Design Comparison of Traditional Application and SaaS

R. Vidhyalakshmi
Dept. of Computer Science, JJT University
Jhunjhunu - 333001,
Rajasthan, India.
vidhyapartha@yahoo.com

Dr. Vikas Kumar
Asia-Pacific Institute of Management,
3&4 Institutional Area, Jasola, Sarita Vihar,
New Delhi, India.
prof.vikaskumar@gmail.com

Abstract— ***Software as a Service (SaaS) is an on-demand software delivery model, where the software and its related data are placed at remote location and are accessed as a web based service using a thin client such as web browser over the Internet. The salient features of SaaS includes low startup cost, easy provisioning at the time of need, shorter release cycle, easy maintenance, cost saving on maintenance and upgradation, multi tenant architecture, etc. The advantages and the working pattern surely make it as a best option for Small and Medium Business Enterprises, when compared to traditional web applications. However the consideration of wide consumer base with different requirements specification for a single application leads to the complexity of SaaS design. The SaaS applications are based on protocols such as HTTP, SOAP and architecture design concept such as REST. Many development models have been used for SaaS design however most of the models have very little focus on the reliability. A perspective comparison has been presented in the paper work for the traditional applications and the SaaS applications. A number of desired characteristics have been outlined to increase the configurability, portability and reliability of the SaaS applications designs.***

Keywords— Design of SaaS, Mashups, Modularization, NoSQL, SaaS DRESS code , Service Oriented Architecture.

I. INTRODUCTION

The cloud concept has to be viewed as an integration and development of the technologies from the past rather than as a new technology [2].

The three distinct categories of Cloud computing often called as Cloud stack are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [3]. The demand for Cloud has always seen a steady growth with the development of technology. Cloud market is believed to be initiated in 2005 with nearly $26 million investment with 5 cloud related investments. According to Forrester, the Cloud computing market was valued at $40.7 billion in 2010 and is expected to grow to $241 billion by 2020 [5].

The main concentration of this paper will be on SaaS and its design. The SaaS applications are designed for end users (i.e.) they are in the application layer of the stack. These are the applications that have an integrated code base and are delivered as service to consumers simultaneously, securely over the Internet [2]. National Institute of Standards Technology (NIST) defines SaaS as the capability provided to the consumers to use the provider's application running on the cloud which are accessible from various client devices through thin client interface such as web browser [1]. Examples of SaaS applications are web based email, Google sites, Salesforce CRM, Youtube video streaming etc. SaaS offers advantage for both service providers and consumers. The investment done by the providers on a single hardware is used to host application that are shared multiple users (i.e.) clients and from the consumers perspective the applications can be used by them at their convenience without the overhead of maintenance [12]. The base of SaaS could be traced back to 1960s, when IBM and other mainframe providers offered computer power and databases to banks and big organizations as utility computing. This was followed by the ASP. This is an innovation due to the advent of Internet in 1990s. ASPs offered services of hosting and managing small specialized business application. Due to the massive development in the speed of Internet and also due to the increased Internet customer base around 2003s, SaaS became popular in delivering web based on-demand software subscription basis. Usage of SaaS shifts the infrastructure demands from the organization to the service provider. The rate of SaaS application changes is high. Quick and iterative development cycles that suit different business model is expected from SaaS. Due to this reason SaaS always supports agile development life cycle [4].

This paper provides an outline for an efficient design for SaaS application so as to leverage the complete advantage for sustainable business. The paper is divided into six sections with the section II describing the relation between SaaS and Service Oriented Architecture (SOA). Section III deals with a brief discussion on traditional and Saas design comparison. Section IV discusses about some of the related work on the SaaS design and section V gives the desired and some of the existing design features of SaaS application. The final section VI holds the conclusion of this paper.

978-93-80544-12-0/14/$31.00

II. SAAS AND SOA

The Cloud applications (SaaS) are based on SOA. Designing and developing software with interoperable services feature is the main principle of SOA. SaaS is a software delivery model and SOA is a software construction model [8]. The guiding principles of SOA are standardized service abstraction, loose coupling, reusability, composability, statelessness, service contract and granularity. SOA concentrates on architectural design which enables application development to be based on service discovery and composition. It can also be viewed as an architecture that uses the model of building applications from the interaction of loosely coupled, coarse grained autonomous software units called services [10]. The cloud computing on the other hand focuses on effective delivery of services through efficient sharing of resources through virtualization and load balancing. Service Oriented Software Engineering (SOSE) incorporates the working style of SOA and Cloud [6]. In services computing context the software is viewed as a service with features such as well defined standard interfaces, platform independent with a URL, stateless and XML usage for function description. Standard bodies such as Organization for the Advancement of Structured Information Standard (OASIS) and World Wide Web Consortium (W3C) have established a variety of protocols and service interfaces for developing loosely coupled, reusable, composable, stateless and discoverable applications using SOA. Some of the protocol and interfaces used in SOA are given in Table 1 [6].

TABLE I. : PROTOCOLS AND INTERFACES USED IN SOSE

Protocol or Interface	Purpose Description
XML	Used for textual data representation.
Java Script Object Notation (JSON)	Used for serializing and transmitting secure data over network.
Simple Object Access Protocol (SOAP) / Representational State Transfer (REST)	Used for service invocation from remote location across network and platforms.
Web Services Description Language (WSDL)	Used for the description of service functions and interfaces.
Universal Description, Discovery and Integration (UDDI)	Used for the automatic publishing and discovery of services.
Electronic Business XML (ebXML)	Secure & consistent use of business information.
Business Process Execution Language (BPEL)	Used for the co-ordination of the workflow.

The Cloud application such as SaaS includes both software components used by the developers as well as the consumers. These are service oriented programs as the integration with other SaaS applications is inevitable and the service oriented feature will ease out the integration process. The combined working of SOA and Cloud will facilitate each other to strengthen their vulnerable areas. The Quality of Service (QoS) of loosely coupled services challenge of SOA can be handled by the resource allocation and virtualization feature of Cloud computing. Application deployment and maintenance are the areas in which SOA lacks behind and can be assisted by Cloud using its service virtualization. The Cloud challenges of providing interoperability across different clouds can be assisted with the standard protocols. The rapid development and adaption of fast changing business challenge of Cloud can be solved using the dynamic service discovery features of SOA as the software development process in SOA is highly flexible due to publish and reuse feature for application, business models and application templates [6]. Adopting SOSE for cloud application development has some challenges due to the development and deployment features of Cloud.

III. DIFFERENCE BETWEEN SAAS AND TRADITIONAL APPLICATION DESIGN

Software Engineering is divided into generations and SaaS fits in to the third generation of it which is viewed as development + execution + automated runtime management of resources sharing and security. Rapid customization and deployment is the key aspect of this generation [11].

SaaS design differs from the traditional web based application as the service providers play crucial role in it. The traditional application development deals mainly with the functional requirements as they are deployed as on-premise application with implicit security and control operations. The non-functional requirement of SaaS differs from traditional application. Traditional application development is hardware specific where as SaaS provides integrated solution over the Internet. Frequent code changes and customization is needed in traditional application which is replaced by a single code base in SaaS. The innovation cycle of traditional applications is annual where some SaaS applications need weekly innovation and changes. Technical consulting is required in traditional application developments but functional and business consulting is required for SaaS development [9]. Due to wide usage the Saas developers provide services with different levels of granularity. The SaaS development should include R&D activities to cover the changing consumer needs, rigid control on operations such as security, performance, logging and monitoring and deployment automation along with high customer service activity to cater to the needs of all consumers across the globe [9]. SaaS product development does not only deal with developing a software product alone but it deals with running and maintaining the software on behalf of the consumer. The value chain in the customer view also changes from parallel transformation to serial transformation [16].

The success of SaaS design highly depends on the way it is built on. Normally any Cloud service needs to be built on the fundamental principles such as Discoverability, Reachability, Economy, Scalability and Supportability. These apply to SaaS also and are also called as SaaS DRESS code [7]. SaaS design is expected to be agile, dynamic and highly flexible. The SaaS design differs from traditional design in terms of architecture, database partitioning, database architecture, scalability issues, user interface design, use of APIs and workflow [12] [13].

IV. RELATED WORK

Jiang et. al. have designed SaaS based on model driven approach and SOA with model layer describing the data, business process and user interface and core model driven engine layer which is in-charge for parsing models [17]. Tsai et. al. have proposed layered service oriented cloud computing architecture with inclusion of a layer named Cloud Ontology Mapping Layer that is used to mask the differences among the cloud service providers which helps in easy migration of cloud

applications [19]. Cloud agnostic and cloud aware are the two designs given by Sodhi and Prabhakar. In the former the application is developed like a traditional application without any concern about the cloud deployment and cloud aware design which is based on the understanding of the highly distributed, virtual and scalable platform to exploit the non functional quality attributes of the application [15]. Guochun Tang has designed four layers of SaaS maturity model. Using the Iterate and Rational Unified Process model, a five layer design has been proposed for SaaS which includes target layer, configuration layer, business logic layer, realizing layer and deployment layer [16]. Service Oriented Software Engineering view has also been proposed which the cloud service providers will publish their services in the service directory cloud which would then be discovered and composed by the cloud application developer to build a SaaS application [6]. A novel design pattern called Platform Level Aspect-Orientation has been proposed utilizing the property "software abstraction of hardware resources" of the virtualized platform for examining, controlling and monitoring the resources [18].

V. DESIRED DESIGN FEATURES OF SAAS

SaaS applications which are based on SOA are mostly assembled from the components such as web services, APIs, workflows and business rules that are available rather than being built from the scratch. The traditional way of synchronizing data using extract, load and transformation method that involves huge cost and process definition is being replaced by "mashups". These are the new class of light weight solutions having minimal data movement between the systems with real time and on-demand integration facility to provide consolidated and cross silo information to the consumer [10].

Compartmentalization is a feature that is inevitable in a SaaS product which deals with separating the parts of a system such that failure of a subsystem has minimal or no impact on the other parts of the system. This is achieved in SaaS using time-tested concepts of "separation of concern" and "abstraction" [10]. The advantage of compartmentalization leads to security issues which are dealt with distributed responsibility and trust partitioning. Modularization and reusable user interface designs have saved 85% of the front-end development process which in turn contributed to the reduced cost and faster product release in the SaaS market [14]. The key characteristics of SaaS - such as configurability, security, multi-tenancy and integration can be achieved easily through designing the SaaS product by following the metadata architecture. Metadata of an enterprise defines the complete working of an enterprise including details to represent workflow, policies, logic and data is stored in XML format or in the database. Usage of metadata in developing a SaaS product includes the advantages of reusability, faster time-to-value and consistency [10]. Aspect oriented programming and declarative programming can be used to implement the requirement of a Saas product.

SaaS product having a global reach needs to have customization of the same service according to the different consumer needs. Product polymorphism is expected to be done in a fast pace. This is achieved in a SaaS product with the help of an engine that triggers different workflow based on the business rules of the consumer. Involving a light weight open source rule engine and a work flow component with a clear separation between the two is the desired solution to enhance reusability [10]. SaaS product users have to tolerate less product frustration on comparison with on-premise application. This is achieved through modularization and facility for customization and this has also helped in lowering or complete avoidance of customer churn [14].

Mere deployment of an application on the cloud platform will not bring in scalability. The suitable architectural design should be done to harness the properties of the cloud oriented computing platform [18]. Maintainability and modifiability of SaaS are related to the design and development process maturity [15]. Some of the considerations for a good SaaS design are [13]

- Configurable interface design.
- Distributed infrastructure and platforms.
- Device independent user interface.
- Identification and representation of service through the use of APIs
- Endless upgrades

No schema and NoSQL are the preferred components for data usage in SaaS product as the main aim is not to store the business logic in the database. NoSQL works on BASE (Basically Available, Soft state Eventual Consistent) property and not on ACID (Atomicity, Consistency, Isolation and Durability) property. The characteristics of NoSQL are that they neither have fixed column table relationship nor SQL interface for data manipulation. APIs are used to define schemas and they load the data directly. The main reason for using this for maintaining data is because of their scalability, fault tolerance, availability, security, privacy, easy query interface and interoperability. Some of the features of RDBMS are also used to leverage their advantages. For example views are used to have a secure access of the data from the underlying table. The tenants can be given permission to views alone instead of giving access to the whole table. Sharding, a concept of splitting an entity and storing in multiple databases is an approach in SaaS to build scalable products with higher availability and faster queries. Amazon simple DB, Yahoo's PNUTS, Mongo DB and CouchDB are some of the examples of popular NoSQL databases. These also have some disadvantages of not providing transactional integrity, querying, SQL usage and easy connectivity with the popular Business Intelligence tools. A compromising solution for this is the usage of MegaStore which is a blend of SQL convenience and NoSQL scalability. It is a flexible data model with user-defined schemas, full-text indexes and queues [20].

Elimination of vendor lock-in can be guaranteed if the SaaS applications are based on customization and open source standards. A good SaaS design should have iterative nature incorporated into it as the upgradation is a perennial process for a SaaS product. SOA is an inevitable base for a SaaS design due to its synchronization between business and technology, loose coupling, increased interoperability and federation [10].

VI. CONCLUSION AND FUTURE WORK

This paper has outlined the important points to be considered during the design of SaaS applications. The intention is to deliver a SaaS product to consumers in such a way that they have to be able to utilize the product without any tolerance to the unwanted modules. Different levels of detailing of a single module are to be presented for the user to choose the required one according to their needs. Elimination of vendor lock-in and removal of vendor change overhead in case of business development or workflow change should also be the key point of consideration. The current need is to have a user friendly, scalable and customizable SaaS product with assured reliability. The further work is to create a model that incorporates the desired characteristics mentioned in the paper.

REFERENCES

[1] Mell, P., & Grance, T. (2011), "The NIST definition of cloud computing", special Publication 800-145, National Institute of Standards and Technology, available at http://csrc.nist.gov/ publications/ PubsSPs.html#800-145.

[2] Luit Infotech article, "Difference between the ASP model and SaaS model", retrieved from www.luitinfotech.com/kc/saas-asp-difference.pdf on October, 2013.

[3] "Understanding the Cloud Computing Stack: PaaS, SaaS, IaaS", Cloud U understanding article from www.rackspaceclouduniversity.com retrieved on Feb 2013.

[4] Haojie Hang and Ogheneovo Dibie, "Software as a Service", 2012, retrieved from www.cs.colorado.edu/~kena/classes/5828 /s12/presentation-materials/dibieogheneovohanghaojie.pdf on August 2013

[5] "Is Cloud Computing and Storage Environmentally sustainable" an article retrieved from www.earthtechling.com/2013/09/is-cloud-computing-and-storage-environmentally-sustainable/ published on Sep., 2013 retrieved on October 2013.

[6] Stephen S. Yau and Ho G. An, "Software Engineering Meets Services and Cloud Computing", an article published by IEEE Computer Society and InfoQ, 2011, pp 47-53

[7] Hanu Kommalapati and William H. Zack, "The SaaS Development Lifecycle", a white paper from www.infoq.com, Oct 2011, accessed on Sep. 20, 2013.

[8] Phillip A. Laplante, Jia Zhang,Jeffrey Voas, "What's in a name? Distinguish between SaaS and SOA". IT Pro May/June 2008 article published by IEEE Computer Society.

[9] Jaroslav Gergic, "Software Engineering in the age of SaaS and Cloud Computing", a whitepaper from www.gooddata.com, August 2013, retrieved on Sep, 2013.

[10] Karthik Vishwanaathan, "Right Engineering SaaS : Successfully deploying Software as a Service Models", a white paper from Aspire Systems, retrieved from www.sandhill.com/assets/pdf/Right_Engineering_SaaS.pdf on September 2013.

[11] Yinong Chen, "A Dream of Software Engineers – Service Orientation and Cloud Computing", 6th IEEE Joint International Information Technology and Artificial Intelligence Conference Keynote, accessed from www.public.asu.edu/~ychen10 /activities/jicsit11/ChenKeynote11.pdf on August 2013.

[12] Eyad Saleh, Nuhad Shaabani, and Christoph Meinel, "A Framework for Mitigating Traditional Web Applications into Multi-Tenant SaaS", INFOCOMP 2012, The Second International Conference on Advanced Communications and Computation.

[13] David Brumbaugh, "Developing Cloud based SaaS Applications", an article from news.dice.com, June 2013, accessed on September 2013.

[14] Catalyst Resource white paper, "SaaS UI Design Principles", Oct 2011, retrieved from www.catalystresources.com on August 2013.

[15] Balwinder Sodhi and T.V. Prabhakar, "Application Architecture Considerations for Cloud Platforms", proceedings of 2011 Third IEEE International Conference on Communication Systems and Networks (COMSNETS 2011), January 2011, pp. 1-4.

[16] Gouchun Tang, "The Saas Architecture and Design on the Five Layers Driving Model", Journal of Management and Engineering 02 (2011) pp. 61 - 65.

[17] X. Jiang, Y. Zhang and S. Liu, "A Well designed SaaS Application Platform Based on Model-driven Approach", Ninth International Conference on Grid and Cloud Computing, IEEE Computer Society, 2010, pp. 276-281

[18] Balwinder Sodhi and T.V.Prabhkar, "Cloud-Oriented Platforms: Bearing on Application Architecture and Design Patterns", 2012 IEEE Eighth World Congress on Services, 2012, pp. 278-284.

[19] W. Tsai, X. Sun, J. Balasooriya, "Service-Oriented Cloud Computing Architecture", Seventh International Conference on Information Technology, IEEE, 2010. pp. 684-689.

[20] Json Baker, Chris Bond, James C. Corbett, JJ Furman, "Megastore: Providing Scalable, Highly Available Storage for Interactive Services", 5th Biennial Conference on Innovative Data Systems Research (CIDR '11), 2011, pp. 223-234.