አዳማ ሳይንስ እና ቴክኖሎጂ ዩኒቨርሲቲ
Adama Science & Technology University

School of Graduate Studies

Department of Computing

Variability Challenges and Design Approach of SaaS-based Inventory and Sales data management system in the case of Small and Medium Enterprises in Adama City

A thesis submitted to School of Graduate Studies of Adama Science and Technology University in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering

Wobetu Shiferaw

September, 2015

Adama, Ethiopia

አዳማ ሳይንስ እና ቴክኖሎጂ ዩኒቨርሲቲ
Adama Science & Technology University

School of Graduate Studies

Department of Computing

Variability Challenges and Design Approach of SaaS-based Inventory and Sales data management system in the case of Small and Medium Enterprises in Adama City

A thesis submitted to School of Graduate Studies of Adama Science and Technology University in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering

By Wobetu Shiferaw

Names and Signature of Members of the Examining Board

Chair Person, Examining Board	Signature
Dr. Mesfin Kifle Advisor	 Signature
Examiner	Signature

Declaration

The thesis is my original work and has not been presented for a degree in any other university or conferences and all sources of materials used have been acknowledged.

_____

Wobetu Shiferaw

Acknowledgement

Above and first of all, Lord, Glory to you. Let my praise and heartfelt thanks come unto your Kingdom. With you, I believe that, nothing is impossible and as so, I became a man as I am today.

Next, I would like to express my sincere gratitude to my advisor Dr. Mesfine Kifle (PhD, Assistant Professor of Computer Science, Addis Ababa University) for your patience, motivation and valuable comments you provided to me throughout the research. Your comments and directions helped me a lot to successfully complete this work. Thanks a lot.

Finally, I would like to express my special thanks to my instructors, classmates and close friends who helped me directly and indirectly to complete this research paper successfully.

Table of Contents

List of figures

List of tables

Glossary

API	-	Application Programming Interface
ASP	-	Application Service Provider
AWS- EC2	-	Amazon Web Services-Elastic Compute Cloud
CC	-	Cloud Computing
CPU	-	Central Processing Unit
CRM	-	Customer Relationship Management
CSP	-	Cloud Service Provider
CV	-	Curriculum Vitae
ERP	-	Enterprise Resource Planning
ETB	-	Ethiopian Birr
EU	-	European Union
FDRE	-	Federal Democratic Republic of Ethiopia
HR	-	Human Resource
HTTP	-	Hypertext Markup Language
IaaS	-	Infrastructure-as-a-Service
IBM	-	International Business Machines
ICT	-	Information Communication Technology
IDE	-	Integrated Development Environment
IMS	-	Inventory Management System
IT	-	Information Technology
LAMP	-	Linux Apache MySQL PHP
LAN	-	Local Area Network
mBaaS	-	Mobile Backend-as-a-Service
MSME	-	Micro, Small and Medium Enterprise
NIST	-	National Institute of Standards and Technology, United States
PaaS	-	Platform-as-a-Service
PC	-	Personal Computer
PDA	-	Personal Digital Assistant
PLC	-	Private Limited Company

PSA	-	Processional Services Automation
REST	-	REpresentational State Transfer
SaaP	-	Software-as-a-Product
SaaS	-	Software-as-a-Service
SAP	-	Systems, Applications and Products (a German Software Company especially well known for ERP)
SDK	-	Software Development Kit
SLA	-	Service Level Agreement
SMB	-	Small and Medium Business
SME	-	Small and Medium Enterprises
SOAP	-	Simple Object Access Protocol
UI	-	User Interface
UNCTAD	-	United Nations Conference on Trade and Development
US	-	United States
VMM	-	Virtual Machine Monitor
XML	-	eXtensible Markup Language

Abstract

Software-as-a-Service (SaaS) is one of the important models of Cloud Computing (CC), in which software can be delivered as-a-service online on a one-to-many basis which can be an important step for individuals and organizations to have access to software products as on-demand service instead of purchasing a full-fledged software product. This way of getting software reduces the cost of having the software and other running resources. This kind of service delivery has become most popular especially for Small and Medium Enterprises (SMEs) in the world since it greatly decrease cost of having ICT infrastructure and automated software system and so that they can outsource ICT related burdens for service providers and concentrate on direct business activities.

However, since SaaS is about sharing of centralized single instance software and data architecture, service providers face problems of providing services of heterogeneous customer requirements. Since, most business customers have different business requirements; it is hard to satisfy all customers by developing standard application software. So an approach should be devised to develop a fully tailored and customizable application so that users can use an on-demand, self-service shared application.

In this thesis, we have identified advantages of adopting cloud computing solutions for Small and Medium Enterprises in Ethiopia and explored the various requirement challenges facing in SaaS service offerings from different level of software architecture, conducted data collection and analysis based on taken case study automation system (development approach of online Inventory Management system). Then we developed a framework and development approach that helps SaaS developers in the development of business applications targeted SMEs. Finally, we have proposed SaaS-based Inventory management system and recommended to apply it in all cases off SaaS-based business automations.

Keywords: CC, SaaS, Multi-tenancy, SMEs/SMBs, ASP, SaaP

CHAPTER ONE

1. INTRODUCTION

1.1. Background

Cloud Computing (CC) is a model for enabling ubiquities, 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 [1]. It is a new computing paradigm that allows IT management to become the most effective in terms of cost and performance as well as to be available everywhere and at any time. CC shifts the traditional desktop computing approach ("desktop as a platform") into internet based computing ("Internet as a platform") [2] where users and organizations have access and use the services provided by the Cloud Service Provider (CSP) through Internet as a communication link. CC [3] has been widely recognized as the next generation computing infrastructure. CC offers some advantages by allowing users to use **infrastructure** (e.g., servers, networks, and storages), **platforms** (e.g., middleware services and operating systems), and **software** (e.g., application programs) provided by cloud providers (e.g., Google, Amazon, and Salesforce) at low cost. Different companies in the world ranging from small business organizations to big named ones nowadays are moving their IT service management into the cloud infrastructure renting either of virtualized hardware (IaaS), hardware and platforms (PaaS) or the whole cloud service including hardware, platforms and application software mainly referred to as SaaS from different cloud service providers which are either of public, private, community or hybrid cloud service providers. However, in developing countries like Ethiopia, CC is still in infant stage and no much work has been done to take advantage of the opportunities it brings. Organizations in Ethiopia are using the traditional desktop computing systems even in a very little scale while many of them still using the very traditional paper based computing. Lack of finance to purchase hardware and suitable application software as well as lack of professional manpower to manage the IT infrastructure and software development process is the main reason for this. These problems are more prevalent in small and medium sized business (SMB) institutions where the government put strong emphasis as a hope to the countries growth and development base. They face initial capital investment problem, IT service

management is additional requirement for them and it is even thought as a luxury. To have IT service facility, they have to have computers, purchased application programs and network communication links to collaborate the work between different departments of the institution. But as pointed out above, having all this facility requires to invest much and even impossible. On the other hand leveraging IT automation should not be continued as a luxury since it is a focal point of economic development and growth. Then the best solution is moving into the cloud more specifically adopting **Software-as-a-Service(SaaS)** model can become the best solution for these kinds of business where an organization can use all types of services (infrastructure, deployment platform and application programs installed in the cloud provider premises). These customers can then access the application services from their laptop, desktop, tablet or even from smartphones via the web application interface. However this kind of computing paradigm basically has its own drawbacks and challenges. Among other things security, privacy, payment models, market targets, multi-tenancy and network performance issues are the most prevailing challenges of SaaS. In addition to these, the most interesting challenge that attracts more researchers in the area is the existing of variant customer business requirements. This means that different SaaS customers have no common business requirements and so no common standard interface and business logic. This in turn requires an application to handle different user interface, different business logic and different database structure. As the software application becomes more complex, business requirement become more complex and diverse. SaaS customers may raise the following challenges to the service provider [4].

- I need more fields to describe my business documents
- Our manager wants a different report/dashboard to analyze sales data
- Our organization has no roll of procurement manager
- The workflow of our business is different with what you support

Any of these challenges can be divided into implications to different perspectives of the SaaS application. E.g. data, business logic, user interface, organization structure, business rules etc. These conclusions indicate us that, we have to have some tailoring techniques and tools to harmonize the services provided by the SaaS vendor. This issue gains major focus in this research. Accordingly, a study of customer business requirement challenges will be conducted then a solution approach will be proposed.

1.2. Basic Concepts of Cloud Computing

The term cloud is Internet-based computing, whereby shared resources, software, and information are provided to computers and other devices on demand – pay per use. It's a model for enabling on-demand access to shared pool of computing resources such as servers, applications and services, networks, security utilities etc. Data and software packages are stored in servers; its structure allows access to information as long as an electronic device has access to the web. This allows employees to work remotely at any time anywhere in the world.

1.2.1. Service Delivery models of Cloud computing

Cloud computing is all about delivering different services such as shared data storage, CPU cycles, Operating Systems, development environments and software application services. The model of providing these services has been designed as a layered service oriented architecture. This service oriented architecture has 3 hierarchical components and the services provided at each layer clearly identified. The figure below clearly shows this CC layered service delivery model (the so called service oriented architecture of CC) [3].

Infrastructure-as-a-Service (IaaS)

A service model that involves outsourcing the basic infrastructure used to support operations--including storage, hardware, servers, and network equipment. The service provider owns the infrastructure equipment and is responsible for housing, running, and maintaining it. The customer typically pays on a per-use basis. The customer uses their own platform (Windows, UNIX), and applications

Platform-as-a-Service (PaaS)

A service model that involves outsourcing the basic infrastructure and platform (Windows, UNIX). PaaS facilitates deploying applications without the cost and complexity of buying and managing the underlying hardware and software where the applications are hosted. The customer uses their own applications**.**

Software-as-a-Service (SaaS)

Also referred to as "software on demand," this service model involves outsourcing the infrastructure, platform, and software/applications. Typically, these services are available to the customer for a fee, pay-as-you-go, or a no charge model. The customer accesses the applications over the internet. The customer seems like using only the software online but actually he/she uses the platform where the software is running on, and the hardware and virtualized resources. That means SaaS user uses all resources and services of cloud without having any resource at hand. The only thing the client needs to have is an end point device may it be desktop/laptop PC or mobile phones to get access from the services provided. This is the main interesting point of this research study since in developing countries like Ethiopia, small and medium sized business have limitations in all the IT infrastructure resources and there is a need to outsource their automation and get benefit from CC SaaS delivery model.

1.3. Small and Medium Sized business enterprises

"Small and medium enterprises" (**SME)** or "small and medium-sized businesses" (**SMBs**) are businesses whose personnel numbers fall below certain limits. There is no universally accepted definition of an SMB enterprise. Different standardizations to define businesses as micro, small, medium sized or large scale enterprises exist in different regions in the world. The prominent factors in every kind however are similar, total number of employees and turn over (total revenue minus indirect tax) or total investment capital. The European Union (EU) [5] put total employee boundary to less than 10, 50 and 250 to micro, small and medium sized enterprises respectively and in terms of total investment capital, it has put less than or equal to €2, €10 and €43-€50 Million respectively. The United States [6], scale it based on turn over revenue as <$7 Million,

<$25 Million and <$250 Million respectively. The Ethiopian government, as of 2012, clearly defined it based on maximum number of employee and total investment capital. According to this declaration micro business enterprise means an enterprise with total number of employees (including family members) not exceed to 5 and maximum capital bound to 50,000(ETB) for service delivery sectors or up to 100,000(ETB) for industry sectors. And "Small Scale enterprise" means an enterprise with total number of employees (including family members) bound between 6 and 30 while total capital investment is 50,001 – 500,000 (ETB) for service delivery sectors and 100,001 – 1,500,000 (ETB) for industry sectors. So this time on in this research proposal and future research document, while we say SMB or SME, we are referring to these kinds of businesses as stipulated on [7].

1.4. Statement of the problem

Software as a Service (SaaS) provides software application vendors a Web based service delivery model to serve big number of clients with multi-tenancy based infrastructure and application sharing architecture so as to get great benefit from the economy of scale [4]. This kind of service model is very useful especially for organizations which have no interestingly enough capital investment to purchase hardware, software and to afford having a network infrastructure as well as skillful human resource to manage the IT services. These applications are often designed in highly standardized software architecture, but many clients have diverse business requirements and this makes SaaS-based application delivery much more challenging to adopt it. For example, different organizations have different process and requirements to manage human resource, finance, procurement, employee recruitment etc. for instance, one organization may want a candidate to submit his/her CV as a pdf while the other want him/her to fill the online recruitment application asking first name, last name, sex, age, work experience, specialization etc. (filling each field for each attribute). Another example, in the context of SaaS based human resource system, arriving late of an employee matters for one organization while the other organization do not care about it. Language is also another barrier. If the user interface supports only English language, for instance, it means that it can't satisfy maximum target customers; it can't achieve maximum market penetration since most of the business persons are illiterate and can't interact. So it should be configurable to support multi-lingual functions. One may need to have additional data fields and labels to manipulate his business data while the other

does not, on the available user interface. These all cases will result in different functionalities among different customers such as different user interfaces, different business logic, different database structure etc. These are the major challenges to adopt SaaS-based software service into SME's since they are varied from simple, medium, complex and highly complex organizations and their functionalities. In this case fully configurable and customizable software should be built and, to achieve this, there should be some tailoring mechanism, procedure and tool to support the development of this kind of systems. Configuration and customization are the two most important aspects of tailoring a SaaS-based software development. But "what level of configuration and customization required" and "are these methods will affect the user interface, the source code and database structure?" and "is code customization necessary and feasible" are the major question while adopting of SaaS services and developing of the software. Different authors have conducted a number of researches [8] [9] [4] to study the challenges in designing of SaaS in general and requirement variability in particular. Most of these works mainly focus on migration of a single tenant web application into multi-tenant enabled applications as well as configuration-customization aspects of multi-tenant enabled software and none of which, as to our understanding didn't touch design approach of SaaS-based application from implementation point of view.

At the end of the day, the result of this research answers the question.

1. Is the existence of requirement variability is a challenge to use a cloud computing service and if so, how can we design an approach for developing SaaS-based software for Small and Medium business that copes variability.

1.5. Objective of the study

The main objective of this research is to study the variability challenges of customers' business requirements in SaaS-based software service delivery and designing a service and development approach to solve the problems specified. The research concentrates more specifically on:

- Adopt and modify a SaaS-based software architecture for SME's

- Analyze variability of customers' business requirements raising some sample SME's on the case of a "stock and inventory control" automation system
- Design an enhanced configuration and customization model to develop and use SaaS-based software application

1.6. Scope and limitations

This research is limited only to study the existence of variance in business requirements across different SMEs and designing of framework that realizes a true multi-tenant SaaS architecture. It deals with SaaS-based software configuration and customization issues that can satisfy the requirements of a large group of customers. In contrast, it doesn't deal with a number of cloud computing service delivery issues and challenges such as Security, costing and payment models. Plus, it concentrates only on showing developers the challenges and desirable solutions to follow in the development of the software and doesn't deal with a full-fledged application development and testing.

1.7. Research methodology

To achieve the solutions to the problems specified, the following research methods have been conducted throughout the progress of the research.

1.7.1. Document analysis

Business documents of Sample business enterprises such as forms and reports has been examined, identified and compared to test whether there is a difference in business requirement among them so that we can decide to develop the software whether as a standard product or a highly customizable one as well as to what level of customization and configuration is required.

1.7.2. Interview

We have conducted a randomly raised open ended interview with company owners, financial auditors and officers from Adama trade and urban development agency as well the city's micro and small enterprises development agency to collect important information about current statistics of SMEs and SMBs as well as current business operations of these companies.

1.7.3. Literature review

Review of related journals, books and web sources has been conducted to examine either of similar works have been done before and to support our idea with already worked investigations. This helps us to have better understanding of the subject matter, to identify the gaps and to develop a better solution to the problem identified.

The research has been conducted mainly in the following procedures

- Sample organizational data was collected from sample SME's with respect to SaaS-based Inventory and sales management automation requirements
- The data was analyzed and compared from data and business perspectives
- Then currently available tailoring models and architectures has been studied and gaps identified
- A suitable software architecture was proposed and designed
- An enhanced programming model developed to leverage customization and configuration techniques in using and development of the software

In general the overall activities performed in the research can be summarized step by step as shown in the diagram below.

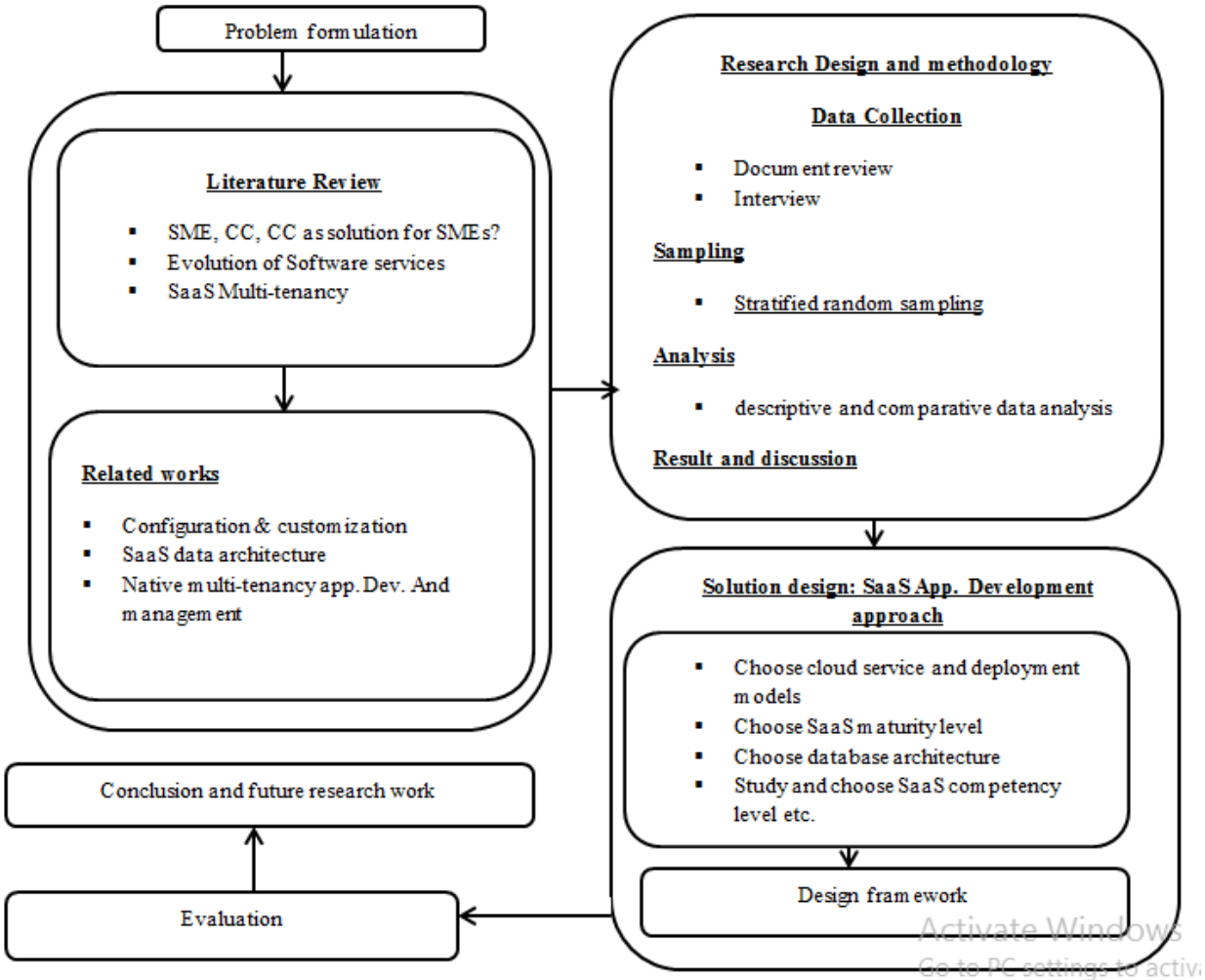

Figure 1-1 Research methodology diagram

1.8. Significance and Expected results of the study

SME's are a major hope to the government as a center for growth and economic development of the country. To achieve this goal ICT should play an important role and it should become a necessity for these enterprises to be supported by it. But what we are seeing in this kind of organizations is, there is lack of IT infrastructure such as software, hardware and network facilities as well as skillful manpower to manage the infrastructure. In these environments, adopting SaaS-based automation systems become the best solution to achieve economy of scale. Since many organizations have different software requirements out of the same automation

system, a highly configurable and customizable software offering greatly enhances their utilization based on their unique interest. On the other hand the service vendors of this kind of software can become greatly profitable since they can attract very large customers rather than offering a highly standardized system. To achieve this, the proposed software tailoring model greatly directs them to develop a high performance software solution.

1.9. Organization of the paper

The rest of this thesis comprises of five chapters. The next chapter contains review of problem related literatures and background information. Chapter three covers review of related works and chapter four discusses about research design and methodology which includes research design, research approaches and methods, research techniques, requirements gathering and data analysis as well as discussion of analysis results. Then chapter five covers solution design of the proposed architecture of SaaS based software development, chapter six discusses an implementation proposal on how the solution can be implemented and applied. Finally, chapter seven concludes the thesis and suggests further research directions.

CHAPTER TWO

2. CONCEPTUAL DISCUSSION AND LITERATURE REVIEW

2.1. Overview

Having background information on the area around the problem is an important step in any research activity. Accordingly, we have conducted a study of the important scientific background and literature review around the problem under study. In this chapter we have included scientific and background information on SMEs, Cloud Computing, CC Service and deployment models, CC components, enabling technologies, evolution and comparison of software system services as well as multi-tenancy architecture and its implementation models.

2.2. Small and Medium Sized Businesses

SMEs' role in the overall economy cannot be over-emphasized, because of its contribution to creating more jobs and development of the social-economy for the local community. They are at the heart of efforts to address key socio-economic challenges such as poverty and unemployment. In South Africa, as is the case in most African economies, the SME sector is one of the largest employment-generating sectors, particularly among the country's poor. In Ethiopia, an estimated 50% of the urban workforce is projected to be engaged in the micro and small-enterprises sector. In Ghana, SMEs have become critical agents in enabling the countries attain their development objectives, notably poverty alleviation and wealth creation [10].

According to the United Nations Conference on Trade and Development (UNCTAD), SMEs account for 60 to 70 per cent of all employment in developing countries, and hence contribute to poverty reduction. In this regard, many countries in Africa have given high priority to the growth of SMEs. For instance, the Government of Ethiopia, in its Growth and Transformation Plan (2010/11-2014/15), has given top priority to micro and small enterprises, targeting to create employment opportunities for more than three million people by end of the plan period and aiming to boost access to ICTs. [11]. It has identified the development of micro, small and medium enterprises (MSMEs) as a key industrial policy direction for creating employment opportunities for millions of Ethiopians.

Micro & Small Enterprise Development Program in Ethiopia meaningfully has been given due attention by government since 2004/2005 (Konjit Debela). Since then, the government puts strong attention to the development of the business and believes it as the main way to create employment opportunities thereby can alleviate poverty. SMEs in Ethiopia are main important parts of the country's Growth and Development strategy which is believed to be genuine policy to push the country become a middle income nation in 2025.

2.2.1. The role of ICT for SMEs

SMEs are best places where entrepreneurship, innovation and adoption of new technology can be easily entertained. To be successful, then, they need to be supported by Information Communication Technology (ICT) and services, so that they can enhance the business automation culture. ICT has become a major catalyst for change and economic transformation. It's a foundation to enhance every aspect of life of society such as education, business, banking and healthcare. Through the introduction of powerful mobile technologies, ICT become a necessity by which it can play significant role in the day to day activities of society. Adopting ICT for SMEs is difficult especially for developing countries since ICT infrastructure requires significant capital budget and high profile expertized personnel to build, in contrast to their capacity. SMEs by definition are low powered businesses which are assumed to be the way for creating big industries starting from investing less. For instance, maximum initial capital of SME in Ethiopian economy is 1.5m ETB for industry firms and this amount become even less (50000 ETB) (Konjit Debela) when applied to service sectors. Since, ICT is a major tool for change and development, SME firms should be supported by it to create more business opportunities. Even though, Ethiopia is very young in the technology, other African countries such as Kenya, Nigeria Ghana and South Africa have also experienced the start and become a model to other neighbors. These countries have already adopted different ICT solutions like e-commerce, e-agriculture, e-learning, e-health services in their SMEs. More than this, adopting cloud computing technologies is even bring tremendous remarks in the transformation of enhanced ICT services which become affordable for any type of business in any country in the world since it's all about cost effective deployment and utilization of ICT computing resources and services.

2.3. Cloud Computing

The application of the term is everywhere. Pick up any tech magazine or visit almost any IT website or blog and we will be sure to see talk about cloud computing. The only problem is that no one agrees on what exactly it is. There is no universally accepted definition of cloud computing among different authors and stake holders. Even though the core concepts are almost similar, different authors defined CC in different ways. Some of these definitions are:

- The US National Institute for Standards and Technology defines it as [1]

Cloud Computing is a model for enabling ubiquities, 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 characteristic (on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service), three service models and four deployment models.

- According to Bhaswati Hazarika et al. [12] CC can be defined as:

Cloud Computing is a way of leveraging the Internet to consume software or other IT services on demand. Users share processing power, storage space, bandwidth, memory, and software. The resources are shared and so are the costs. Users can pay as they go and only use what they need at any given time, keeping cost to the user down. Cloud Computing is very much a business model as well.

- And [13] defines CC as:

A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers.

CC involves deploying groups of remote servers and software networks that allow centralized data storage and online access to computer services or resources using the internet. It rapidly

become popular and a newly standard computing fashion in the IT industry and become the most effective means of computing in terms of technical and economic perspectives. Because of CC, companies now don't need to have their own data centers and on-premise enterprise applications. They can rent servers, processors, storage, applications and services online from a service provider. In most organizations, IT service management is an indirect activity of the business. Since so, leveraging the cloud service can help them outsource the complex IT related burdens and be effective in terms of cost and reduced technical manpower to administer the IT infrastructure as well as to spend their precious time on the directly responsible operations of the business. CC is a great way for IT professionals to focus less on their data centers and more on the work of Information Technology. Nowadays, CC has a great influence on giant to small and medium sized business companies and most of them are moving into this trend. Amazon, force.com, Google, Oracle and Microsoft are among the giant companies that are already adopted CC paradigm and provides services for their customers. The beauty of CC is that another company hosts your application, handles the cost of servers, and manages the software updates, and you pay less for the service.

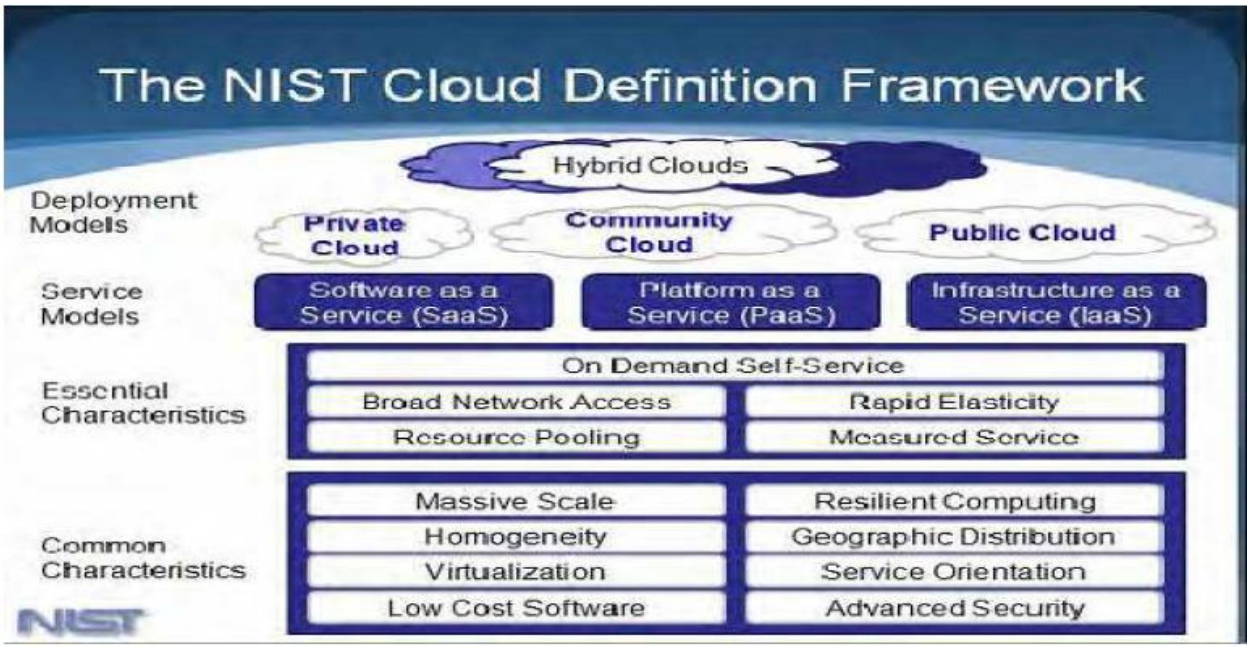

Figure 2-1 The NIST definition of cloud computing [1]

2.3.1. *Cloud Computing Components*

Whenever we see from a high level point of view, a cloud solution is built from several elements: application, client, infrastructure, platform, service, distributed servers and storage resources.

Application

A *cloud application* leverages the Cloud in software architecture, often eliminating the need to install and run the application on the customer's own computer, thus alleviating the burden of software maintenance, ongoing operation, and support. Web applications such as (twitter and Facebook) and SaaS (Google apps, salesforce and SAP) are applications provided based on cloud.

Clients

Cloud clients are devices that the end users can interact with to manage their information in the cloud. These include desktop PC, laptops, Mobiles, tablets or PDAs. They are also known as end point devices and generally fall into three categories [14].

- **Mobile client:** Mobile devices include PDAs or smartphones
- **Thin client:** computers that do not have internal hard drives, but rather let the server do the entire job, but then display the information itself.
- **Thick client:** this type of client is a regular computer, using a web browser like Firefox or Internet Explorer to connect to the cloud.

Thin clients are becoming an increasingly popular solution, because of their price and effect on the environment. Some benefits to using thin clients include

- **Lower hardware costs** thin clients are cheaper than thick clients because they do not contain as much hardware. They also last longer before they need to be upgraded or become obsolete.
- **Lower IT costs** thin clients are managed at the server and there are fewer points of failure.

- **Security** Since the processing takes place on the server and there is no hard drive, there's less chance of malware invading the device. Also, since thin clients don't work without a server, there's less chance of them being physically stolen.
- **Data security** since data is stored on the server, there's less chance for data to be lost if the client computer crashes or is stolen.
- **Less power consumption** thin clients consume less power than thick clients. This means you'll pay less to power them, and you'll also pay less to air-condition the office.
- **Ease of repair or replacement** If a thin client dies, it's easy to replace. The box is simply swapped out and the user's desktop returns exactly as it were before the failure.
- **Less noise** without a spinning hard drive, less heat is generated and quieter fans can be used on the thin client.

Infrastructure

This includes the cloud infrastructure which incorporates interconnected physical and virtualized resources as well as Network installations within the data center.

Cloud computing isn't a one-size-fits-all affair. There are several different ways the infrastructure can be deployed. The infrastructure will depend on the application and how the provider has chosen to build the cloud solution. This is one of the key advantages for using the cloud. Your needs might be so massive that the number of servers required far exceeds your desire or budget to run those in-house. Alternatively, you may only need a sip of processing power, so you don't want to buy and run a dedicated server for the job. The cloud fits both needs.

Services:

The term services in cloud computing is the concept of being able to use reusable, fine-grained components across a vendor's network. This is widely known as "as a service". Offerings with *as a service* as a suffix include traits like the following:

- Low barriers to entry, making them available to small businesses
- Large scalability
- Multitenancy, which allows resources to be shared by many users
- Device independence, which allows users to access the systems on different hardware

A *cloud service* includes "products, services and solutions that are delivered and consumed in real-time over the Internet or via Local Area Network. These services are given in a fashion of Software-as-a-Service, Platform-as-a-Service and Infrastructure in general.

Storage

Cloud storage involves the delivery of data storage as a service, including database-like services, often billed on a utility computing basis, e.g., per gigabyte per month. For example:

- Database (Amazon SimpleDB, Google App Engine's BigTable)
- Web service (Amazon Simple Storage Service, Nirvanix)

2.3.2. *Features, deployment and service models of cloud computing*

As indicated in [1] cloud computing services are based upon five basic features, four deployment models and three service models. The following section discuses each of these issues one by one.

2.3.2.1. Features

On-demand Self-Service

A consumer can provision computing capabilities such as server time and network storage as needed automatically without requiring interaction with each service provider.

Broad Network Access

Is one of the basic features of CC in which services are available through the network and accessed via heterogeneous end points (client devices) such as desktop PCs, laptops, mobile phones, tablets and PDAs.

Resource Pooling

This feature of cloud computing dictates the providers' computing resources are pooled to serve multiple consumers using a multi-tenant architecture, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).

Rapid elasticity

Another important feature of CC is its capability to provision resources elastically with consumer's requirement. This is the basic feature that differentiates it from traditional computing approach. In traditional computing systems every resource is pre-allocated and the consumer works only until the allocated resource is fully capable off. In a traditional system such as in traditional web hosting, for instance, if consumer runs out of disk space, service will be halted or additional space provisioned upon request manually. But in CC, capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly up and down with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

Measured Service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service. Another new aspect of CC is application of usage based billing model (pay-per-use or pay-as-you-go model). Customers simply pay for the services they use while providers bear the costs of hardware and software provision. Pricing may vary depending on the time of day due to peaks in demand or varying electricity costs and institutions may therefore carry out certain activities when costs are cheaper. However distributed cloud networks may enable providers to smooth out demand globally and offer uniform pricing strategies not dependent on timing.

2.3.2.2. Deployment Models

As indicated in the [1] definition, a CC system has different models based on either of deployment or service delivery. Based on how the cloud infrastructure is deployed or installed, todays cloud platforms available in any corner of the world are either of private, community, public or hybrid.

Private Cloud: is a secure IT infrastructure which is built, managed and used by a single organization. Traits that characterize private clouds include the ring fencing of a cloud for the sole use of one organization and higher levels of network security. They can be defined in contrast to a public cloud which has multiple clients accessing virtualized services which all draw their resource from the same pool of servers across public networks. Private cloud services draw their resource from a distinct pool of physical computers but these may be hosted internally or externally and may be accessed across private leased lines or secure encrypted connections via public networks. The private cloud model is closer to the more traditional model of individual local access networks (LANs) used in the past by enterprise but with the added advantages of higher security and virtualization, more control, cost and energy efficiency, improved reliability and cloud bursting.

Public Cloud: the most recognizable model of cloud computing to many consumers is the public cloud model, under which cloud services are provided in a virtualized environment, constructed using pooled shared physical resources, and accessible over a public network such as the internet. To some extent they can be defined in contrast to private clouds which ring-fence the pool of underlying computing resources, creating a distinct cloud platform to which only a single organization has access. Public clouds, however, provide services to multiple clients using the same shared infrastructure. The most salient examples of cloud computing tend to fall into the public cloud model because they are, by definition, publicly available. Software as a Service (SaaS) offerings such as cloud storage and online office applications are perhaps the most familiar, but widely available Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) offerings, including cloud based web hosting and development environments, can follow the model as well. Public clouds are used extensively in offerings for private individuals who are less likely to need the level of infrastructure and security offered by private clouds. However, enterprise can still utilize public clouds to make their operations significantly more efficient, for example, with the storage of non-sensitive content, online document collaboration and webmail.

Hybrid cloud: is an integrated cloud service utilizing both private and public clouds to perform distinct functions within the same organization. All cloud computing services should offer certain efficiencies to differing degrees but public cloud services are likely to be more cost efficient and scalable than private clouds. Therefore, an organization can maximize their

efficiencies by employing public cloud services for all non-sensitive operations, only relying on a private cloud where they require it and ensuring that all of their platforms are seamlessly integrated.

Hybrid cloud models can be implemented in a number of ways:

- Separate cloud providers team up to provide both private and public services as an integrated service
- Individual cloud providers offer a complete hybrid package
- Organizations who manage their private clouds themselves sign up to a public cloud service which they then integrate into their infrastructure

Community cloud: is a hybrid form of private clouds built and operated specifically for a targeted group. These communities have similar cloud requirements and their ultimate goal is to work together to achieve their business objectives. Community clouds are often designed for businesses and organizations working on joint projects, applications, or research, which requires a central cloud computing facility for building, managing and executing such projects, regardless of the solution rented.

2.3.2.3. Service Models

CC is a large-scale distributed network system implemented based on a number of servers in data centers. The cloud services are generally classified based on a layer concept. In the upper layer of this paradigm Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS) are stacked [3]. The figure below shows this layered Service-Oriented architecture of CC.

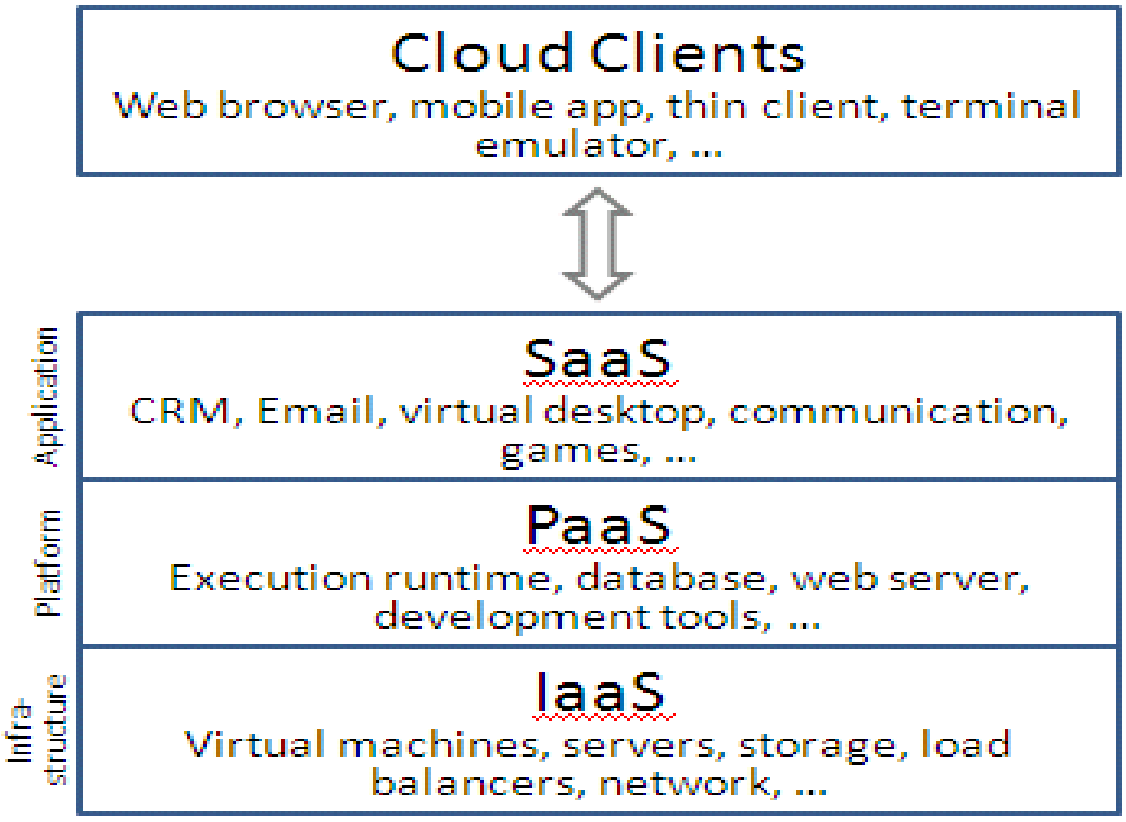

Figure 2-2 Service Oriented Cloud Computing Architecture [3]

Datacenter Layer

This layer provides the hardware facility and infrastructure for clouds. In data center layer, a number of servers are linked with high-speed networks to provide services for customers. Typically, data centers are built in less populated places, with high power supply stability and a low risk of disaster.

Infrastructure-as-a-Service (IaaS)

Infrastructure as a Service is built on top of the data center layer. IaaS enables the provision of storage, hardware, servers, and networking components. The client typically pays on a per-use basis. Thus, clients can save cost as the payment is only based on how much resource they really use. Infrastructure can be expanded or shrunk dynamically as needed. The service provider owns the infrastructure equipment and is responsible for housing, running, and maintaining it and the

customers use their own platform and application programs and services. Amazon Elastic compute cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3) are among suitable examples of this kind of Cloud services. Below is a list of some IaaS services explained in brief [15]:

Amazon AWS- EC2: Amazon is the granddaddy of IaaS providers. It was the first household brand name to provide such service. It is also the biggest provider. Originally, Amazon simply sold excess capacity from its own enormous infrastructure that it uses to power Amazon.com and all the associated services and products. However, its Amazon Web Services, or AWS, offering has quickly grown into a standalone business used both by numerous big name startups, and small users. Like many other IaaS providers, Amazon also offers numerous additional cloud services beyond straight infrastructure under the AWS banner. Amazon's Elastic Compute Cloud, or EC2, is the company's most direct IaaS product. Pricing is based on the size and type of instance created and the amount of time it runs.

Google Cloud Platform – Compute Cloud: Like Amazon, Google already had a massive world-wide infrastructure when it decided to get into the IaaS game. The IaaS part of Google's Cloud Platform is Google Compute Engine. Compute Engine offers numerous operating systems including Debian, CentOS, Red Hat Enterprise Linux, Windows Server and more. Google pricing offers different machine class types including: Standard, High CPU, High memory, and Small Machine. Cloud computing is one of the places where Google is playing catch-up to more entrenched competitors.

IBM Cloud- Softlayer: IBM's business model shifted long ago from just selling servers, to selling servers packaged with service and support to install and run that hardware. So, in one way, the move to selling virtual servers already installed and supported by IBM as IaaS was an easy move. On the other hand, like Microsoft IBM didn't start with a ready-made international network of servers and data centers, so the company bought Softlayer. IBM / Softlayer offers pure IaaS in form of bare metal servers, or virtual servers, with bare metal servers starting at $0.47/hr ($159/mo) versus just $0.04/hr or $28/mo for virtual servers. IBM also offers a fully managed IaaS, which it positions for use by corporate style IT departments.

AT&T Cloud Services: If Amazon and Google are the big names that leveraged their existing infrastructure to get to the cloud, and IBM, HP, and Microsoft are the big names that leveraged their existing expertise in server systems to get into the cloud, then AT&T is the provider that leveraged its existing worldwide network to jump into the cloud. AT&T Cloud Services' IaaS offering is called Synaptic Compute. Virtual machines start at $0.030 vCPU/HR, plus memory and storage. Image offerings are more limited than other providers and include just Red Hat Enterprise Linux 5 and Microsoft Windows Server 2008.

HP Cloud: The IaaS part of HP Cloud is the HP Helion Public Cloud. HP is a big contributor to the OpenStack community and the company integrates "key elements of OpenStack technology" in its offering. The best part of this, is that you can call your instance types "flavors" like they do in OpenStack. HP offers Standard and High Memory instance types (flavor types?), with Linux (CentOS, Debian, Fedora, Ubuntu) and Windows Server 2008 operating system images.

Platform-as-a-Service (PaaS)

Platform as a Service offers an advanced integrated environment for building, testing, and deploying custom applications. This service model involves outsourcing basic infrastructure and platforms (such virtualized Operating System, IDE and runtime environment). As we indicated earlier, since the service model is layered architecture, customers using platform services are also use the infrastructure services. Customers are required only to come with their own application services. The service provider is responsible for administering the data center infrastructures, Operating System and runtime environment. Some examples of world-class PaaS Services include [16]:

Google App Engine: is designed for distributed web applications and developers using Java, Python, PHP and Go. The Java environment supports other languages that make use of the JRE and there is a SDK for each of the four main supported languages as well as a plugin for Eclipse. The PaaS offers managed infrastructure and runtime environments that are guaranteed to scale, but only if the applications fit the restrictions of Google App Engine. The Data store, a

transactional, schema-less data store based on key-value pairs, handles the complex management of data that's accessible to multiple machine instances.

Engine Yard: is designed for web application developers using Ruby on Rails, PHP and Node.js who want to take advantage of cloud computing without the operations management responsibility. Engine Yard provides a set of services on top of Amazon AWS. In fact, Engine Yard runs its platform in the Amazon cloud, so the value of the PaaS rests more with orchestration and management than with providing software components. Engine Yard takes care of key operations tasks such as performing backups, managing snapshots, managing clusters, administering databases and load balancing.

Windows Azure: With Windows Azure, Microsoft is blurring the lines between IaaS and PaaS. Windows Azure Cloud Services supports .NET, Node.js, PHP, Python, Java and Ruby. In addition to software development kits, developers can use Visual Studio for creating and deploying applications. Developers can choose between a SQL Database, Tables and Blobs when it comes to persistent storage. Applications are administered through the Windows Azure dashboard or through a command line interface.

Amazon Web Services (AWS): Although Amazon Web Services is primarily an IaaS, many of the services available in AWS are comparable to PaaS offerings. we can use the platform services available in AWS without having to create or maintain our own application servers. AWS readily supports Java, Python, Ruby, Perl and other languages. Oracle, MySQL and SQL Server can be set up and managed, but AWS offers RDS web service as well, which eliminates database administration tasks. Developers can take advantage of Amazon Elastic Beanstalk for automatic load balancing, auto-scaling, and application health monitoring.

mBaaS Providers: Mobile App Development Platforms: backend platforms for mobile application development, known as mobile backend as a service (mBaaS), can be a great productivity boost. mBaaS is a specialized type of PaaS with an emphasis on supporting mobile app developers. It can relieve the need to code common functions speeding up the development process. While some IaaS/PaaS providers, like Google and Microsoft, offer mBaaS functions, there are many that are pure-play mBaaS vendors. Typical mBaaS services that you can expect from them include APIs for user management, push notifications, social network integration and cloud storage.

Software-as-a-Service (SaaS)

Also referred to as "software on demand," this service model involves outsourcing the infrastructure, platform, and software/applications. Typically, these services are available to the customer for a fee, pay-as-you-go, or a no charge model. The customer accesses the applications over the internet. The customer seems like using only the software online but actually he/she uses the platform where the software is running on, and the hardware and virtualized resources. That means SaaS user uses all resources and services of cloud without having any resource at hand. The only thing the client needs to have is an end point device may it be desktop/laptop PC or mobile phones to get access from the services provided through thin or thick clients [14]. Since the main focus of this research revolves around SaaS, a detail discussion was made in the upcoming sections of the chapter. Below shows some list of world-class SaaS vendors:

Salesforce: the founder of SaaS industry, this American-based company is best known for its Customer Relationship Management (CRM) software, although it also offers other services, like PaaS (Platform as a Service), AppExchange (a market for cloud computing applications), or Work.com, a social performance management platform. They also offer a SOAP/REST Web service API that enables integration with other systems [17].

NetSuite: the well-known American software company offers SaaS integrated business management software that includes software for ERP (Enterprise Resource Planning), accounting, order management, inventory, CRM, PSA (Professional Services Automation) and e-commerce applications [18].

Concur Technologies: Provides travel and expense management solutions from its Washington-based headquarters in the US. Concur's SaaS solutions include **Concur Travel & Expense**, a travel and expense management solution for companies, and **TripIt**, a mobile travel organizer for individuals [19].

Cornerstone OnDemand: is another American-based software company that provides cloud-based talent management solutions to more than 12 million users in 190 countries worldwide. All of its products are SaaS, and include from learning management systems to performance management tools to talent acquisition management and tracking solutions [20].

2.3.3. Enabling Technologies

Cloud computing is emergent based on years achievement on Grid computing, Virtualization, Utility computing, Web computing and related technologies [2]. It is a coordination of different pre-existing technologies working in a seamless manner to provide services to end users. Apart from the functional needs of high speed low cost and scalable computing there are some technological forces which paved the way for the evolution of cloud computing. These technological enablers are [21]:

Virtualization Technology

Virtualization of hardware and software resources plays a prominent role in the development of Cloud Computing environment. Virtualization traces its roots back in the 1960's where it was used in Mainframe computers to divide resources for different applications [21]**.** Since Cloud computing is all about providing computing resources as utility services and as shared infrastructures for multiple end users, virtualization should be the first thing to think of and in fact, it is impossible to enable cloud computing technology without virtualization. Virtualization is a technique of logically divide a single physical machine into multiple virtual machines to enable sharing of the machine resources for various end users or vice versa (i.e. making multiple physical machines scattered in different areas to work as a single virtual devices ; the concept of cluster computing).

Virtualization gives user the illusion of full access of system resources which in fact may be shared by multiple users. Virtualization achieves this illusion by separating hardware from the operating systems. It installs an abstraction layer known as Hypervisor or Virtual Machine Monitor (VMM) between hardware and operating system which emulates the set of operating system hardware and user level instruction. Hypervisor allows multiple operating systems (Virtual machines) to run on same hardware. It provides system level multi-tenancy by allowing a number of users to use the hardware resources as a whole.

Web Services

The W3C defines a Web service as: “a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards”

Web services provide interoperability in the heterogeneous World Wide Web environment. Web services are programming language and platform independent information exchange systems employing XML language for message (request/response) transfer. These Web Services uses the XML technology over the HTTP protocol for communication between client and server. Web services framework like SOAP/WSDL and REST API are used in cloud environment to invoke almost all web applications.

Grid Computing

It is a form of distributed computing that implements a virtual supercomputer made up of a cluster of networked or internetworked computers acting in unison to perform very large tasks. Thus grid computing offers to cloud the capabilities for resource sharing, heterogeneity and ability to de-centralize resource control [22].

Cluster Computing

A cluster is a set of multiple interconnected computers. Classifying a set of computer as a cluster requires software that makes that computers work together. Couple of very reason for clustering is performance and high availability, which means a fault tolerant hard and software configuration. Performance clustering is a natural way to add performance if one node configuration is not enough. High availability configuration adds reliability by avoiding a single point failure. These configurations are also used together in an active cluster configuration [22].

Software Technology Innovations

Key players of cloud computing have innovated new technologies to exploit benefits of cloud computing. For example Google has its proprietary MapReduce framework which can be employed in parallel and distributed environment to process unstructured data [22]. Bigtable a storage management system is innovated to store petabytes of data. Hadoop is an open source implementation of MapReduce framework built on top of Hadoop file system. Cloud computing uses this framework to process huge amount of data in cloud clusters. For example Hadoop is used by Yahoo, Facebook, LinkedIn, and Netflix in their cloud environment.

2.4. Cloud computing as a Small business enterprise solution

CC is all about cost savings in using IT solutions as a utility for an organization, may it be large or small enterprise. It provides a convincing opportunity for organizations to outsource their ICT related service to a third party provider. CC model is delivered as pay-per-use service oriented manner to end users. This converts computing power into public utility such as water and electricity [23]. SMEs normally have not a large ICT department or budget, so they are not likely to have access sophisticated IT architectures and supporting tools. CC appears to be the solution to many of their business requirements, while at the same time avoiding conventional IT maintenance costs. It allows them to improve IT support for their commercial activities and keeps pace with new technologies. For the SMEs defined in the context of developing countries, these companies have a far limited initial investment capital which is basically targeted to direct business investment, adopting IT solution is very hard to achieve since every IT resource, may it be hardware, software, networking is unaffordable for them. In this regard CC seems to become the default computing solution and initially designed to alleviate this IT cost barriers to business success. From the standpoint of SMEs, when compared to the traditional IT systems, the cloud, by nature, provides major advantage such as lowest upfront infrastructure, location independent computing culture, low cost for periodic use, ease of management, scalability, and rapid innovation. From customers point of view the CC service delivery model brings capital expenditure reduction, increased IT agility, faster return on investment, removal of barriers to entry and a more resilient and robust infrastructure, leading to better business continuity. CC technology and services generate promising opportunities for SMEs to collaborate and create new competitive advantage in the current digital business context [24]. The attractiveness of CC

lies in its ability to show SME entrepreneurs immediate cost savings, increased productivity and improved responsiveness to the business, by incorporating cloud infrastructure as part of their IT strategy. According to [25], CC can act as a key enabler for future innovation, adding value and enhancing growth, wealth and employment in the global economy, thus creating healthy and sustainable societies. Understanding the big necessity of adopting cloud services to SMEs, the next task is what kind of service model and deployment model is suitable and should be selected from. This requires analysis of the characteristics and features of this deployment and service models as well as the characteristics of the SME which needs to migrate its IT services into the cloud. Accordingly different authors made a comparison among the different deployment and service models of the cloud and the behavior of the requirement of the business.

For example, according to [26],

IaaS abstracts the underlying infrastructure and data center capabilities so that consumers no longer have to rack and stack hardware, power and cool data centers, and procure hardware.

PaaS takes us one level higher in the stack and abstracts operating system, database, application server, and programming language. Consumers using PaaS can focus on building software on top of the platform and no longer have to worry about installing, managing, and patching LAMP stacks or Windows operating systems. PaaS also takes care of scaling, failover, and many other technical design considerations so that developers can focus on business applications and less on the underlying IT "plumbing".

SaaS is the ultimate level of abstraction. With SaaS, the entire application or service is delivered over the web through a browser and or via an API. In this service model, the consumer only needs to focus on administering users to the system.

SaaS is very common for non-core competency type applications like customer relationship management (CRM), human resources applications, and financial and accounting applications. Many companies are now going away from the legacy model of shipping software to clients or delivering software internally over the internal network to a SaaS model where the software is available 24 by 7 over the internet. In this model, software is updated in one place and immediately available to end users as opposed to the old ship and upgrade method of the past.

Hence, the first decision to choose the service model lies on whether it is feasible to buy the software or renting it. Should we write the code ourselves or pay for a SaaS solution that provides the functionality on demand? If the service is not a core competency, SaaS is usually a very good alternative to building as long as the service is affordable, mature, and meets the business requirements.

The PaaS vs. IaaS decision typically is determined by the performance and scalability requirements of the application. PaaS solutions have limitations on their ability to achieve very high scale due to the fact that these platforms must provide auto scaling and failover capabilities for all tenants of the platform. Besides this technical consideration, choosing PaaS or IaaS depends on, whether the SME can manage as well as administer the other high level resources with its own. For example, when we say it's feasible to outsource the IaaS components only, we are saying that the SME organization can manage the PaaS and SaaS by itself. Now, let's come to our context. As we have discussed the scenario from the beginning of this research, SMEs in Ethiopia are very low investment powered organizations and so they have no any computing resources (like hardware, software, and networking). In this scenario, the fact discussed in the paragraphs above, outsourcing all IT related service, which mainly referred as SaaS is feasible solution. From the deployment models perspective, since the service needs to be provided by a third party provider which is dedicated only to provide IT solutions to its customers, it's most likely be a public cloud. According to the given scenario, SMEs will not have their own private, community or hybrid cloud. Taking up a case study of public SaaS application system to be delivered by third party providers to SMEs, we come to our target to study issues and challenges of developing these kind of SaaS-based application that can serve a large number of customers (or tenants). The next section of this chapter discusses Software-as-a-Service in detail, issues and challenges of having, accessing, managing and developing it.

2.4.1. Evolution of Software Systems

For a long period, [27] the first Information Systems were projected to be installed and run on self-contained and autonomous platforms, independent from other systems. As a first and accepted approach to market these systems, software was distributed in a product-based licensing manner. In time, with the

evolution of ICT, new platforms arose that allowed for automatic information exchange between various systems. Among other things, this allowed for the development of new kinds of software architectures (i.e. client-server, multi-tier architectures), that allowed even further optimization of business efficiency. As such, many organizations adopted these new technologies, building enterprise networks to broadcast information quickly and effectively between various workstations and implementing enterprise Information Systems to harness these technologies. While larger enterprises were able to develop, deploy and maintain their personal Information Systems, smaller enterprises struggled to do so in their own.

Due to this barrier, the Service Provider family of models – Application Service Provider (ASP) business model – emerged, providing customers with outsourced IT Services. This model provides customers specially those involved with small business operations to outsource their IT related services. In this kind of model, an application is deployed on the remote server independently to other services and application and then can be accessed through the network (either LAN or WAN). [27]Application virtualization plays an important role which allows every application can run independently in a single physical machine. Web hosting and application hosting can be taken as the most important examples of ASP based service model. [28] Although ASP application could help to reduce users' initial investment on software and hardware, but from technical point of view, each application tenant had to have their own application instance, which must be developed according to individual demand respectively by ASP vendors. So each customer had a database instance, a Web site or virtual directory. As a result, the system scalability incurred greater restrictions and more cost of maintenance and upgrades were needed.

Finally as a result of the drawbacks found in ASP model, SaaS-based model has been discovered, where application service is targeted a larger customer base through the internet. In this model the application is designed as a single instance interface, single database and single instance data structure. This single instance service can be used by multiple users (tenants, multi-tenant aware architecture), which in turn improves the scalability and allows low cost of maintenance and upgrades. Customer uses the SaaS services installed in the cloud computing platform in a pay-per use licensing scheme rather than purchasing, installing, upgrading and

maintaining it. Google services, salesforce.com, Facebook, flikr.com, Microsoft Live Mesh are the best examples of this kind of software services on the internet today [3]. João Samuel Nunes Lopes [27] discussed and analyzed each of the three evolutional software systems as summarized below.

Software-as-a-Product (SaaP) - legacy system

Table 2-1 Features of Legacy software Systems

Operational characteristics ▪ Software is developed as a product rather than a service ▪ Development cost is high ▪ Can't produce steady income revenue stream	**Payment model** ▪ Customers pay a flat fee for permanent product usage
Integration ▪ Software is designed and developed to be installed and run on customers' end-point devices ▪ Each user works on a single app. copy in equipment ▪ IT teams have to install the software solution for every device ▪ No inter-device interaction; applications deployed via external drives	**Intellectual protection** ▪ Purchasing copyrighted SW or acquiring open source product
Maintenance, Support & updates ▪ IT teams must sustain application functionality in every device with an application installation ▪ Customization delivered through new patches and updates	**Service Level Agreements (SLA)** ▪ Not relevant

Application access ▪ Self-contained programs ▪ Developed to be executed alone and are directly accessed by the end user	**Market target** ▪ Usually targets niche markets (specific group of clients)
Licensing ▪ Vendor sell products by selling licensing agreements ▪ "One application per user or per device" licensing ▪ Cost increases as user or device increases ▪ Hardware vending agreements	**Marketing and product promotion** ▪ Customers granting trial period to install and evaluate the app. To decide purchasing ▪ Promotion could be word-of-mouth, internet, workgroups, consulting groups
	Unresolved Issues ▪ Hard to apply in large scale environments ▪ Costly to maintain ▪ Takes much time to be integrated and upgraded ▪ Products and licenses cannot be monitored

ASP Model

Table 2-2 Features of ASP model

Operational characteristics ▪ Separate SW possession from usage ▪ Multi-instance architecture ▪ SW solutions provided as Services ▪ Introduction of app. Virtualization	**Intellectual property protection** ▪ several protection approaches ▪ more rigorous product licensing compliance monitoring ▪ central subscription and authentication capability
Integration ▪ Core SW product installed on high-power central device such mainframes	**Service level agreements** ▪ SLAs negotiated during service contract ▪ Negotiations generally entail financial

▪ Applications are separated from hardware and software environment where they need to be run on ▪ And accessed by users' end device ▪ Needs networking to access the application ▪ The network may be corporate LAN or internet	penalties and the right to cease service provision when one of the parties disrespected the agreement ▪ SLAs include the definition of payment model, service performance assurances and problem management assurances
Maintenance, Support and Updates ▪ Clients pay regular fee	**Market targets** ▪ ASP target specific markets and can be classified into either of local or consumer ASPs (deliver apps to individuals. E.g. email service), functional or specialist ASPs (deliver single app. To organizations. E.g. Credit Card), vertical market ASPs (deliver a solution package to a specific organization. E.g. Inventory management solutions) or enterprise ASPs (delivers a wide variety of applications to organizations)
Application access ▪ Application delivered based on LAN or internet networks ▪ Data can be accessed locally and securely through client application installed on client devices	**Marketing and product promotion** ▪ New users can be easily in minutes ▪ Client can experiment the service before subscribing it ▪ Promotion is done through word-of-mouth, internet, workgroups etc. inadequate servicing can lead to business liabilities
Licensing ▪ Application purchased in regular bases like legacy models. However, a single license	**Unresolved issues** ▪ Security greatly depends on the provider ▪ Customers rely on providers to supply critical business functions, thus limiting

can be virtually shared by multiple users. i.e. customer needs to buy a single license then it can access the application on multiple devices ▪ Clients don't have to but the product license directly, instead they pay for the service	control of those functions ▪ Provider faces difficulty answering to customer problems and provides application customizations.

SaaS Model

Table 2-3 Features of SaaS model

Operational characteristics	**Payment model**
▪ Successor of ASP model ▪ One application instance used by multiple clients (Multi-tenancy architecture) ▪ Used to achieve better economy of scale while practicing more competitive prices ▪ Following web 2.0 technology, SaaS applications can interact with other apps on the internet and can access other SaaS services ▪ This allows different services integrate to provide larger and complex services	▪ Service has low cost of entry or less expensive depending on the SLA agreed ▪ Various service plans – Fixed-fee, subscription-based, usage-based, transaction-based, value-based models
Integration	**Intellectual protection**
▪ Unlike ASP, it doesn't install any code outside the provider ▪ Service accessed through the internet via web browsers	▪ Several SW protection approaches ▪ Customer can decide between copyrighted vs. open source solution ▪ Continuous investment on the development of the product- more effective customer oriented applications

Maintenance, Support & updates ▪ Updating is a provider's internal task ▪ Clients pay a regular for that ▪ SaaS apps provide simple personal interfaces so that client can customize and configure as needed	**Service Level Agreements (SLA)** ▪ Similar with ASP model
Application access ▪ Like ASP apps are delivered through the network but in this case the network is always the internet ▪ Services are accessed through common browsers ▪ Nothing gets shipped or installed ▪ Users need only to run a browser, load provider's link and access login interface	**Market target** ▪ Primary market targets are standard consumers and Small and Medium enterprises (SMEs) ▪ SaaS providers classified as one of the following businesses; Personal providers (email providers), Collaborative market providers (inventory management), enterprise providers (CRM, ERM, Tax Analysis etc.)
Licensing ▪ Clients pay for a service they have used for. ▪ Clients don't have obtain directly, but pay for the service delivered by the provider ▪ Customers can opt to provide their hired services to intermediary affiliates	**Marketing and product promotion** ▪ SaaS benefits from marketing advantages the internet provides ▪ Services can be promoted easily and offered to anyone on a trial bases ▪ Promotion can be through word-of-mouth and internet
	Unresolved Issues ▪ Security issues, customers delegate their IT systems to a third party, so their security and privacy greatly dependent on their provider ▪ Hard to implement multi-tenancy. Multi-tenancy is the default approach to deliver SaaS applications, but it's hard to implement it.

2.4.2. Software-as-a-Service

Designed to leverage the benefits brought by economy of scale, SaaS is about delivering software functionalities to a big group of clients over a Web with one single instance of software application running on top of a multi-tenancy platform [29]. The Gartner IT glossary defines SaaS as software delivered and managed remotely by one or more providers. The provider provides software based on one set of common code and data definition that is consumed in a one-to-many model by all contracted consumers at any time anywhere on pay-for-use bases as a subscription based on use metrics. Java played an important role since in 2000s by proclaiming itself as a language that would run on any platform [30]. While this is true, the "As a Service" model has abstracted this concept by delivering software using the cloud. Instead of having to deal with the downtime associated with locally installing an application, SaaS allows users to access any application published to their account as long as they have a network connection. SaaS is one of the service delivery models where it will change the way people build, sell, buy and use software. SaaS applications are known as Web-based software, on-demand software or hosted software. Cloud provider maintains the application its security, availability and performance. SaaS cloud computing delivers end user desired application through the internet to thousands of customers using a multitenant architecture. On the other side, cloud user can run application or software with no upfront costs or investment in database, servers and software licensing. With the advent of Web 2.0 & faster HTML5 standards, graphically rich applications can be run smoothly and in high speed just like running our software on our own personal computers [31].

SaaS System Architecture

Much like any other software, Software as a Service can also take advantage of Service-Oriented Architecture to enable software applications to communicate with each other. Each software service can act as a service provider, exposing its functionality to other applications via public brokers, and can also act as a service requester, incorporating data and functionality from other

services. It is important to understand that the SaaS methodology requires system architecture capable of supporting peak usage demands and the ability to process large numbers of transactions in a secure and reliable environment.

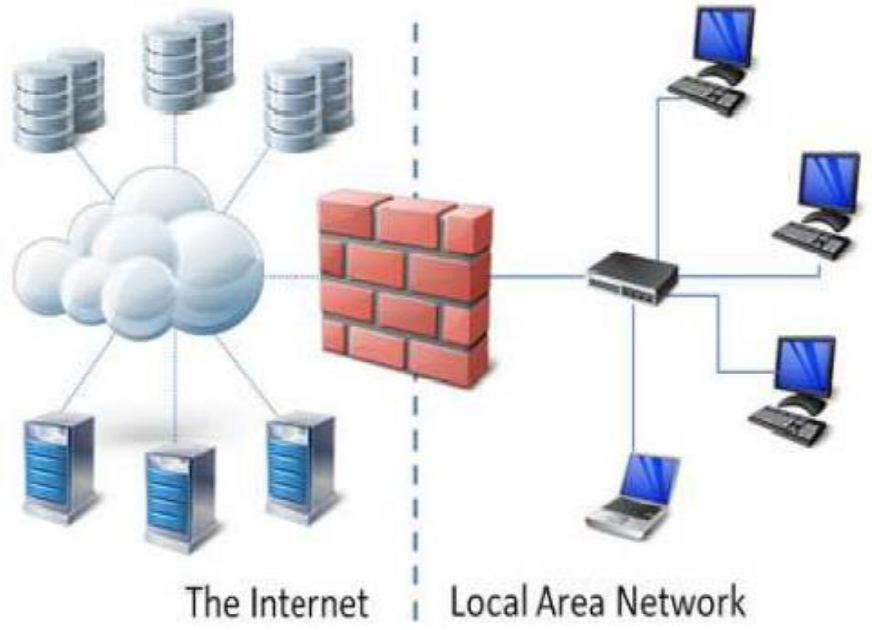

Figure 2-3 SaaS system architecture [31]

The software would need to meet certain criteria's to work on a model such as this. The application would need to be well architected to sustain and provide the scalability, ease of use of the traditional desktop applications. There are three key points which would differentiate a successful SaaS application from an unsuccessful SaaS application:

- **Scalability:** Scaling the application means maximizing concurrency and using application resources more efficiently-for example, optimizing locking duration, statelessness, sharing pooled resources such as threads and network connections, caching reference data, and partitioning large databases.

- **Multi-tenant efficient:** Multi-tenancy may be the most significant paradigm shift that an architect accustomed to designing isolated, single-tenant applications has to make. For example, when a user at one company accesses customer information by using a CRM application service, the application instance that the user connects to may be accommodating users from dozens, or even hundreds, of other companies-all completely

abstracted to any of the users. This requires an architecture that maximizes the sharing of resources across tenants, but that is still able to differentiate data belonging to different customers.

- **Configurable:** if a single application instance on a single server has to accommodate users from several different companies at once, you can't simply write custom code to customize the end-user experience. Anything you do to customize the application for one customer will change the application for other customers as well. Instead of customizing the application in the traditional sense, then, each customer uses metadata to configure the way the application appears and behaves for its users. The challenge for the SaaS architect is to ensure that the task of configuring applications is simple and easy for the customers, without incurring extra development or operation costs for each configuration.

There can be four ways of hosting an application on the SaaS architecture. These are also called maturity models of SaaS:

Ad-hoc/Custom: It is similar to the traditional application service provider (ASP) model of software delivery. Each customer has its own customized version of the hosted application, and runs its own instance of the application on the host's servers.

Configurable: The vendor hosts a separate instance of the application for each customer (or tenant). Unlike the previous one, each instance is individually customized for the tenant, at this level, all instances use the same code implementation, and the vendor meets customers' needs by providing detailed configuration options that allow the customer to change how the application looks and behaves to its users. Despite being identical to one another at the code level, each instance remains wholly isolated from all the others.

Configurable, Multi-tenant-efficient: The vendor runs a single instance that serves every customer, with configurable metadata providing a unique user experience and feature set for each one. Authorization and security policies ensure that each customer's data is kept separate from that of other customers; and, from the end user's perspective, there is no indication that the

application instance is being shared among multiple tenants. This approach eliminates the need to provide server space for as many instances as the vendor has customers, allowing for much more efficient use of computing resources than the second level, which translates directly to lower costs. A significant disadvantage of this approach is that the scalability of the application is limited.

Scalable, configurable, Multi-tenant-efficient: The vendor hosts multiple customers on a load-balanced farm of identical instances, with each customer's data kept separate, and with configurable metadata providing a unique user experience and feature set for each customer.

2.4.3. Multi-tenant aware Software Architecture

Multi-tenancy is the core of SaaS; it is the ability to use a single-instance of the application hosted by a provider to serve multiple clients (tenants). Multi-tenancy is different from multi-instance architecture where separated instances of the same software are hosted on different servers to serve different tenants.

Figure 2-4 Multi-instance vs. Multi-tenant Architecture [8]

Multi-tenant architecture introduces the concept of single application which can be used for multiple customers. Each customer is called a tenant. Multi-tenant architecture runs the application on the infrastructure of the service vendor, and multiple tenants are then allowed to access the same instance of the application with customized configurations. Optimized use of

hardware resources, highly customizable and extensible application is one of the major concerns [32] in multi-tenancy.

Multi-tenancy is the default paradigm for SaaS-based software application delivery. To bring quality of service in terms of cost, security, data privacy among the clients (tenants), researchers proposed various implementations of enabling multi-tenancy architecture into SaaS. Among this, most of them agreed on three approaches. Based on database deployment strategies, they have classified it into "separate database, separate schema", "shared database, shared schema" and "shared database, shared schema". Microsoft's Frederick Chong et al [33] explained each of them as follows.

Separate Databases, separate schema

Involves storing tenant data in separate databases, computing resources and application code are generally shared between all the tenants on a server, but each tenant has its own set of data that remains logically isolated from data that belongs to all other tenants. Metadata associates each database with the correct tenant, and database security prevents any tenant from accidentally or maliciously accessing other tenants' data.

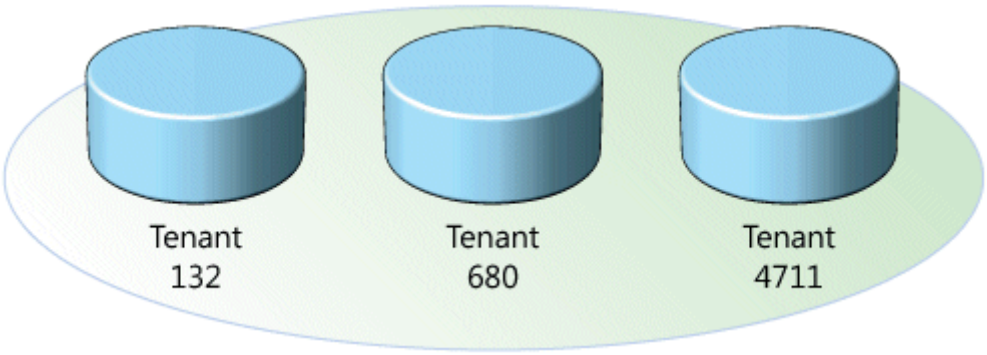

Figure 2-5 Separate database data architecture [33]

Giving each tenant its own database makes it easy to extend the application's data model (discussed later) to meet tenants' individual needs and restoring a tenant's data from backups in the event of a failure is a relatively simple procedure. Unfortunately, this approach tends to lead

to higher costs for maintaining equipment and backing up tenant data. Hardware costs are also higher than they are under alternative approaches, as the number of tenants that can be housed on a given database server is limited by the number of databases that the server can support. (Using auto close to unload databases from memory when there are no active connections can make an application more scalable by increasing the number of databases each server can support.)

Separating tenant data into individual databases is the "premium" approach, and the relatively high hardware and maintenance requirements and costs make it appropriate for customers that are willing to pay extra for added security and customizability. For example, customers in fields such as banking or medical records management often have very strong data isolation requirements, and may not even consider an application that does not supply each tenant with its own individual database.

Shared database, Separate schema

Another approach involves housing multiple tenants in the same database, with each tenant having its own set of tables that are grouped into a schema created specifically for the tenant.

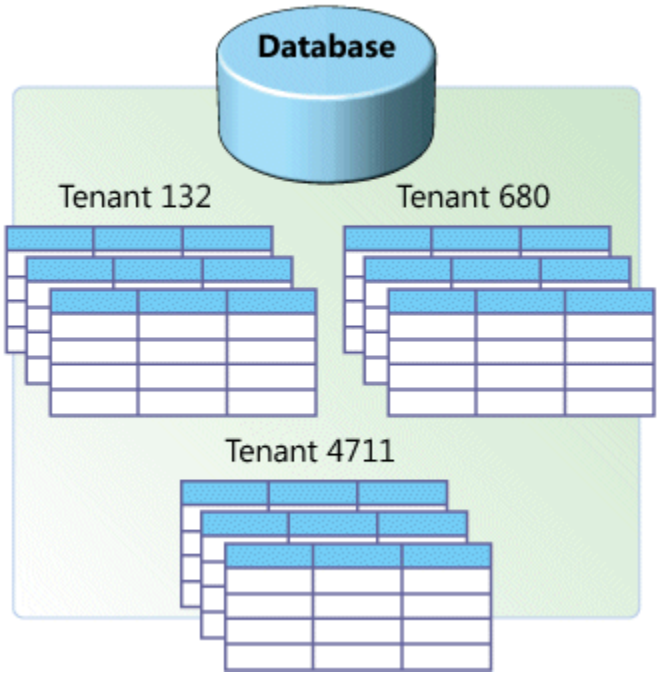

Figure 2-6 Shared database, separate schema [33]

When a customer first subscribes to the service, the provisioning subsystem creates a discrete set of tables for the tenant and associates it with the tenant's own schema.

A significant drawback of the separate-schema approach is that tenant data is harder to restore in the event of a failure. If each tenant has its own database, restoring a single tenant's data means simply restoring the database from the most recent backup. With a separate-schema application, restoring the entire database would mean overwriting the data of every tenant on the same database with backup data, regardless of whether each one has experienced any loss or not. Therefore, to restore a single customer's data, the database administrator may have to restore the database to a temporary server, and then import the customer's tables into the production server—a complicated and potentially time-consuming task.

The separate schema approach is appropriate for applications that use a relatively small number of database tables, on the order of about 100 tables per tenant or fewer. This approach can typically accommodate more tenants per server than the separate-database approach can, so you can offer the application at a lower cost, as long as your customers will accept having their data co-located with that of other tenants.

Shared database, Shared Schema

A third approach involves using the same database and the same set of tables to host multiple tenants' data. A given table can include records from multiple tenants stored in any order; a Tenant ID column associates every record with the appropriate tenant.

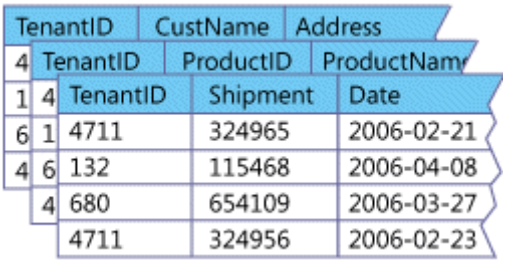

Figure 2-7 Shared database, shared schema [33]

Of the three approaches explained here, the shared schema approach has the lowest hardware and backup costs, because it allows you to serve the largest number of tenants per database server. However, because multiple tenants share the same database tables, this approach may incur additional development effort in the area of security, to ensure that tenants can never access other tenants' data, even in the event of unexpected bugs or attacks.

The procedure for restoring data for a tenant is similar to that for the shared-schema approach, with the additional complication that individual rows in the production database must be deleted and then reinserted from the temporary database. If there are a very large number of rows in the affected tables, this can cause performance to suffer noticeably for all the tenants that the database services.

The shared-schema approach is appropriate when it is important that the application be capable of serving a large number of tenants with a small number of servers, and prospective customers are willing to surrender data isolation in exchange for the lower costs that this approach makes possible.

Each of the three approaches described above offers its own set of benefits and tradeoffs that make it an appropriate model to follow in some cases and not in others, as determined by a number of business and technical considerations. Some of these considerations are economic, security, tenant as well as regulatory considerations.

CHAPTER THREE

3. Related Works

3.1. Overview

Different researches have been conducted by different authors to study the challenges in adoption of SaaS in general and requirement variability issues in particular. Most of these works mainly focus on migration of a single tenant web application into multi-tenant enabled applications as well as configuration-customization aspects of a multi-tenant enabled software and none of which, as to our understanding, didn't touch design approach of SaaS based application from implementation point of view. In addition, some of which provide different level of abstraction where we can do application tailoring activities. That is, presentation layer, application/business logic layer and database abstraction for multitenant architecture. They see each of the modules independently and they didn't show any linkages how one module affects the other. With respect to this, our work focuses on flow of how each of the layers affect each other, design a programming model from implementation point of view. Among these literatures, some of them have been discussed in the following sections.

3.2. Native Multi-tenancy application development and management

Chang Jie Guo et al. [29] explored the requirements and challenges of a native multi-tenancy pattern to develop and manage an application. This paper identifies two ways of enabling multi-tenancy framework which are known as "multiple instances multi-tenancy mode and Native Multi-tenancy" as illustrated in figure 3-1**,** made a comparison between them in terms of cost, security, scalability, administration to provide quality of service.

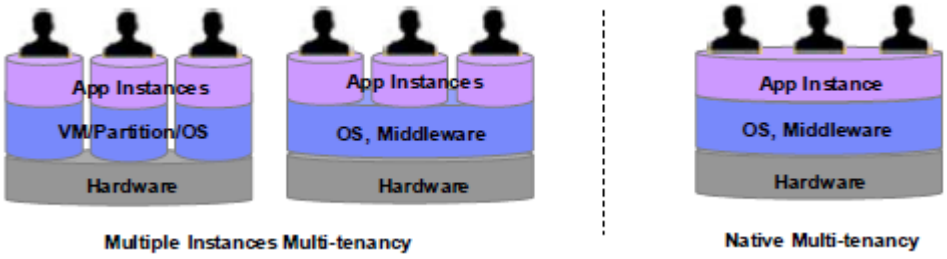

Figure 3-1 two kinds of multi-tenancy patterns [29]

According to their analysis, they recommended the Native multi-tenancy enablement approach and developed a framework which helps application developers how to build and deploy the kind of a native multi-tenant application service. However, their analysis focuses on major challenges of isolating services to enhance quality of service among many tenants, in terms of security, performance, availability and administration. Customization and configuration is one part among the components of the framework and it suggests only a simple parameter based configuration where predefined isolation points are identified and defined in the configuration file. Here the intention of the paper is, providing quality of service by bringing enhanced security, performance and scalability options among different tenant requirements and doesn't deal in detail, the requirement variability challenges and didn't suggest any on tailoring application services which is the integral part of enabling a multi-tenant aware software architecture.

3.3. Configuration and Customization perspectives of SaaS offerings

Wei Sun et al. [4] analyzed different perspectives and implementation varieties of configuration and customization in a SaaS based service offering. In this research they tried to identify complexity requirements of configuration and customization and provide a comparison of using

either of configuration or customization. According to the paper, configuration can be performed through from fully standardized offering up to pre-defined parameterized tools and does not involve source code changing. However, customization involves SaaS application source code changes to create functionality that is beyond the configurable limit.

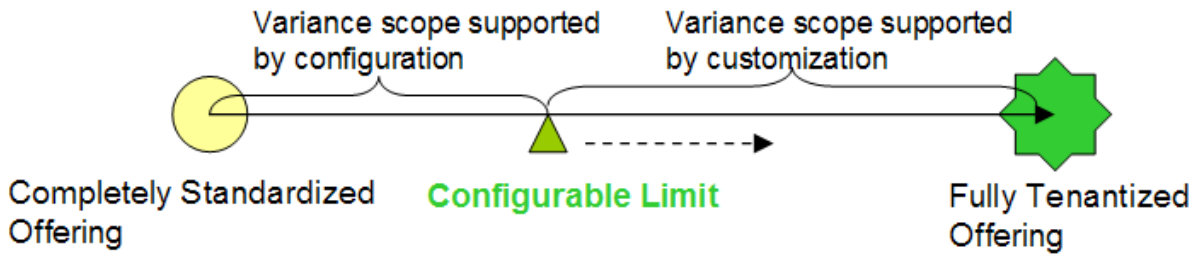

Figure 3-2 Configuration vs. Customization

Accordingly, they don't recommend a vendor to approach fully customization since it means high development and deployment cost in terms of high profile human power as well as hardware resource costs.

They also argued that there are different requirement level for configuration and customization of the application. The more complex become the application software, the more requirements for configuration and customization. Accordingly they developed a competency model and a methodology framework to help SaaS vendors to plan and evaluate their capabilities and strategies for configuration and customization. This competency models and the methodology framework have been demonstrated in the following figures.

Table 3-1 SaaS Configuration competency model [4]

Level of Competency	Description	Approach	Variance Level Supported
Entry	Highly standardized offering without any configuration and customization support	Well design the functionalities as standardized offering to cover targeted customers	None
Aware	Relatively standardized offering with pre-defined variance points	Offer parameterized configuration	Low
Capable	Relatively standardized offering with user defined configuration	Offer self serve configuration tool to empower customers	Medium
Mature	Base offering with programable enviroment to enable user preferred customization	Offer scripting based programming for very flexible customization	High
World Class	Offer a platform supported by programming model and tools to enable extremely strong customization or even new application development	Offer well defined programming model and tools to enable extensive customization and new application development	Extremely High

Completely Standardized Offering

Fully Tenantized Offering

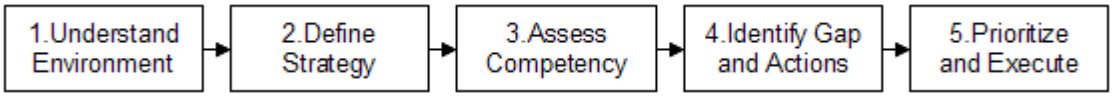

Figure 3-3 A framework to plan and execute configuration and customization strategy [4]

This research can help us decide our competency level of configuration and customization requirement with respect to our case. But it simply shows competency model to decide and does not cover a framework that allow configuration or customization of a fully multi-tenant aware SaaS offering. Our research tries to solve this problem.

3.4. Migrating a traditional web application into Multitenant SaaS

Hong Cai et el. [9] propose an end-to-end methodology for migration based on identifying isolation points between tenants, such as UI, constant fields, and configuration files. Once these points are identified, the development team can modify these points to be configurable per tenant using a toolkit which was built for the purpose of their work. This approach addresses only the look-and-feel and basic configuration options between tenants; however, business logic and data-model are not covered.

Eyad Saleh et el. [8], also proposed a complete framework for migration based on works previously done by Hong Cai et el. [9] and other authors. In this proposal, they attempted to

provide a complete framework which provides tenant specific customization and configuration at different layers of the software such as presentation layer, business logic layer and database architecture layer. In this research, also, they attempted to show the sequential workflow of how the transaction takes place starting from authentication and authorization up to backend application configuration. But their architecture should be modified in some parts of it. For instance, the architecture recommends separate database instances as shown on figure 3-4, which is criticized by other authors. Instead, shared database and shared schema architecture is the most effective in terms of cost, scalability, resistance to update anomalies and it fulfills a true multitenant framework. One more thing, they didn't show how one layer can be affected by configuration-customization of the other layer. This architecture doesn't allow a highly customizable software application such as dynamically adding tenant specific form fields and, based on that, altering the business logic computations and changing query optimization on the data access layer. In contrast to this, our proposed approach enables to allow a fully customizable application which allows clients to configure their own environment. Based on the working environment the client set, the system can generate source code and optimized query processing on the data access layer as well as updates the database schema.

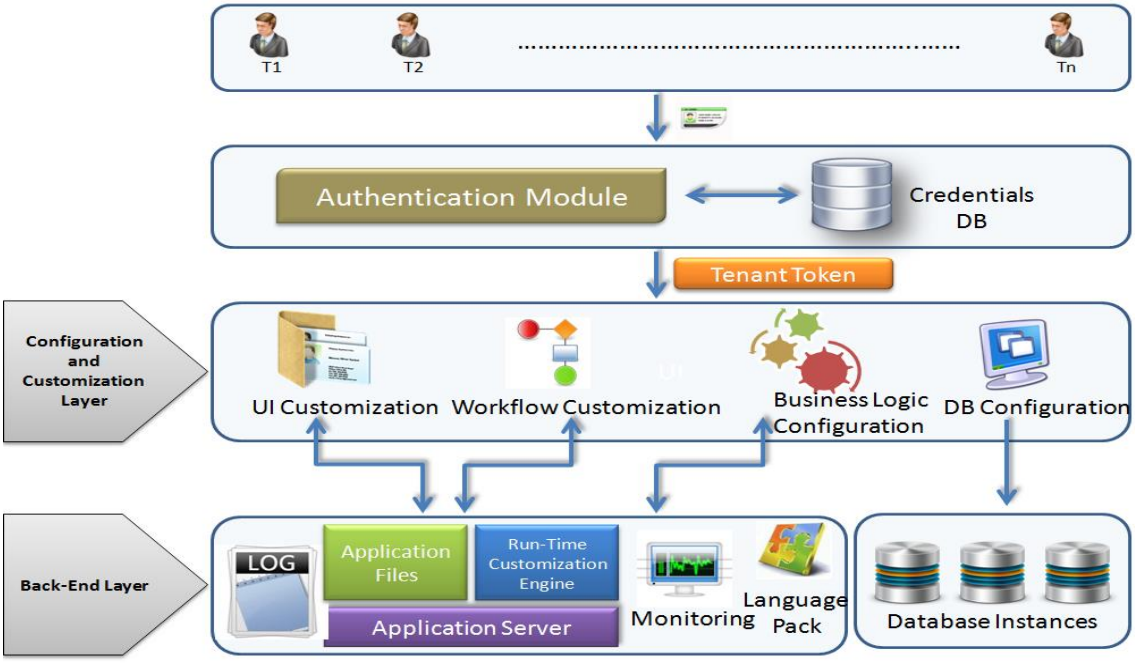

Figure 3-4 A framework for migrating into Multi-tenancy architecture [4]

CHAPTER FOUR

4. Research Design and Methodology

Overview

Selecting suitable research design and methodology, based on the nature of the research, should be performed before actually starting doing a research. In this chapter, we have presented our research design and the selected research methodologies. We have covered the study area (Geographical area and population where the data collected). We also discussed the type and approach of the research being conducted, sampling techniques, data analysis as well as discussion of final analysis results. To collect the require primary and secondary data, we have conducted different data collection methods such as document analysis and interview and review of related literatures.

4.1. Study area

This research is conducted mainly in Adama city targeting SMEs as well as small and medium income business (SMBs) organizations. The study focused mainly on private businesses which start from a single owner business centers to private limited companies (companies which are owned by two or more persons), PLCs.

4.2. Nature of the study

The nature of this study is limited only by describing and identifying the possible requirements which are fundamental to search a solution for designing a customizable SaaS framework for the development of business automation systems of SMEs or SMBs. The study is descriptive in nature and has intention to conduct comparative analysis between businesses on their transaction trends and document structures of how they process their business data. It is based on conducting a case study of Inventory and sales data management system of various businesses/organizations. Since SaaS is about software sharing among multiple clients, the study investigates variability challenges and levels, of business enterprises across different sector organizations and yet has no intention to search main functional and non-functional requirements for the development of automated Inventory and sales management system.

4.3. Research Design and Approaches

Since the purpose of this study is to identify facts, taught, ideas, and feelings about the current trends of business data management and it adopts a case study, descriptive research design is appropriate for the problem at hand.

To gather the needed primary and secondary data, we have used qualitative approaches. We conducted open ended interviews and analyzed business documents of businesses focusing on Inventory and sales management practices.

4.3.1. Sampling Techniques

We defined study population at Adama city Kebele-06 SME and SMB organizations and centers. Once the target population was defined, the next task performed was taking representative samples. We used stratified random sampling techniques to select representative samples. Since all SMEs and SMBs are governed by the Federal economic and trade policy, we assume that the samples taken at Adama city can represent all population across the country.

Samples were classified based on sectors or business areas. The main reason of sector based classification is, to study existence of requirement variability; inter-sector comparison is an important

method since inter-sector businesses are expected to have more variation in business requirement and trend in data management.

We conducted stratified random sampling to select sample businesses from Medical supplies, Electronics and general trading, Ceramics, Tyre and clothing business centers in Adama city Kebele-06.

4.3.1.1. Source Population

The sampling units of the study are micro and medium business centers (SME and SMB) of Adama city Kebele-06. There are a total of approximately 8000 actively registered business organization in the city of Adama in general. From this, we are interested with studying business data management behaviors of Kebele-06.

4.3.1.2. Target Population

The target population of this study is employees of the business centers and contractor auditors of the selected business center.

4.3.1.3. Sample Size

A total of 7 respondents were involved in the actual study, 6 directly from the business organization and 1 from the auditors' representative personnel. In using the stratified method, there were steps which we have followed. These steps of stratification were:

a. Dividing the population into different strata (sector based classification).
b. Taking samples randomly from each stratum

To divide the population into different strata, sector or business area classification was used. First we have collected the interested data from Adama city administration Trade and Urban development agency as well as from the urban micro and small enterprise development agency. We found 1121 actively registered business organizations in Adama Kebele-06 from FDRE Ministry of Trade online database system.

Then business area (sectors) identified and samples taken randomly from each business area. But since our intention is to investigate whether there is a variation in business requirements among different business areas, we didn't take a sample from each and every business area. Because our aim is just showing whether there is a variation between at least two sectors, no need to investigate requirement of every business sector. If there exists even one variation, it is feasible to study design and development of a customizable SaaS based software application.

4.4. Methods of data analysis

Qualitative data was collected and analyzed using simple descriptive data analysis techniques. We have followed the following steps in analyzing of the data collected.

Coding of data under interest

Selected sample business areas were coded using three letters and two digits concatenated. The letters intended to refer to the business area of the company. Example Medical Supplies can be coded to Med01, Med02, and Med03 etc. which stand for Medical facilities distributer and wholesaler, Pharmacy and drug store respectively. Table 4-1 shows coding of sample business organizations.

Table 4-1 Company/business type and corresponding coding

Type of business	**Given code**	**Sample size**
Medical Supplies wholesale	Med01	1
Pharmacy and Drug store	Med02	1
Electronics/computer trading	Elc01	1
General trading	Gen01	1
Ceramics	Cer01	1
Tyre	Tyr01	1
Clothing	Cls01	1

a. Identifying major application services (modules)

Major application services or modules of an automated Inventory and Sales control system requirements identified. These major services are very important in studying basic functional and non-functional requirements of businesses in using this kind of developed system and their current traditional trend. So the proposed SaaS based Inventory and Sales control system assumed to have services such as "Inventory List", "Purchasing", "Sales", "Supplier management", "Customer management", "Stock and bin card management", "Balance Sheet" and "Reports". Then requirements of business analyzed whether they necessarily require each module or they need as optional support service. The following shows analysis of this service support requirements of each business area.

Table 4-2 Current Interest in Support service requirements among businesses

Service/module Name	**Med01**	**Med02**	**Elc01**	**Gen01**	**Cer01**	**Tyr01**	**Cls01**
Inventory List	Yes	Yes	No	No	No	Yes	No
Purchasing	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sales	Yes	Yes	Yes	Yes	No	Yes	No
Supplier Management	Yes	No	No	Yes	No	No	No
Stock and Bin card management	Yes	Yes	No	No	No	No	No
Balance Sheet	Yes	Yes	Yes	Yes	No	Yes	Yes
Report	Yes	Yes	Yes	No	No	No	No

b. Grouping each sample and compare them

Related business sectors were grouped and compared based on how many data attributes included in each application service and how many of which a business document of a particular business area includes and excludes according to review of their current Inventory and Sales processing documents. For example when we compare Med01 and Med02, in "purchasing"

module, total estimated attributes counted to be 21. Of these, total number of attributes relevant in both business areas found to be 15 and those that are relevant only to Med01 found to be 6. Based on this, we calculated variation as percentage of differentiated requirements. That is (6/21)*100 which is 28.6%. Table 4-3 to table 4-5 summarized such variability analysis between grouped business areas and estimated variation with major variation reasons.

Table 4-3 Attribute variability analysis between Med01 and Med02 (intra-sector comparison)

Module Name	**No of Estimated attributes**	**No of Similar attributes**	**No of Different attributes**	**Variation (%)**	**Reason**
Inventory list					
Purchasing	21	15	6	28.6	Missing attributes
Sales attachment	17	8	9	53	Missing attributes
Supplier	-	-	-	100	Module missing
Stock and bin card handling	-	-	-	100	Module missing

Table 4-4 Attribute variability analysis between Electronics and General Trading business items (Elc01 and Gen01)

Module Name	No of Estimated attributes	No of Similar attributes	No of Different attributes	Variation (%)	Reason
Inventory list	-	-	-	-	Missing of module
Purchasing	30	19	11	33.3	Missing of attributes
Sales	20	10	10	50	Missing of Attributes
Supplier	-	-	-	100	Missing of attributes
Stock and bin card handling	-	-	-	100	Missing of modules

Table 4-5 Attribute variability analysis between ceramics and Car tyre business items (Tyr01 and Cer01)

Module Name	No of Estimated attributes	No of Similar attributes	No of Different attributes	Variation (%)	Reason
Inventory list	-	-	-	100	Missing of module
Purchasing	11	8	3	27.3	Missing of attributes
Sales	11	8	3	27.3	Missing of attributes
Supplier	-	-	-	-	Missing of module
Stock and bin card handling	-	-	-	-	Missing of module

All of the above analysis conducted based on the companies' formal and informal working documents of Inventory and Sales processing. Much of the findings show variability in data and file structure which will be translated later to database schema in designing of the system.

However, variability in business processing logic and company presentation should also be treated. For this reason we have also conducted randomly chosen open ended interviews. Accordingly, we have identified that there are some variation points in different layers from the application or system point of view.

4.5. Results and discussion

In this section we have discussed and interpreted the findings found by analyzing the documents collected and the interviews conducted from the sample business companies. We have classified the discussion according to the variation points identified at different layers of the system thinking the system as three tier architecture. I.e. presentation layer, business logic layer and data access layer (database layer) [8].

A. Presentation Layer

It is obvious that different companies have their own logo, motto, and unique color patterns to represent company profile. The presentation layer refers to the User Interface (UI) components displayed on the client's computer (end point device). Different companies have their own unique interest in the look and feel of the SaaS software service. They need the system to profile their own existence. They want to display their own logo; they want every UI element (such as general layout, colors, themes, and buttons) to represent their company. So, the SaaS software should have a mechanism to support tailoring the service to suite all the clients (tenants) by enabling tenant specific UI customization.

B. Business Logic Layer

Business rules are part of the business logic layer and these rules varies from one company to another. As we have observed, the variety increases when it is applied to much heterogeneous companies. (That is, companies in different sectors). For example, Med01 classifies its customers into three levels. These are pharmacy drug vendors, Medium Clinic owners and Higher Clinic owners. Since drug should be controlled and limited in delivering to these customers, a customer should show his trade permission certificate while he wants to purchase drug. But for Gen01, everyone is a customer and no limitation for customer to purchase. In addition, since it works

with medical supplies, Med01 authenticates its buyers with appropriate certifications. That is, renewed trade permission, updated tax payment certificate and license level etc. while this is not a concern for Gen01. Gen01 also has different branches across the country and so there is additional information processing across branches. Here, for instance, product shipping can be classified into two kinds, intra-shipping and shipping while they import product, while in the case of Med01, information processing for intra-shipping transaction is not required since the company has only a single working office. With connection to this, the finance and accounting system will have also a different way of processing. One more, when we think of very small scale businesses, they will not need all modules of the Inventory Management System (IMS). For instance, take a small computer and accessories sales institution (such as Elc01), it may only need sales, purchase and balance information about its products and services and it may not care for information about where the product is stored, how many packages available, stock and bin card information, customer and supplier information etc. while a larger company needs all these.

C. Database Layer

Database layer is the major layer where data structure and schema variation exists across different SMEs. In the following sections, we have summarized some variations which exist in different working modules of sample companies. Analysis data has been presented in terms of existence and non-existence of attributes from different modules of inventory processing document such as “inventory list”, “sales”, “purchasing” and so on. In the analysis we have compared two closely related samples by using similarity and variation attributes quantified by variation percentage and we have identified the major reasons for the variations to occur. While the variation percentages and reasons discussed in the tables shown in the analysis section, some of exact variation points discussed in the table below as example. The table shows the data attributes of the specified module of a company and the shaded attributes are those missed in module of a company under comparison with.

Purchasing

Table 4-6 Purchasing (Goods Received Note) of Gen01.

▪ Supplier	▪ Item Details

▪ Delivered to Warehouse No. ▪ Debit Acc. No ▪ Credit Acc. No ▪ Date ▪ RequestNumber ▪ PurchaseOrderNumber ▪ ItemCategory	▪ SerialNo. ▪ ItemIdNumber ▪ Description ▪ PartNumber [model] ▪ Quantity ▪ UoM ▪ Unit Cost ▪ Total Cost
▪ Transported by, Received & Stored by, Approved by: …	

Table 4-7 Purchasing (Goods Receiving Note) of Med01

▪ Supplier ▪ Classification: ▪ Raw materials ▪ Spare Parts ▪ Fixed Assets ▪ Purchase requisition ▪ Suppliers Invoice No ▪ Purchase Order No. ▪ Letter of Credit No.	▪ Item Details • SerNo • Location • ItemCode • Description • UoM • Qty • UnitPrice • Total ▪ A/C • Remark ▪ Delivered by, received by approved by…

Sales Invoice

Table 4-8 Sales Invoice sheet (Med01)

▪ Shop/Store ▪ Invoice No ▪ Fs. No. ▪ Customer/buyer • Buyer's Name • Buyer's trade name • Buyer's Tin	▪ [ItemName][Description][PER UoM] ▪ BatchNo ▪ ExpiryDate ▪ ManfDate ▪ Quantity ▪ UnitPrice ▪ TotalPrice

• Buyer's VAT Reg. No.	▪
SubTotal, Tax1, GrandTotal,	
Prepared by…, Approved by…... Received by……	

Table 4-9 Sales Invoice sheet (Gen01)

▪ Fs. No, Date, Time ▪ CustomerName ▪ CustomerTradeName ▪ CustomerVatNumber ▪ CustomerTin ▪ Address[Zone, SubCity, Kebele, HouseNo]	▪ ItemDetails • SerNo • CodeNo • Description • UoM • Quantity • UnitPrice • TotalPrice
▪ AmountInWords ▪ ModeOfPayment ▪ DeliveryNo ▪ ReasonForAttachent ▪ CustomersSign	▪ MachineNumber(MRC) ▪ PreparedBy ▪ ApprovedBy ▪ Casher's Sign ▪ Saler's Sign

Based on the analysis, we have reached into the conclusion that most of the variation occurred because of missing of attributes or modules (services) and doesn't lead us to follow a complex design pattern.

CHAPTER FIVE

5. Solution design: SaaS Application Development Approach

5.1. Overview

As we have discussed so far, the main goal of this research is studying requirement variability challenges and design issues in the development of cloud based business applications (such as online inventory and stock management, HRM (human resource management), customer relationship management and finance systems etc.) targeting small and medium sized enterprises in the Ethiopian context. For this research, we have taken cloud-based online inventory and stock management system as a case study. Then data collection and analysis has been done and results discussed in the previous chapter. In this chapter, we tried to bring together inputs from the previous chapters (objective study, literature review, related works and results from data analysis) and tried to design configuration and customization framework and software architecture for SaaS based business automation systems in general and Inventory and Sales control system in particular.

5.2. Variability challenges in SaaS-based Software architecture

In every case in computing, a software development philosophy dictates that, in modeling a system of any kind, we cannot develop a complete and standard software system that can attract and satisfy a large number of customers unless it is a general purpose software application. In using general purpose software such as word processors, presentation software, spreadsheet, database management and multimedia production, most of the users are comfortable with the functionalities provided and almost all of them have the same interest with the services brought to them. But design and development of special purpose automation systems (such as HRM, CRM, Inventory and financial management systems) follows strict rules of the business process logic of an organization. In this case, every client of the software is unique in its requirements and this leads to requirement variance among clients or customer organizations. The fundamental causes of the requirements variance among clients include: industry focus differences; customer behavior differences; product offering differences; regulation differences; culture and language differences and operation strategy differences as well as operational size [4] which leads to business transaction and file structure differences among clients/organizations. Hence, when we go automating a particular service of an organization, we have to deeply study its functional and non-functional requirements of the service and develop it based on the study we have conducted.

In a traditional software development approach (i.e. software as a product or ASP), we develop uniquely attributed systems for each individual or organization and if another request come from another individual, then developers are either develop another system from scratch or modify the previously developed system module by module. Here modularization and template design plays an important role. Developers can either change the behavior of some part of the source code or customize at some module level/templates.

But in cloud based software services, the service is given to clients in one-to-many fashion. I.e. as we have discussed early in this research, a single system is deployed on the server and accessed and give services for many clients. In this type of service delivery, it is very difficult to handle commonality among different client requirements. This idea opens a wide research topic to come up with a powerful tailor made customizable software service. Different authors proposed various implementations and software architectures on this area. The implementations

range from designing easily configurable solutions such as changing logo, appearance and predefined file based configuration up to a fully and completely customizable solution which requires utilizing some level of programming tools and techniques. Wei Sun et al. [4] proposed these classifications into hierarchical levels namely entry, aware, capable, mature and world class. Our work includes evaluation of our clients' requirements and proposes what level of customization/configuration is suitable. Frederick Chong et al. [33] and Chang Jie Guo et al. [29] also discussed on how to design a multi-tenant aware data architecture. These are "separate database" vs. "shared database and separate schema" or "shared database shared schema". With related to this, we also discus, in this section, most suitable data architecture for the problem under study. We also discussed about suitable cloud deployment strategies and service models to build an infrastructure to provide the service for SMEs. Finally according to the criteria discussed, we proposed our framework and software architecture.

5.3. Choosing suitable Maturity level of SaaS

Broadly speaking, SaaS application maturity can be expressed using a model with four distinct levels. Each level is distinguished from the previous one by the addition of one of the three attributes of a well-designed SaaS application (scalable, configurable and Configurable-multitenant-efficient) [34].

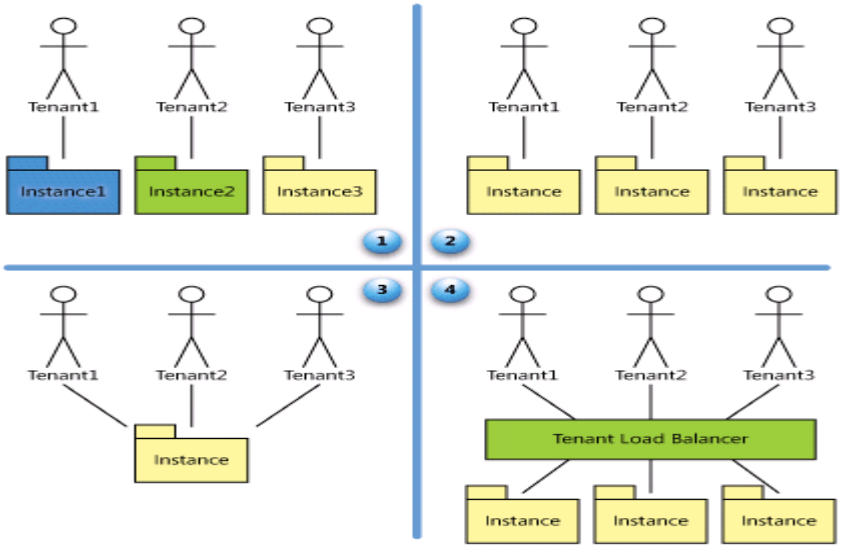

Figure 5-1 the four levels of SaaS maturity model [34]

Level I: Ad Hoc/Custom

The first level of maturity is similar to the traditional application service provider (ASP) model of software delivery, dating back to the 1990s. At this level, each customer has its own customized version of the hosted application, and runs its own instance of the application on the host's servers. Architecturally, software at this maturity level is very similar to traditionally-sold line-of-business software, in that different clients within an organization connect to a single instance running on the server, but that instance is wholly independent of any other instances or processes that the host is running on behalf of its other customers.

Although this level offers few of the benefits of a fully mature SaaS solution, it does allow vendors to reduce costs by consolidating server hardware and administration.

Level II: Configurable

At this level of maturity, the vendor hosts a separate instance of the application for each customer (or *tenant*). Whereas in the first level each instance is individually customized for the tenant, at this level, all instances use the same code implementation, and the vendor meets customers' needs by providing detailed configuration options that allow the customer to change

how the application looks and behaves to its users. Despite being identical to one another at the code level, each instance remains wholly isolated from all the others.

Moving to a single code base for all of a vendor's customers greatly reduces a SaaS application's service requirements, because any changes made to the code base can be easily provided to all of the vendor's customers at once, thereby eliminating the need to upgrade individual customized instances. However, repositioning a traditional application as SaaS at the second maturity level can require significantly more re-architecting than at the first level, if the application has been designed for individual customization rather than configuration metadata.

Similarly to the first maturity level, the second level requires that the vendor provide sufficient hardware and storage to support a potentially large number of application instances running concurrently.

Level III: Configurable, Multi-Tenant-Efficient

At the third level of maturity, the vendor runs a *single* instance that serves every customer, with configurable metadata providing a unique user experience and feature set for each one. Authorization and security policies ensure that each customer's data is kept separate from that of other customers; and, from the end user's perspective, there is no indication that the application instance is being shared among multiple tenants.

This approach eliminates the need to provide server space, as many instances as the vendor has customers, allowing for much more efficient use of computing resources than the second level, which translates directly to lower costs. A significant disadvantage of this approach is that the scalability of the application is limited. Unless partitioning is used to manage database performance, the application can be scaled only by moving it to a more powerful server (scaling up), until diminishing returns make it impossible to add more power cost-effectively.

Level IV: Scalable, Configurable, and Multi-Tenant-Efficient

At the fourth and final level of maturity, the vendor hosts multiple customers on a load-balanced farm of identical instances, with each customer's data kept separate, and with configurable metadata providing a unique user experience and feature set for each customer. A SaaS system is scalable to an arbitrarily large number of customers, because the number of servers and instances on the back end can be increased or decreased as necessary to match demand, without requiring additional re-architecting of the application, and changes or fixes can be rolled out to thousands of tenants as easily as a single tenant.

Among these levels of maturity, level IV seems the ultimate goal for most SaaS providers. But business applications fall between a continuum between isolated code and data versus shared code and data. So, our application fall along this continuum depends on the business, architectural and operational needs as well as on customer considerations. An application can also fall between one or more maturity levels at a time or throughout. Moreover, it depends on the progress of the service as how many customers can support at initial level and as it increases to support more. Depending on this logic, our SaaS service application will probably has fewer numbers of customers since reaching to too large customer need more efforts which requires the application service to be known by large target customers. Accordingly, this application initially can be designed to fulfill requirements of level III, and then can be migrated to fit level IV maturity as more tenants subscribe to the service.

5.4. Choosing suitable Configuration and Competency model

To facilitate strategy definition and execution discussion around SaaS configuration and customization, [4] introduced configuration and customization competency model. This model classifies SaaS maturity into five levels namely entry, aware, capable, mature and world class levels.

Entry: is a completely standardized offering without any configuration and customization support. It is achieved through designing the functionalities as standard offering to cover targeted customers.

Aware: provides a relatively standardized offering with predefined variance points and its achieved through parameterized offering. This level is useful when there is low variance level among the requirements.

Capable: it provides a relatively standardized offering with user defined configuration. It offers self-service configuration tool to empower customers and there is medium level of requirement variance is supported.

Mature: provides base offering with programmable environment to enable user preferred customization. It offers scripting based programming for very flexible customization. It is suitable to design services that require tailoring of higher variance requirements.

World Class: it supports to design applications with extremely high variance requirement. It is a fully tenantized offering. It offers a platform supported by a programming model and tools to enable extremely strong customization or even new application development.

Analysis was made to understand the characteristics of each of the competency models. And according to the case study we have conducted on the behavior of inventory management among different enterprises, most of the characteristics of most of the businesses fulfill the third competency models. That is, “capable”. So we have proposed “capable” competency model to design the software service with this level of customization and configuration.

5.5. Choosing a multi-tenant database architecture

We have discussed so far in chapter two that there are three major approaches to manage a multi-tenant data. I.e. Separate databases, shared database separate schema, and shared database shared schema. Designing a SaaS application needs to understand the pros and cons of each approach and the nature of the business to develop a robust and scalable cloud based service.

Separate Databases

In this architecture, a separate database and table instance is designed for each tenant. It provides a shared pool of computing resources and source code among tenants. It is the simplest approach to data isolation.

Pros: easy to

- Extend the applications' data model
- Meet tenant's individual needs
- Restore a tenant data from backups

Cons:

- Leads to higher cost of maintaining equipment and backing up tenant data
- More hardware costs
- Maximum number of tenants to support depends on the maximum number of databases the server can support. That is limited scalability.

It is the premium approach, and its relatively high hardware and maintenance cost make it appropriate for customers who are willing to pay extra for added security and customizability. For example, customers in fields such as banking or medical records management often have very strong data isolation requirements, and may not even consider an application that does not supply each tenant with its own individual database.

Shared Database, Separate Schema

This approach involves housing multiple tenants in the same database, with each tenant having its set of tables grouped into a schema created specifically for the tenant. When a customer first subscribes to the service, the provisioning subsystem creates a discrete set of tables for the tenant and associates it with the tenant's own schema.

Pros:

- Easy to implement and the data model can be easily extended
- Can support a larger number of tenants per database server.
- Offers moderate degree of data isolation for security conscious tenant

Cons:

- Tenant data is harder to restore in the event of failure – restoring the entire database means, overwriting the entire data of every tenant on the same database.

The separate schema approach is appropriate for applications that use a relatively small number of database tables, on the order of about 100 tables per tenant or fewer. This approach can typically accommodate more tenants per server than the separate-database approach can, so we can offer the application at a lower cost, as long as your customers will accept having their data co-located with that of other tenants.

Shared Database, Shared Schema

This approach involves using the same database and the same set of tables to host multiple tenants' data. A given table can include records from multiple tenants. Developers can use "TenantID" to associate every record with appropriate tenant.

Pros:

- Has lowest hardware and backup costs
- Allows us to serve largest number of tenants per database server

Cons:

- Since multiple database share the same database and schema, it may incur additional development effort in the area of security; to insure one tenant can never access the other tenants data.

The shared-schema approach is appropriate when it is important that the application be capable of serving a large number of tenants with a small number of servers, and prospective customers

are willing to surrender data isolation in exchange for the lower costs that this approach makes possible.

Each of the three approaches described above offers its own set of benefits and tradeoffs and choosing the most appropriate depends on the providers requirements to support larger tenants and to provide better performance in some areas of technical and business considerations such as economic considerations, security /tenant, regulatory as well as skill set considerations.

An analysis made in our case to choose which approach is appropriate to design and implement SaaS-based inventory management system for SME enterprises existed in Ethiopia. Economically, the third approach is most appropriate since it costs lowest hardware and software than other systems. And it is better scalable than the first two approaches. It can support large number of tenants than others. It is not limited by the maximum number of databases the server can support because it only uses a single database and shared tables for all tenants. However, as indicated, it needs additional development effort to ensure security and highly customizable software services. The first approach (separate database and schema) incurs high cost of hardware and software. And even though it is simple to implement, it doesn't realize a full cloud based SaaS architecture. Separate database instance is a typical implementation of ASP model which faces potential scalability issues. Plus inventory management of most SMEs is thought to require not more than 100 tables per tenant. And separate schema for each tenant ensures data isolation better. And since it uses a single shared database, the system scalability cannot be limited with maximum number of databases the server can support. For these and other reasons we proposed shared database with separate schema in the development of our solution.

5.6. Choosing Service and deployment models

We have discussed so far, the different classifications and attributes of cloud service and deployment models. Selecting the most suitable service and deployment model is a major task in adopting cloud services to organizations. This needs a better understanding of the business needs as well as distinguishable characteristics of services.

We proposed a cost effective cloud based software service to all SMEs in Ethiopia. The proposed cloud service should be reachable for all SMEs and should be open for everyone to subscribe and use it once they get credentials from the provider. So, everyone can have access to the services

and only those who have the credentials can use the services provided by the application once they have logged in. It is assumed that there is a service provider (May it be private or government own) company, who serves this clients online from anywhere. So, according to the proposed service model, everyone who can pay the service fee can use the system. The appropriate deployment model, hence, is proposed to be public model. So, the service will be available on the internet, then anyone interested with it can request for the service and will be granted accordingly checking if all requirements are met.

With regard to service models, we are already studying about SaaS based software development which is delivered as a service through a given cloud platform. Plus we have already discussed, in the problem statements section, that IT infrastructure installation and management for SMEs by themselves is costly investment for which they cannot afford to achieve by themselves. So outsourcing all the IT related services (hardware, software, management) is the most important cost effective alternative. Accordingly, we mean that we proposed to provide the IT service as "SaaS".

5.7. The framework and software architecture

In this section, we discuss the proposed framework and software architecture for the problem under study. The section discusses overview of how the online system can be accessed from end point devices; general framework of the proposed SaaS based automation development with description of each component as well as software architecture of SaaS based Inventory and Sales control system based on the proposed framework.

5.7.1. Getting access to the service

The SaaS based IMS application is deployed in the provider's datacenter and will be accessed publicly via the public internet. Any client interested in and who is running a business of any kind starting from micro to large one, can subscribe and use the service provided by. Once created an account and accepted by the provider, client uses the software which can be configured and customize based on his needs. The software seen by the client as it installed by his/her machine. He can put his company logo and profile and it works for other clients in the same way. Plus the service can be designed to be accessed by various devices, PCs, laptops,

PDAs, smartphones and tablets. Device independence is the major focus and recommendation in developing this system to enable SMEs a cost effective access of services. For instance, micro business can use smartphone, instead of PC to manipulate their data on the cloud. The overall architecture of the infrastructure is depicted in figure 5-2 below.

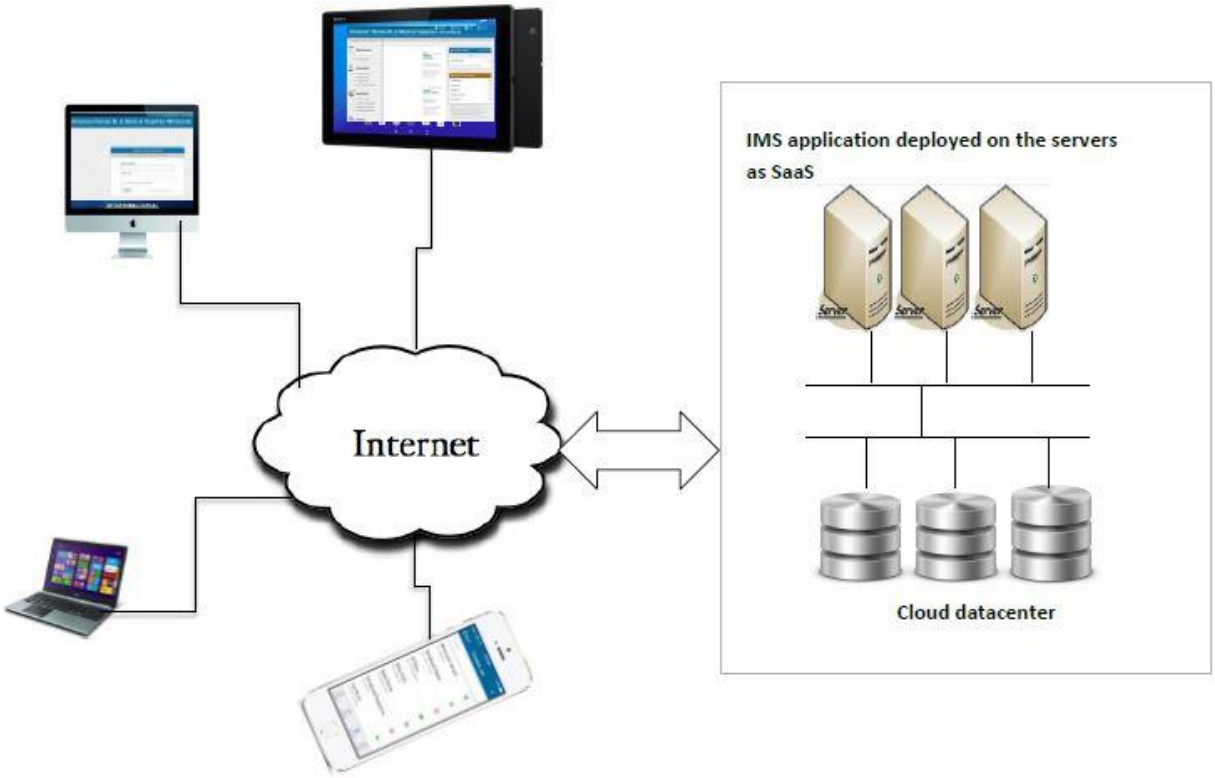

Figure 5-2 High level view of getting access to a cloud service

5.8. The software architecture

Specifications on different parameters have been discussed so far in the previous chapter. Among these specifications we have defined:

- Database architecture – shared database with separate schema approach
- SaaS maturity level – 3rd maturity level (configurable multi-tenant-efficient) to be upgraded into level 4
- Configuration competency model- "capable" [8]
- Configuration and customization done on layers- presentation, business logic, data access layers

The architecture is an improvement model of a paper authored by [8], but significant improvements have been added based on the specifications defined. And the flow of customization has been changed to improve and ease configuration and customization for the client. Figure 5-3 presents high level software architecture of the system. Details will be discussed on the line that follows.

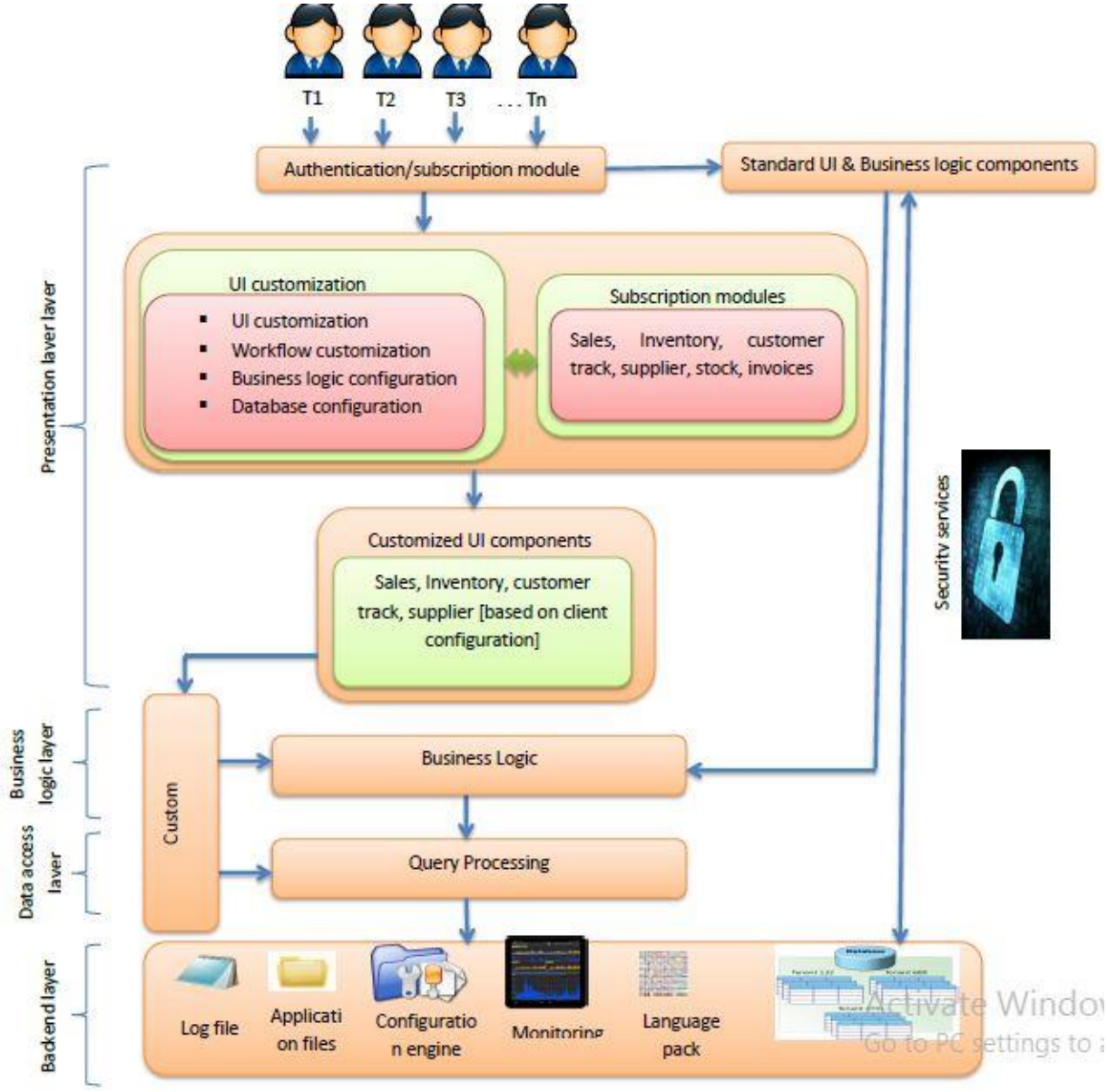

Figure 5-3 High level of IMS software architecture

5.8.1. Basic components of the architecture

A. The Backend layer

The backend layer is the software file system deployed on the provider's infrastructure that includes the following main components.

The Performance Monitoring Service: this component performs how the system resources are utilized by clients and how the software well respond to requests. This will help the vendor getting information to upgrade the services of the software. It can monitor, for example, which queries are responding slowly, which components are heavily used. And which tenant is overusing resources.

Log file: very important component of a software application. It can be used for performance monitoring of the application, user activities, discover software bugs, finding out processing bottlenecks etc.

Language pack: SaaS applications are used by many tenants from different cultures and different language. So, there is a need to define a language engine that enables tenant to choose a language suitable to its working environment. In the Ethiopian context, customers may be from Tigray, Afar, Amhara, Oromia, South, Somalia, Gambella or from Benshangul gumuz. The system we are going to realize, proposed to serve clients in major working languages of Ethiopia. So, a language pack is integrated into the architecture to enable tenant to choose whatever language they want to work with.

Application files: are the main application program files of the IMS system. They are main program files of IMS written using one of most popular web based programming tools such as PHP, Java or ASP.

Configuration files: Is where configuration and customization changes are saved in, for a specific tenant.

Runtime configuration engine: This component provides customization tools and environments. For example, we can generate a form element and specify its data type to hold since these form elements are already defined and specified to use.

Database instance: another important component in this layer is database design. Among the three basic data architectures was comparatively analyzed. Then we chose "Shared database separate schema" architecture. The database component in the framework shows this database schema.

B. Data access layer

Query Processing: is a component where database query scripts such as SELECT, UPDATE, DELETE, INSERT and COMMIT statements written and executed.

Business Logic: this component is where basic business logic scripts written and connected to the query processing to manipulate tenant data and perform application specific operations.

C. Presentation Layer

UI customization: this component allows client to customize company logo, color, profile information etc. as well as advanced features such as adding new form element. Here all customization operations (UI, business logic; data flow and database configuration) depend on UI customization. Client customizes UI elements, and then the effect will be applied on all layers of the software at once.

Subscription Modules: this component consists of list of main modules of the SaaS based IMS system or system modules in general in any SaaS base automated system. It allows the tenant to choose relevant modules and configure his application environment based on his interest.

D. Security

Another important feature of this architecture is implementation of security at all levels. Since security is a major threat in cloud based and any network systems, it should be given high concern and latest security techniques should be applied at all layers of the software. This may range from authentication and authorization from the presentation layer up to data encryption or better technique in backend layer. The security component in the architecture shows this.

5.8.2. The core business process flow of the architecture

One major contribution of this research is introducing a new and improved programming approach which recommends SaaS developers to follow to develop most powerful SaaS-based line-of-business applications which enables self-service user defined customization model. In this section we have tried to show the core business process flow of the steps shown on the proposed framework (IMS system architecture).

Steps:

1. System displays authentication and sign-up interface of the system.
2. A client who is already signed up and have created working environment will input his credentials and try login to the system.

Credentials

- User Name
- Password
- Company Registered ID (tenant ID)

[But the main concern of this topic is about new clients to the system]

3. Client sign-up [with full information of company]

4. After completing client sign-up, system confirms (confirmation via client email), and provides the standard interface (with standard application modules), which user allows to configure some components which require minimum complexity.
5. If client satisfied with the standard service offerings, he/she continue to use it only with some parameterized configuration options, such as company profile [logo, color, UI appearances, language etc.].
6. If he/she is not satisfied with the available environment, the system will allow to go further customization options.
7. System will display application modules to choose to which he/she wants to make customizations.
8. User chooses application module to make customizations on it.

Customization of the user interface

8.1. System provides a form to be filled, which asks field name, field label name, field specification (data type to hold and store to database, values if necessary, require/not required options etc.)

8.2. Filling the form, client submits it. then system:

- 8.2.1. Creates a form field specified automatically,
- 8.2.2. Integrates it into the required application component
- 8.2.3. Generates application code (business logic program integration)
- 8.2.4. Generates customized query to manipulate the database
- 8.2.5. Creates additional table column which is specified in the form.
- 8.2.6. New change will be saved into configuration file with the corresponding tenant ID.

CHAPTER SIX

6. Implementation Proposal

6.1. Overview

This section discusses how the research project can be implemented and become usable to the general public. We put here, the highest level view of the project which looks like and how it works if it will be implemented. We have included snapshots of some of its application interfaces. The name of the hypothetical company who is dedicated to deploy and serve clients is named to be "HALETA". This is why the first snapshot titles the header of the subscription website as "**HALETA Online Inventory Services**".

A. Sign-up and Subscription interface

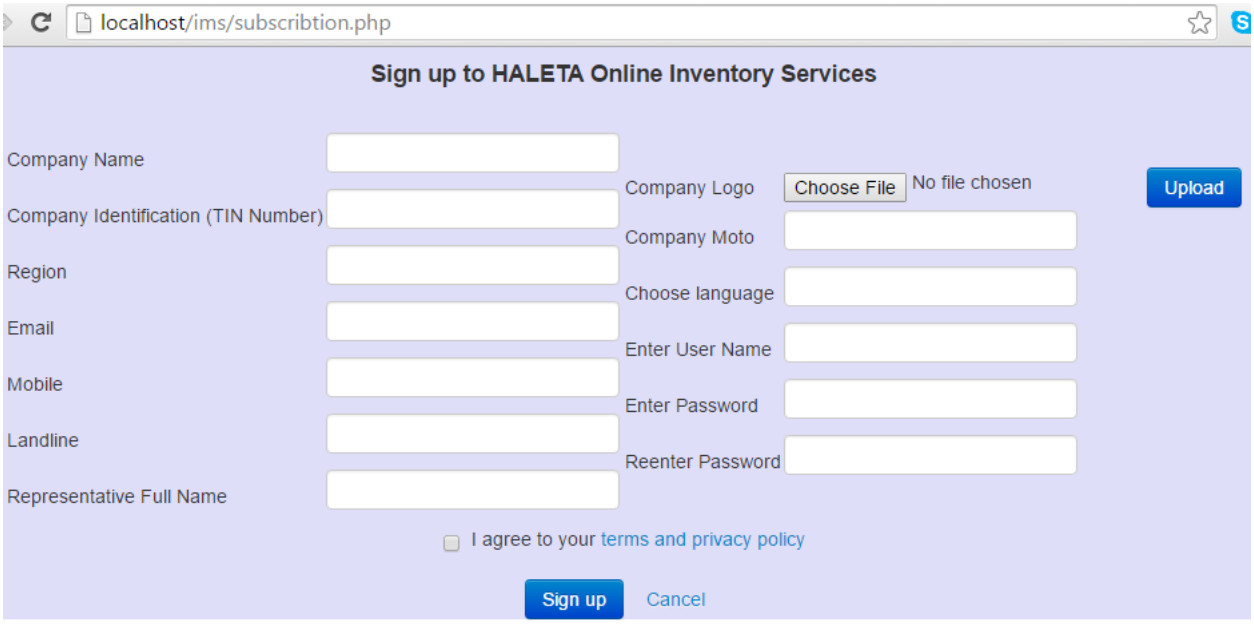

Figure 6-1 Sign-up and subscription user interface

B. Client Login Page

After subscribing and being approved by the provider client can sign and use the service. The login panel is depicted in the figure below. Because its SaaS application, it includes additional

field to authenticate a person as a representative of a particular company (Company identification).

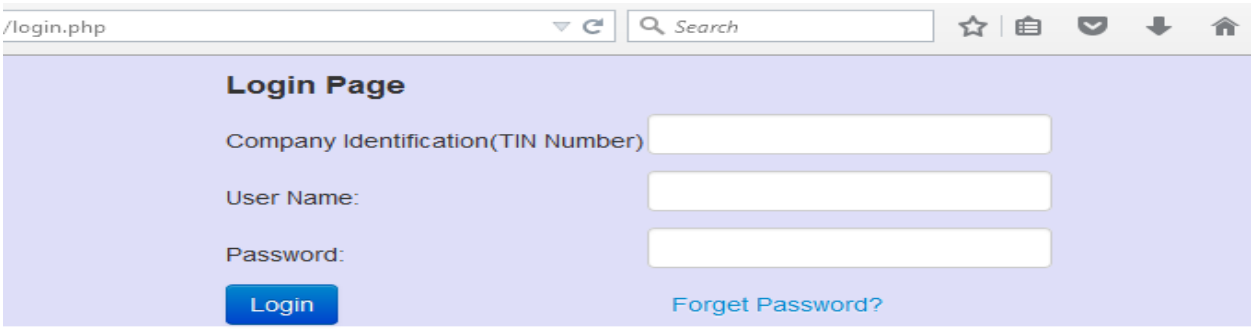

Figure 6-2 Login user interface

C. Clients Home page

After login, clients are presented with a client specific home page, which incorporates all customizable and standard services. The two figures below illustrate, home page of two sample companies.

Figure 6-3 Client home page of HAYAT General trading

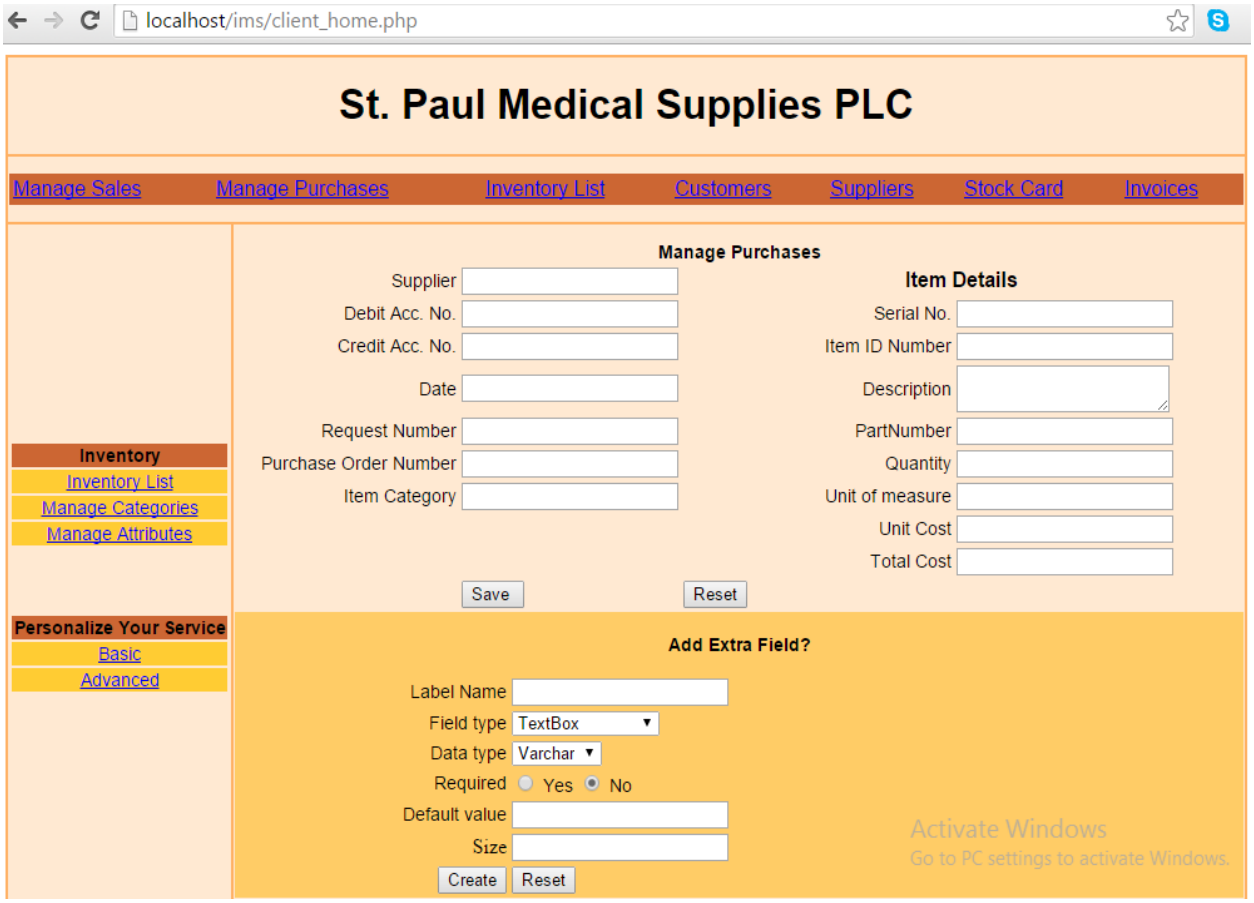

Figure 6-4 Client home page of St. Paul Medical Supplies

D. Basic Configuration

Through basic interface configuration panel, users can manage their client specific basic environment settings

Figure 6-5 Basic configuration

E. Advanced configuration options

Through advanced configuration panel, clients can add additional data fields with their specification and extend their application functionalities beyond the standard interfaces.

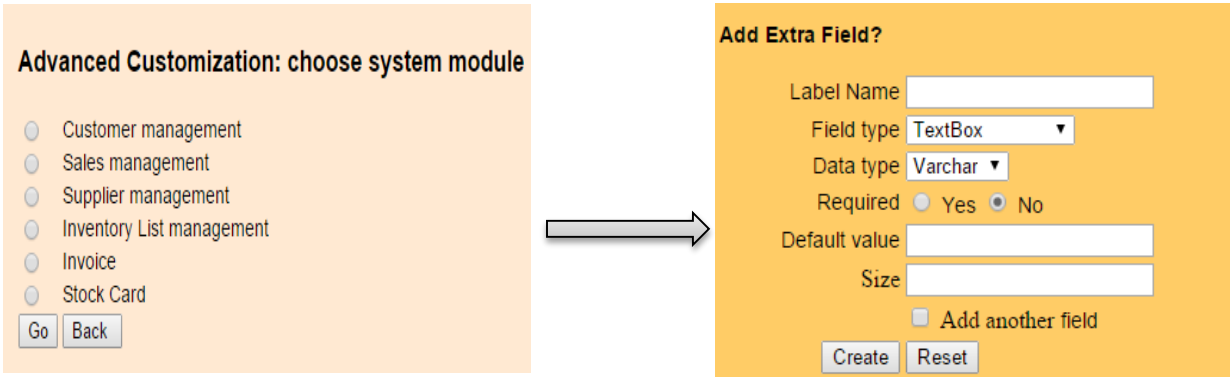

Figure 6-6 Advanced Configuration panel

CHAPTER SEVEN

7. Results and Conclusion

7.1. Main results of the study

Cloud computing become an important part of everyday life of organizations in the computing industry. It has become main tool to provide computing solutions as a service via the internet thereby reduces the cost invested on IT. This computing paradigm plays an important role for organization in every aspect in outsourcing computing utilities and renting from the providers that provide powerful IT service either of Hardware, Networking or hosted application programs. In this research, we have been motivated to study adopting of cloud computing solutions for Small and medium sized businesses in Ethiopia since this area is still not much covered and solved in research. In studying the adoption of cloud based solutions we have studied how the service can be delivered and, most importantly, discussed major challenges of enabling SaaS based software development, requirement variability, among tenants in a multi-tenancy architecture. And then we studied different ways to realize customizable and configurable software solution which can specify variable requirements of tenants. Then we have designed a development framework and system architecture to enable developers to develop powerful SaaS applications that allows tenants entertain easy to use self-service customizable application service.

7.2. Major contribution of the research

The main intention of this research is to develop architecture and programming model to guide SaaS based line-of-business application development more specifically for Inventory management system for SMEs. The designed architecture has the following features and enhancements on the framework previously done by researchers Eyad Saleh et al. [8].

1. It enables to develop fully standard SaaS Software with an additional feature that allows configuring and customizing, if tenant has additional feature requirements. That is, tenant provided two separate environments. Customizable and standardized.
2. Customization is layered (hierarchical), in which all programming components in each layer automatically customized (custom code can be generated) once UI components specified. That is all programing components, depend on UI customization and once UI customization is done, all things will be updated automatically for tenant. This way, customers can configure their own unique environment without taking care of how to program each and every component of the application.
3. Enhanced security at all layers of the application. From authentication up to database access layers. The previous framework which we have used as a base for our architecture has no advanced security mechanisms rather than the basic authentication service.
4. Database architecture is enhanced into Shared database-Separate schema approach which can improve processing power of the service than Separate database approach, while the previous architecture uses Separate database approach.
5. The architecture we have demonstrated is architected for a particular SaaS based system service (cloud based inventory management system), and it can be modified a much variety of business applications such as ERP, HR, financial and accounting services etc.

7.3. Conclusion and future research directions

We have studied adopting SaaS and its challenges specifically, requirement variability challenges in different layers of software, and we have proposed a development framework/architecture, to provide enhanced self-service user-defined customizable application service provided via cloud based computing environment. The proposed architecture enhances the customization process so that eases the burden of tenants since the previously proposed architecture dictates to customize the service layer by layer or component by component independently. Our approach is generating a customized code for business logic and database query processing while tenant customizes User Interface. The previous architecture, since dictates tenant to customize each component independently, it need skilled IT personnel to make

customizations. So, our approach indirectly removes requirement of highly professional IT personnel to make customizations for each client organizations thereby reduces additional cost incurred to have the personnel. In the future, we will test the proposed architecture by developing the project based on the proposed framework and programming model and we will do further study on other areas of SaaS based architecture such as Security, costing or service payment models with respect to the current conditions of business transaction in Ethiopia.

References

[1] NIST, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, US Department of State, 2011.

[2] A. N. E. I. I. Nawsher Khan*, "Cloud Computing: Analysis of various platforms," 2013.

[3] C. L. D. N. a. P. W. Hoang T. Dinh, "Survey of Mobile Cloud Computing: Architecture, applications and approaches," *WIRELESS COMMUNICATIONS AND MOBILE COMPUTING,* p. 4, 2013.

[4] X. Z. C. J. G. P. S. H. S. Wei Sun, "Software as a Service: Configuration and Customization Perspectives," vol. IEEE Congress on Services Part II, 2008.

[5] [Online]. Available: http://ec.europa.eu/enterprise/policies/sme/facts-figures-analysis/sme-definition/index_en.htm. [Accessed 07 March 2015].

[6] "SME: U.S. and EU Export Activities, and Barriers and Opportunities Experienced by U.S. Firms," United States International Trade Commission , July 2010.

[7] FDRE, Ministry of Construction and Urban development, [Online]. Available: http://www.mse.org.et/Pages/default.aspx. [Accessed 07 March 2015].

[8] N. S. a. C. M. Eyad Saleh, "A Framework for Migrating Traditional Web Applications into Multi-Tenant SaaS," in *INFOCOMP 2012 : The Second International Conference on Advanced Communications and Computation*, 2012.

[9] N. W. M. J. Z. Hong Cai, "A Transparent Approach of Enabling SaaS Multi-tenancy in the Cloud," in *2010 IEEE 6th World Congress on Services*, 2010.

[10] A. O. Kuyoro 'Shade O., "ICT solution to Small and Medium Scale Enterprises (SMEs) in Nigeria," *International Journal of Computer and Information Technology (ISSN: 2279 – 0764),* 2014.

[11] UNECA, "Can e-commerce facilitate the growth of small and medium-sized enterprises in Africa?," United Nations Economic Commission for Africa, 2014.

[12] T. J. S. Bhaswati Hazarika, "Survey Paper on Cloud Computing & Cloud Monitoring: Basics," *SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE),* 2015.

[13] C. S. Y. a. S. V. Rajkumar Buyya, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in *The 10th IEEE International Conference on High*

Performance Computing and Communications, 2008.

[14] T. J. V. P. R. E. Anthony T. Velte, Cloud Computing: A practical approach, McGraw-Hill, 2010.

[15] Purch Company, [Online]. Available: http://www.tomsitpro.com/articles/iaas-vendor-list,1-2228.html. [Accessed 2 May 2015].

[16] Purch company , 2 May 2015. [Online]. Available: http://www.tomsitpro.com/articles/paas-providers,1-1517.html.

[17] Salesforce, [Online]. Available: https://www.saleforce.com. [Accessed 2 may 2015].

[18] NETSUITE, [Online]. Available: http://www.netsuite.com/portal/home.shtml. [Accessed 2 may 2015].

[19] Concur Technologies, [Online]. Available: https://www.concur.com/. [Accessed 2 may 2015].

[20] CornerStone OnDemand, [Online]. Available: http://www.cornerstoneondemand.com/. [Accessed 2 may 2015].

[21] A. Gajbhiye, "Cloud Computing: Need, Enabling Technology, Architecture, Advantages and Challenges".

[22] B. Getnet, "FRAMEWORK TO ADOPT CLOUD COMPUTING FOR," p. 19, 2013.

[23] M. F. C. C. A. M. R. N. M. d. C. Neves Fátima Trindade, "The adoption of cloud computing by SMEs: identifying and coping with external factors," 6 may 2015. [Online]. Available: http://hdl.handle.net/10362/6166.

[24] A. Petrakou, P. Brandt, R. Gustavsson and P. Jokela, "Collaborative e-Marketplaces Containing Clusters of SMEs: Drivers and Barriers in the Local Food Sector," in *44th Hawaii International Conference on System Sciences (HICSS)*, 2011.

[25] O. Sinan Tumer, "Digital ecosystems & Co-innovation towards sustainable societies," in *4th IEEE International Conference on digital Ecosystems and* , 2010.

[26] Techradar, "SaaS, PaaS and IaaS: which cloud service model is for you?," 2015. [Online]. Available: SaaS, PaaS and IaaS: which cloud service model is for you?. [Accessed 6 May 2015].

[27] J. S. N. Lopes, "SaaS (Software as a Service) – Models and Infrastructures," 2009.

[28] H. LIAO, "Design of SaaS -based Software Architecture," 2009.

[29] W. S. Y. H. Z. H. W. B. G. Chang Jie Guo, "A Framework for Native Multi-Tenancy Application Development and Management," *The 9th IEEE International Conference on E-Commerce,* 2007.

[30] N. Lehrer, "DataCenterPost," 2015. [Online]. Available: http://datacenterpost.com/2014/07/closer-look-iaas-paas-saas/. [Accessed 9 may 2015].

[31] J. G. R. P. Gurudatt Kulkarni, "Cloud Computing-Software as Service," in *International Journal of Cloud Computing and Services Science(IJ-CLOSER)*, 2012.

[32] H. S. K. B. Sunil Kumar Khatri, "Multi-Tenant Engineering Architecture in SaaS," in *International Conference on Reliability, Infocom Technologies and Optimization*, Noida, India, 2013.

[33] G. C. a. R. W. Frederick Chong, "Multi-Tenant Data Architecture," 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479086.aspx. [Accessed 11 may 2015].

[34] G. C. Frederick Chong, "Architecture Strategies for Catching the Long Tail," 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479069.aspx. [Accessed 21 May 2015].

[35] SalesBinder, "Online Inventory Management Software in the cloud," 2015. [Online]. Available: https://www.salesbinder.com/. [Accessed 20 May 2015].

[36] ClearlyInventory, 2015. [Online]. Available: http://www.clearlyinventory.com/. [Accessed 20 May 2015].

Annexes

A. Interview Questions

The following are the interview questions conducted during the data collection activities.

1. Does your organization have ICT infrastructure?
2. If you have ICT infrastructure fulfilled, what kinds of activities you work with your computers and the internet service?
3. What kinds of data processing you are using for manipulating your business data?
    - Manual or automated/computerized

Questions for those who are using Manual data processing

4. Are you comfortable with this kind of data processing? Tell me comparing it against using software applications, since currently there are a number of automation software available on the market. What if you can a chance to use technologies like mobile or computer software to manipulate your business data.
5. What is your problem having automated System?

- Skill?
- Personnel?
- Cost of software and infrastructure

Skill problem?

Outsourcing is better option. Did you do that? Because, there are external auditor companies to do this for you. For example, to audit your financial information.

- Here some companies told me they are already contracted with external auditors. I have asked them as:

But there is another better way than this. That is getting automated system developed for yourself (i.e. your organization). This way, the system automatically audits everything you need with minimal human intervention. What is the problem getting and using these kinds of software?

6. I want to know something on how you are handling information on Inventory and stock management data. Something such as basic business processes and requirements. Can you show me some formal documents related to this?

Questions for those using automated system

7. Are you comfortable with the system you are using?
8. Can you tell me the main functionalities your system can do?
9. Is the software developed more specifically in-line with your business requirements or it considers general business process of other different other organizations? That is, have you purchased from market or got developed for your specifications?
10. Can you invite me to see overview of how the system works?
11. Currently there is a modern software service delivery trend. That is, single software can be deployed centrally on software provider networks such as Ethiotelecom (to describe the trend in a more understandable way). Customers can subscribe online to use the software. The software system and its data are stored on the providers' central computer. Then customers can access the system interface and its data through mobile phones or PCs. In this case, they pay only for the service instead of purchasing the original software

so that they can decrease their IT cost. Take your cash register, for instance, it handles all trade customers data of the Federal Revenue Authority. In this system customer data is completely isolated with one another. You may also be familiar with Facebook and Gmail services. They work that way. What can you feel if your system can be optimized like this? What is your filling on its advantages and disadvantages such as data security and isolation, virus and more exposure to unfair tax collection systems (unfair in your sense)?

B. Sample Data lists of Small and Medium Enterprise in Adama City

lakk	Maqaa Interpay	Maqaa Miseensa	Gosa gurm'in	Seektar	Bar gurm'in	Kappitalaa		Bayyina miseensa			Ganda
						Ka'u	amma	Dhi	Dub	Id a'a	
1	Tsahaayi Nigatuu	Tsahaayi Nigatuu	Taji	Taji	2006	30000	60000		1	1	M/Adama
2	Seenaa Rayyaa	Seenaa Rayyaa	Taji	Taji	2006	10000	15000		1	1	M/Adama
3	Roozaa Takaliny	Roozaa Takaliny	Taji	Taji	2006	5000	8000		1	1	M/Adama
4	Sofiya Sayida	Sofiya Sayida	Taji	Taji	2006	30000	45000		1	1	M/Adama
5	Fannayee Eliyaas	Fannayee Eliyaas	Taji	Taji	2006	40000	5000		1	1	M/Adama
6	Balchoow Tashomaa	Balchoow	Ij	Ij	2006	10000	170000	2		2	M/Adama
		Tashomaa									
7	Marat Tazaraa	Marat Tazaraa	Dal	Dal	2006	3000	2000		1	1	M/Adama

8	Balayinesh Taxasoo	Balayinesh Taxasoo	taji	taji	2006	5000	6500		1	1	M/Adama
9	Shifara Adunyaa	Shifara Adunyaa	taji	taji	2006	15000	18000	1		1	M/Adama
10	Kh	Hiwot W/senbat	Ij	Ij	2006	10000	10000	3	1	4	M/Adama
		Iteenash Tarikuu									
		Kinfuu Nammartuu									

B. Wanofi finance and auditing office accounting data for different SMEs using Peachtree accounting System

Inventory List

xyz Company		
Inventory List		
Thursday, June 18, 2015		
Item ID	Description	Item Class
01	MRF 750 X 16 M77 Tyre	Stock item
0139	9-20 CARGO FSR TYRE	Stock item
02	MRF 750 X 16 LUGTyre	Stock item
03	MRF 750 X 16 Miller Tyre	Stock item
04	MRF 750 X 16 Traction Tyre	Stock item
05	MRF 825 X 16 SML Tyre	Stock item
06	MRF 825 X 16 LUG Tyre	Stock item
07	MRF 560 X 13 Tyre	Stock item
08	MRF 195 X 15Tyre	Stock item

09	MRF 185 X 14Tyre	Stock item
10	MRF 900 X 20 M77 Tyre	Stock item
100	QUANTM 12 V 50AM	Stock item
101	QUANTM 12 V 45 AM	Stock item
102	Bridgstone185/70R13	Stock item
103	Ling Long10:00-20 Tyres	Stock item
104	Aelous 165-70-13	Stock item
105	Aelus 185 X 13	Stock item
106	Aelus 175 X 13	Stock item
107	Ling long 175 X 13	Stock item
108	Ling Long 900 X 20	Stock item
109	Bridgestone 235 X 15	Stock item
11	MRF 900 X 20 Miller Tyre	Stock item

Inventory Unit Activity Report

Inventory Unit Activity Report

For the Period From Apr 1, 2015 to Apr 30, 2015

Filter Criteria includes: 1) Stock/Assembly/Serialized. Report order is by ID. Report is printed with shortened descriptions.

Item ID	Item Description	Item Class	Beg Qty	Units Sold	Units Purc	Adjust Qty	Assembly Qty	Qty on Hand
01	T340	Stock item	-509.00	971.00	960.00			-520.00
010	T526	Stock item	-4.00	2.00				-6.00
011	H5	Stock item	-6.00					-6.00
012	T611	Stock item	-7.00					-7.00
013	P5	Stock item	-31.00	40.00	30.00			-41.00
014	T635	Stock item	-6.00					-6.00
015	Machinary rent	Stock item						
016	T605	Stock item	-8.00					-8.00
017	T718	Stock item						
018	Mobile Accesory	Stock item						
019	S9	Stock item						
02	T463	Stock item	-5.00					-5.00
020	T345	Stock item						

021	H7	Stock item	-1.00					-1.00
03	M3	Stock item	14.00	11.00				3.00
04	T535	Stock item	13.00	5.00				8.00
05	T630	Stock item	-25.00	1.00				-26.00
06	T420	Stock item	-35.00	11.00				-46.00
07	T430	Stock item	-54.00					-54.00
08	T545	Stock item	6.00					6.00
09	TV51	Stock item	-17.00	61.00	80.00			2.00
115	Itel 230	Stock item	3.00					3.00
116	Ok 98 Oking	Stock item	19.00					19.00
126	Ok 367	Stock item	20.00					20.00
127	Ok M31	Stock item	40.00					40.00
128	Smadl A10	Stock item	165.00					165.00

C. Different business forms of Sample organizations

Attachment of Amanuel Human Medicine and Medical Supplies Wholesaler

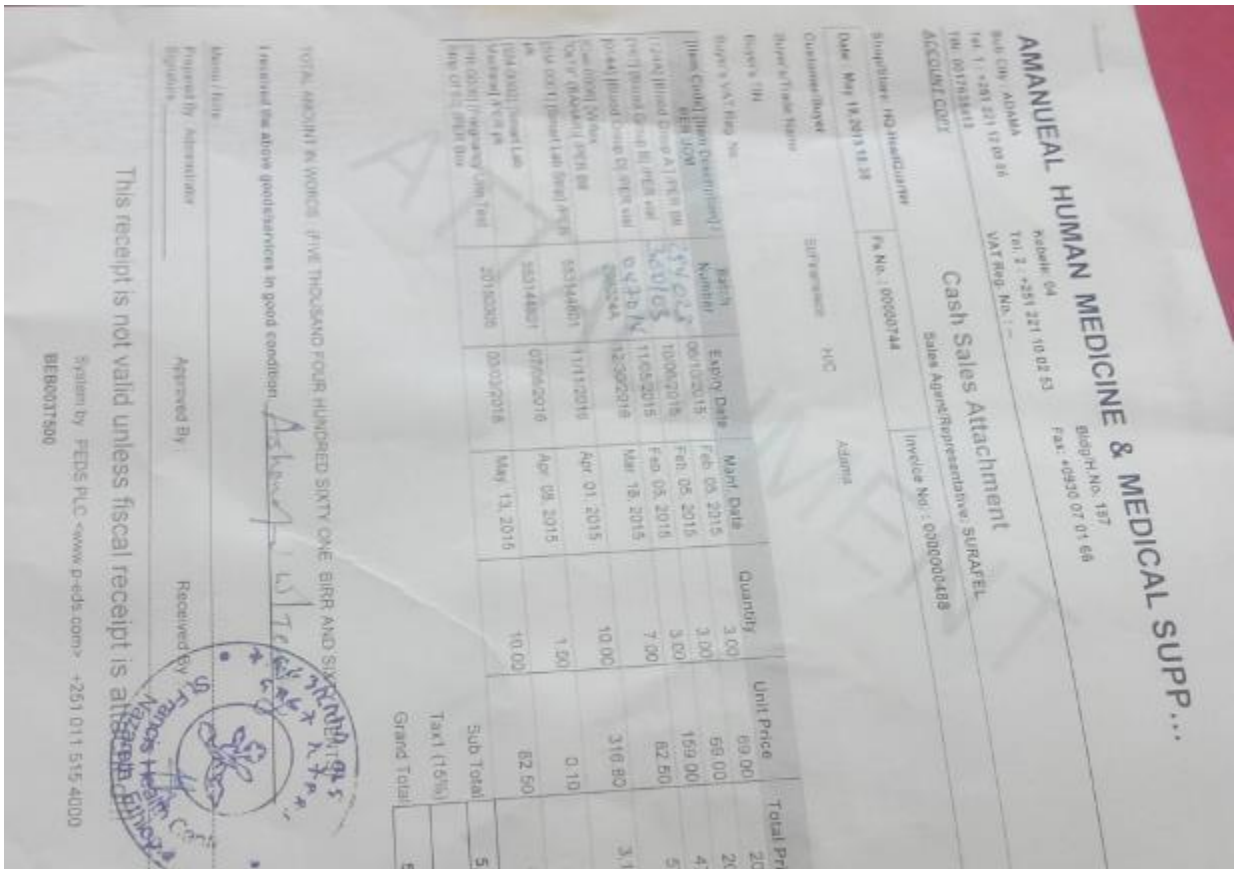

Figure 0-1 Attachment of Amanuel Human Medicine and Medical Supplies Wholesaler

Cash Sales Attachment of OMEDAD General Trading

Figure 0-2 Cash Sales Attachment of OMEDAD General Trading

OMEDAD Pvt. Ltd "Delivery Order"

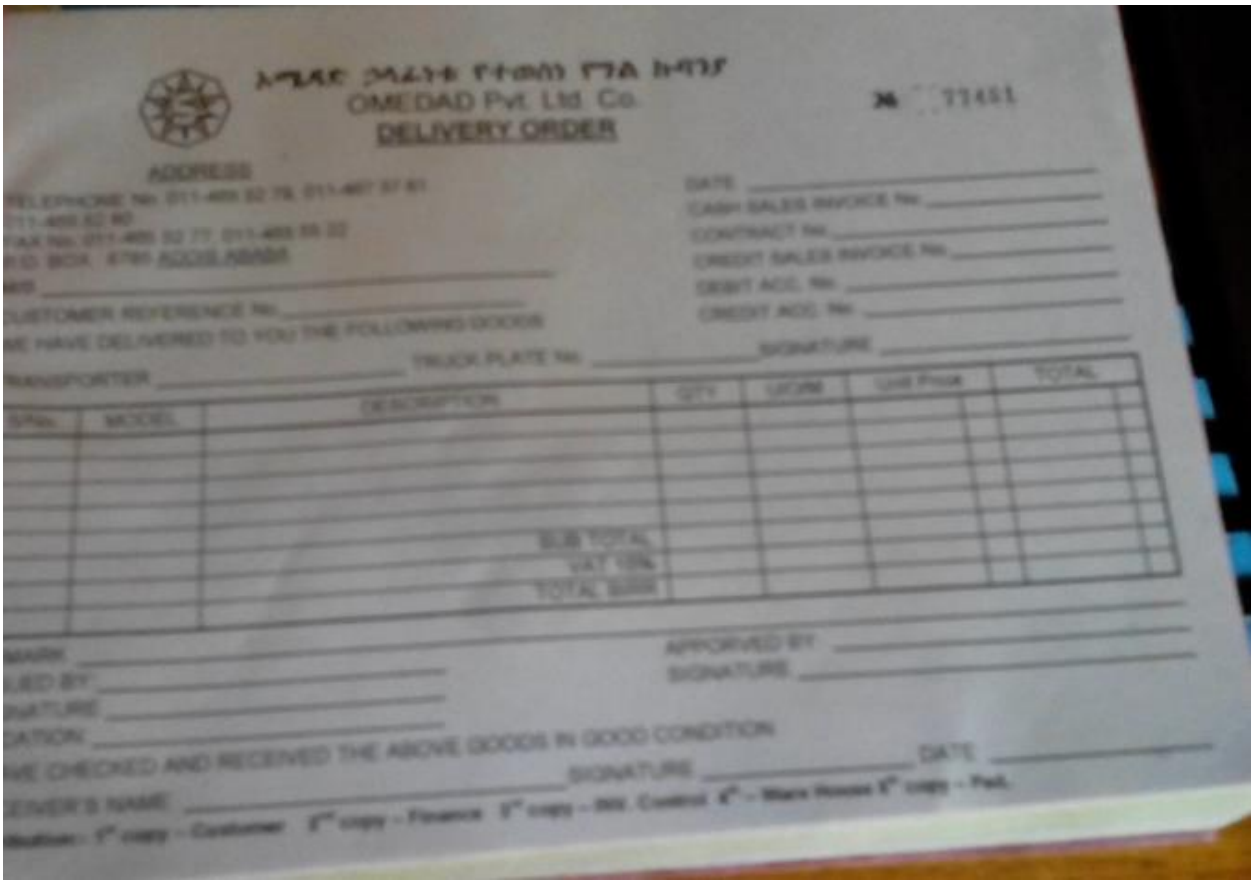

Figure 0-3 OMEDAD Pvt. Ltd “Delivery Order”