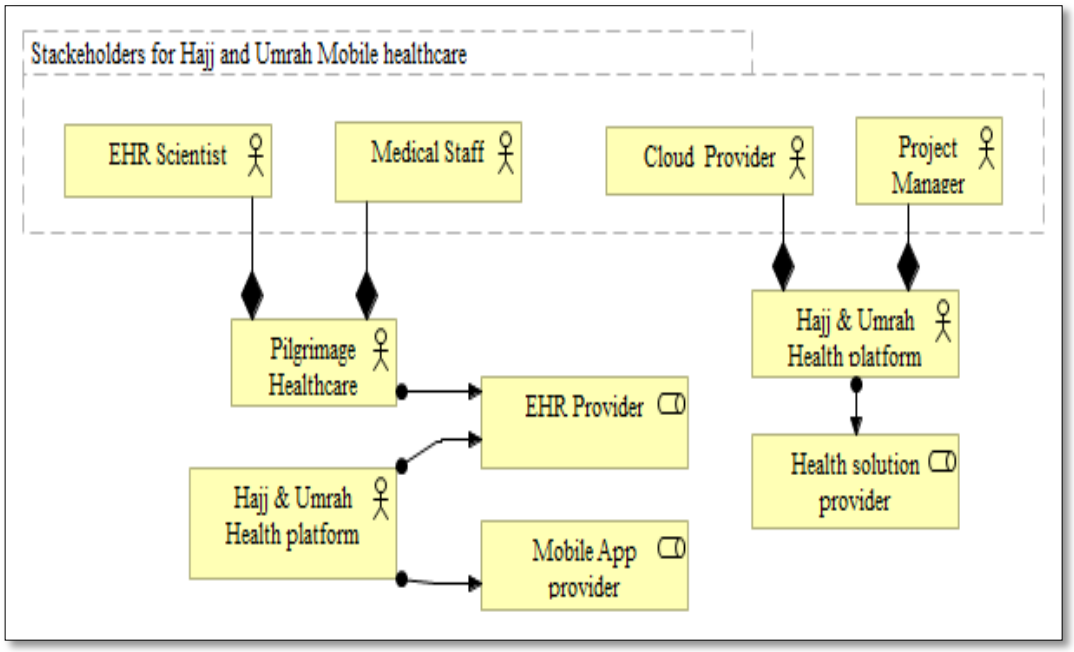


Figure B.4: Hajj and Umrah Mobile Healthcare analysis roles.

B.5 Phase 2: Business Architecture Services Integration

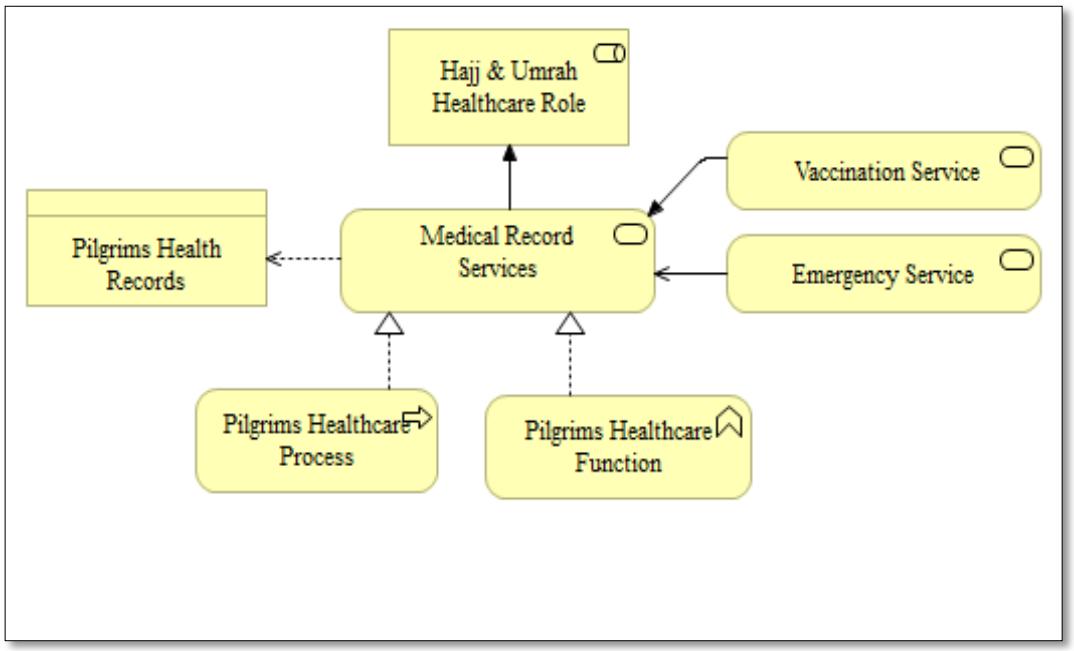


Figure B.5: Hajj and Umrah Healthcare Services.

B.6 Phase 2: Business Architecture Capabilities and Resources

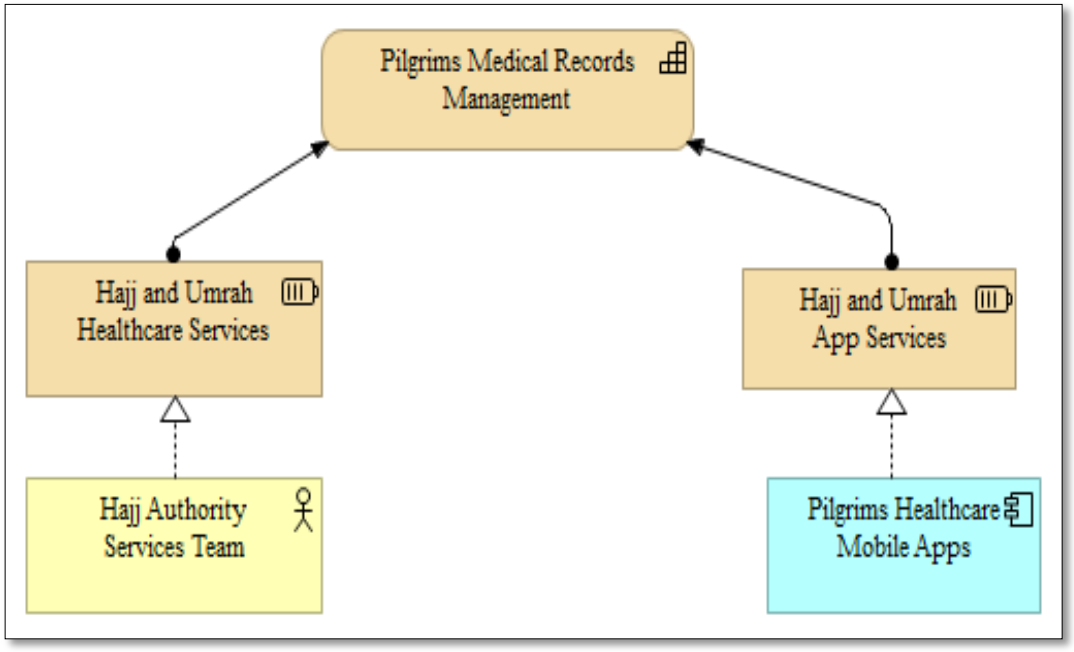


Figure B.6: Hajj and Umrah Mobile Health management capabilities.

B.7 Phase 3: Application Architecture

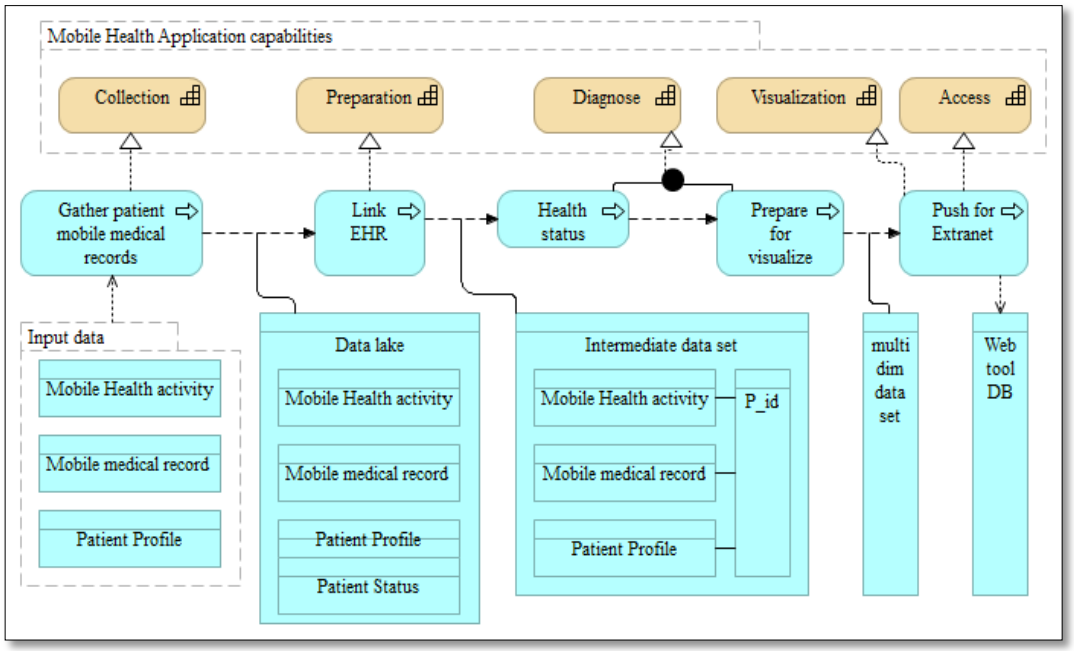


Figure B.7: Mobile Application design and development process.

APPENDIX C

CDO Verification uses CloudMiG Xpress for migrating Online Pet Store Application to the cloud

CloudMIG Xpress provides tool support for the comparison and planning phases to migrate software systems to PaaS or IaaS-based clouds. It originates from an academic prototype and is built to support research in cloud migration, aims at supporting SaaS providers to semi-automatically migrate existing enterprise software systems for scalability and resource efficient PaaS and IaaS-based applications.

The deployment options steps for migrating on multiple clouds are explained as follows:

C.1 Main Interface

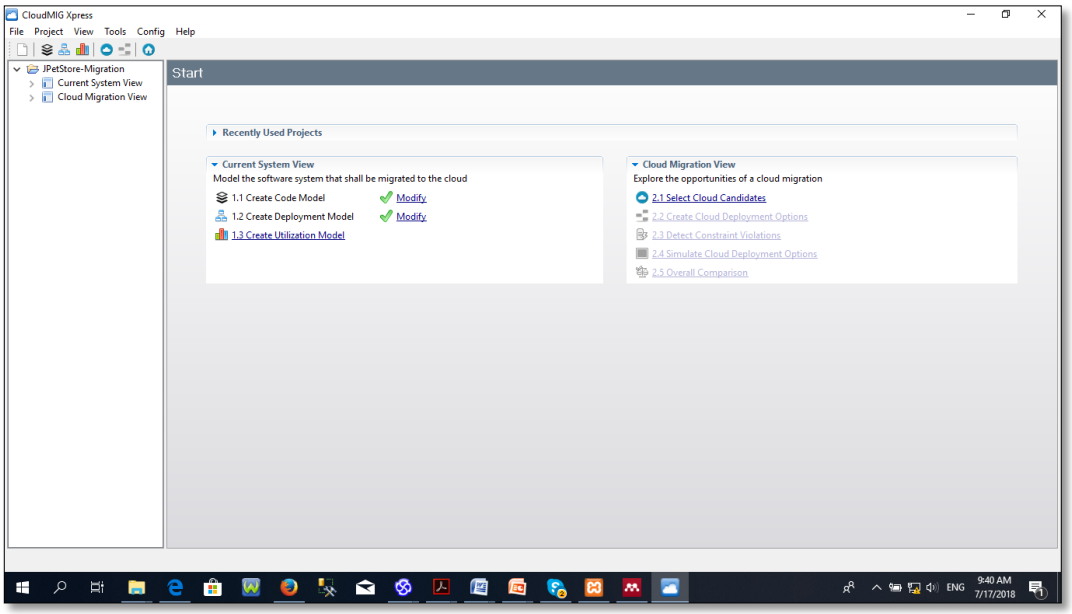


Figure C.1: CloudMIG Xpress Main GUI.

C.2 Source code Extraction

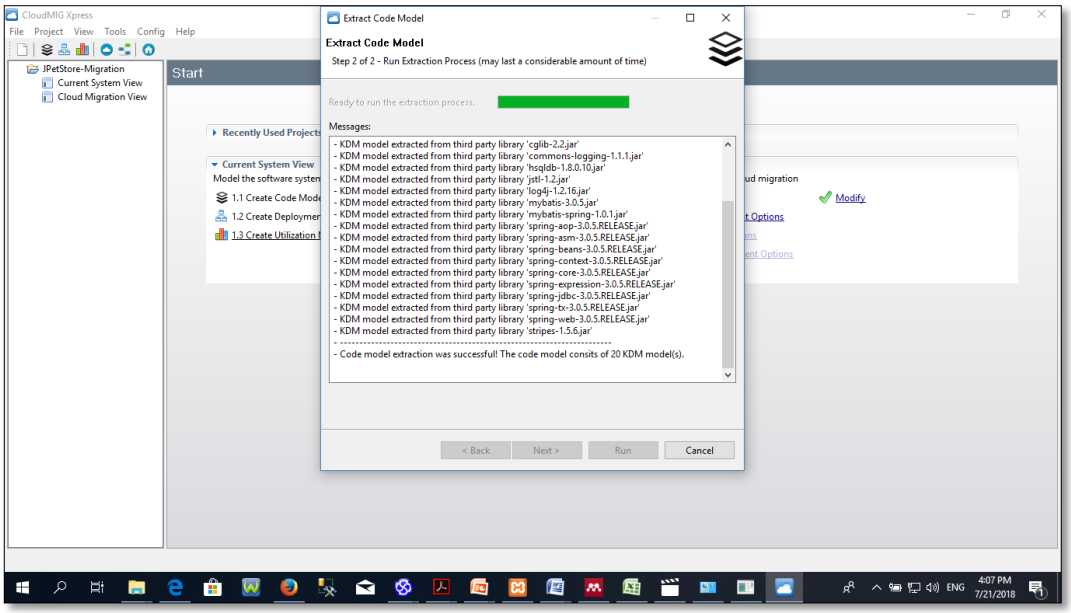


Figure C.2: Extract Knowledge Discovery Meta model (KDM) from source code.

C.3 Deployment model creation

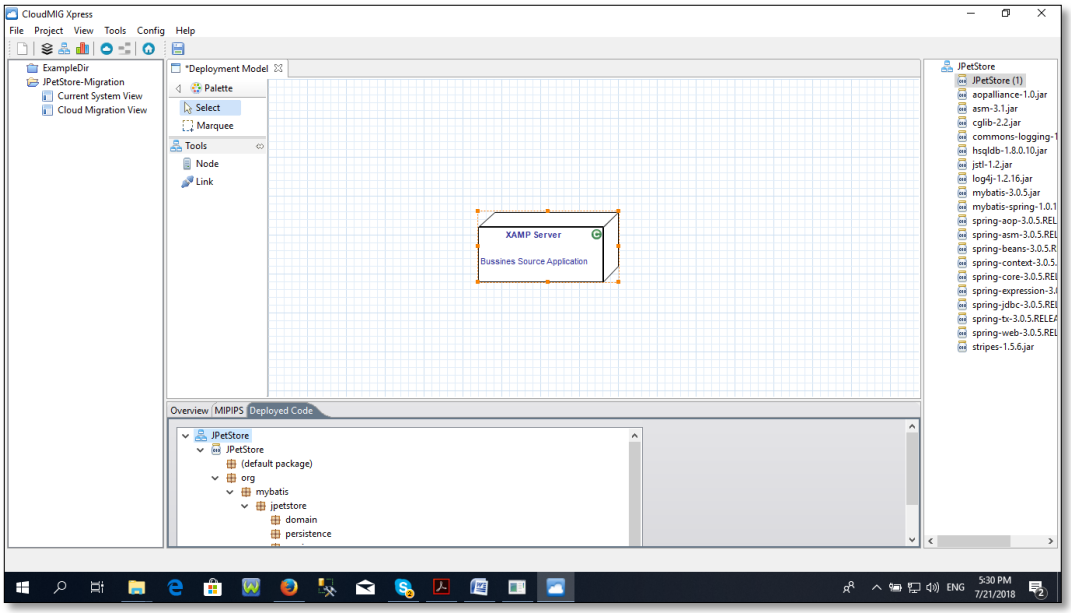


Figure C.3: Deploy Extracted model to server machine.

C.4 Create Utalization Model

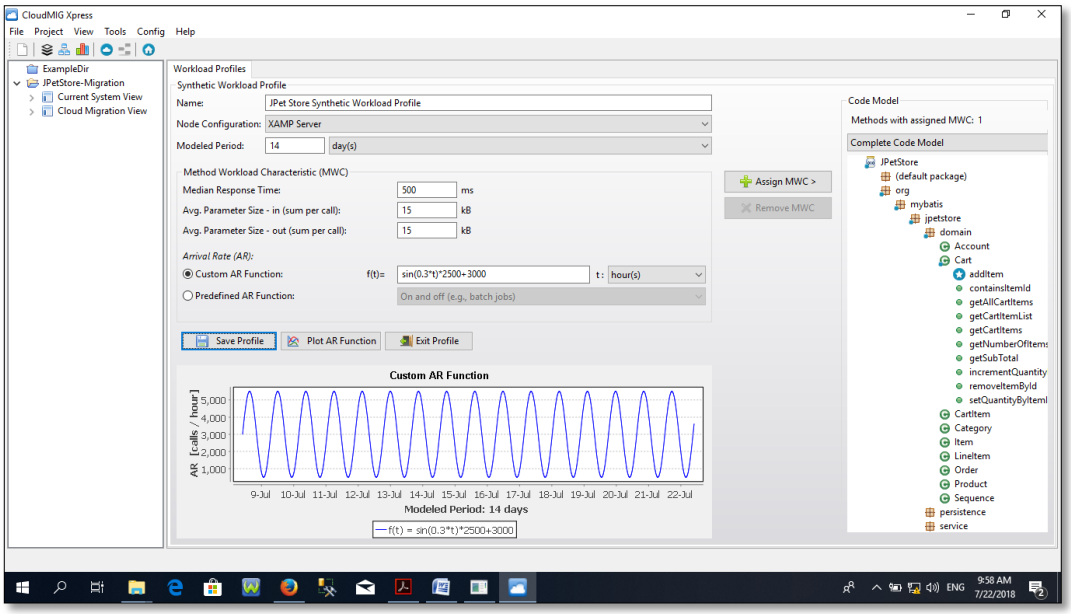


Figure C.4: Synthetic Workload profile with Arrival Time (AR) simulation result.

C.5 Cloud candidates Selection

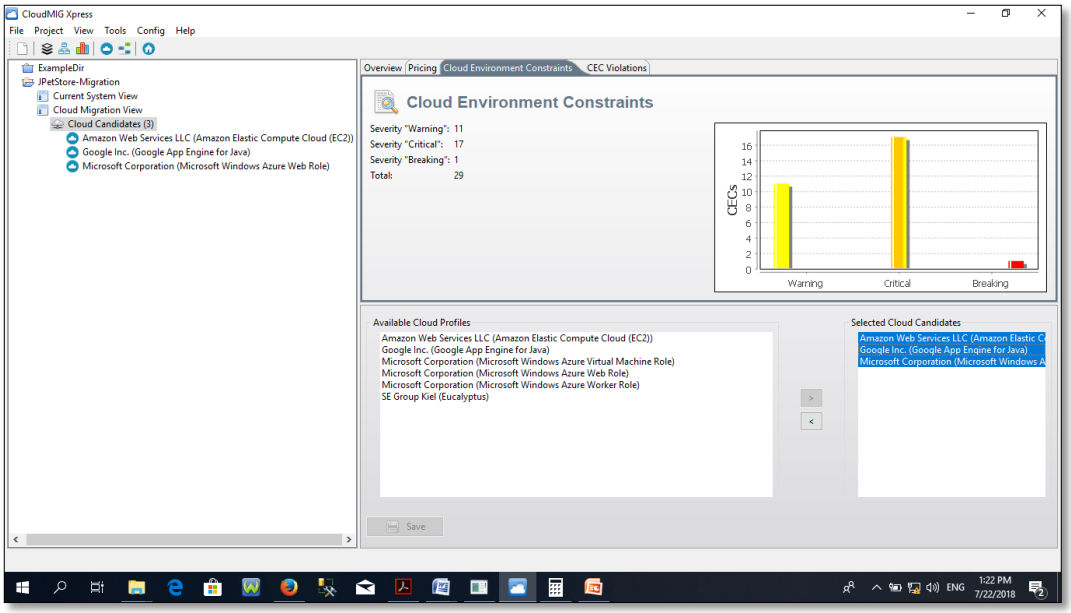


Figure C.5: Candidates cloud selected , and Cloud environment Constrains (CEC) defined in the cloud environment profile.

C.6 Cloud Deployment Option (CDO) Creation

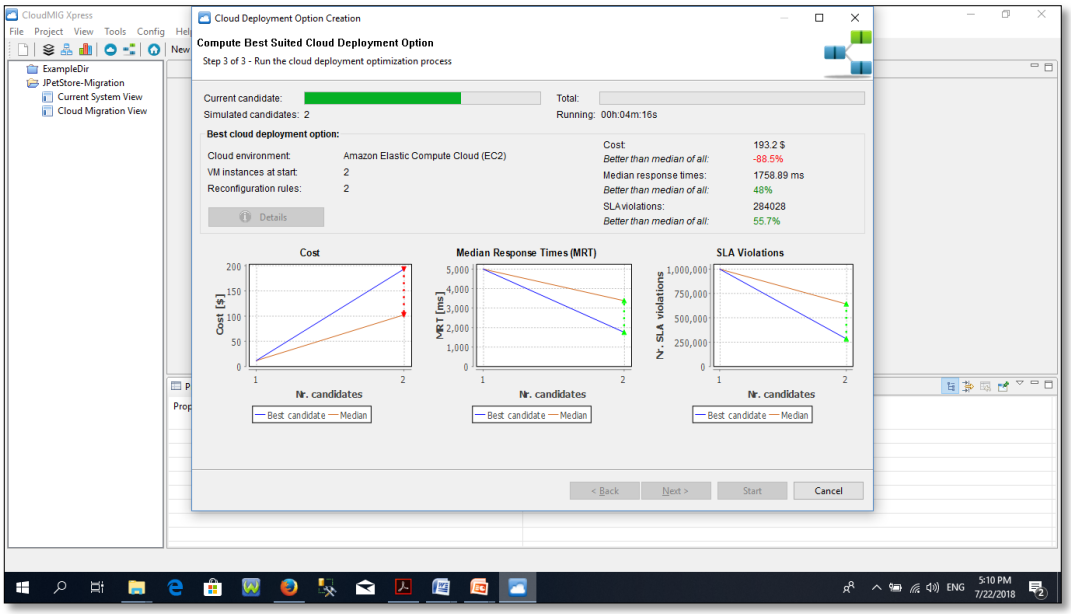


Figure C.6: Simulation result for 2 candidates cloud deployment options in comparison.

C.7 Cloud Deployment Option (CDO) creation

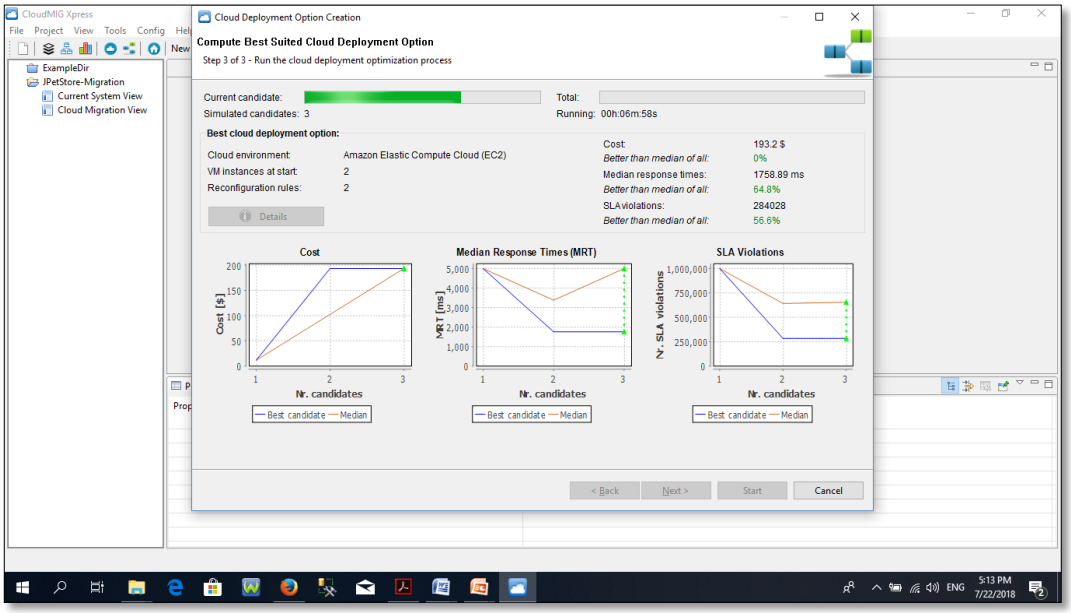


Figure C.7: Simulation result for 3 candidates cloud deployment options in comparison.

APPENDIX D

A summary of Cloud Application Architecture Design Patterns

#	Category	Pattern Name	Problem	Solution
1	**Fundamental Cloud Architectures**	Loose Coupling	Reduce dependencies between Distributed Applications and between individual components.	Broker: to communicating components and decouple multiple integrated applications from each other.
2		Distributed Application	How can application functionality be decomposed to be handled by separate application components?	The functionality of the application is divided into multiple independent components that provide a certain function
3	**Cloud Application Components**	Stateful Component	A synchronized internal state	Replication Internal state
4		Stateless Component	Increase Elasticity (Failures)	Storage Offerings (External)
5		User Interface Component	Reduce Dependencies	Elastic Load Balancer
6		Processing Component	Processing functions to meet different customers' requirements	Split into separate functional blocks (stateless)
7		Batch Processing Component	Asynchronous processing delayed	Stored (asynchronous) until conditions are optimal for their processing
8		Data Access Component	Hide & isolate data while ensuring data configuration.	Integrity of data and coordinates
9		Data Abstractor	How consistent data be presented and inconsistencies hidden from another application (components and users)?	Consistent state is unknown by approximating values or abstracting them into more general ones, such as change tendencies (increase / decrease).
10		Idempotent Processor	Presented of consistent data Duplicate function execution	Adjusted to allow retrieved data to be consistent (inconsistency detection)
11		Transaction-based Processor	Ensure messages receive are processed successfully and altered data successfully after processing	Transaction-based Delivery
12		Timeout-based Message Processor	Process messages, guaranteeing all messages handled or processed at-least-once	Timeout-based message processor sends acknowledgement after has successfully processed the message.
13		Multi-Component Image	Virtual server provides functionality of multiple application components	Multiple application components hosted on a single virtual server (to ensure running virtual servers may be used for different purposes without making provision or decommissioning operations necessary).

#	Category	Pattern Name	Problem	Solution
14	**Multi-Tenancy**	Shared Component	Shared between multiple tenants(individual configuration)	Optimize portion of the app_ stacks and app components deployed equally to all tenants.
15		Tenant-isolated Component	Shared between multiple tenants enabling individual configuration and tenant-isolation regarding performance, data volume, and access privileges	Ensure isolation between tenants by controlling tenant access, processing performance used, and separation of stored data.
16		Dedicated Component	Not shared components be integrated into a multi-tenant app	Dedicated application components are provided exclusively for each tenant using the application.
17	**Cloud Integration**	Restricted Data Access Component	Need component alter provide DB on Accesses restriction on different environments	Defined privileges for each data element, separate restriction data access
18		Message Mover	Message queues of different providers be integrated	Message Mover integrates message queues hosted in different environments, receiving messages from one queue and transferring it to a queue in other environments.
19		Application Component Proxy	Application component be accessed directly to its hosting environment is restricted.	Synchronous and asynchronous communication with proxy component is initiated and maintained from the restricted environment access the unrestricted environment directly.
20		Compliant Data Replication	Replicated between environments, how some environments, handle subsets of the data due to laws and corporate regulation?	Message filters are used to delete and obfuscate certain data elements in these messages. Information about the data manipulations stored in a storage offering.
21		Integration Provider	How can application components in different environments, belonging to different companies, be integrated through a third-party provider?	Using integration components offered by a third party provider.

APPENDIX E

List of Participations

The research study detailed in this thesis has yielded several conference and workshop presentation participant, these include:-

E.1 Conferences

1. Amar Ibrahim Eldein, Hany Ammar ,and Dale Dzielski, 2017, December. "**Enterprise architecture of mobile healthcare for large crowd events**." Information and Communication Technology and Accessibility (ICTA), 2017 6th International Conference on, Muscat, Oman, Sultan Gabous University, 19-21 Dec.2017.
2. Amar Ibrahim Eldein, E.S. & Ammar, H.H., 2017. "**Requirements Model For Hajj and Umrah Mobile Healthcare System (HUMHS)**", 4thInternational Conference on Islamic Applications in Computer Science And Technology, 20-22Dec 2016, Sudan. http://unitechkl.com/iman/pres/37.ppsx, online.

E.2 Work Shop

1. Amar Ibrahim Elhaj Sharaf Eldein. "**A framework for Development Cloud Application Architecture Environment**", Software Engineering Applications, Challenges Workshop, 10 TH International Computing in Arabic (ICIA), 12-14 March 2016, Khartoum, Sudan.

عمار ابراهيم الحاج شرف الدين, **"إطار تطوير بيئة تطبيقات الحوسبة السحابية", ورشة تحديات و تطبيقات هندسة البرمجيات** , المؤتمر الدولي لعلوم و هندسة الحاسوب: الدورة العاشرة, 12-14 مارس 2016 الخرطوم السودان.

LIST OF PUBLICATIONS

The research study detailed in this thesis has yielded several internationally recognized peer-reviewed journals in the areas of cloud computing, information and communication technologies (ICT), and software architecture. These include:-

[1] Eldein, A.I.E.S., Ammar, H.H. and Dzielski, D.G., 2017, December. Enterprise architecture of mobile healthcare for large crowd events. In 2017 6th International Conference on Information and Communication Technology and Accessibility (ICTA) (pp. 1-6). IEEE.

[2] Ibrahim, A., Eldein, E.S. & Ammar, H.H., 2017. Requirements Model For Hajj and Umrah Mobile Healthcare System (HUMHS). , 5(1), pp.53–62.

[3] Ibrahim, A. & Eldein, E.S., 2016. Pattern Oriented Analysis for Web Based Applications on Cloud. , 9(1), pp.1–12.

[4] Ibrahim, A. & Hany, A., 2015. Model-Driven Architecture for Cloud Applications Development , A survey. , 4(9), pp.698–705.