بسم الله الرحمن الرحيم

Islamic University – Gaza
Deanery of Graduate Studies
Faculty of Information Technology

الجامعة الإسلامية – غزة
عمادة الدراسات العليا
كلية تكنولوجيا المعلومات

A Cloud-Based Applications Transformation Framework for the Palestinian e-Government

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology

By

Ibrahim AbuJalanbo

120110631

Dr. Rebhi Soliman Baraka

Supervisor

(1437 H) October, 2015

إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

A Cloud-Based Applications Transformation Framework for the Palestinian e-Government

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification

Student's name:	اسم الطالب/ة: إبراهيم خليل محمد أبو جلنبو
Signature:	التوقيع:
Date:	التاريخ: 26 / 12 / 2015

بسم الله الرحمن الرحيم

الجامعة الإسلامية – غزة
The Islamic University - Gaza

مكتب نائب الرئيس للبحث العلمي والدراسات العليا
هاتف داخلي: 1150

الرقم ج.س.غ/35/..... Ref

التاريخ 2015/11/16 Date

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **إبراهيم خليل محمد أبوجلنبو** لنيل درجة الماجستير في **كلية *تكنولوجيا المعلومات*** **برنامج تكنولوجيا المعلومات** وموضوعها:

إطار تحويل تطبيقات الحكومة الإلكترونية الفلسطينية إلى الحوسبة السحابية

A Cloud-Based Applications Transformation Framework for the Palestinian e-Government

وبعد المناقشة التي تمت اليوم الاثنين 04 صفر 1437هـ، الموافق 2015/11/16م الساعة الثالثة مساءً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. ربحي سليمان بركة	**مشرفاً و رئيساً**	.....
أ.د. علاء مصطفى الهليس	**مناقشاً داخلياً**	.....
د. تامر سعد فطاير	**مناقشاً خارجياً**	.....

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

نائب الرئيس لشئون البحث العلمي والدراسات العليا

أ.د. عبدالرؤوف علي المناعمة

P.O. Box 108, Rimal, Gaza, Palestine fax: +970 (8) 286 0800 فاكس Tel: +970 (8) 286 0700 ص.ب. 108 الرمال، غزة، فلسطين هاتف
public@iugaza.edu.ps www.iugaza.edu.ps

Dedication

To my father (Allah please his soul)

To my family

To my teachers

To my friends

To my colleagues

To Palestine

Acknowledgment

I thank Allah for giving me the strength and ability to complete this study despite all the difficult circumstances. I would like to express my sincere gratitude to my advisor Dr. Rebhi Baraka, without his help, guidance, and continuous follow-up; this research would never have been.

Also I would like to extend my thanks to the academic staff of the Faculty of Information Technology who helped me during my Master's study and taught me different courses. I cannot forget to express my thanks to the database administrator, system administrator, and software development staff in Government Computer Center and Ministry of Telecom and Information technology. I would like to thank my colleagues and classmates for making my study a great experience, useful, enjoyable, and full of warm atmosphere.

Last but not least, I am greatly indebted to my family for their support during my course studies and during my thesis work.

Abstract

The acceleration in the pace of software development in the Palestinian e-Government shows a critical issue in delivering services due to the lack of resources and due the geographically distributed government institutes which leads to a problem in integrating government applications and services. The cloud computing has been adapted on large scales and enables public sectors to achieve speed, agility, security, and integrability by providing services and applications as software as a service Saas.

While cloud-based solutions are adopted in industry with a great pace, and applications have been developed rapidly in e-government, there appears a serious need for a cloud transformation framework that can be used for the Palestinian e-government applications transformation to the cloud.

This research presents and analyzes the current Palestinian e-Governemnt cloud environment, and discusses the software development model, The research proposes a framework for applications transformation to the cloud for the Palestinian e-Government cloud architecture, defining its components and designing the software development lifecycle as a component of the proposed framework which will be used to transform applications to the cloud environment.

The proposed framework is evaluated using a scenario based software architecture evaluation method and shows that it achieves the quality attributes set as goals for the framework which are: Integrability, and agility. Moreover, a scenario of application transformation is adapted and validated. A specific usage scenario for the framework is discussed and further shows that the framework accomplishes its functionality and quality attributes.

Keywords e-Government, Cloud Computing, Software Engineering, Service Engineering Models, Service Oriented Architecture (SOA), Integrability

عنوان البحث

اطار تحول تطبيقات الحكومة الالكترونية الفلسطينية الى الحوسبة السحابية

الملخص

ان التسارع الملحوظ في تطوير البرمجيات في الحكومة الالكترونية الفلسطينية يظهر مدى كبير في قصور هذه العملية نظرا لقلة الموارد والتباعد الجغرافي بين المؤسسات الحكومية مما يؤدي الى ظهور مشكلة التكاملية بين التطبيقات الحكومية.

ظهور مفهوم الحوسبة السحابية وانتشاره بشكل كبير بما يمكن القطاع الحكومي من الاستفادة من مميزاته كالسرعة في التطوير والامان والتكاملية بين البرامج من خلال تقديم الخدمات الحكومية بمفهوم Software as a Service.

وحيث ان مفهوم الحوسبة السحابية تم الاستفادة منه بشكل كبير في مجالات عدة وحيث ان النمو الملحوظ في الحكومة الالكترونية ظهر جليا فقد ظهرت حاجة ملحة لايجاد اطار للتحول نحو البرمجة السحابية في الحكومة الالكترونية الفلسطينية.

يبدأ البحث بتقييم ودراسة الوضع الحالي للبيئة السحابية في الحكومة الالكترونية الفلسطينية ودراسة دورة حياة تطوير تطبيقات الحكومة الالكترونية الفلسطينية, ثم يقدم البحث مقترح لاطار تحول للتطبيقات الحكومية الى الحوسبة السحابية ودراسة مكونات هذا الاطار ودورة حياة التحول.

تم تقييم هذا الاطار باستخدام طريقة "تقييم معمارية انظمة البرمجيات بالاعتماد على السيناريوهات" والتحقق من ان هذا الاطار يحقق الاهداف المرجوة وهي agility, integrability

كما تم تطبيق نموذج لبعض جوانب هذا المقترح والتحقق من فاعليته وتحقيقه للاهداف وتطبيق عملية التحول لاحد تطبيقات الحكومة الالكترونية الفلسطينية واثبات ان هذا الاطار يحقق اهدافه.

Contents

List of Figures

List of Tables

List of Abbreviations and Glossaries

ATAM Architecture Tradeoff Analysis Method is a Scenario Based Software Architecture Evaluation Method.

ESB Enterprise Service Bus (ESB) is a software architecture model used for designing and implementing the interaction and communication between mutually interacting software applications in Service Oriented Architecture.

HTTP Hyper Text Transfer Protocol (HTTP) is a networking protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

SLA Service Level Agreement (SLA) is a part of a service contract where the level of service is formally defined. The term SLA is sometimes used to refer to the contracted delivery time or performance.

SOA Service-Oriented Architecture (SOA) is a set of principles and methodologies for designing and developing software in the form of interoperable services. These services are well-defined business functionalities that are built as software components (discrete pieces of code and/or data structures) that can be reused for different purposes.

ACL Is a list of permissions attached to an object. An ACL specifies which users or system processes are granted access to objects, as well as what operations are allowed on given objects.

SaaS Is a cloud computing model in which software is licensed on a subscription basis and is centrally hosted. It is sometimes referred to as "on-demand software".SaaS is typically accessed by users using a thin client via a web browser. SaaS has become a common delivery model for many business applications, it is a software delivery method that provides access to software and its functions remotely as a Web-based service.

PaaS Is a cloud computing model that delivers applications over the Internet. In a PaaS model, a cloud provider delivers hardware and software tools - usually those needed for application development - to its users as a service. A PaaS provider hosts the hardware and software on its own infrastructure. As a result, PaaS frees users from having to install in-house hardware and software to develop or run a new application.

IaaS Is a form of cloud computing that provides virtualized computing resources over the Internet.

Chapter 1 Introduction

Recently, the applications development in the Palestinian e-Government increased significantly, the e-services and applications are developed and hosted in the e-Government cloud.

The increasing in software development raises retreat in integrability between e-Government applications and e-services, this issue is critically need to be solved, because the e-Government applications have many interactions among each other's [1].

In addition its appear the absence of the agility in software development, especially with those applications changed frequently in user requirements, or changes applied on their architecture.

On the other hand, cloud computing has been spread, and adapted in business and public sector organizations obviously, and introduce solutions for various issues such as speed, security, agility, reduced costs, and ability to integrate easily [2].

Despite of the readiness of the Palestinian cloud infrastructure, the e-Government applications still does not benefit from the cloud benefits, and issues such as integrability and agility needs to be solved. In this research we introduced agility and integrability as the main issues to be studied and solved.

Transformation to cloud environment needs considering technical requirements to make sure that the application reside on a stable, reliable infrastructure, data will be protected, developing application will be agile, and the application will be integrated with others applications. These requirements can be achieved by adapting a transformation model or framework. Currently there are various service engineering models and frameworks used for transforming applications to the cloud to be presented as software as a service [3]. These models and frameworks will be discussed in details in thesis work, and their pros and cons will be introduced.

Currently the Palestinian cloud infrastructure has been installed and virtual servers are used for hosting e-Government applications. The cloud service models are adapted such as software as a service (Saas), platform as a service (Paas), and infrastructure as a service (Iaas), and the cloud community deployment model is adapted according to the e-Government policies.

To benefit from cloud features, there is a serious need for a cloud transformation framework that can be used for the Palestinian e-government applications transformation to the cloud to overcome the limitations in agility and integrability of the current e-Government applications development [1].

The current cloud environment needs to be studied, and the current software development model needs to be analysed to define its pros and cons and its shortcoming in transformation process, and design the

Palestinian e-Government cloud application transformation framework. In this framework, the Palestinian e-Government cloud architecture needs to be defined, some of the cloud components need to be built, and the others need to be integrated, in addition the process lifecycle for transforming applications needs to be clarify.

Next we introduce the statement of problem, objectives, importance of the research, scope and limitations, methodology, and finally the thesis structure.

1.1. Statement of the Problem

In spite of the significant acceleration in the development of government applications in the Palestinian e-Government, these applications still suffer from deficiencies such as security, reliability, availability, and integrability among them as well as other government services. In addition to that, the development process in the e-Government applications suffer from the lack of the agility, since the applications are developed depending on the traditional software engineering models.

In this research we introduce integrability, and agility as a target attributes to be solved, because the moving to the cloud shows critical issues in applications and services integrability, whereas the applications will be introduced as SaaS. In addition the transformation process need to maintain the application architecture, and the customer needs to achieve agility transformation.

There appears a serious need for a cloud transformation framework that can be used for the Palestinian e-Government applications transformation to the cloud. This framework needs to be architected, its components need to be defined, and the lifecycle for transformation needs to be defined.

1.2. Objectives

1.2.1. Main Objective

To design a framework for transforming the Palestinian e-Government applications into a cloud-based environment that achieves integrability and agility.

1.2.2. Specific Objectives

The specific objectives of the project are:

- Analyse the current Palestinian e-Government cloud environment and determine the shortcomings and requirements against the integrability and agility in transforming applications to the cloud environment.

- Design the Palestinian e-Government cloud transformation framework.
- Realizing the framework through applying and transforming a case of e-Government applications.
- Evaluate the proposed framework for integrability and agility using ATAM method.

1.3. Importance of the Research

The proposed transformation framework will benefit the e-Government applications and their development in the following ways:

- Solving the integrability issues between the e-Government applications.
- The cloud based software development cycle needs more study and adaptation in order to design special transformation framework for the Palestinian e-government cloud applications transformation.
- The proposed framework would have a great value towards applying software engineering models in the governmental applications transformation.
- The framework would allow an easy software management and good time estimation for large and medium projects transformation.
- The framework would add a new model to the Software as a Service SaaS software engineering community.

1.4. Scope and Limitations of the Research

- The designed framework will address the targeted characteristics such as the integrability and the agility.
- The framework will not be fully implemented but rather a scenario will be applied.
- The proposed software lifecycle component will cover the software lifecycle faces used for transforming applications to the cloud.
- The framework components will function in the Software as a service cloud model.
- The platform model and the infrastructure model will be a part of the framework but will not be implemented in this research.

1.5. Methodology

To achieve the objectives of the research the following methodology will be followed:

1. Study and analyse the current software engineering model of the Palestinian e-government

application transformation to the cloud.

2. Study and investigate frameworks used for applications transformation to the cloud.
3. Develop the proposed framework:
   a. Specify the requirements and framework architecture.
   b. Define the components of the framework.
   c. Specify the interactions between framework components.
   d. Evaluate the framework against the objectives which are the integrability and the agility by applying the designed framework on a suggested application as a case study.

1.6. Thesis Structure

The thesis consists of eight chapters:

Chapter 2 Technical and theoretical Foundation: describes the technical foundations needed for thesis work, cloud computing, software engineering from Cloud perspective.

Chapter 3 Related Works: presents works related to service engineering methodologies, and the impacts of cloud on software engineering and strategies.

Chapter 4 Current Palestinian cloud environment: presents the current situation of the cloud infrastructure, hardware installed, software types, and counts, and the readiness for transformation to cloud.

Chapter 5 Cloud applications transformation Framework: presents the proposed framework for the Palestinian e-Government applications transformation to the cloud and describes the components and their interaction.

Chapter 6 Framework Scenario Applying: is devoted to the presenting the applying of the framework scenario.

Chapter 7 Framework Evaluation: presents the evaluation of the framework using scenario based software architecture evaluation method, and validates the scenario applying using the proof of concept validation approach.

Chapter 8 Conclusions and Future Work: discusses the final conclusions and presents possible future works.

Chapter 2 Technical and Theoretical Foundation

This chapter describes the technical foundations needed for thesis work, including cloud computing, software as a service in cloud computing, Government SOA based framework, Agility and integrability in application transformations, and finally evaluation methods for software architecture and proof-of-concept.

2.1. Cloud Computing

Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications [2], which could solve the problem of lack of resources. The National Institute of Standards and Technology's (NIST) defines cloud computing as " 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". This cloud model is composed of five essential characteristics, three service models, and four deployment models [28].

Recently, cloud computing has captured significant attention as both business and computing model that enables public sector organizations to achieve objectives such security, speed, agility, integrity, and maintainability [1].

Cloud computing refers to flexible self-service, network-accessible computing resource pools that can be allocated to meet demand. Services are flexible because the resources and processing power available to each can be adjusted on the fly to meet changes in need or based on configuration settings in an administrative interface, without the need for direct IT personnel involvement. These resources are assigned from a larger pool of available capacity (for examples, memory, storage, CPUs) as needed, allowing an organization to spin up a proof-of-concept application, expand that to a full prototype, and then roll it out for full use without having to worry about whether existing hardware, data centre space, power, and cooling are capable of handling the load. Cloud computing allows the allocation of resources to be adjusted as needed, creating a hardware-independent framework for future growth and development [4].

For many years, the adoption of cloud computing has brought a massive change and it has become a trend in the field of information technology because it initiates significant cost reductions and provides new business opportunities for its users and providers [5].

The benefits of using cloud computing are: reduced cost for hardware and maintenance, global accessibility, and flexibility and highly automated processes, in which customers do not need to worry about routine concerns, such as software upgrading [5].

2.1.1. Cloud Computing Service Models

Cloud computing offers many different levels of services, from individual Software as a Service (SaaS) to Platform as a Service (PaaS) development environments and even Infrastructure as a Service (IaaS) complete solutions resident in the cloud. Some vendors now term even Everything as a Service (XaaS) as an offering, although this is more of a marketing term melding traditional and cloud computing than an established standard [4].

Depending on what resources are shared and delivered to the customers, there are four types of cloud computing. In cloud computing terminology when hardware such as processors, storage and network are delivered as a service it is called infrastructure as a service (IaaS). When programming platforms and tools like Java, Python, .Net, MySQL and APIs are delivered as a service it is called platform as a service (PaaS). When applications are delivered as a service it is called software as a service (SaaS).

Cloud services are aligned with their service model, in which each level of service abstraction will be associated with the term as a Service (aaS). Consumers of cloud resources access these "as a Service" resources via their favourite web browser without considering whether they are consuming an application or an entire infrastructure within the cloud.

Cloud vendors often describe their products as Backup as a Service (BaaS), Database as a Service (DBaaS), or even Everything as a Service (XaaS) to fit their particular product's function [4].

The service models are often represented in the form of a pyramid like that shown in Figure 2.1 because IaaS provides the most fundamental service category and each successive level includes elements of the lower-level service categories.

Currently the Palestinian cloud infrastructure has been installed the cloud service models are adapted as mentioned in details in section 4.1.

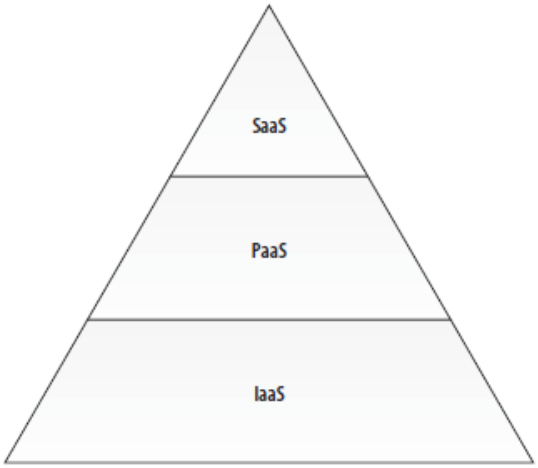

Figure 2.1 Cloud Service Models [4]

a. Infrastructure as a Service (IaaS)

IaaS is located at the bottom level of the service chart. It consists of all logical and physical resources needed to build the cloud. The foundation of cloud key services is virtualized platforms, which evolves from a virtual private server. Core IaaS elements include: data centre housing, virtual computing, storage, and backup.

Customers are offered with the possibility of buying the all package of memory, software. Due to this, the services by IaaS are aimed to reduce the time and costs to install a system. Despite being unable to manage the fundamentals of cloud, consumers are still able to control on their own OS, installed software and storage capacity completely [6].

The IaaS catalogue is comprehensive, enabling agencies to pick and choose from a range of services.

There are some core characteristics which describe what IaaS is. IaaS is generally accepted to comply with the following;

- Resources are distributed as a service.
- Allows for dynamic scaling.
- Has a variable cost, utility pricing model.
- Generally includes multiple users on a single piece of hardware [6].

b. Platform as a Service (PaaS)

Platform as a Service is another application delivery concept where resources needed to build applications and services should not to be downloaded and installed, but are accessible through the Internet. It is not so simple to draw a distinct line between PaaS and IaaS, also companies that

provide infrastructure services offer platform services as well [7]. Services that PaaS provides include: application design, development, testing, deployment, hosting, team collaboration, web service integration, database integration, security, scalability, storage, state management, and versioning.

c. Software as a Service (Saas)

Software as a Service (SaaS) is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the Internet. This research focuses on this cloud service model as we introduce a transformation framework for applications to run on the e-Government cloud as a software as a service.

The software as a service model composes services dynamically, as needed, by binding several lower-level services, thus overcoming many limitations that constrain traditional software use, deployment, and evolution [8].

SaaS focuses on separating the possession and ownership of software from its use. Delivering software's functionality as a set of distributed services that can be configured and bound at delivery time can overcome many current limitations constraining software use, deployment, and evolution [8].

SaaS is becoming an increasingly prevalent delivery model as underlying technologies that support web services and service-oriented architecture (SOA) mature and new developmental approaches. Meanwhile, broadband service has become increasingly available to support user access from more areas around the world [9]. benefits of the SaaS model include:

- Easier administration
- Automatic updates and patch management
- Compatibility: All users will have the same version of software.
- easier collaboration, for the same reason
- Global accessibility.

Saas is based on service oriented architecture (SOA) and virtualization of hardware and software resources. Because of the virtualization technique, physical resources can be linked dynamically to different applications running on different operating systems. In Section 2.2 we introduce the SOA based framework for the Palestinian e-Government which will be integrated to the proposed framework as the data service integrability component.

Software engineering in applications development starts with an explicit process model having framework of activities which are synchronized in a defined way, in Saas applications

development , there are already several service engineering methodologies for service design, we introduce these methodologies section 3.1.

2.1.2. Cloud Computing Deployment Models

With most organizations focusing on leveraging the cloud in order to cut capital expenditure and control operating costs, there is aggressive growth in business for cloud adoption. However, the cloud can bring security risks and challenges for IT Management, which can be more expensive for the organization to deal with, even considering the cost saving achieved by moving to the cloud. Therefore, it is very important for businesses to understand their requirements before opting for various deployment models available on the cloud [4].

There are primarily four cloud deployment models, which are discussed below, along with scenarios in which a business could opt for each. These models have been recommended by the National Institute of Standards and Technology (NIST).

a. Private Clouds

This model doesn't bring much in terms of cost efficiency: it is comparable to buying, building and managing infrastructure. Still, it brings in tremendous value from a security point of view. During their initial adaptation to the cloud, many organizations face challenges and have concerns related to data security. These concerns are taken care of by this model, in which hosting is built and maintained for a specific client. The infrastructure required for hosting can be on-premises or at a third-party location. Security concerns are addressed through secure-access VPN or by the physical location within the client's firewall system [5].

Several SaaS applications, provide options to their clients to maintain their data on their own premises to ensure data privacy is maintained according to the requirements of the particular business.

b. Community Clouds

In the community deployment model, the cloud infrastructure is shared by several organizations with the same policy and compliance considerations. This helps to further reduce costs as compared to a private cloud, as it is shared by larger group.

Various state-level government departments requiring access to the same data relating to the local population or information related to infrastructure, such as hospitals, roads, electrical stations, etc., can utilize a community cloud to manage applications and data [4].

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

According to the policies and Government Computer Centre, the deployment model in the e-Government cloud is the community cloud (see section 4.1.2).

c. Public Clouds

The public cloud deployment model represents true cloud hosting. In this deployment model, services and infrastructure are provided to various clients. Google is an example of a public cloud. This service can be provided by a vendor free of charge or on the basis of a pay-per-user license policy. This model is best suited for business requirements wherein it is required to manage load spikes, host SaaS applications, utilize interim infrastructure for developing and testing applications, and manage applications which are consumed by many users that would otherwise require large investment in infrastructure from businesses. This model helps to reduce capital expenditure and bring down operational IT costs.

d. Hybrid Clouds

This deployment model helps businesses to take advantage of secured applications and data hosting on a private cloud, while still enjoying cost benefits by keeping shared data and applications on the public cloud. This model is also used for handling cloud bursting, which refers to a scenario where the existing private cloud infrastructure is not able to handle load spikes and requires a fallback option to support the load. Hence, the cloud migrates workloads between public and private hosting without any inconvenience to the users.

Many PaaS deployments expose their APIs, which can be further integrated with internal applications or applications hosted on a private cloud, while still maintaining the security aspects. Microsoft Azure and Force.com are two examples of this model.

2.2. The SOA Based Framework for the Palestinian e-Government Integrated Central Database

The proposed framework depend on the e-Government SOA framework as the data service integration component, this component is responsible for the techniques used for the service implementation.

The SOA Based Framework for the Palestinian e-Government introduced by [1] has four layers: the front end user interface layer which presents the access interface that the end user interacts with, the common service layer which provides front end services that commonly needed by e-Services, the data access layer which addresses database access gateway either centralized or decentralized, and the infrastructure layer which includes physical and low level software components. The Central Database is one of the components of the data access layer.

2.2.1. The SOA-Based Framework

SOA is adopted in the transformation of the Central Database due to its open architecture and standards that cope with heterogeneous systems and applications with high degree of interoperability and flexibility. A typical architecture of SOA includes three main roles that interact using standard messaging. The roles are service provider, service registry and service client [1].

The service is first published by the service provider to the service registry which is a repository that holds service interfacing information. The service client searches the service registry for a specific service, and gets its binding information. The client uses binding information to consume the service provided by the service provider [1].

Web Services are used as a tactical realization of the SOA architecture. A Web Service is a software component that another software application can access automatically on the Web.

ESB provides the SOA solution with the necessary integration infrastructure for Web Services. It integrates applications, services, and the registry. ESB is event driven and provides standard messaging between services. It routes and transports service requests to the appropriate service provider. A typical ESB provides capabilities such as: routing, message transformation, protocol transformation, service mapping, Service orchestration, transaction management, and Security [1].

2.2.2. SOA-Based Integrated Central Database Architecture

The SOA based framework as shown in Figure 2.3 consists of a number of components. Each component satisfies one or more of the requirements and leads to the achievement of the goals of the framework.

Central Database Service Bus: Is considered the central platform of integration between different Web Services, and provides routing and transportation features for Web Service requests as well QoS feature for the framework. It is used and accessed by government institutions over the government private network as well as over the Internet for non-government institutions. It satisfies the reachability requirement of the framework.

Service Registry: Is used to provide a search point of access to services and database definitions and metadata for all services provided by the Central Database model. The registry is based on Universal Description Discovery and Integration (UDDI).

Government Informational Service: These Web Services provide access to basic informational queries and allow consumers to benefit from the government Central Database along with its presentation logic. This reliefs them from invoking services that interacts directly with the Central Database and return record sets that need to be manipulated by the developer. An example Web Service is a one that returns the social information of a citizen or the administrative record of an employee.

Service Orchestration: This component is responsible for managing composite services. The composite service is invoked by a client and in turn it invokes and orchestrates different services to achieve the requirement of the composite service.

Database Management Adapter: This adapter allows the Central Database Service Bus to accept requests for data sources from client systems and then invokes the relevant adapter to retrieve the data and return it in a standard format to the requester. It is used to hide the database management details from the rest of the Web Services. It communicates directly with the underlying data sources and provides database specific connectivity capabilities. This component achieves the accessibility requirement.

Database Replication Service: This service is used to manage replication between the Central Database and ministries databases. It handles connection types, mode of replications, access permissions. This service achieves the replication requirement.

Systems Management Service: This service is used to manage and monitor the Central Database service bus, and Web Services. It collects metrics, provides framework performance reporting capabilities.

Security Assurance Service: This service insures that security policies are adhered to. It is invoked by different services to add security layer to their functionality. Provided security functionalities include authentication, authorization, and nonrepudiation.

The interaction among these components is performed through the Central Database Service Bus which integrates the components and acts as the glue that connects them together. It routes, transports, formats requests and responses of services and provides service discovery through the registry.

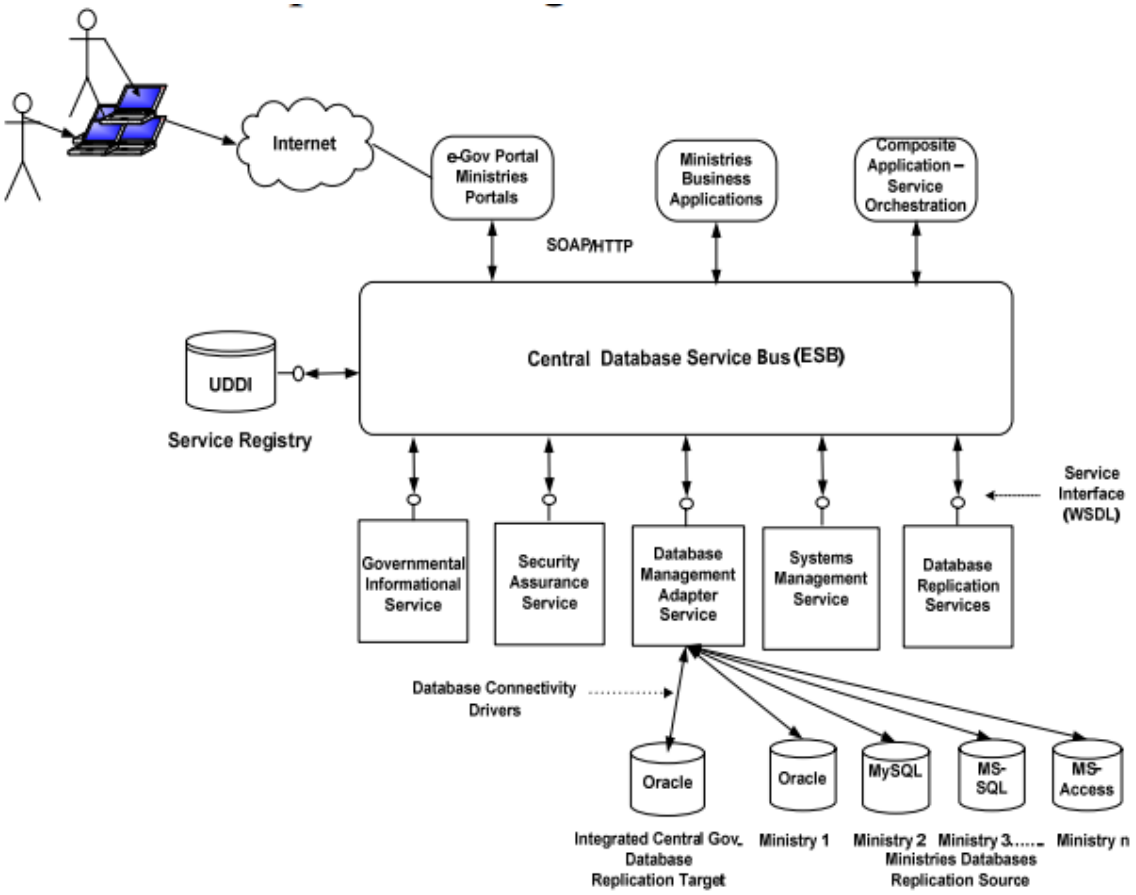

Figure 2.2 The SOA-based Integrated Central Database Framework [1]

2.3. Cloud Attributes in Applications Transformation to Cloud

Transforming applications to cloud has many benefits, including lower costs, better support, Reliability, availability, scalability, and infrastructure flexibility.

In this research we focus on agility, and integrability, as quality attributes to solve the introduced issues.

2.3.1. Agility

Agile methodology is an alternative to traditional software development methodologies. It helps application development to response to customer changes, and improves the software development process [10].

The great benefit of transforming applications to the cloud is the agility, Today, businesses have to respond with flexibility and speed to ever-changing customer demand, market opportunities and external threats. Software as a service is a special way to look develop cloud applications, focusing on their adaptability - the ability to respond to changing and new requirements. It is more than evident that agile approaches to software development seem to be a natural fit for developing such systems[11].

With agile development, the application is constantly subjected to the reality check of actual users putting it through its paces. As a result, developers are less likely to get ahead of themselves by guessing what people will want.

To achieve greater agility in software as a service development, agile methodologies are seemed to be fit to develop such systems. Although agile development methodologies are successful in dealing with changes, but they don't act well against complexities which are the nature of Saas projects because of the lack of the pre-defined design of system. For developing each system, a structure or architecture is needed for better communication between stakeholders and when the system is larger and more complex, the architecture is required more.

To realize the agility concept and achieves the research goals, a well-defined framework architecture need to be design, the components need to be clarify, and during application transformation the application architecture should be discussed with stockholders.

The Agile Manifesto [10] established a common set of overarching values and principles for all of the individual agile methodologies at the time. The manifesto details four core values for enabling high-performing teams to achieve agile development.

- Individuals and their interactions.
- Delivering working software.
- Customer collaboration.
- Responding to change.

2.3.2. Integrability

Integrability means an ability to make separately developed components of a system to work correctly together. Integrability is related to interoperability and interconnectivity.

Interoperability is the ability of software to use the exchanged information and to provide something new originated from exchanged information whereas interconnectivity is the ability of software components to communicate and exchange information [12].

Integrability is critical to the success of any application, especially within enterprise organizations. Its prerequisites are a bridge between Saas applications, data warehouses, and cloud framework components [13].

SaaS approach to integration leverages a set of standard web service application programming interfaces published by the Saas solution provider (SOA framework in this research). All data integration is executed through these APIs over the internet, enabling Saas solution providers to continuously provide upgrades to functionality without breaking existing integrations.

Complex application integration requirements challenge even the best Saas applications today, there are still limitations and pitfalls that organizations must be wary of. While organizations encounter some new integration challenges with Saas, they still faced with many of the same challenges of traditional application integration.

The cloud framework architecture can increase the integration between applications, as the most components of the framework works and interacts with all cloud applications, this make the integration process more stable.

2.3.3 Integrability in Cloud Application Transformation Framework Design

Once the fundamental integrability requirements have been established, the process of designing the integrability can begin. Given that SaaS application integrability typically occur over the internet, the integrability architecture must consider the locations of the different on premise source and target systems within the network. Understanding the connections and the interactions between the framework components will offer immediate insight into integrability.

The functionality offered by SaaS applications can be leveraged only if the data to be acted upon is stored within the SaaS application's data tier. This is fundamental to most SaaS applications and often leads to data replication and synchronization between cloud applications.

In the Palestinian e-Government, all application's data are stored in the government data center, and the SOA based framework responsible for managing the data access process for all cloud applications (Section 2.2).

2.4. Software Engineering Models

A software process model is the set of activities and associated results that produce a software product. There are four fundamental process activities that are common to all software processes:

1. Software specification where customers and engineers define the software to be produced and its constraints.
2. Software development where the software is designed and programmed.
3. Software validation where the software is checked and validated against customer requirements.
4. Software evolution where the software is modified to adapt it to changing customer and market requirements.

A software process model is a simplified description of a software process that presents one view of that process. Process models may include activities that are part of the software process, software products and the roles of people involved in software engineering.

Most software process models are based on one of three general models or paradigms of software development:

1. The waterfall approach this takes the above activities and represents them as separate process phases such as requirements specification, software design, implementation, testing and so on. After each stage is defined it is 'signed-off', and development goes on to the following stage.
2. Iterative development this approach interleaves the activities of specification, development and validation. An initial system is rapidly developed from very abstract specifications. This is then refined with customer input to produce a system that satisfies the customer's needs. The system may then be delivered. Alternatively, it may be re-implemented using a more structured approach to produce a more robust and maintainable system.
3. Component-based software engineering: this technique assumes that parts of the system already exist. The system development process focuses on integrating these parts rather than developing them from scratch.

2.5. Evaluation Methods

In this section we introduce evaluation methods used to testify software architecture and validate the proposed framework. The evaluation considers quality attributes and is based on a method used for analyzing software architectures against quality attributes and is called Architecture Tradeoff Analysis Method (ATAM).

2.5.1. Software Architecture Evaluation Using ATAM

Here we address software architecture evaluation methods and the ATAM approach as it is the method to be used for the architecture evaluation. Architectural design decisions determine the ability of the system to meet functional and quality attribute requirements. In the architecture evaluation, the architecture should be analysed to disclose its strengths and weaknesses, while eliciting any risks [14]. Two comparison criteria for software architecture are identified, namely, early software architecture evaluation and late software architecture evaluation. In the thesis work we are using the former one for evaluation, since we are proposing a framework and it does not have detailed architecture components. Early software architecture evaluation has the following features:

- It is a scenario-based evaluation and do not need data measured from implementation.
- It does not require metric usage.
- It can be based on mathematical model, simulation based or experience based.
- It requires the participation of the stakeholders.
- It has several methods that are discussed in a large volume of software engineering literature, among these are: Software Architecture Analysis Method (SAAM), Architecture Level Maintainability Analysis (ALMA), Architecture Tradeoff Analysis Method (ATAM), and Performance Analysis of Software Architecture (PASA)

ATAM is the most suitable for thesis framework evaluation among the mentioned methods since its superior to SAAM, and both ALMA and PASA evaluate for attributes other those to be evaluated in the proposed framework. The ATAM method -developed by the Carnegie Mellon Software Engineering Institute (SEI)-, relies on the elicitation of quality attribute scenarios from a diverse group of system stakeholders. The ATAM is an enhanced method for the SAAM. The purpose of the ATAM is to assess the consequences of architectural decisions in light of quality attribute requirements [15].

The ATAM gets its name because it not only reveals how well an architecture satisfies particular quality goals (such as performance or modifiability), but it also provides insight into how those quality goals interact with each other—how they trade off against each other. Such design decisions are critical; they have the most far-reaching consequences and are the most difficult to change after a system has been implemented [15]. The method was created to uncover the risks and tradeoffs reflected in architectural decisions relating to those quality attribute requirements. Quality attributes, also known as nonfunctional requirements, include usability, performance, scalability, and Interoperability, and so on. One of the positive consequences of using the ATAM

is a clarification and concretization of quality attribute requirements. Quality attribute scenarios give precise statements of usage, performance and growth requirements, various types of failures, and various potential threats and modifications [16]. Once the important quality attributes are identified, the architectural decisions relevant to each high-priority scenario can be illuminated and analysed with respect to their appropriateness [17]. The resulting analysis yields:

- Risks: architectural decisions that might create future problems for some quality attribute.
- Non-risks: architectural decisions that are appropriate in the context of the quality attribute that they affect.
- Tradeoffs: architectural decisions that have an effect on more than one quality attribute.
- Sensitivity points: a property of one or more components, and/or component relationships, critical for achieving a particular quality attribute requirement.

The ATAM analysis of the quality attribute scenarios gives insight into how well a particular transformation framework architecture satisfies the particular quality attribute goals of these scenarios and how certain quality attributes interact with each other in the framework context. The ATAM focuses on quality attribute requirements. The major goals of ATAM are to [15]:

- Elicit and refine a precise statement of the architecture's driving quality attribute requirements.
- Elicit and refine a precise statement of the architectural design decisions.
- Evaluate the architectural design decisions to determine if they satisfactorily address the quality requirements.

In ATAM there are three types of scenarios [15]: use case scenarios (these involve typical uses of the existing system and are used for information elicitation); growth scenarios (these cover anticipated changes to the system), and exploratory scenarios (these cover extreme changes that are expected to "stress" the system). These different types of scenarios are used to probe a system from different angles, optimizing the chances of surfacing architectural decisions at risk.

Examples of each type of scenario follow.

- **Use Case Scenarios**

Use case scenarios describe a user's intended interaction with the completed, running system.

For example,

1. There is a radical course adjustment during weapon release (e.g., loft) that the software computes in 100 ms. (performance)
2. The user wants to examine budgetary and actual data under different fiscal years without re-entering project data. (usability)

3. A data exception occurs and the system notifies a defined list of recipients by e-mail and displays the offending conditions in red on data screens. (reliability)
4. User changes graph layout from horizontal to vertical and graph is redrawn in one second. (performance)
5. Remote user requests a database report via the Web during peak period and receives it within five seconds. (performance)
6. The caching system will be switched to another processor when its processor fails, and will do so within one second. (Reliability).

- **Growth Scenarios**

  Growth scenarios represent typical anticipated future changes to a system. Each scenario also has attribute-related ramifications, many of which are for attributes other than modifiability.

  For example, Scenarios 1 and 4 will have performance consequences and Scenario 5 might have performance, security and reliability implications.

  1. Change the heads-up display to track several targets simultaneously without affecting latency.
  2. Add a new message type to the system's repertoire in less than a person-week of work.
  3. Add a collaborative planning capability where two planners at different sites collaborate on a plan in less than a person-year of work.
  4. Migrate to a new operating system, or a new release of the existing operating system in less than a person-year of work.
  5. Add a new data server to reduce latency in use case Scenario 5 to 2.5 seconds within one person-week.
  6. Double the size of existing database tables while maintaining 1 second average retrieval time.

- **Exploratory Scenarios**

  Exploratory scenarios push the envelope and stress the system. The goal of these scenarios is to expose the limits or boundary conditions of the current design, exposing possibly implicit assumptions. Systems are never conceived to handle these kinds of modifications, but at some point in the future these might be realistic requirements for change. And so the stakeholders might like to understand the ramifications of such changes. For example,

1. Add a new 3-D map feature, and a virtual reality interface for viewing the maps in less than five person-months of effort.
2. Change the underlying Unix platform to a Macintosh.
3. Re-use the 25-year-old software on a new generation of the aircraft.
4. The time budget for displaying changed track data is reduced by a factor of 10.
5. Improve the system's availability from 98% to 99.999%.
6. Half of the servers go down during normal operation without affecting overall system availability.
7. Tenfold increase in the number of bids processed hourly while keeping worst-case response time below 10 seconds.

2.6. Technical Foundations Summarization

Technical foundations presented so far can be summarized in Figure 2.4. It is clear how much theories and techniques can be used and related to the thesis problem. The interactions between these techniques can lead to the design of the transformation framework.

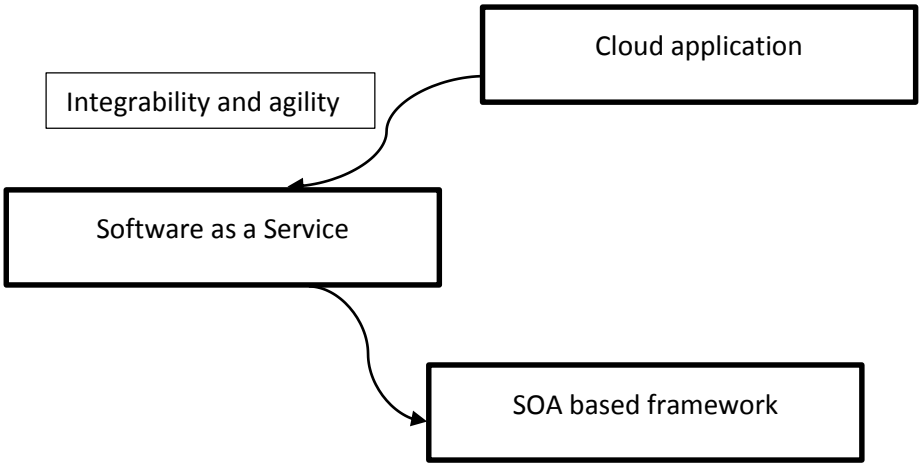

Figure 2.3 Technical Foundations Dependancies

Chapter 3 Related Works

In this chapter we study and investigate frameworks used for cloud computing, show how cloud computing environment may affect the process of transformation or running of applications.

The related works are introduced in two sections, the service engineering methodologies and frameworks which we study and analyse against the research objectives which are the integrability and agility and will help in designing the cloud transformation framework. The other researches are about the impacts of cloud on software engineering and strategies which help in designing the transformation lifecycle component.

3.1. Service Engineering Methodologies and Frameworks.

Software engineering in applications development starts with an explicit process model having framework of activities which are synchronized in a defined way. This process model describes or prescribes how to build software with intermediate visible work products (documents) and the final finished product i.e. the operating software. The whole development process of software from its conceptualization to operation and retirement is called the software development lifecycle (SDLC). SDLC goes through several framework activities like requirements gathering, planning, design, coding, testing, deployment, maintenance and retirement. These activities are synchronized in accordance to the process model adopted for a particular software development. There are many process models used for service engineering like Service Architecture Engineering (SAE), Service-Oriented Analysis and Design (SOAD), Service-Oriented Modelling and Architecture (SOMA) , and Service Migration and Reuse Technique (SMART).

In the next sections we introduce a literature review for these service engineering methodologies, and listing their properties. These properties will be discussed against criteria such as classifications of framework components, service architecture, process lifecycle, project profile, and integrability and agility during transformation process.

3.1.1. Service Architecture Engineering (SAE)

The SAE reference framework is designed to provide a comprehensive framework defined approach for service architecture including taxonomy, classification and policies together with repeatable service engineering processes that guide the delivery of the agile enterprise,

implemented in a knowledgebase with integrability between the architecture concepts, processes, tasks, techniques and deliverables [18].

a. SAE architecture and Components

The SAE components are classified to: Organization, Process, and Architecture as shown in Figure 3.1. These three parts form a triad that describe key aspects of any methodology framework. The process component describes a structure of business processes or activities that a service provisioning organization should follow in order to successfully analyze, plan, design, provision, and run services.

The organization component describes the roles and responsibilities, project profiles, and funding models recommended in order to successfully support the service lifecycle [18].

The Architecture component provides the detailed description of the various views, models and other elements used and created during the execution of the method and how they relate to one another.

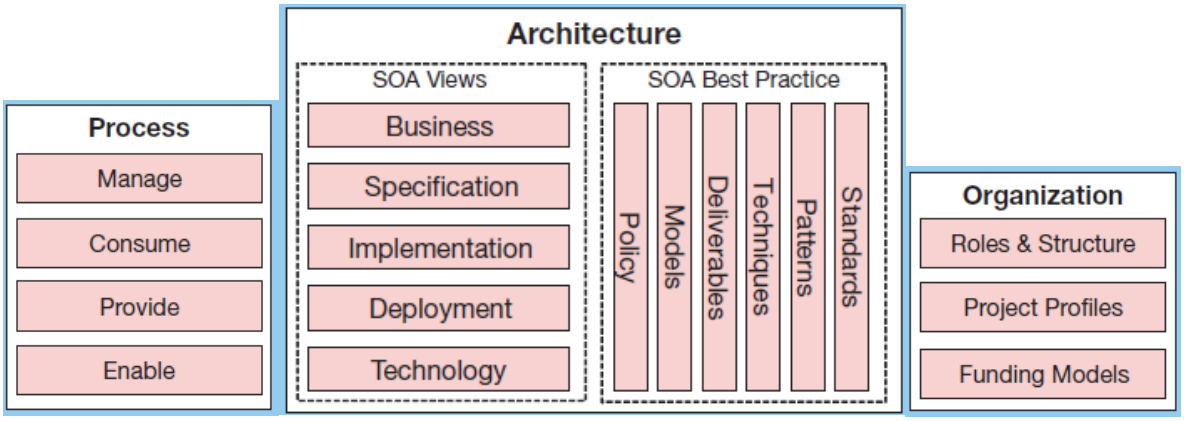

Figure 3.1 SAE SOA reference framework [18]

The architecture component includes five views – Business, specification, Implementation, Deployment, and Technology.

These views comprise a consistent level of abstraction for deliverable artifacts that relate to distinct set of stakeholders.

This provides an effective mechanism for grouping related SOA best practices based on a particular part of the enterprise under study. Each view defines and clusters together the standards, patterns, techniques, deliverables, models, and polices that apply to appropriate view as illustrated in Figure 3.1.

Table 3.1 SAE View Descriptions [18]

View	Purpose	Primary Roles(s)	Enterprise Layer
Business	To understand and analyze business needs and how the business operates in terms of goals and objectives, organizational structure, processes, information, etc.	Business Architect	Business
Specification	To plan and specify software services from a platform independent perspective.It provides a means of thinking in depth about logical services and their interrelationships.	Service Architect	Software
Implementation	To package services into automation units, identify dependencies between the automation units, and to determine the implementation constraints that will govern the internal design and deployment of these units	Service Architect, Software Designer	Software
Deployment	To explore alternative and finally capture deployment choices for run time services. To map implementation view services to deployment units and to construct an optimum configuration on the computing infrastructure.	Infrastructure Architect, Operations Management	Software/Infrastructure
Technology	To ensures technologies are in place to enable software lifecycle in all levels from planning through specifications, design and execution and retirement.	Infrastructure Architect, Operations Management	Infrastructure

Table 3.1 provides a first level of detail on each of the five views:

The business view includes best practices such as policies, patterns, and techniques that provide guidance as to how to capture knowledge about the business.

The specification view comprises the artifacts and models required by architects to specify the functional and non-functional requirements of software solutions and services as well as the architectural dependencies between them. This view is meant to be independent of any particular platform such as application server, operating system or even enterprise service bus.

The specifications view artifacts are: service specification architecture, service dependency diagram, service orchestration diagram, service information model, service description, service specification, and security specification.

The primary artifact of the implementation view is the service implementation architecture that captures the structure of the automation units that implement the services identified in the service specification architecture.

The deployment view provides the mapping of automation units onto nodes or services platforms allowing services or solution architects to communicate with infrastructure architects about how services will run in the production environment.

This ensures that services required for runtime will be available on the platforms that will run the automation units. Second it provides a mechanism for these same service and infrastructure architects to analyse the processing and bandwidth capacity required for each segment of the infrastructure.

The technology view is the last piece in the overall enterprice layering. The purpose of this view is to nail down exactly what the network will look like, polices that will govern service operations and to ensure that the technology base required by the services running in the production environment have all the pieces require.

b. Discussion

The architecture component of the SAE reference framework is been structured into views and best practices in order to support a number of key architectural principles.

The most critical of these principles is separation of concerns. By dividing the structure into views, architects can separate business concerns from software concerns, logical concerns from technology concerns and so on. This separation in addition to allowing the architect to focus on a particular concern without having to remember all the others, also improves the maintainability by “chunking” the architecture into manageable pieces.

Adaption of SAE reference framework is an evolutionary process. Techniques, and particularly patterns and policies will evolve with SOA maturity. In the early stages many polices will probably be advisory but with more experience they may well become strongly recommended or mandatory.

The SAE framework provides an easy way in integrability between framework components, since the architecture is defined well, and the clear sequence in framework adaption activities can

support the agility in transformation process. This methodology can be used as a base model in designing the Palestinian e-Government cloud framework.

3.1.2. Service-Oriented Analysis and Design (SOAD)

A service oriented analysis and design (SOAD) process has been defined as shown in Figure 3.2 for developing Saas based applications in an enterprise context [19]. The basic concepts of Service-Oriented Architectures (SOAs) and Web services are becoming part of our everyday language and recognized as a suitable architectural style for crafting modern enterprise applications. In this context, the underlying issues of: what makes good services are becoming increasingly critical for ensuring the successful implementation of SOAs [20].

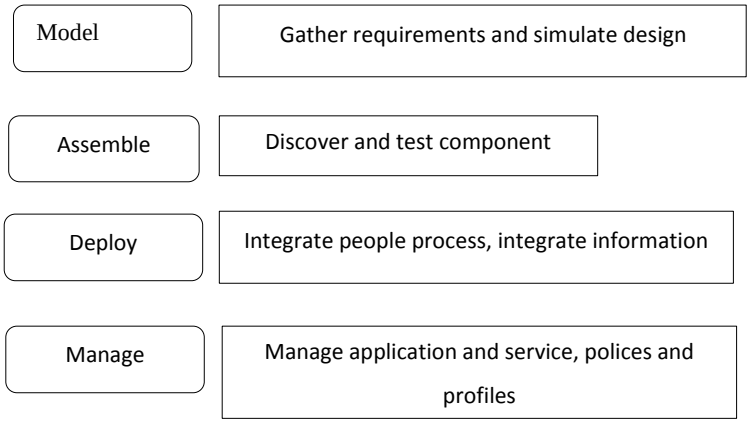

Figure 3.2 SOAD Architecture and Components[20]

a. SOAD architecture and components

The definition of SOAD model contains the following four phases wherein with each phase, the development of the design gets tuned up and the final formalized structure can be derived. [21].

Model: This phase collects the requirement specifications from the consumer and models the suitable services with each requirement specified. After the accumulation of requirements and modelling, the simulation of the complete structure can be prepared in order to get the basic idea of the architecture.

Assemble: Once the requirement model is prepared, then corresponding services which performs the required task are identified and testing the same can be performed. A locator can perform the task of service identification and the verification of the same. This phase is preparing the full proof model of services with verification and testing.

Deploy: The deployment phase describes the integrability of various security parameters with the required specifications. The integrability of services in a specific process and the application environment is possible only when they are compatible with each other and the security parameters should be embedded with each service. The federated interface can be used to provide the connectivity among the similar user groups.

Manage: This phase is enabling the architecture to manage the processes and the individual services in the application environment. The management of the performance level is an important feature of the proposed architecture in order to provide sustainability to the process execution. This helps the workflow to perform the activity successfully without any further verifications or the analysis. Any degradation in the performance level is supplemented by another service with the same specifications.

With the SOAD framework, the services present in the proposed model can be classified into the following categories.

- Educational services: Supports the educational purposes. The services like form filling, mark sheet generation can be classified into this category.
- Management services: These are the services which helps the system to maintain the service collaboration within the application environment which also takes care of the security maintenance. Example: SAS, AAA services.
- Process services: These services help the system to have a correlation and collaboration among the services. This enables the smooth process choreography. Example: Orchestration services, service brokers
- Interaction services: These services are enabling the interaction among various services and processes in a single working environment. Example: Interfaces & federated services.

b. Discussion

The SOAD model is a simple architecture which contains the components in every level in a transparent way. The structure can be prepared on runtime by choosing the services independently and combined to produce the complete system architecture.

Any addition and deletion of services within the process choreography can be performed independently without affecting the existing working environment.

In a SOA, all services follow the same design philosophy (which is articulated through patterns and templates) and interaction patterns; the underlying architectural style can easily be identified which leads to a good integrability services.

The development of the services and service consumers requires only basic programming language skills in addition to domain knowledge; middleware expertise is only required for a few specialists, that in an ideal world, work for tool and runtime vendors, and not for the companies crafting enterprise applications as SOAs.

3.1.3. Service Oriented Modelling and Architecture (SOMA)

Service Oriented Modelling and Architecture, referred to as SOMA, is a method proposed by a group of IBM architects and members of prestigious SOA-related organizations, to guide the adoption of SOA within a company. In this process, several stages could be identified: analysis, design, implementation and deployment of services. For each of these, SOMA specifies techniques to be used, roles and a work breakdown structure (WBS) consisting of tasks, their specific input and output, and several rules regarding their execution [22].

The method is based on a fractal software development model which principles influence the manner of using SOMA:

- The seven phases consisting of several capabilities are applied incrementally and iteratively. Therefore, managing risks becomes a continuous process based on the analysis of the service-dependencies in each iteration.
- The capabilities are applied in a similar way, independent of the scope (enterprise, business unit, project) but not identical as each scope has its own characteristics.

As previously mentioned the rules of executing the phases are dictated by the fractal model. Two of the phases, Business modeling and transformation and Solution management, are related to the preparation stage in adopting service oriented architecture. Their output is the creation of enterprise architecture documentation, solution templates, transformation roadmap and project management planning. The other phases, Identification, Specification, Realization and Implementation, have as final result the instantiation of the SOA Reference Framework. As phases are applied, their produced deliverables are used for populating SOA reference framework. This main deliverable offers several advantages.

a. SOMA architecture and components

At the heart of SOMA is the identification and specification of services, components, and process flows. At a high level, SOMA is a three-phased approach to identify, specify, and realize services as

shown in Figure 3.3, components, and flows (typically, choreography of services). The first phase is that of service identification, where various techniques are used to identify an exhaustive list of candidate services. The second phase is that of service specification, in which a detailed design of services and components is completed. The realization phase focuses on making architectural decisions, justifying the most prudent approach to implement the services [23].

One of the main outputs of the SOMA method is a service model. It is recommended that a service model constitute of the following artifacts about services [22]:

- Service portfolio—List of all the enterprise services
- Service hierarchy—A categorization of services into service groups
- Service exposure—An analysis and rationalization of which services should be exposed and which should not
- Service dependencies—Representing the dependencies of services on other services
- Service composition—How services take part in compositions to realize business process flows
- Service NFRs—The nonfunctional requirements that a service must comply with
- State management—The various types of states that a service should maintain and implement
- Realization decisions—Architectural decisions, for each service, around the most justified mechanism to implement the service

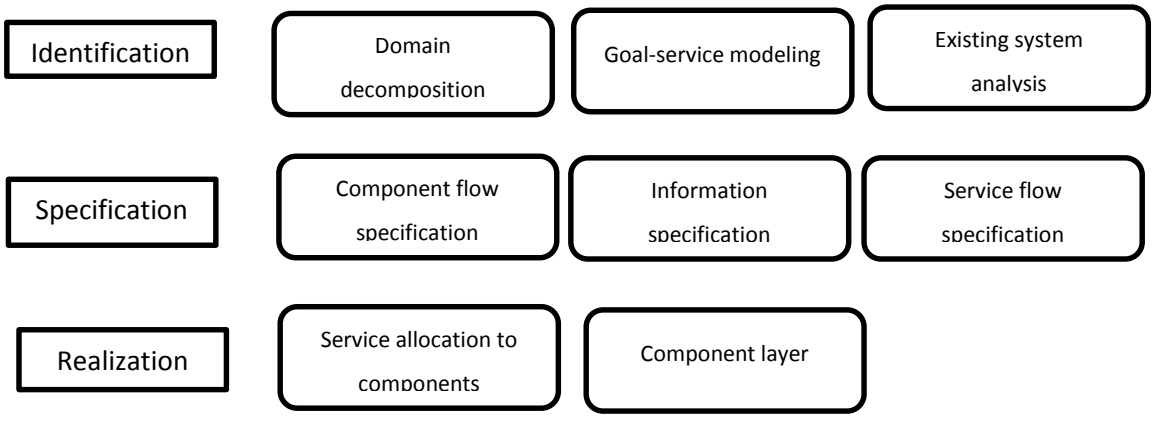

Figure 3.3 SOMA Architecture and Components [22]

b. Discussion

SOMA provides specific guidance on the analysis and design activities for determining the three fundamental aspects of a service-oriented architecture: services, flows, and components that realize the services. The limitation in SOMA framework are includes: the lack of the software lifecycle development, and there are no clear relation between components. This limitation will reduce the integrability and make the agility in development weak.

3.1.4. Service Migration and Reuse Technique (SMART)

The Service-Oriented Migration and Reuse Technique (SMART) [24] was developed to assist organizations in analysing legacy capabilities for use as services in an SOA.

SMART gathers a wide range of information about legacy components, the target SOA, and potential services to produce a service migration strategy as its primary product. However, SMART also produces other outputs that are useful to an organization whether or not it decides on migration. SMART input (from documentation and interviews) and output are depicted in Figure 3.4.

a. SMART architecture and components

SMART consists of five major components, each divided into several tasks. The components and generalized process and information flows of SMART are depicted in Figure 3.5. However, the number of artifacts considered, the time required, and the specific activities of a given application of SMART depend on previous activities and expectations of the requesting organization. For example, if the requesting organization has specific legacy components in mind for migration, SMART activities will be focused on those components.

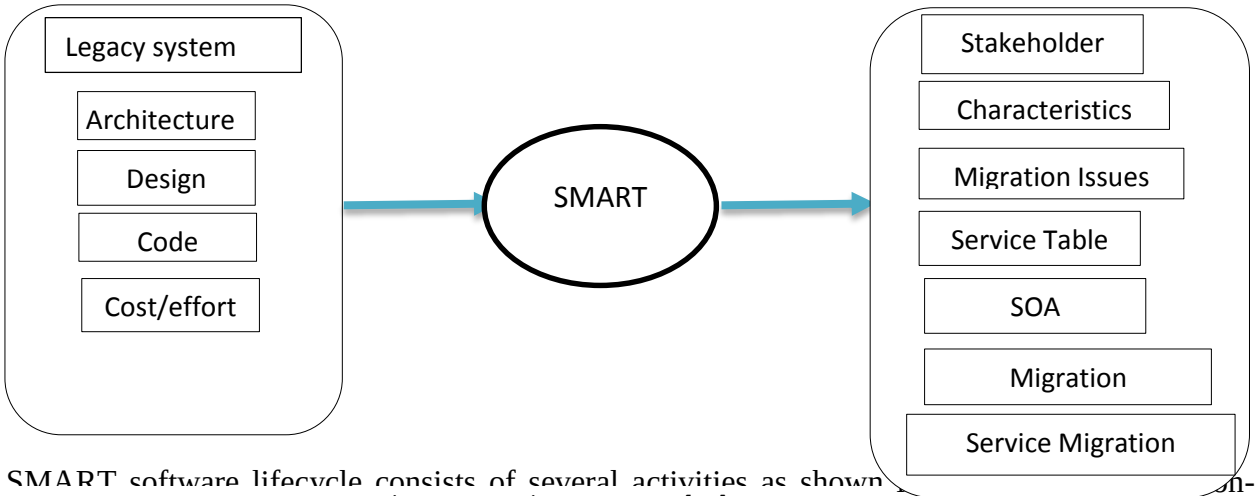

Figure 3.4 SMART Architecture and Components [24]

SMART software lifecycle consists of several activities as shown [illegible] Service Migration Interview

Guide (SMIG). The SMIG contains questions that directly address the gap between the existing and target architecture, design, and code, as well as questions concerning issues that must be addressed in service migration efforts. Use of the SMIG assures broad and consistent coverage of the factors that influence the cost, effort, and risk involved in migration to services.

It is not necessary for the team to complete all data gathering during these initial activities. Additional opportunities are provided during the Analysis activity.

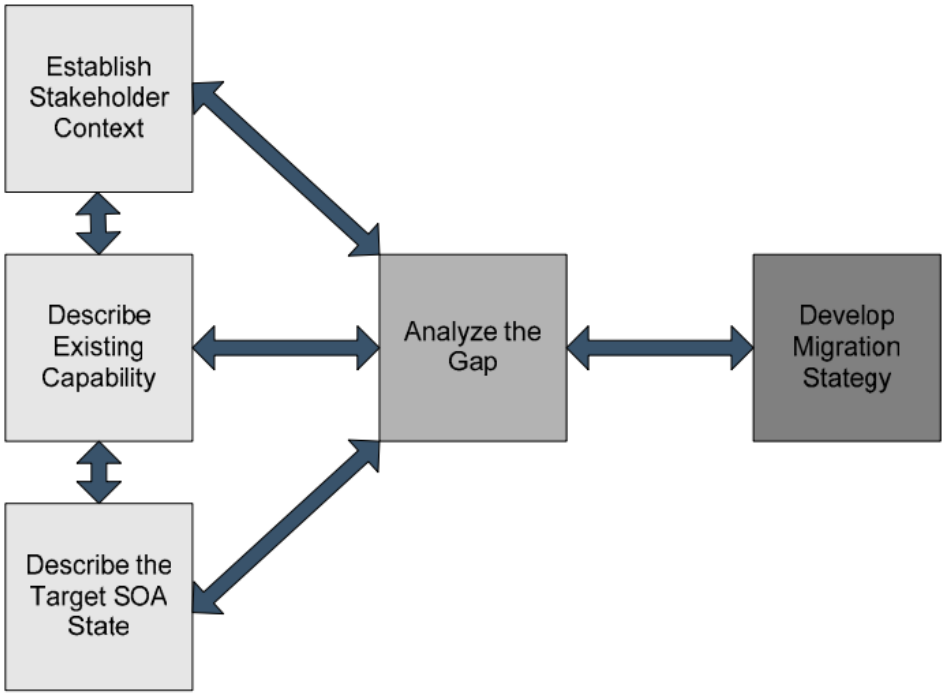

Figure 3.5 SMART Activities [24]

b. Discussions

As introduced previously, the SMART framework consists of several components which used for transforming legacy applications to software as a service, these components interacts as input and output to the model, and the result will be a full transformed application.

The software lifecycle is described as several activities and tasks, these activities explained well and make the transformation process agile.

3.1.5. Discussions and Conclusions

As mentioned previously, there are many process models used for service engineering like SAE, Service Architecture Engineering, SOAD, Service-Oriented Analysis and Design, SOMA, Service-Oriented Modelling and Architecture , and SMART (Service Migration and Reuse Technique).

Some of these models interested in the service lifecycle, and the others introduces a full solution for service transformation starting from analysis to implementation.

The architecture component of the SAE reference framework is been structured into views and best practices in order to support a number of key architectural principles.

Adaption of SAE reference framework is an evolutionary process. Techniques, and particularly patterns and policies will evolve with SOA maturity. In the early stages many polices will probably be advisory but with more experience they may well become strongly recommended or mandatory.

The SAE framework provides an easy way in integrability between framework components, since the architecture is defined well. The Reference framework triad – Organization, Process, and Architecture describe key aspects of any methodology framework. The process component describes a structure of business processes or activities that a service provisioning organization should follow in order to successfully analyze, plan, design, provision, and run services [18].

Service Migration and Reuse Technique (SMART) [24] consists of five major components, each divided into several tasks. However, the number of artifacts considered, the time required, and the specific activities of a given application of SMART depend on previous activities and expectations of the requesting organization. For example, if the requesting organization has specific legacy components in mind for migration, SMART activities will be focused on those components.

In Service Oriented Modelling and Architecture , referred to as SOMA[22], several components could be identified: analysis, design, implementation and deployment of services. For each of these, SOMA specifies techniques to be used, roles and a work breakdown structure (WBS) consisting of tasks, their specific input and output, and several rules regarding their execution.

Service-oriented analysis and design SOAD [21] notation and process remain to be defined in detail, components such as service conceptualization (or identification), service categorization and aggregation, policies and aspects, meet-in-the-middle process, semantic brokering, and service harvesting (for reuse) can already be identified.

SOAD will require enhancements to existing software engineering methods, further improving their usability and applicability to enterprise application development projects. Related best practices and patterns will evolve over time.

Table 3.2 discusses these service models compared and listing properties and advantages.

Table 3.2 Service Engineering Model Properties.

Criteria	SAE	SOAD	SOMA	SMART
Classifications of the framework components depending on its jobs	Classified to 3 main components, each component classified to views	Classified to 3 layers and each layer classified to components	Classified to 3 types of service classifications	The framework classified to input and output components.
Service architecture including taxonomy, classification and policies	The SOA view discusses the service architecture in detailed steps.	Service layer discusses the service architecture.	The seven phases consisting of several capabilities	In the input component the service layer is classified
Process lifecycle	In the process component the framework discuss the process lifecycle.	Meet-in-the-middle processes The bottom-up approach tends to lead to poor business-service abstractions in case the design is dictated by the existing IT environment, rather than existing and future business needs	The seven phases consisting of several capabilities are applied incrementally and iteratively	The SMART activities list the process lifecycle.
Project profile, polices, and monitoring	The organization component list the project profile and monitoring models	Service harvesting and knowledge brokering model	The final phase, Deployment, monitoring and management, could trigger various changes in the services landscape	Develop Migration Strategy discusses project profiles

3.2. Impacts of Cloud on Software Engineering

3.2.1. Federal Cloud Computing Strategy

Cloud computing describes a broad movement to treat IT services as a commodity with the ability to dynamically increase or decrease capacity to match usage needs. By leveraging shared infrastructure and economies of scale, cloud computing present's Federal leadership with a compelling business model. It allows users to control the computing services they access, while sharing the investment in the underlying IT resources among consumers [25].

a. IT will be simpler and more productive

Cloud computing also provides an indirect productivity benefit to all services in the IT stack. For example, less effort will be required to stand up and develop software testing environments, enabling application development teams to integrate and test frequently in production-representative environments at a fraction of the cost of providing this infrastructure separately.

b. Agility improvements will make services more responsive

The impact of cloud computing will be far more than economic. Cloud computing will also allow agencies to improve services and respond to changing needs and regulations much more quickly.
With traditional infrastructure, IT service reliability is strongly dependent upon an organization's ability to predict service demand, which is not always possible.
Cloud computing provides an important option for agencies in meeting short-term computing; agencies need not invest in infrastructure in cases where service is needed for a limited period of time.

c. Services will be more scalable

With a larger pool of resources to draw from, individual cloud services are unlikely to encounter capacity constraints. As a result, government services would be able to more rapidly increase capacity and avoid service outages. Given appropriate service level agreements and governance to ensure overall capacity is met, cloud computing will make the government's IT investments less sensitive to the uncertainty in demand forecasts for individual programs, which frequently emerge rapidly in response to national program needs which cannot be foreseen in the early stages of the Federal budget cycle.

d. Decision Framework for Cloud transformation

The broad scope and size of the cloud transformation will require a meaningful shift in how government organizations think of IT. Organizations that previously thought of IT as an investment

in locally owned and operated applications, servers, and networks will now need to think of IT in terms of services, commoditized computing resources, agile capacity provisioning tools, and their enabling effect for citizens. This new way of thinking will have a broad impact across the entire IT service lifecycle – from capability inception through delivery and operations. The following structured framework presents a strategic perspective for agencies in terms of thinking about and planning for cloud transformation [25].

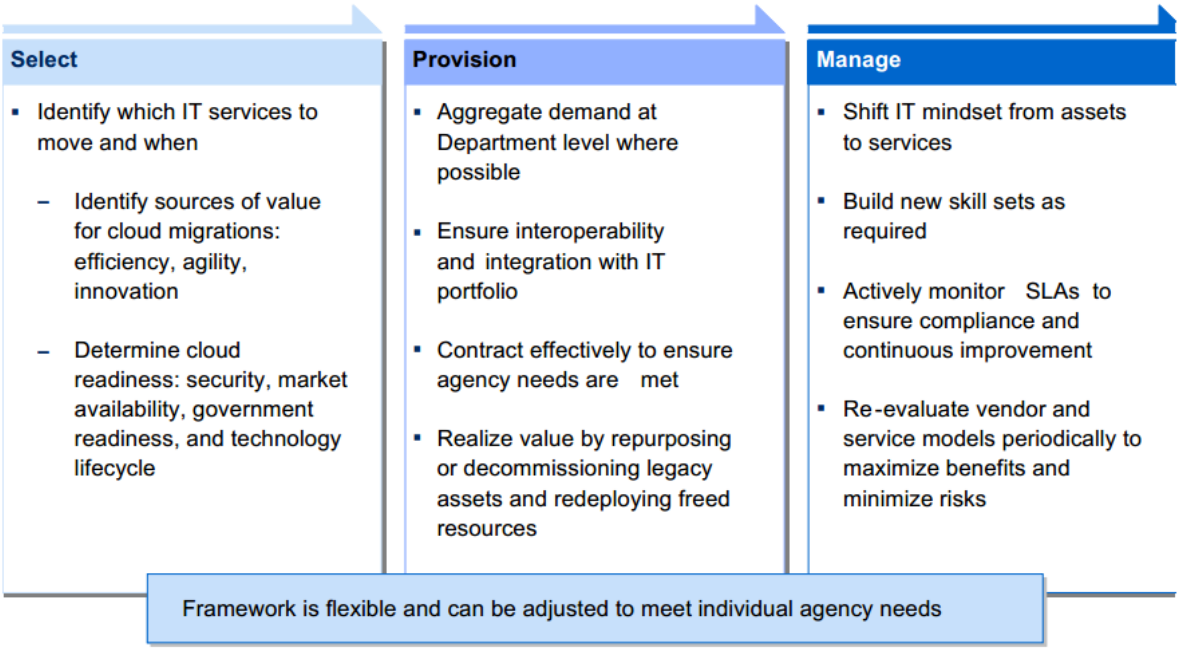

Figure 3.7 Decision Framework for Cloud transformation [25]

e. Selecting services to move to the cloud

Successful organizations carefully consider their broad IT portfolios and create roadmaps for cloud deployment and transformation. These roadmaps prioritize services that have high expected value and high readiness to maximize benefits received and minimize delivery risk.

Defining exactly which cloud services an organization intends to provide or consume is a fundamental initiation phase activity in developing an agency roadmap.

f. Provisioning cloud services effectively

To effectively provision selected IT services, agencies will need to rethink their processes as provisioning services rather than simply contracting assets. Contracts that previously focused on metrics such as number of servers and network bandwidth now should focus on the quality of service fulfillment.

Aggregate demand: When considering "commodity" and common IT services, agencies should pool their purchasing power by aggregating demand to the greatest extent possible before transforming services to the cloud. Where appropriate, demand should be aggregated at the departmental level and as part of the government-wide shared services initiatives such as government-wide cloud-based email.

Integrate services: Agencies should ensure that the provided IT services are effectively integrated into their wider application portfolio. In some cases, technical experts may be required to evaluate architectural compatibility of the provided cloud service and other critical applications. Rather than a one-time event, this principle should be followed over time to guarantee that systems remain interoperable as individual IT services evolve within the portfolio. Business process change may similarly be required to properly integrate the systems.

Contract effectively: Agencies should also ensure that their contracts with cloud service providers set the service up for success. Agencies should minimize the risk of vendor lock-in, for instance, to ensure portability and encourage competition among providers. Agencies should include explicit service level agreements (SLAs) for security, continuity of operations, and service quality that meet their individual needs. Agencies should include a contractual clause enabling third parties to assess security controls of cloud providers [25].

Realize value: Agencies should take steps during transformation to ensure that they fully realize the expected value. From an efficiency standpoint, legacy applications and servers should be shut down and decommissioned or repurposed. Data center real estate used to support these systems should be closed down or used to support higher value-add activities. Where possible, staff supporting these systems should be trained and re-deployed to higher-value activities. From an agility and innovation standpoint, processes and capabilities may also need to be refined in order to fully capture the value of the investment.

g. Discussion

This Federal Cloud Computing Strategy is designed to articulate the benefits, considerations, and trade-offs of cloud computing, It provide a decision framework to support agencies in transformation towards cloud computing, It also highlight cloud computing implementation resources, identify federal government activities and roles and responsibilities for analysing cloud adoption.

Following the publication of this strategy, each agency will re-evaluate its technology sourcing strategy to include consideration and application of cloud computing solutions.

3.2.2. Analytical Study of Agile Methodology with Cloud Computing

Agile development methodologies and Cloud Computing complement each other very well. Cloud Services take pride in meeting user requirements rapidly.

The agile system of software development aims to break down project requirements into little, achievable segments. This approach guarantees user feedback on all task of the project [26]

a. Cloud Computing and Agile Development

Cloud computing is the perfect environment for agile development. It lets you get valuable functionality to your customers quickly, collect instantaneous feedback, and make quick changes based on that feedback. This rapid development cycle, an inherent benefit of cloud computing, are impossible to implement in the conventional development model because of the huge cost of distribution.

b. Advantages of Cloud Computing with Agile Development:

- Shortened development cycle-time of 75%.
- Higher stability of work-loads.
- Higher utilization of work-load that is, developing large-scale, software systems with a fixed number of developers.
- Higher quality by earlier feedback from the customers.
- Higher flexibility to change of Management and development plans.
- Reduce the cost of moving information between people.
- Place people physically closer.
- Reduce the elapsed time between making a decision to seeing the consequence of that decision.
- Replace documents with talking in person and at whiteboards.

c. Agile development Lifecycle in Cloud Computing

Agile development methodology attempts to provide a lot of opportunities to assess the direction of a project during the development lifecycle. This is achieved through regular cadences of work, well-known as sprints or iterations as shown in Figure 3.8, at the end of which teams must present a shippable increment of work. Thus by focusing on the repetition of shortened work cycles as well as the functional product they yield, agile methodology could be explained as “iterative”

and "incremental." In waterfall, development teams just have one chance to get each aspect of a project right. In an agile paradigm, every aspect of requirements, development, design, etc. is continually revisited during the lifecycle.

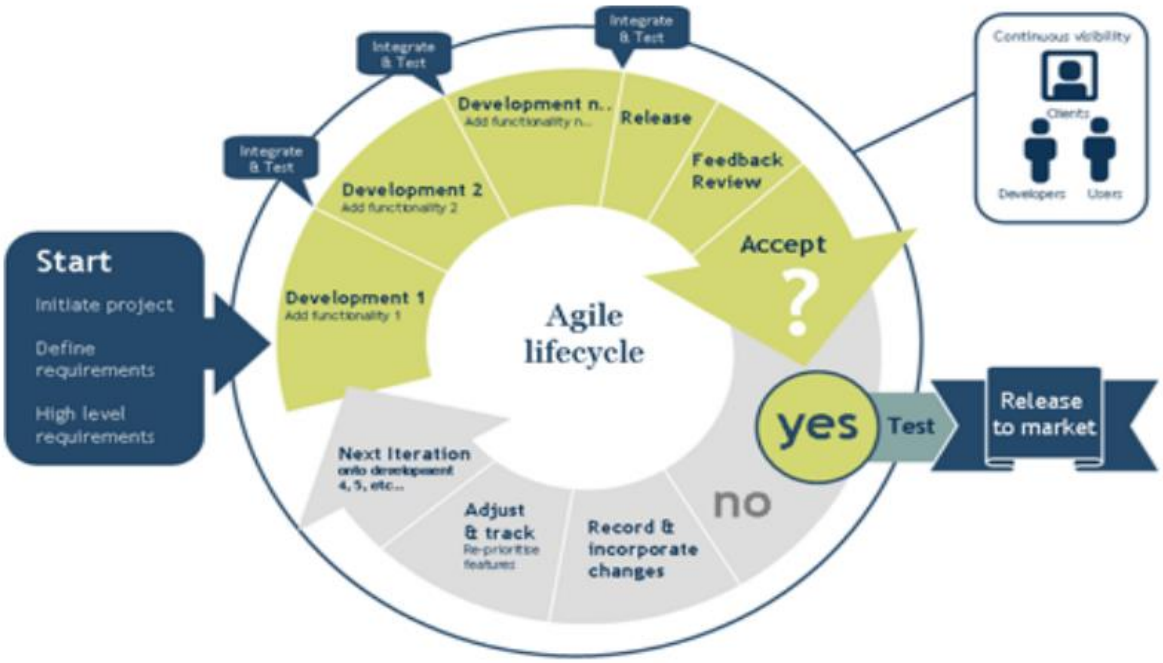

Figure 3.8 Agile Development Lifecycle [26]

3.2.3. Impact of Web 2.0 and Cloud Computing Platform on Software Engineering

Current era of Web 2.0 is enabling new business models for using the semantic web. One such Business model is leasing out computing platform of hardware and software over the internet to the tenants and is dubbed as Cloud Computing. The anticipated future trend of computing is believed to be this cloud computing as it promises a lot of benefits like no capital expenditure, speed of application deployment, shorter time to market, lower cost of operation and easier maintenance for the tenants [27].

a. Impact of Cloud Computing on Software Engineering: Challenges

In the rapidly changing computing environment with web services and cloud platform, software development is going to be very challenging. Software development process will involve heterogeneous platforms, distributed web services, multiple enterprises geographically dispersed all over the world.

Existing software process models and framework activities are not going to be adequate unless interaction with cloud providers is included.

Requirements gathering phase so far included customers, users and software engineers. Now it has to include the cloud providers as well, as they will be supplying the computing infrastructure and maintain them too. As the cloud providers only will know the size, architectural details, virtualization strategy and resource utilization of the infrastructure, planning and design phases of software development also have to include the cloud providers. The cloud providers can help in answering these questions on: How many developers are needed, Component Reuse, Cost estimation, Schedule Estimation, Risk Management, Configuration Management, Change Management, and Quality Assurance [27].

Because of the component reuse of web services the size of the software in number of kilo- lines of code (KLOC) or number of function points (FP) to be newly developed by the software engineer will reduce but complexity of the project will increase many folds because of lack of documentations of implementation details of web services and their integrability requirements. Only description that will be available online is the metadata information of the web services to be processed by the computers automatically.

Only coding and testing phases can be done independently by the software engineers. Coding and testing can be done on the cloud platform which is a huge benefit as everybody will have easy access to the software being built. This will reduce the cost and time for testing and validation.

But software developers have to use the web services and open-source software freely available from the cloud instead of procuring them. Software developers should have more expertise in building software from readily available components than writing it all and building a monolithic application. Refactoring of existing application is required to best utilize the cloud infrastructure architecture in a cost effective way.

Maintenance phase also should include the cloud providers. There is a complete shift of responsibility of maintenance of the infrastructure from software developers to cloud providers. Now because of the involvement of the cloud provider the customer has to sign contract with them as well so that the “Software Engineering code of ethics” are not violated by the cloud provider. In addition, protection and security of the data is of utmost importance which is under the jurisdiction of the cloud provider now [27].

Now we analyze how difficult will be the interaction between cloud providers and the software engineers, The amount of interactions between software engineers and cloud providers will depend on type of cloud like public, private and hybrid cloud involvements. In private cloud

there is more control or self-governance by the customer than in public cloud. Customer should also consider using private cloud instead of using public cloud to assure availability and reliability of their high priority applications.

b. Proposed Software Process Model

Innovative software engineering is required to leverage all the benefits of cloud computing and mitigate its challenges strategically to push forward its advances. [27] Propose an extended version of Extreme Programming (XP), an agile process model for cloud computing platform and name it Extreme Cloud Programming Figure 3.10.

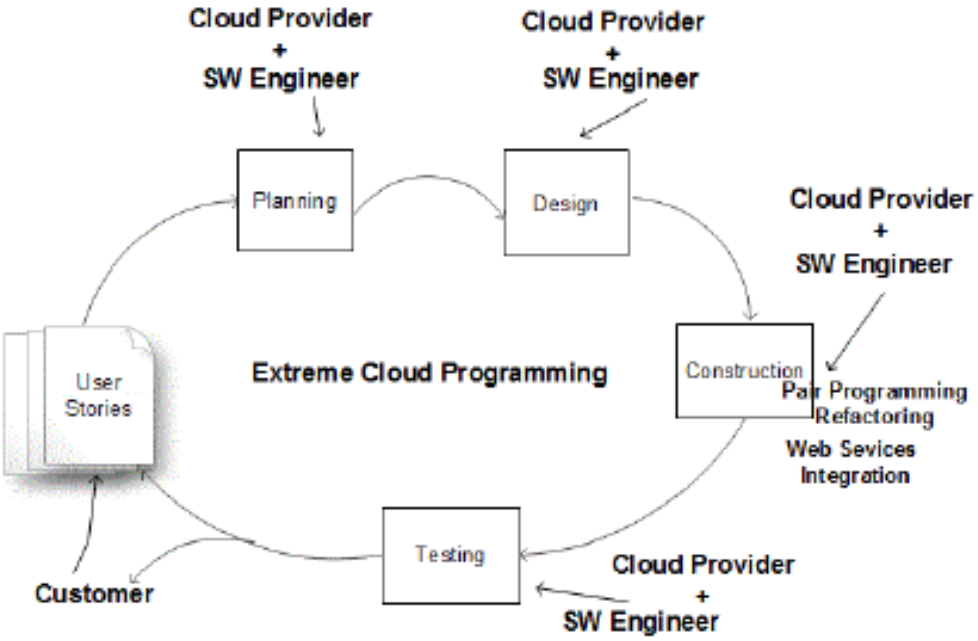

Figure 3.9 Extreme Cloud Programming [27]

All the phases like planning, design, construction, testing and deployment need interaction with the representatives from cloud provider. Resource accounting on cloud platform will be done by the cloud provider in the requirement gathering phase. Software architecture, software architecture to hardware architecture mapping, interface design, data types design, cost estimation and schedule estimation of the project all should be done in collaboration with the cloud provider.

During the construction phase of the application if web services are integrated where many different enterprises are involved then error should be mitigated with the mediation of the cloud provider. Maintenance contract with cloud provider will be according to the Quality of Service agreement.

c. Discussion

[28] Analyses how cloud computing on the background of Web 2.0 is going to impact the software engineering process to develop quality software. As the cloud provider is an external entity or third party, how difficult will be the interaction with them? How to separate the roles of SW engineers and cloud providers? SW engineering should include framework activities to leverage all the benefits of cloud computing systematically and strategically, The main thesis was that the prevalent SW process models should involve the cloud provider in every steps of decision making in software development lifecycle to make the software project a success.

[28] Focuses on the roles of cloud provider in software development cycle, and not mentioned the changes made in the model of the software engineering and they did not design a real cloud transformation framework neither the cloud platform.

3.2.4. Model-Based Migration of Legacy Software Systems to Scalable and Resource-Efficient Cloud-Based Applications: The CloudMIG Approach

Newly developed enterprise software may easily be designed for utilizing cloud computing technologies in a Greenfield project. Though, SaaS providers may also consider to grant responsibility of operation and maintenance tasks to a cloud provider for an already existing software system. Running established enterprise software on a cloud computing basis usually involves extensive reengineering activities during the migration [29].

Nevertheless, instead of recreating the functionalities of an established software system from scratch for being compatible with a selected cloud provider's environment, a migration enables the SaaS provider to reuse substantial parts of a system. The number of system parts which might be migrated is dependent on the weighting of several parameters in a specific migration project.

a. The CloudMIG actitivities

CloudMIG is composed of six activities for migrating an enterprise system to a cloud environment. It provides model driven generation of considerable parts of the system's target architecture. Feedback loops allow for further alignment with the specific cloud environment's properties and foster resource efficiency and scalability on an architectural level. Its activities (A1-A6) are briefly described in the following including the involved models.

Activity A1 - Extraction

The knowledge about the internal structure is often incomplete, erroneous, or even missing. As CloudMIG utilizes a model transformation during generation of its target architecture, a representation of the software system's actual architecture has to be available first. Concerning this issue, an appropriate model is extracted by means of a software architecture reconstruction methodology.

CloudMIG includes the extraction of an established software system's utilization model acting as a starting point. The utilization model includes statistical properties concerning user behavior like service invocation rates over time or average submitted datagram sizes per request.

Activity A2 - Selection

Common properties of different cloud environments are described in a cloud environment meta-model. Selecting a cloud provider specific environment as a target platform for the migration activities therefore implies the selection of a specific instance of this meta-model. For example, this meta-model comprises entities like VM instances or worker threads for IaaS and PaaS-based cloud environments, respectively. As a result, for every cloud environment which shall be targeted with CloudMIG a corresponding meta-model instance has to be created once beforehand. Transformation rules define possible relationships to the architecture metamodel.

Activity A3 - Generation

The generation activity produces three artefacts, namely a target architecture, a mapping model, and a model characterizing the target architecture's violations of the cloud environment constraints.

Activity A4 - Adaptation

The activity A4 allows the reengineer to adjust the target architecture manually towards case-specific requirements that could not be fulfilled during generation activity A3.

Activity A5 - Evaluation

For being able to judge about the produced target architecture and the configured capacity management strategy, A5 evaluates the outcomes of the activities A3 and A4. The evaluation involves static and dynamic analyses of the target architecture.

Activity A6 - Transformation

This activity comprises the actual transformation of the enterprise system from the generated and improved target architecture to the aimed cloud environment. No further support for actually accomplishing the implementation is planned at this time.

3.3. Discussions and Conclusions

As presented previously the cloud computing development has an important issue in software engineering, transforming applications to cloud needs a good software development lifecycle process, the architecture of cloud environment needs to be structured well.

Most of the presented researches presented the cloud architecture but most of them did not discuss the cloud components.

The architecture of the cloud presented in these researches has been identified depending on the specific situation, and could not be adapted in other situation.

The researches that presented the cloud architecture did not presented a transformation lifecycle, neither mention the software engineering model for development in cloud.

Finally, in this research we will present a cloud computing framework for transforming Palestinian e-government applications, define its components, design a software model, and define transforming lifecycle.

Chapter 4 Current Palestinian Cloud Environment

Currently there are multiple of software and applications developed for the Palestinian e-Government, some of these applications are hosted as standalone applications in the cloud and does not need to be transformed to the cloud, but most of these applications are need to be transformed into cloud application.

Next we introduce the current Palestinian cloud infrastructure according to the government computer centre system manager, the essential characteristics, the service models, the deployment models, and we introduce the current software engineering model and discuss the model applicability for applications transformation to cloud.

4.1. Palestinian Cloud Infrastructure

According to the structure of the Government Computer Centre (GCC), the current infrastructure of the Palestinian e-Government cloud is installed and the applications are ready for transforming to the new cloud environment. The cloud infrastructure as shown in Figure 4.1 consists of a main data centre that is composed of two clustered physical servers with a shared storage.

The current e-Government applications are hosted on the virtual machines and benefits from the cloud infrastructure as discussed in Section 4.2. This cloud model is composed of three service models, and four deployment models.

4.1.1. Service Models

a. **Software as a Service (SaaS).** The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

Currently there are various applications provided to the customers and hosted on the government cloud, these applications are provided as software as a service.

As a case the government email system as shown in Figure 4.2 provided as a software as a service and serves more than 20 ministries and institutes. Every ministry account has access to the email system, manage, monitor, and use the email system as it owns the system.

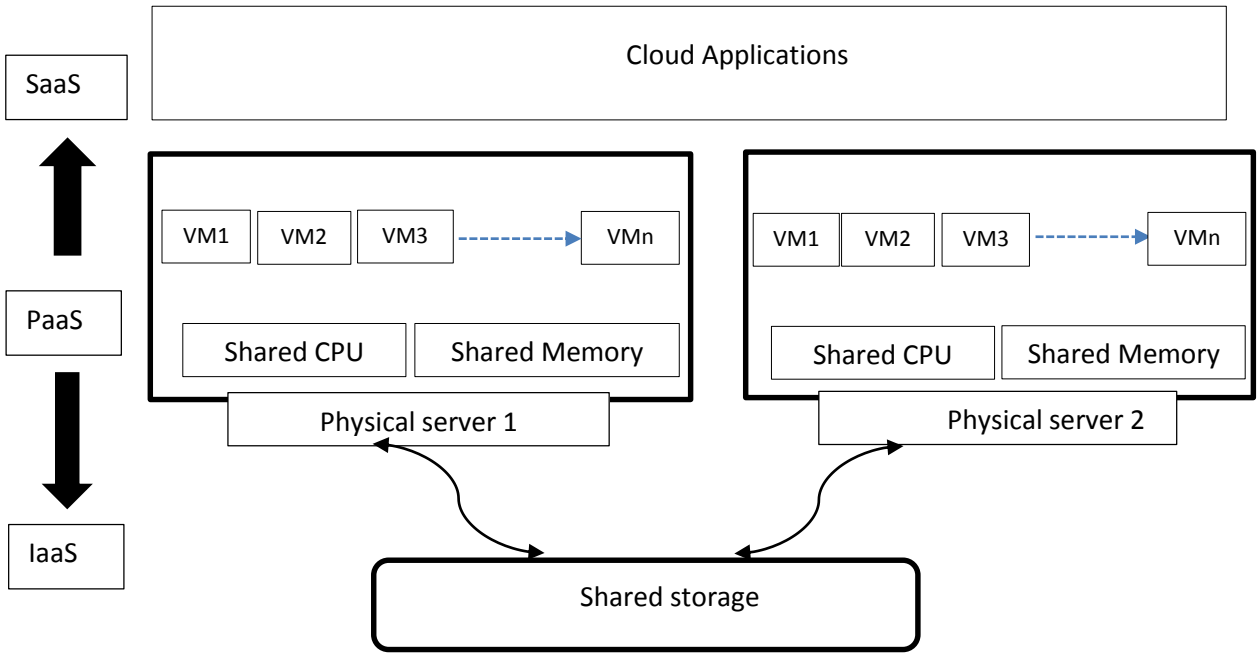

Figure 4.1 The Palestinian Cloud Infrastructure

The government computer centre has the full responsibility for installing the infrastructure, manage the email's protocols, securing email sending and receiving, and the email backup and storage. The ministry email account has the rights to create special email domain accounts or use the global government domain (.gov.ps).

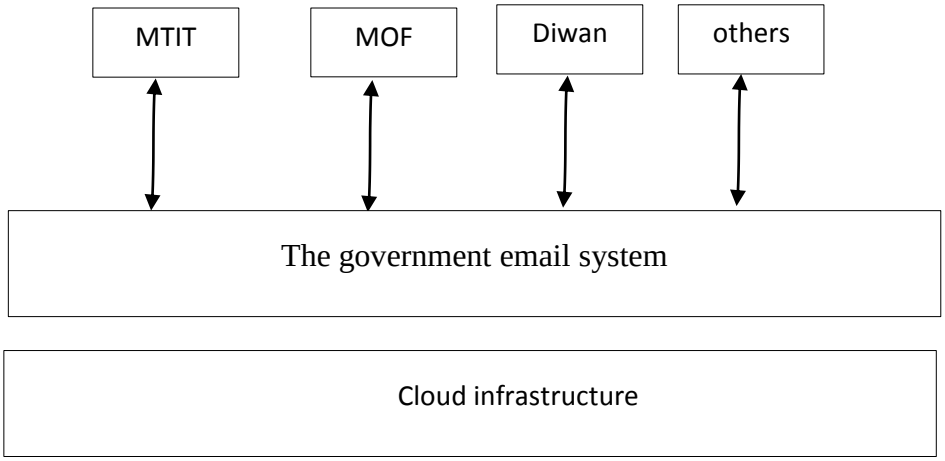

Figure 4.2 The Government Email System (Saas Case)

b. **Platform as a Service (PaaS).** The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control

the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

Currently there are various ministries and institutes benefit from the cloud infrastructure as platform as a service, the Palestinian cloud provides platform services such as php application server, mysql database, oracle database, and provides interface to manage the platform.

As a case the ministry of health develop its own applications and services using the platform of the government cloud. According to the general manager of the information technology centre in the ministry of Health (MOH), the Pharmacy management system is developed in MOH by MOH developers, and hosted on the e-Government cloud environment to benefit from the infrastructure provided by the cloud environment such as the database system, the development environment, the programming platform as shown in Figure 4.3.

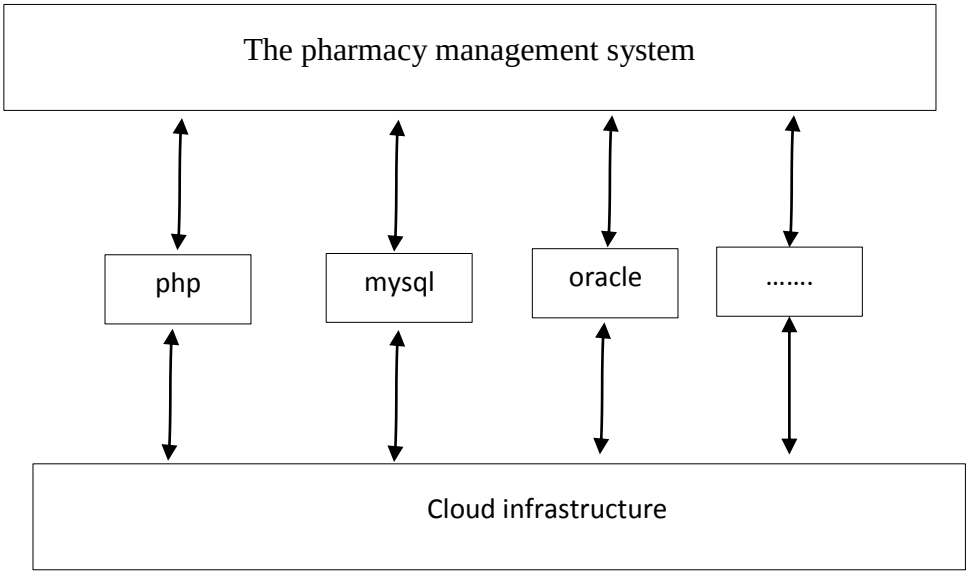

Figure 4.3 The Pharmacy System (Paas Case)

c. **Infrastructure as a Service (IaaS).** The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls). Currently there are various ministries and institutes benefit from cloud infrastructure as infrastructure as a service.

As a case the ministry of interior (MOI) benefit from the cloud infrastructure and install their operating systems and manage their applications as if it is standalone. e.g the MOI webserver is a virtual server hosted in the e-Government cloud to benefit from the infrastructure of the cloud.

4.1.2. Deployment Model

Based on the well-known deployment models introduced in section 2.1.2, and according to the policies and the Government Computer Centre, the deployed model in the e-Government cloud is the Community cloud. In the deployed model, the cloud infrastructure is shared by several ministries and institutes with a government policy.

The community deployment model increases the integrability in cloud applications, and the data integrity as various ministries requiring access to the government central database.

4.2. Current Palestinian e-Government Software Engineering Model

Refer to section 3.1, the transformation to the cloud needs a well-defined software development lifecycle, as the transformation framework has a main component for software development lifecycle. The current software engineering model adapted in the Palestinian government is a combination of the waterfall model and the iterative model according to the manager of web applications development, The main functionalities of this model is to document the user and system requirements, in addition to clarify the system design flow during development cycles, There is a system analyst who is responsible for preparing the system requirements specification document, and coordinating the development cycles between divisions, finally an SRS template is designed to be used in defining system requirements specifications.

Figure 4.3 shows the development cycle applied in the e-Government, the applied model is designed base on the e-government application development needs, but doesn't take into consideration the cloud transformation in addition to cons introduced in section 4.2.1.

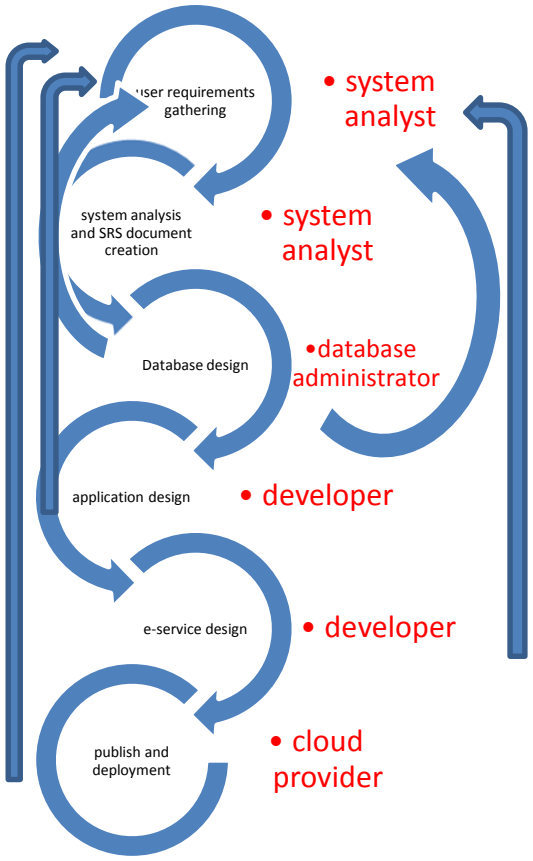

Figure 4.4 The Palestinian e-Government Development Process Model [The manager of web applications development division]

As shown in Figure 4.4 (the manager of web applications development division) the software development lifecycle consists of six phases, user requirements gathering, system analysis, database design, application design, e-services design, and deployment phase.

The tasks associated to each phase are separated depending on the divisions of the e-Government architecture:

1- Users requirements gathering:

   The system analysis division is responsible for meeting stakeholders and gathering the user requirements. The system analyst write down everything the user interested in, taking in consideration the user experience.

   This task take too much time in some cases because of the issues in coordinating meetings with customers and stakeholders.

2- System analysis and SRS document creation

   After the system analyst gather the user requirements, he will analyse the system and prepare the SRS document. The SRS document contains the user requirements and the system specifications presented using user cases and charts.

3- Database design

After the SRS document is created, the document will be delivered to the database division, and the database designer will design and create the database schema and tables, create required procedures and assign required privileges.

After database is created the system analyst will be inform to validate the design and to coordinate delivering the database to the web application division.

4- Application design

The web application division will receive the SRS document and the database to start developing the system. This step is critical as the web developer needs to return to the system analyst for each step, and the requirements changes needs to be discussed well.

5- E-service design

The e-service division in Palestinian e-Government is separated from the applications division because the e-service has special cases due to the concern of publishing e-services and the different stakeholders.

The e-services developer will receive the SRS document and start to develop the e-service and coordinate with the system analyst and the web application developer.

6- Publish and deployment

As shown in Figure 4.3 the deployment phase is the latest step in developing applications but actually it starts during the web application developing, when the main interface of the system are ready to be presented.

4.2.1. Palestinian e-Government Software Engineering Model Pros and Cons

As presented previously, the Palestinian e-Government software engineering model consists of six phases, these phases' delivers the milestones using the waterfall model approach, and depending on requirements changes the requirements updated during development using iterative model approach.

a. Model pros:

First, the staged development cycle enforces discipline: every phase has a defined start and end point. The emphasis on requirements and design before writing a single line of code ensures minimal wastage of time and effort and reduces the risk of schedule slippage, or of customer expectations not being met.

Getting the requirements and design out of the way first also improves quality; it's much easier to catch and correct possible flaws at the design stage than at the testing stage, after all the components have been integrated and tracking down specific errors is more complex. Finally, because the first two phases end in the production of a formal specification, the model can aid efficient knowledge transfer when team members are dispersed in different locations.

b. Model cons:

The most drawback of this model is that the customers don't really know what they want up-front; rather, what they want emerges out of repeated two-way interactions over the course of the project. In this situation, the model, with its emphasis on up-front requirements capture and design, is seen as somewhat unrealistic and unsuitable for the vagaries of the real world. Further, given the uncertain nature of customer needs, estimating time and costs with any degree of accuracy (as the model suggests) is often extremely difficult. In general, therefore, the model is recommended for use only in projects which are relatively stable and where customer needs can be clearly identified at an early stage and this is impossible especially in the e-Government projects.

Another drawback revolves around the model's implicit assumption that designs can be feasibly translated into real products; this sometimes runs into roadblocks when developers actually begin implementation. Often, designs that look feasible on paper turn out to be expensive or difficult in practice, requiring a re-design and hence destroying the clear distinctions between phases of the model which will reduce the agility in the development.

Also this model suffer from the lack of agility, when the user requirements changed or additional features added to the application, it will be difficult to redesign the required components and the changes will take too much time.

In addition to the previous issues, this model is not suitable for transforming applications to the cloud because none of its phases take in consideration the cloud environment and it’s requirements, so there are various software engineering issues that have to be addressed when the e-government cloud becomes the target deployment environment, These issues include the integrability between cloud components, and reused components of cloud applications, designing the database and relation to the SOA framework, designing system components, components hosted in cloud environment, customizing general purposes applications , deployment application in cloud environment, configuration management , schedule estimation, and cloud based applications maintenance [2].

Software development cycle of the cloud applications transformation may be affected due to the hosting on the e-government cloud environment, since cloud development is different and poses its

own challenges to the development team, so we need to design a software development lifecycle component in the e-Government cloud framework.

4.3. Conclusion

Software development lifecycle has great challenges in adopting applications in cloud based environment, software development process will involve heterogeneous platforms, distributed web services, multiple enterprises geographically dispersed, while existing software process models and framework activities are not going to be adequate unless considering the cloud architecture and the interaction between its components to affect software development process.

The acceleration in the pace of software development in the Palestinian e-Government shows a critical need for a software development model to make the development easy and faster, and to be used for transforming applications to the cloud, since the current software development model suffer from the lack of agility and late in application development.

In addition to the previous issues, the integrability process between the applications become more complex due to the lack of integrability between these applications, the lack of integrative environment.

Next we introduce the proposed transformation framework, and discusses how the cloud transformation framework would overcome these issues and limitations.

Chapter 5 The Palestinian e-Government Cloud Applications Transformation Framework

The Palestinian transformation framework will take into consideration the agility in development and the integrability between applications.

To achieve these goals we will apply the following properties in the proposed framework and check their suitability to achieve our goals:

- Classifications of the framework components depending on their jobs.
- Service architecture including taxonomy, classification and policies.
- Process lifecycle.
- Project profile, polices, knowledge base, and monitoring.
- Customer involvement in all framework component implementation.

Transforming applications to the cloud is not a deployment action, it is a full lifecycle solution from analysis through implementation to retirement.

As introduced in Chapter 4, the Palestinian e-Government applications developments has its special concerns which include:

- The lack of software engineering model for applications transformation to cloud.
- The geographically distributed government institutes, necessitate transforming applications to cloud and to introduce them as Saas.
- The integrability between government application and services also necessitate transforming applications to cloud and transforming application's functionality into services.

The proposed framework as shown in Figure 5.1 is divided into components, each component consist of multiple models, and every model has its tasks and functionalities.

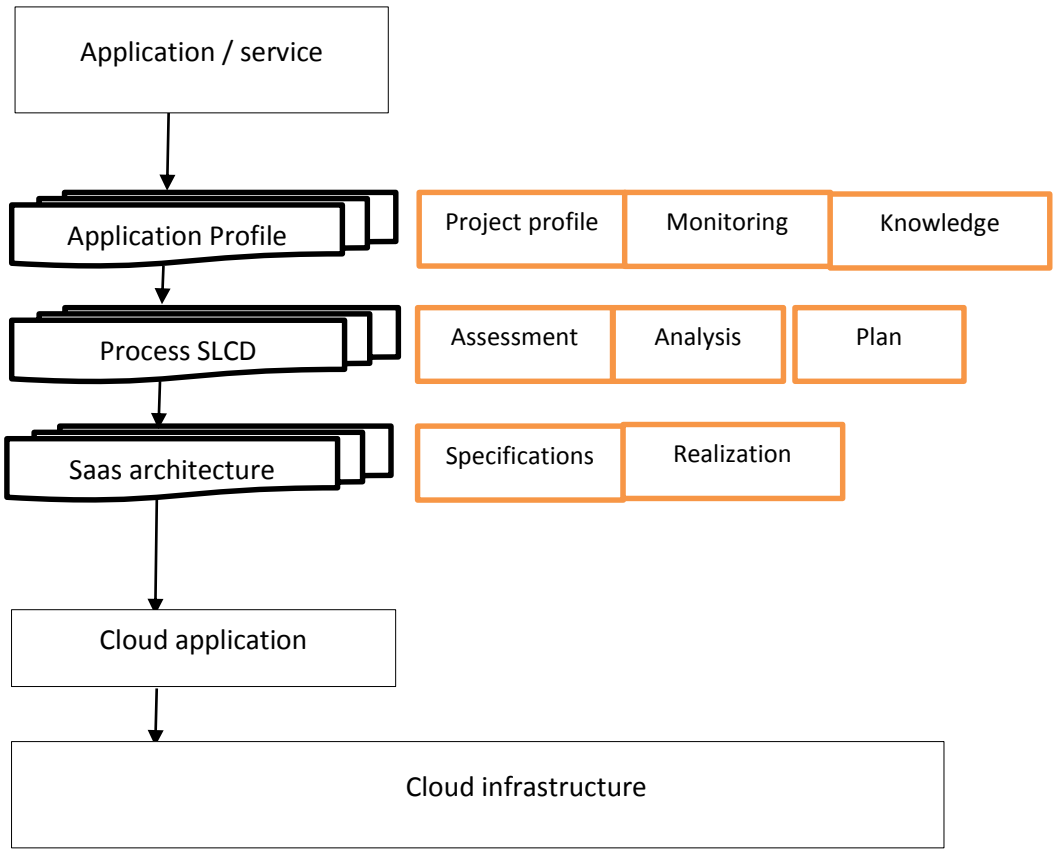

Figure 5.1 The Proposed Palestinian e-Government Cloud Transformation Framework

5.1. Application Profile

The application profile component describes the project profile and characteristics, the monitoring functionalities, and the knowledge base.

5.1.1. Project Profile

In project profile component, the analyst simplified a description of the application, in addition to define the purpose and ownership of the project, a first estimation of activities and functions of the project.

Certain criteria, such as service-level agreements (SLAs), data portability, and long-term costs, must be carefully evaluated when considering a SaaS deployment.

SLAs: The SaaS government provider should provide a SLA for application overall availability, scalability, and performance, as well as provide clear polices and guidelines for application maintenance and use. The Palestinian e-Government publishes the policies concerns the agreements

of using the e-Government application, service consumers has to sign the terms of use agreement, agree to use the service, and agree to provide the service provider with any information required. SLA is applied in the e-Government but it is not detailed here because it need additional related issues such as legislative, council of ministers, and ministry of justice.

Data portability: The SaaS government provider should provide a way to allow other government institutions to own and control their application data. SaaS customers should have the ability to export all application data that belongs to them, in a format that can be easily parsed and migrated to other internal or external applications.

Security: The SaaS application can contain sensitive corporate data when stored on the SaaS provider's infrastructure. You should require transparency in the service provider's security policies to be able to determine whether adequate security is provided, based on the nature of application data.

5.1.2. Monitoring

This component will be responsible for all the activities related to the monitoring of the application process and services function.

Monitoring is used to measure and collect real usage data of an application before it is transformed. This data can help size the application deployment in a cloud. Ideally, application data should be collected for at least 10 to 15 days to allow capture of variances in daily and weekly usage patterns.

The following data should be collected:

- CPU usage
- Memory usage
- Storage data such as throughput, latency, and input/output operations per second (IOPS)
- Network data such as throughput, connections per second, and dropped connections.
- Interaction with other applications and services to study and analyse the integrability of the application.

In addition to these statistics, it is also important to profile user activity, such as the total number of connected users, request and transaction rates, and request latencies. The usage data can also be used to build automated tests for the application to make sure of the same or an improved level of service after the application is transformed.

5.1.3. Knowledge Base

In knowledge base component complex structured and unstructured information used for the government decision makers and managers, depending on this component the application will be integrated as a model in the e-Government decision making systems.

Currently the decision making systems has not been built despite of the government data is hosted in the datacenter, so it is important to design the knowledge base component in the transformation framework in order to take the data analysis into consideration when transforming legacy applications to the cloud.

The knowledge base component will provide the decision making system with statistics, reports, data interaction, and graphs e.g info graphs.

5.2. Process software Lifecycle Development

In this component a full lifecycle for developing the Saas application from analysis to design, as shown in Figure 5.2 the process software lifecycle component is composed of assessment, analysis, and plan components.

5.2.1. Assessment

We can't just take any application to the cloud, we should move only those that will yield value from running on a cloud infrastructure.

During this assessment, we will evaluate several business and technical characteristics of the application.

During our applications transformation approach, we will assess applications and determine if they're suitable for running in the cloud, In addition to that the assessment component will determined the components and the services that need to be moved to the cloud.

Highly suitable applications are those that can be replaced by existing software as a service offerings. Or they're applications whose business and technical characteristics align with the target cloud platform's attributes.

The following assessment criteria's will be used for assessing application's transformation, and we will evaluate the proposed framework against them in chapter 7.

- The application needs to integrate with other applications.
- The reused of existing components.
- The application uses a centralized authentication service.

- The application uses supporting services such as the reporting service, and the knowledge base services.
- The application uses the Central e-Government database and interacts with SOA framework.
- The application's architecture changed frequently according to the customer needs.
- The application's development needs involvement of the customer.

And for those applications not suitable for the cloud, we propose three options:

- Retire application.
- Leave it as is.
- Migrate to a traditional infrastructure.

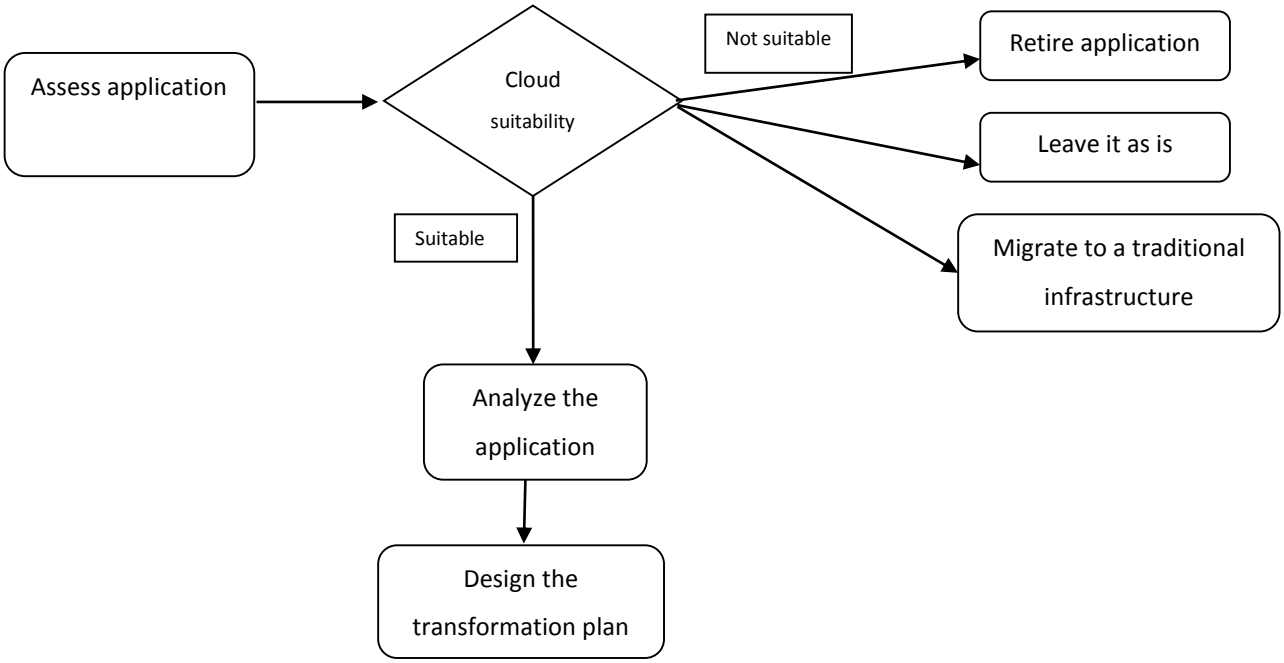

Figure 5.2 The Cloud Application Transformation Lifecycle Component

5.2.2. Analysis

Analyzing the application, defining its user and system requirements, and cloud transformation requirements whereas the legacy application has special consideration in transforming to the cloud, the application components need to be defined, and the interactions between these components need to be clarified.

During the analysis process, the service analysis is presented in the early stages before moving to service specifications component (Section 5.3.1), service analysis process begins with information gathering that are results in creation of conceptual service structure.

The service analysis process is commonly carried out iteratively for each business process to achieve the agility in development and transformation, and to guarantee the integrability between application services and other applications services.

In addition, the service analysis component determines the scope of service use and the service interfaces for use.

Figure 5.3 introduces the service analysis activities, a complete top-down processes are carried out, compromised of numerous iterations through service analysis.

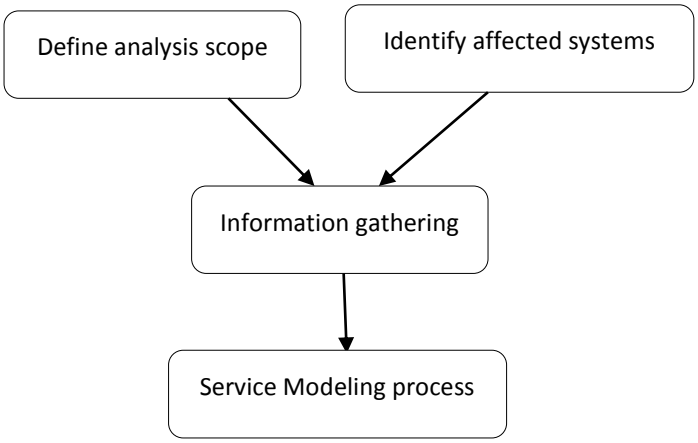

Figure 5.3 Service Analysis Activities

Figure 5.3 shows how service analysis actually represents a parent process consisting of two information gathering steps and then a third step represented by the service modeling sub-process.

- ***Define Analysis Scope***

  During this step business analysts are asked to clearly establish the boundary of the analysis. Most commonly, there is a ratio of one analysis process to one business process definition. However, business processes can be complex or multi-layered (containing nested processes) and may or may not already be representing portions of business logic. Therefore, this step may also require identifying portions of a given business process for which service modeling is not required.

- ***Identify Affected Systems***

It is helpful to have an understanding of what existing parts of the enterprise will be affected by the scope of the planned business process analysis. Especially relevant are legacy systems that may later raise service encapsulation and autonomy challenges. These types of constraints can directly impact the partitioning of logic into services and the ultimate granularity at which service candidates are defined.

- ***Service Modeling Process***

  Service modeling is the process of conceptualizing services and capabilities prior to their actual physical definition and development. Because nothing concrete is defined during this stage, we introduce the service modeling and specifications in Section 5.3.1.

5.2.3. Plan

In this component, the analyst will design the transformation plan, in this plan all the jobs of transformation will be defined and the roles and responsibilities of transformation team including the stockholders to achieve the agility.

To identify applications for transformation to a cloud, it is necessary to first identify and understand the business and technical factors for the transformation. integrability and business agility are typical business factors for application transformation to clouds. Cloud computing can provide significant integrability because of the service classification and integrability.

After an application has been identified as a candidate for cloud transformation, based on business and technical factors, it is necessary to consider for what type of cloud environment SaaS, PaaS, or IaaS the application is best suited.

a. Classified as Software as a Service

Based on the type of application, and if SaaS-based alternatives exist, it is worth considering if the SaaS alternatives can meet both business and technical needs. Such a change is no longer an application transformation but more of a replacement of the existing application with a SaaS option. There might still be a need to migrate existing data to the new application.

SaaS removes the need to manage both the application and the infrastructure on which the application is deployed. This approach can be attractive, but certain criteria, such as service-level

agreements (SLAs), data portability, and long-term costs, must be carefully evaluated when considering a SaaS deployment.

b. Classified as Platform as a Service

Platform as a service might be an option for transformation business applications that are based on standard application server software such as apache, mysql and oracle.

In this model, the service provider manages the application platform software and might provide access to common application services such as SQL databases. The application platform might be shared by multiple applications belonging to different customers. How the application platform is mapped to the physical infrastructure is typically controlled by the cloud service provider.

The decision factors in such a transformation will depend on the type and version of the application server used. Some PaaS environments might not support all features of the application server and might require application changes.

c. Classified as Infrastructure as a Service

Ttransformation of an application to an IaaS involves deploying the application on the cloud service provider's servers. The initial step in making a decision to transform to an IaaS model is to determine whether the cloud-based server hardware and operating system (OS) are compatible with the current server's hardware and OS.

For example, if an application is running on an x86 server, the cloud servers must be able to implement x86 instructions. If the hardware is not compatible, the application might need to be recompiled or redeployed for the new platform. Similarly, if the OS is compatible, few changes will be required when the application is migrated.

5.3. SaaS Architecture

One of the cloud framework components, which capture the relevant aspects of SOA specifications, architecture, and implementation; and the technology component which is responsible for the techniques used for the implementation, the security models, and service standards.

5.3.1. Specifications Component

As mentioned in Section 5.2.2 in order to transform an application to the cloud, the application should be introduced as a Saas, and the services specifications need to be designed.

A service specification is a description of what you want from a service. It is a working tool for the Provider to use to structure how they will deliver the service, and it is a document for you to refer to measure the quality of the service and to take into consideration the goals of the framework such as the agility and integrability.

During service specifications the developer can identify already defined services, reused components, and define the interactions with other application services, which enhance the integrability between systems. The other important achieved goal is that the well-defined service specification considered as a contract with the customer to inform him with what will be developed and implemented, these contracts include the provided and required interfaces, and the roles those interfaces play in the service specification, and the rules or protocol for how those roles interact, this will increase the development agility.

In service specifications component we will define the SOA solution by modelling the specification of each service in detail. A service specification must specify everything that a potential consumer of the service needs to know to decide if they are interested in using the service, as well as exactly how to use it. It must also specify everything a service provider must know to successfully implement the service. Thus, a service specification is a mediator or a contract between what consumers need and what providers provide.

Service specifications include at least the following information:

- The name of the service, suggesting its purpose.
- The provided and required interfaces, thereby defining the functional capabilities that are provided by the service and those that it requires of its consumers.
- Any protocol that specifies rules for how the functional capabilities are used or in what order.
- Constraints that reflect what successful use of the service is intended to accomplish and how it will be evaluated.
- Qualities that service consumers should expect and that providers are expected to provide, such as availability, performance, footprint, suitability to the task, competitive information, and so forth.
- Policies for using the service, such as security and transaction scopes for maintaining security and integrability or for recovering from the inability to successfully perform the service or any required service.

5.3.2. Realization Component

The realization component is responsible for realizing the services.

The realization will depend on the SOA-Based Integrated Central Database that introduced by [1], as discussed in Section 2.2, The main components of the framework are: ESB, Web Services, databases, e-Government portals, governmental business applications, and front-end applications.

5.4. Cloud Infrastructure

The cloud infrastructure as introduced in Chapter 4 consists of a main data centre that is composed of two clustered physical servers with a shared storage.

Currently the Palestinian government has a datacentre for hosting the e-Government cloud, in this datacentre the servers and resources are virtualized in order to present a full cloud platform.

The cloud applications interact with these virtual servers as it is single logic resource, this virtualization grantee the separation of services and easy the interactions in cloud environment.

The cloud infrastructure module is explained in details in Section 4.1.

5.5. Conclusion

The transformation framework for the Palestinian e-Government applications is a suitable architectural style for transforming applications to the cloud environment as it take into considerations all the Palestinian e-Government environment needs.

The framework components interaction flow guarantees the objectives of this research which are the agility and integrability, the integrability between cloud applications will be achieved throw the service specifications in the SaaS architecture component, and the agility will be achieved by the separation between the components and the ordered steps in transformation process lifecycle component.

The proposed framework composed of three main components: the application profile which describes the application profile and characteristics, and the software lifecycle component in which the assessment and analysis components are applied, and the Saas architecture component for realizing the Saas.

Chapter 6 Framework Scenario Applying

In this chapter we present applying a scenario of the framework, we introduce the scenarios as a use case scenarios.

In this applying we provide a proof of concept to provide a specific usage scenario for the framework, and through such scenario the framework is validated to perform its requirements.

We use two scenarios for applying the framework, the first scenario introduces a full transformation process for the Administrative Correspondence System (ACS) which is a legacy application. The second scenario will introduce an assessment of the National Frequency System (NFS) that fails to transform to the cloud.

The framework components that are applied include: application profile where the project profile and characteristics and functionalities are described, the process software lifecycle component in which we discuss the assessment process and the analysis process, and the SaaS architecture to describe the service specifications and implementations.

6.1. Scenario 1: Administrative Correspondence System (ACS)

ACS is a legacy application responsible for all the administrative management, task flows, and monitoring, currently the ACS system is running in many ministries, and benefits from the government central database and interact with some other services such as single sign on, and the access control list system. According to the manager of web applications development division, there is a serious need to overcome the integrability among ACS services, and the rapid changes in user requirements. There for we suggested to apply the cloud transformation framework and assess if the transformation can solve these issues.

6.1.1. Application Profile

Referring to Section 5.2, we will introduce a full description of the project in this section, The Administrative Correspondence System proposed by the Ministry of Information Technology to follow-up paper operations that are traded in the Ministry of Communications and Information Technology and computerization to facilitate access them from all parties to existing users in the system and this leads to increased work efficiency and facilitate access to the references.

a. Project Profile

The ACS system is a web application developed using php codeignitor framework depending on the MVC design pattern and based on oracle database and uses government central database to use the employee data.

All web forms are based on a standard design and user experience to less the errors and help users to interact easily with the system, using the html5 and bootstrap techniques to make the design responsive for all devices. The Main system functionalities include:

- Users and groups privileges. The administrator can add, edit, and change users and groups privileges. This module uses the e-Gov ACL system which is a central access control list system, designed to serve central applications to ease the ACL management, and to central permissions granting process.
- Contacts. Used to add, edit contact data for employee, this module depends on the govdata to retrieve contact data.
- Messages. Concerns sending administrative messages, receiving messages, and forwarding messages.
- Attachments. Concerns with attachments manager for all messages types, this module works as an archive repository for formal messages.
- Tasks. Focusses on managing the employee tasks that assigned depending on a particular message. The employee and the manager can monitor tasks progress, and export monthly reports.
- Reports. Responsible for reporting in the ACS system, the user can export various types of reports in various formats.

Certain criteria, such as service-level agreements (SLAs), data portability, and long-term costs, must be carefully evaluated when considering a SaaS deployment.

In this scenario applying we will not list these parts of Application profile component.

6.1.2. Process Software Lifecycle

a. Assessment

During this assessment, we evaluate several business and technical characteristics of the application. According to the assessment component introduced in Section 5.2.1, we assess the

application and determine if it is suitable for running in the cloud, In addition the assessment component determines the components and the services that need to be moved to the cloud.

The ACS application interacts with several web service and data providers, the employee profile service reads the data of employees that stored in the government data centre, the contact information reads the single sign on contact data, etc.

The ACS application uses several central services such as the e-Government Access Control List system (ACL), the Single sign on, and the reporting system. The users and groups module will depend on the e- Government ACL system depending on its services, and will uses the single sign on system for authentication process. The reporting module depends on the e-reporting service, which build using the jasper reporting server [30].

As mentioned previously, the ACS application highly suitable for transforming to the cloud as it has several modules can be introduced as a service.

The other main reason for transforming the application to the cloud, that the application itself will serve several ministries, institutes, and private sector. This will lead to introduce its services as software as a service.

In next section we analyse the system services and modules, and break down its functionalities.

b. Analysis

1- Application Analysis

In analysis module, we analyse the ACS application, define the context diagram, and the use case diagram.

Figure 6.1 shows the context diagram of the application components, the context components are divided into 3 types, the system services which are responsible for the system functions like messages, attachments, and tasks; the support services like single sign on, access control list service, and the reporting service; and finally the government central data.

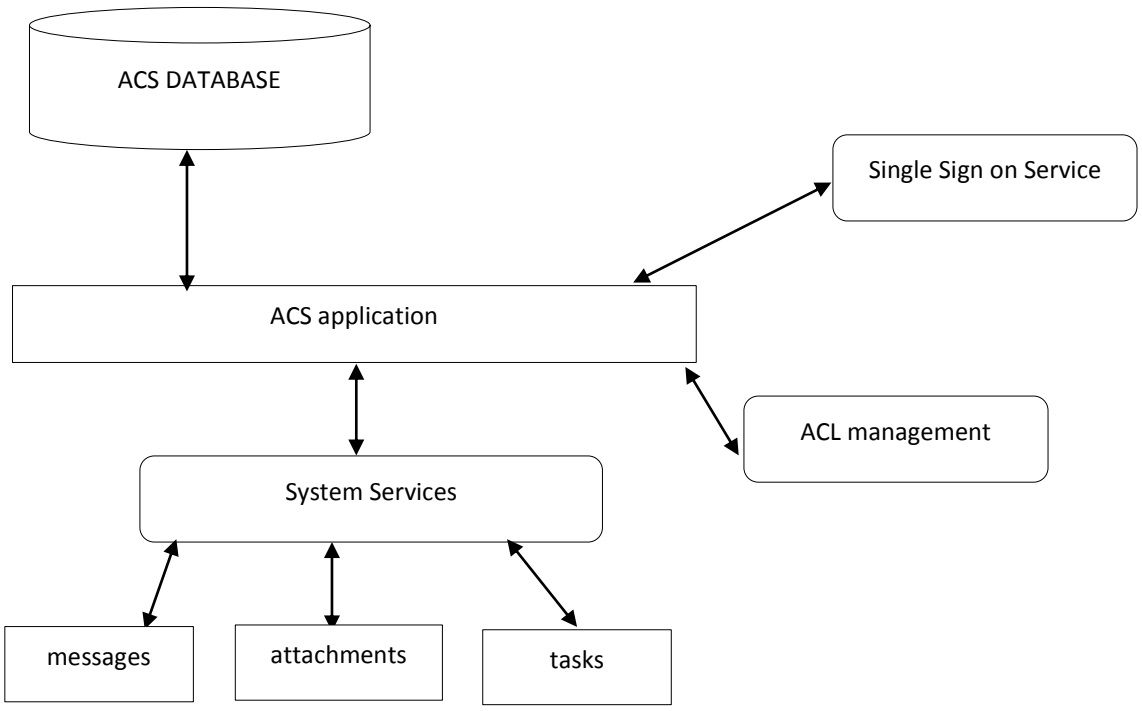

Figure 6.1 The ACS Context Diagram [as introduced in system requirments documents]

Figure 6.2 shows the use case diagram of the ACS application, there are 3 main types of users, the administrator who will be responsible for creating a new instance project for the ACS application, load the employee hierarchy or read it from govdata, and define the groups and privileges of users and groups. The second user type is the message sender user or the guest who will send a message or a task to the ministry, and the message will be forwarded to a particular division. The third user type is the employee, who will receive and send messages depending on his position in the institute, define contacts groups, define tasks, create plans, and export reports.

2- Service Analysis

During analysis process, we present the service analysis (see Section 5.2.2), first in the analysis scope we present the scope of the analysis which is the first layer of the application services, in the scenario applying we don't present the sub services or the multi-layered services.

The affected systems by the transformation of the ACS application includes: the standard applications database as the application database will be moved to the standard applications database, the decision making system, e-Government central database, and the central access control list system ACL.

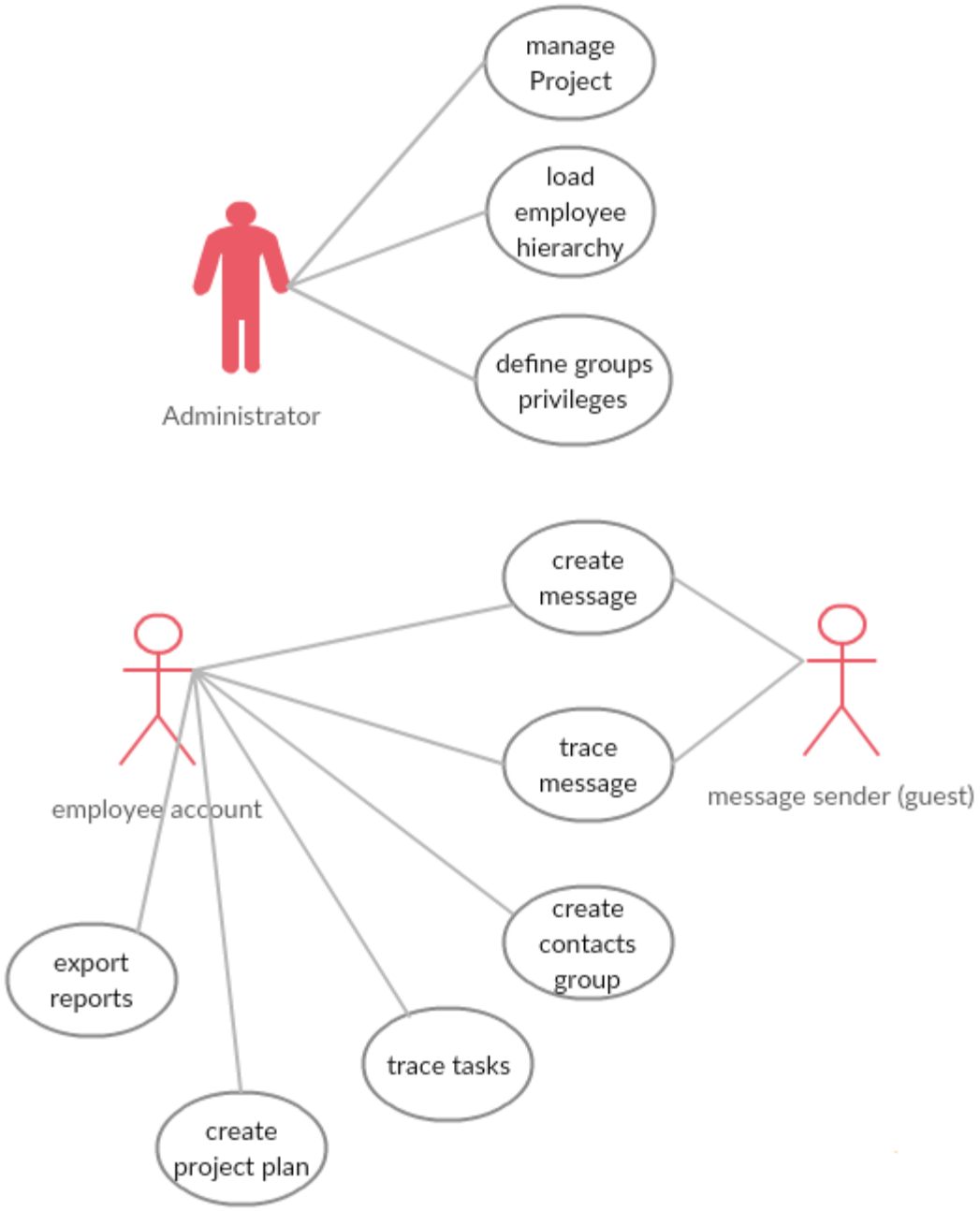

Figure 6.2 The Use Case Diagram of the ACS Application

System services presented in Figure 6.3 include the following services:

- Messages Services
  - Send Message.
  - Forward Message.
  - Show Message.
  - List Messages.

- Attachments Services
  - Add Attachment.
  - Read Attachment.
  - Archive Attachment.
  - Search Attachment.
  - List Attachment.
- Tasks Services
  - Add Task.
  - Forward Task.
  - List Tasks.
  - Monitor Task Flow.

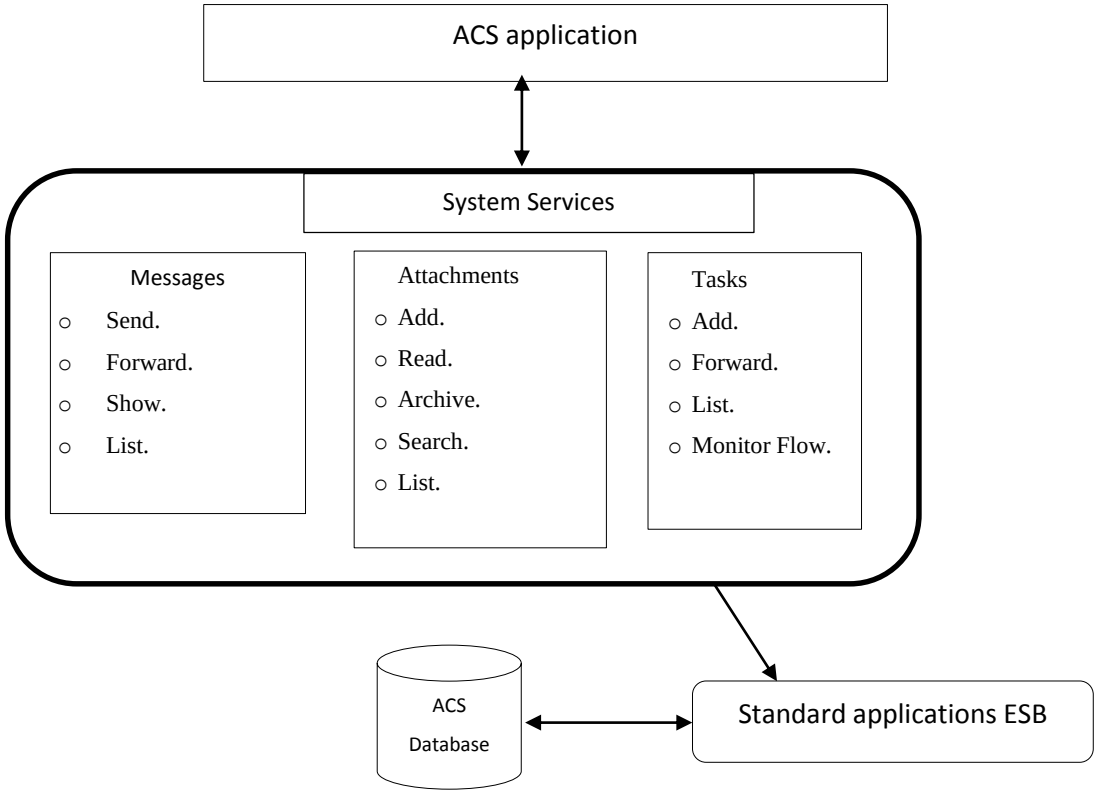

Figure 6.3 ACS Service Analysis

c. Plan

In this component, we design the transformation plan, in this plan all the jobs of transformation are defined.

Whereas the transformed application will replace the existing ACS legacy application, and the application functionalities introduced as services, the application benefits from cloud supporting services such as SSO, ACL, reporting system, and the application data migrated to the central database and standard applications database. The ACS transformation classified as a Saas application.

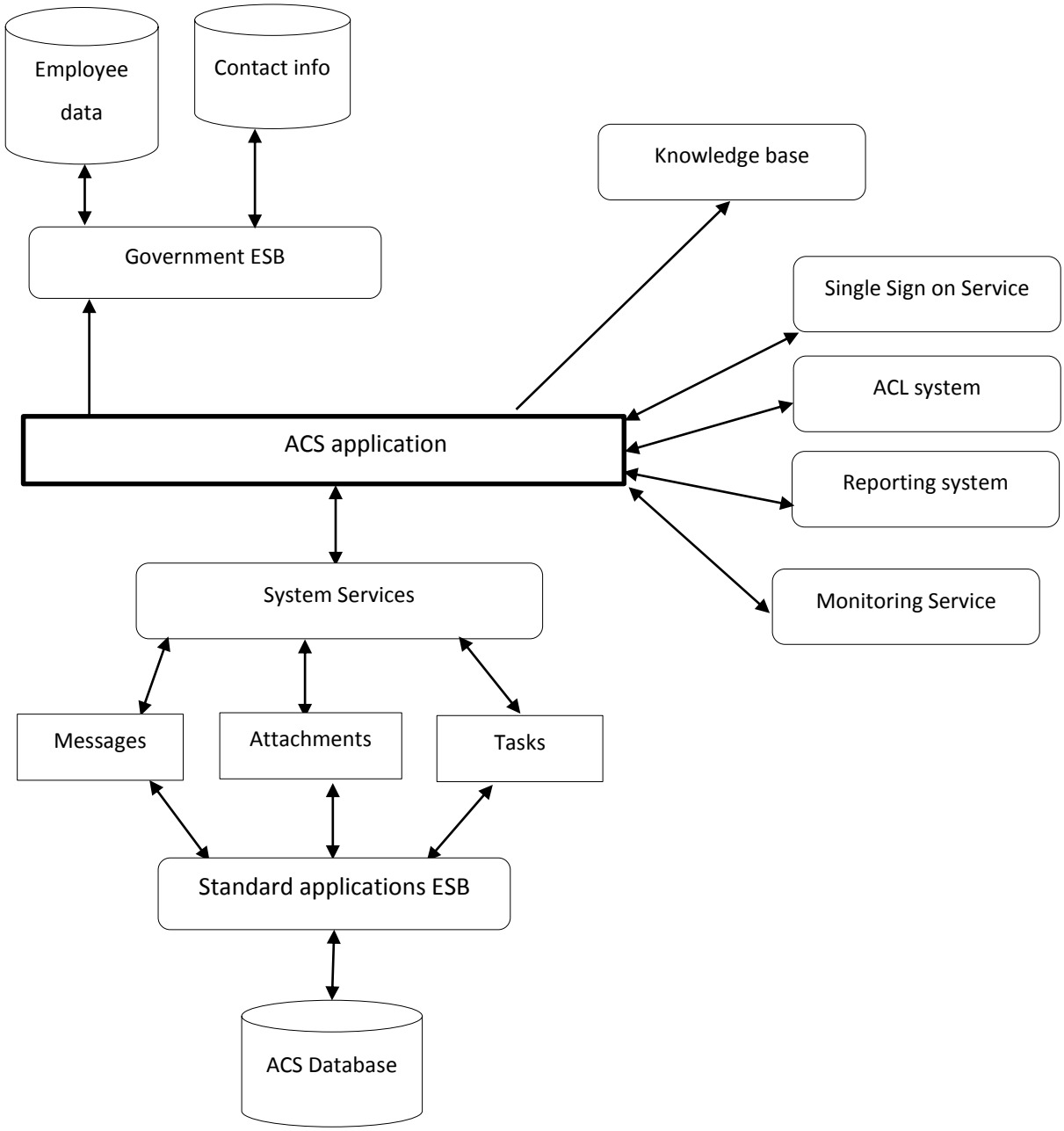

Figure 6.4 ACS Cloud Application Context Diagram

To identify applications for transformation to a cloud, it is necessary to first identify and understand the business and technical factors for the transformation as mentioned previously in section 5.2.3.

The ACS system has three types of services, the system services which are responsible for the system functions like messages, attachments, and tasks; the support services like single sign on, access control list service, the knowledge base service, the monitoring service, and the reporting service; and finally the government central database.

As the system will be introduced as a software as a service, and it will serve many institutes, we need to separate the services as shown in Figure 6.4.

The database will be defined as an application database which will use the standard application enterprise service bus, and the employee data and contact data which will use the govdata enterprise service bus.

This classification in data sources will increase the integrability of the system as it will help other applications to deal with ACS data easily using the government SOA architecture [1].

Then the supporting services will be separated from the ACS application, the ACS application will use their API to benefit from their functions.

This classification in databases and services will achieve the integrability and business agility for application transformation to cloud. Cloud computing can provide significant integrability because of the service classification and integrability as mentioned previously.

6.1.3. SaaS Architecture

This component capture the relevant aspects of SOA specifications, architecture, and implementation; and the technology component which will be responsible for the techniques used for the implementation, the security models, and service standards.

a. Service Specifications

As mentioned previously the services will be classified into three types, the cloud supporting services, the data services which will be designed and deployed using the government central database enterprise service bus, and the system services.

- **Cloud Supporting Services Specifications**
  - Single sign on service

    Single Sign-On (SSO) service allow a single authentication process for deferent service providers, currently the Palestinian e-Government adapted the security markup language for single sign on service.

Security Assertion Mark-up Language (SAML) provides a secure, XML based solution for exchanging user security information between an identity provider (General Administration of e-Government) and a service provider (governmental institutes) [31].

The SAML standard defines rules and syntax for the data exchange, yet is flexible and can allow for custom data to be transmitted to the external service provider.

The SSO service provide an API used for authentication based on SSL/TLS secure channel for exchanging security data between service provider and identity provider that achieves the integrability between the SSO service and the cloud application. The Service Level Agreement of the SSO is provided with the service API.

- Access control list service

  ACL service allow a central management form for granting permissions for all e-Government applications. ACL provides a web service to check user rights. The ACL provides an API used for integrating with the ACL service based on government SOA.

- Reporting service

  Reporting Services provides a full range of ready-to-use tools and services to create, deploy, and manage reports for applications. Reporting Services includes programming features that enable you to extend and customize your reporting functionality.

  Reporting Services is a server-based reporting platform that provides comprehensive reporting functionality for a variety of data sources. Reporting Services includes a complete set of tools to create, manage, and deliver reports, and APIs that enable developers to integrate or extend data and report processing in custom applications. The reporting service uses the JasperReports Server [32] which is a stand-alone and embeddable reporting server. It provides reporting and analytics that can be integrated to the cloud applications. JasperReports Server is optimized to share, secure, and centrally manage your Jaspersoft reports and analytic views.

- **System Service Specifications**

  System service specifications are presented in Table 6.1

Table 6.1 System Service Specifications

Service name	**input parameters**	**Protocols**	**Constraints and policies**
Send Message	Message number, Message date,	http, JSON	SSO, SOA ESB, and Service level agreement

	Message sender id, Message content, Sender signature, Receivers ids, Notes, End date		
Forward Message	Message number, Message date, Message sender id, Message content, Sender signature, Receivers ids, Message Type Notes, End date	http, JSON	SSO, SOA ESB, and Service level agreement
Show Message	Message number, Message date, Message sender id, Message content, Sender signature, Receivers ids, Message Type Notes, End date	http, JSON	SSO, SOA ESB, and Service level agreement
List Messages	Receiver Id, Message Type	http, JSON	SSO, SOA ESB, and Service level agreement
Add Attachment	Message Id, Attachment Type, Attachment Title	http, JSON	SSO, SOA ESB, and Service level agreement
Read Attachment	Attachment Id	http, JSON	SSO, SOA ESB, and Service level agreement
Archive Attachment	Attachment Id	http, JSON	SSO, SOA ESB, and Service level agreement
Service name	**input parameters**	**Protocols**	**Constraints and policies**
Search Attachment	Attachment Title, Message Title	http, JSON	SSO, SOA ESB, and Service level agreement

List Attachments	Message Id	http, JSON	SSO, SOA ESB, and Service level agreement
Add Task	Message Id, Task Title, Employee Id, Manager notes, End Date	http, JSON	SSO, SOA ESB, and Service level agreement
Forward Task	Task Id, Employee Id	http, JSON	SSO, SOA ESB, and Service level agreement
List Tasks	Employee Id, Date	http, JSON	SSO, SOA ESB, and Service level agreement
Monitor Task Flow	Message Id, Task Id, Employee Id	http, JSON	SSO, SOA ESB, and Service level agreement

6.2. Scenario 2: National Frequency System (NFS)

NFS is a legacy application responsible for management the national frequency in Palestine, the NFS application serves different types of stakeholders and according to the manager of web applications development, the applications faces many issues in development due to the changes in user needs. Details about NFS is discussed in next section, and the assessment component of the application transformation suitability.

6.2.1. Application Profile

Referring to Section 5.2, we will introduce a full description of the project in this section, The National Frequency System proposed by the Ministry of Information Technology to manage the licenced frequency of the working companies in Palestine.

a. Project Profile

The NFS system is a web application developed using php codeignitor framework depending on the MVC design pattern and based on oracle database.

All web forms are based on a standard design and user experience to less the errors and help users to interact easily with the system, using the html5 and bootstrap techniques to make the design responsive for all devices.

- **Main system functionalities:**
  - Frequencies management. The system user can add, edit, and manage the frequencies range for every registered company.
  - Frequencies Services. Used to add, edit services provided by telecommunication companies.
  - Banned Types. To specify the banned frequencies numbers and types.
  - Companies. To register the licenced campanies.

Certain criteria, such as service-level agreements (SLAs), data portability, and long-term costs, must be carefully evaluated when considering a SaaS deployment. In this scenario we will not list these parts of Application profile component.

6.2.2. Process Software Lifecycle

a. Assessment

During this assessment, we will evaluate several business and technical characteristics of the application. According to the assessment component introduced in Section 5.2.1, we will assess application and determine if it is suitable for running in the cloud.

As mentioned previously the NFS is composed of 4 types of services, that considered as standalone services, these services will not interact with other governmental applications, and the application data stored in a database separated from the government central database, as this data is used for licencing purposes in the Ministry of Telecom and Information technology.

The NFS is a standalone application that doesn't need to interact with other government services, and it has a low number of audience and stockholders.

As mentioned previously, the NFS application is not suitable for transforming to the cloud as it has no services can be introduced as a service.

The other main reason for failing in transformation assessment is that the application itself will not serve other ministries, institutes, nor private sector. This will lead to leave the application as it is according to the assessment result introduced in section 5.2.1.

6.3. Discussions and Conclusions

The scenario applying as discussed in sections 6.2 and 6.3 provide evidence on how far it achieves the goals of the framework. It is clearly seen that the applying in scenario 1 performs the main functionalities of the proposed framework which are:

The application profile, the process software lifecycle, and the SaaS architecture, the scenario 2 performs the application profile, and the process software lifecycle, and it fails in the assessment process.

In scenario 1 applying the framework achieves the goals of the proposed framework which are: the agility in transforming applications to the cloud. It was seen that main components in the framework are included in the scenario, and the components of the realization are mapped to their counter parts in the framework to achieve this goal. The other goal achieved during realizing the framework is the interaction between the application components, and the ability to interact with other cloud applications as clarified in the discussion in components implementation. The framework realization acts as a proof of concept for the validation of the framework, since it showed that the framework can be realized and implemented and hence the concept of the framework is validated.

Evaluating the framework is discussed in next Chapter, the evaluation will be against the targeted research goals which are integrability and agility.

Chapter 7 Framework Evaluation

In this chapter we present the evaluation of the framework and its scenario applying. The evaluation will be conducted against the targeted quality attributes which are agility, and integrability. The chapter discusses the quality attributes and presents the evaluation of the framework based on ATAM (see section 2.3) and validation of the framework concept using a usage scenario.

7.1. Framework Quality Attributes

The Software Engineering Institute (SEI) believe that the suitability of architecture is determined by the quality attribute requirements that are important to the stakeholders of a system. And hence a given software architecture is suitable for its intended purpose in case of fulfilling the quality attributes. Measurement plays a critical role in effective and efficient software development, as well as provides the scientific basis for software engineering evaluation that makes it a true engineering discipline [33]. Quality attribute scenarios are usually used to specify quality attribute requirements. Also many quality concerns are primarily handled or strongly affected by the runtime environment. In the thesis work, the framework quality attributes to be evaluated need to be clearly defined, as follows:

7.1.1. Integrability

Software system integrability refers to the ability of combining individually tested software components into an integrated whole. Software is integrated when components are combined into subsystems or when subsystems are combined into application [34] .

Integrability is bound up in the concept of component interface, In our case, the integrability concept refers to the ability of other application services reused and supports cloud application services.

In cloud applications transformation frameworks, cloud applications usually interact with other applications, and introduce their services as Saas.

7.1.2. Agility

Transformation to cloud exhibit more structured in application services to introduce applications as a software as a service, the key point in transformation process is the agility in implementing the cloud framework components. It is also important in service implementation to be agile.

As introduce in Section 2.3, although agile development methodologies are successful in dealing with changes, but they don't act well against complexities which are the nature of Saas projects because of the lack of the pre-defined design of system. For developing each system, a structure or architecture is needed for better communication between stakeholders and when the system is larger and more complex, the architecture is required more.

To realize the agility concept and achieves the research goals, a well-defined framework architecture is designed, the components are defined, and during application transformation the application architecture should be discussed with stockholders.

Next we evaluate the applied scenario framework against these quality attributes using ATAM method.

7.2. Framework Evaluation using ATAM Method

What we aim to do in the ATAM, in addition to raising architectural awareness and improving the level of architectural documentation, is to record any risks, sensitivity points, and tradeoff points that we find when analysing the architecture (see section 2.6).

Risks, sensitivity points, and tradeoff points are areas of potential future concern with the architecture. These areas can be made the focus of future effort in terms of prototyping, design, and analysis.

A prerequisite of an evaluation is to have a statement of quality attribute requirements, in this research are: agility and integrability. The requirements to conduct the evaluation are evaluation team and stakeholder staff. The Evaluation team typically check the architectural approaches used to address the important quality attribute requirements specified in the scenarios. The goal is to assess whether these quality attribute requirements can be met. In our case the evaluation team is the team manager responsible for transformation process and all his team members.

7.2.1. Integrability Evaluation Scenarios

The main features that provide the integrability of the proposed framework are presented in table 7.1.

Table 7.1 : Integrability supporting Features

#	**Integrability supporting Features**	**Satisfied(yes,no)**
1.	The framework supports integrability with other application's services	yes
2.	The framework allow integrates and reuse components.	yes
3.	The system services can be integrate to other applications	yes
4.	The framework allows having service users and providers to use different implementation languages and platforms.	yes
5.	The authentication mechanism will be centralized and realized using Web Service.	yes
6.	The framework introduces supporting services such as the reporting services, the knowledge base service.	yes
7.	All application's web services implemented using the SOA based framework and the government ESB to maintain the integrability feature.	yes
8.	Service Specifications component provides interfaces, protocols, and message formats	yes

The proposed framework satisfy these features as follow:

- The framework architecture supports application integrability with other application's service, where the framework architecture is classified into three components, and the service component is separated in a standalone component.
- As introduced in Section 6.1.3, the framework allows integrates with existing components and services, the ACS application reused the Single Sign On service, and the framework introduced supporting services.
- The system services can be integrated to other applications (e.g. the knowledge base service can be integrated to the application services).
- The framework implement its services as web services, the SaaS architecture is a component of the framework, this leads to the ability to use different implementation languages.
- The framework services implemented in the SaaS component using the SOA based framework and the government ESB.

7.2.2. Agility Evaluation Scenarios

The main features that provide the agility in transformation process framework are presented in table 7.2.

Table 7.2: Agility supporting Features

#	Agility supporting Features	Satisfied(yes,no)
1.	The framework supports customer changing in user requirements.	yes
2.	The framework supports customer engagement in transformation process	yes
3.	The framework architecture is clear to the customer	yes
4.	The framework components' interfaces are clarified and discussed with stockholders.	yes
5.	The framework contains a transformation plan including an assessment phase.	yes
6.	The framework contains service classification component.	yes
7.	The transformation can be done partialy.	yes

The proposed framework satisfy these features as follow:

- The framework architecture responds to the application architecture changes easily.
- The framework supports customer engagement in transformation process as introduced in Application Profile component, the customer has main roles in project profile definition.

7.3. Showing Quality Attributes Achievement Through a Usage Scenario

To evaluate the proposed framework and the applied scenario, we consider a usage scenario (see section 2.6). Figure 7.1 introduces the flow of this scenario and the interactions between components as follows:

1- The application profile is prepared and discussed with stockholders.

2- The application is assessed to check its suitability for transforming to the cloud.

3- The application is analyzed and its components are defined.

4- The transformation plan is designed, and services are defined.

5- The application services, and cloud supporting services, and interactions between services are defined.

6- The application is transformed to cloud and introduced as Saas.

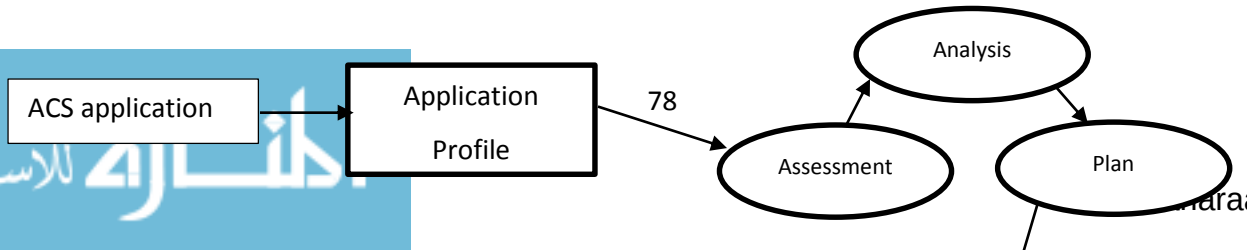

Figure 7.1 Usage Scenario for Applying Framework

The presented scenario outlines how applying the framework fulfills the quality attributes which are agility and integrability.

For the quality attributes discussion, first, agility achievement is clear in this scenario, this is because the well-defined framework architecture. The integrability attribute, achieved in the process flow of transformation, and the interaction between components.

7.4. Conclusion and discussion

The proposed framework is evaluated using a scenario based software architecture evaluation method and proves that it achieves the quality attributes set as goals for the framework which are: Integrabality, and agility. Moreover, a scenario of application transformation is adapted and validated. A specific usage scenario for the framework is discussed and further proves that the framework accomplishes its functionality and quality attributes.

Chapter 8 Conclusions and Future Work

8.1. Conclusions

In this research we introduced a cloud transformation framework, the current situation of the e-Government cloud infrastructure was analysed and discussed.

The proposed framework contains three main components, the Application profile component which describe the application and introduces its characteristics and functionalities, the software development lifecycle component, in which the assessment of the cloud suitability is presented, the analysis of the application components, and the transformation plan is presented; and finally the Saas specification component which discuss the service specifications and implementation.

The framework has been applied through applying the transformation of a legacy application, parts of the framework was applied to achieve the research goals which are the agility, and the integrability.

The framework was evaluated based on the ATAM evaluation method, the scenario based method was introduced, and the framework achieves the goal of the research.

8.2. Future Work

In this research we designed a cloud transformation framework, and define its components, and interactions. The framework is not fully applied, but a scenario was introduced.

The framework doesn't address features such as interoperability and interconnectivity, Interoperability is the ability of software to use the exchanged information and to provide something new originated from exchanged information whereas interconnectivity is the ability of software components to communicate and exchange information. Thus, interconnectivity is a prerequisite for interoperability and those two – interconnectivity and interoperability - are intertwined with functionality and visible at runtime, and may increase the benefits of cloud transformation.

Moreover the framework did not address the issue of scalability of enterprise applications

Future direction in this research could be summarized as follows:

- Full and complete appyling of the framework.
- Enhancing the framework by adding features interconnectivity, and interoperability.
- Providing semantic capabilities to the framework for scalability.

References

[1] R. S. Baraka and S. M. Madoukh, "A conceptual SOA-based Framework for e-Government Central Database," in *Computer, Information and Telecommunication Systems (CITS), 2012 International Conference on*, 2012, pp. 1-5.

[2] S. Venkatraman and B. Wadhwa, "Cloud computing: A research roadmap in coalescence with software engineering," *SEIJ,* vol. 2, pp. 7-18, 2012.

[3] Z. Mahmood and S. Saeed, *Software engineering frameworks for the cloud computing paradigm*: Springer, 2013.

[4] K. K. Hausman, S. L. Cook, and T. Sampaio, *Cloud Essentials: CompTIA Authorized Courseware for Exam CLO-001*: John Wiley & Sons, 2013.

[5] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao*, et al.*, "Cloud computing: a perspective study," *New Generation Computing,* vol. 28, pp. 137-146, 2010.

[6] C. N. Hoefer and G. Karagiannis, "Taxonomy of cloud computing services," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, 2010, pp. 1345-1350.

[7] V. Matveev, "Platform as a Service-new opportunities for software development companies," 2010.

[8] M. Turner, D. Budgen, and P. Brereton, "Turning software into a service," *Computer.,* vol. 36, pp. 38-44, 2003.

[9] B. Waters, "Software as a service: A look at the customer benefits," *Journal of Digital Asset Management,* vol. 1, pp. 32-39, 2005.

[10] L. Lindstrom and R. Jeffries, "Extreme programming and agile software development methodologies," *Information systems management,* vol. 21, pp. 41-52, 2004.

[11] D. Thomas, "Enabling Application Agility-Software as A Service, Cloud Computing and Dynamic Languages," *Journal of Object Technology,* vol. 7, pp. 29-32, 2008.

[12] K. Henttonen, M. Matinlassi, E. Niemela, and T. Kanstren, "Integrability and Extensibility Evaluation from Software Architectural Models- A Case Study," *The Open Software Engineering Journal,* vol. 1, pp. 1-20, 2007.

[13] H. Hai and S. Sakoda, "SaaS and integration best practices," *Fujitsu Scientific and Technical Journal,* vol. 45, pp. 257-264, 2009.

[14] P. Bianco, R. Kotermanski, and P. F. Merson, "Evaluating a service-oriented architecture," 2007.

[15] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for architecture evaluation," DTIC Document2000.

[16] L. Bass, *Software architecture in practice*: Pearson Education India, 2007.

[17] M. Barbacci, P. C. Clements, A. Lattanze, L. Northrop, and W. Wood, "Using the Architecture Tradeoff Analysis Method (ATAM) to evaluate the software architecture for a product line of avionics systems: A case study," 2003.

[18] J. Butler, "The architecture component of the SAE reference framework for SOA," *http://everware-cbdi.com/document_34,* 2015.

[19] S. Kambhampaty, "Service oriented analysis and design process for the enterprise," in *7th WSEAS International Conference on Applied Computed Science, Venice, Italy*, 2007.

[20] R. Kamatchi and A. Rakshit, "Service Oriented Analysis and Design with educational information system," in *2011 World Congress on Information and Communication Technologies*, 2011.

[21] O. Zimmermann, P. Krogdahl, and C. Gee, "Elements of service-oriented analysis and design," *IBM developerworks,* 2004.

[22] C. Handling, "SOMA–Service Oriented Modeling and Architecture."

[23] A. Arsanjani and A. Allam, "Service-Oriented Modeling and Architecture for Realization of an SOA," in *Services Computing, 2006. SCC '06. IEEE International Conference on*, 2006, pp. 521-521.

[24] G. Lewis, E. Morris, and D. Smith, "Service-oriented migration and reuse technique (smart)," in *Software Technology and Engineering Practice, 2005. 13th IEEE International Workshop on*, 2005, pp. 222-229.

[25] V. Kundra, "Federal cloud computing strategy," 2011.

[26] A. J. Rani, "Analytical Study of Agile Methodology with Cloud Computing," *IJCA Proceedings on National Workshop-Cum-Conference on Recent Trends in Mathematics and Computing 2011,* 2012.
[27] R. Guha and D. Al-Dabass, "Impact of Web 2.0 and Cloud Computing Platform on Software Engineering," in *Electronic System Design (ISED), International Symposium on*, 2010, pp. 213-218.
[28] R. Guha and D. Al-Dabass, "Impact of web 2.0 and cloud computing platform on software engineering," in *Electronic System Design (ISED), 2010 International Symposium on*, 2010, pp. 213-218.
[29] S. Frey and W. Hasselbring, "Model-based migration of legacy software systems to scalable and resource-efficient cloud-based applications: The cloudmig approach," in *CLOUD COMPUTING 2010, The First International Conference on Cloud Computing, GRIDs, and Virtualization*, 2010, pp. 155-158.
[30] T. Danciu, "The JasperReports Ultimate Guide," ed, 2002.
[31] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra, "Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps," in *Proceedings of the 6th ACM workshop on Formal methods in security engineering*, 2008, pp. 1-10.
[32] M. LaMonica, "Open source meets business intelligence," *CNET News. com, published: April,* vol. 23, p. 2006, 2006.
[33] S. H. Kan, *Metrics and models in software quality engineering*: Addison-Wesley Longman Publishing Co., Inc., 2002.
[34] S. S. Alhir, "Understanding the Unified Process (UP)," *Methods & Tools,* vol. 10, pp. 2-17, 2002.
[35] D. Das and K. Vaidya, "an agile process framework for cloud application development," Computer Sciences Corporation, leading edge forum2011.
[36] J. Butler, "The architecture component of the SAE reference framework for SOA," *CBDi Journal, http://www. cbdiforum. com/secure/interact/2007-03/the_architecture_component. php,* 2007.
[37] S. Patidar, D. Rane, and P. Jain, "Challenges of software development on cloud platform," in *Information and Communication Technologies (WICT), 2011 World Congress on*, 2011, pp. 1009-1013.
[38] A. Sitalakshmi Venkatraman1 and Bimlesh Wadhwa2 University of Ballarat, "Cloud Computing A Research Roadmap in Coalescence with Software Engineering," *Software Engineering : An International Journal (SEIJ),* vol. 2, 2012.
[39] S. A. Almulla and Y. Chan Yeob, "Cloud computing security management," in *Engineering Systems Management and Its Applications (ICESMA), 2010 Second International Conference on*, 2010, pp. 1-7.
[40] Y. Jiang, X. Zhang, Q. Shen, J. Fan, and N. Zheng, "Design of E-Government Information Management Platform Based on SOA Framework," in *Networking and Distributed Computing (ICNDC), 2010 First International Conference on*, 2010, pp. 165-169.
[41] J. Yunliang, Z. Xiongtao, S. Qing, F. Jing, and Z. Ning, "Design of E-Government Information Management Platform Based on SOA Framework," in *Networking and Distributed Computing (ICNDC), 2010 First International Conference on*, 2010, pp. 165-169.
[42] E. Brown. (2011, 15/5/2014). *Final Version of NIST Cloud Computing Definition Published*.
[43] M. M. Kherajani and M. A. Shrivastava, "Impact of Cloud Computing Platform Based on Several Software Engineering Paradigm," *International Journal of Advanced Computer Research,* 2011.
[44] R. Guha, "Impact of Semantic Web and Cloud Computing Platform on Software Engineering," in *Software Engineering Frameworks for the Cloud Computing Paradigm*, Z. Mahmood and S. Saeed, Eds., ed: Springer London, 2013, pp. 3-24.
[45] E. K. Vahid Khatibi, "Issues on Cloud Computing: A Systematic Review " *International Conference on Computational Techniques and Mobile Computing (ICCTMC'2012),* 2012.
[46] G. Breiter and M. Behrendt, "Life cycle and characteristics of services in the world of cloud computing," *IBM Journal of Research and Development,* vol. 53, pp. 3:1-3:8, 2009.
[47] A. Patrascu and V.-V. Patriciu, "Logging framework for cloud computing forensic environments," in *Communications (COMM), 2014 10th International Conference on*, 2014, pp. 1-4.
[48] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.
[49] M. Jarrar, A. Deik, and B. Farraj, "Ontology-based Data and Process Governance Framework," *Data-Driven Process Discovery and Analysis SIMPDA 2011,* p. 83, 2011.

[50] A. D. M. a. B. F. M. Mustafa Jarrar (BZU), "Ontology-based Data and Process Governance Framework – The Case of e-Government Interoperability in Palestine," *pre-proceedings of the IFIP International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'11),* pp. 83-98, 2011.

[51] G. H. G.-E. Akram Naser, "Proposed Development Model Of e-Government To Appropriate Cloud Computing," *International Journal of Reviews in Computing* vol. 9, 2012.

[52] S. Abhishek and M. Frank, "A Roadmap for Software Engineering for the Cloud: Results of a Systematic Review," in *Agile and Lean Service-Oriented Development: Foundations, Theory, and Practice*, W. Xiaofeng, A. Nour, R. Isidro, and V. Richard, Eds., ed Hershey, PA, USA: IGI Global, 2013, pp. 48-63.

[53] R. AFONSO FRANCOS, "Socio-technical systems simulation and analysis: a goal-oriented framework," 2011.

[54] I. Sommerville, *Software Engineering*: Addison-Wesley; 9 edition, 2010.

[55] J. Schulte, "A Software Verification & Validation Management Framework for the Space Industry," 2009.

[56] H. Altarawneh and A. El Shiekh, "A Theoretical Agile Process Framework for Web Applications Development in Small Software Firms," in *Software Engineering Research, Management and Applications, 2008. SERA '08. Sixth International Conference on*, 2008, pp. 125-132.

[57] R. Guha, "Toward the Intelligent Web Systems," in *Computational Intelligence, Communication Systems and Networks, 2009. CICSYN '09. First International Conference on*, 2009, pp. 459-463.

[58] B. F. Ben Collins-Sussman, and C. Michael Pilato, *Version Control with Subversion*: O'Reilly Media; 2 edition (June 30, 2009).

[59] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski*, et al.*, "A view of cloud computing," *Communications of the ACM,* vol. 53, pp. 50-58, 2010.