Customization Issues In Cloud Based Multi Tenant SaaS Applications

Mala Dutta¹, Pritesh Gupta²

¹Assistant Professor
Department of Computer Engineering
Institute of Engineering and Technology DAVV
Khandwa Road, Indore, India
mala1958@gmail.com

²Teaching Assistant
Department of Computer Engineering
Institute of Engineering and Technology DAVV
Khandwa Road, Indore, India
pritesh.gupta21@gmail.com

Abstract: Cloud computing is growing rapidly, with applications in almost all the areas. SaaS (Software as a Service) is a modern approach to deliver large scalable enterprise software as a service on Internet. Cloud platform provides the scalability, availability and utility computing for services on Internet. There are many technical challenges involved in SaaS development. One of them is multi-tenancy, which allows single instance of software to serve multiple organizations by accommodating their unique requirements through configuration at the same time. SaaS architecture requires the both configuration and some level of customization to achieve higher maturity model. As every organization requires a Management Software to run their business, as they have been using traditional on-premise software which usually requires many hardware and software resources. There are many organizations that cannot afford such investments and cloud computing is the best solution. In this paper, we will explore the customization issues and challenges to SaaS vendors; we proposed an application model in which we can use xml data type in database to make customizations more flexible.

Keywords: Cloud Computing, Multi-Tenant, Software as a service, Customization.

1. Introduction

With the widespread implementation of information technology, the demand for informatization is no longer a requirement unique to large enterprises. However, there are many substantial challenges faced by these entities and software developers, for example, the numerous amounts of entities, the scattered geographic location, insufficient technological infrastructure within the entities and the lack of funding. Software providers have to address the above issues to successfully develop and deploy software that fits these small and medium entities. Software as a Service (SaaS) is both software delivery model and a business model. Software delivery model involves delivering high scalable multi-tenant application through internet to different organizations which are just a bit different in their business functionality [1][2]. In this new multi-tenant approach, a customer has to log on software provider’s web site to use software product for their business instead of installing it on own premise. Customers do not have to worry about infrastructure like Network, Servers, and Operating Systems etc. to keep applications running. Services are provided through internet, so customers can use it from anywhere at any time. The pricing model used in SaaS is subscription based and customers need to pay on 'pay per use' basis. Customer is relieved from installation of product trials, functionality demonstration, and long term agreement to invest money. Market benefit of SaaS is to cover a large pool of clients to deliver standard domain functionality with a single instance of an application [2]. SaaS software and traditional software are different in terms of application architecture, hosting and deploying environment, required developing skill set and other technical challenges. Some of challenges that SaaS developer has to address are Multi-tenancy, Subscription, Monitoring and billing, Security and privacy, Scalability, high availability and reliability, Managing and Administration for separate tenants, Runtime per tenant customization, Customizable GUI, Customizable Business Logic.

2. Related Work

Multi-tenancy is one of the most important concepts for any SaaS application. Based on it, real cloud computing aims for “better resource utilization” for SaaS providers and "Pay as you go" for consumer [3]. There are many ways to develop SaaS application based on hosting environment, architecture to follow, and available development skills. One can develop SaaS and deploy it using proprietary online framework like Google AppEngine, Redhat’s, Openshift, CloudBees, Microsoft Azure etc. SaaS also can be developed using core programming framework like JDK for java and .NET. Here, one has to focus on monitoring, billing, subscription and most important configurability per tenant. There have been some
researches on SaaS and its configuration, customization and multi-tenant. Nitu [4] addresses the issue of how to effectively and efficiently support configurability in SaaS software and proposes SaaS architecture to support configurability. A set of research issues in the design, testing and maintenance of multi-tenant SaaS systems is outlined in [5]. Cor-Paul Bezemer and Andy Zaidman [6] discuss some potential challenges in implementation and maintenance of multi-tenant SaaS systems and present an architectural approach to separate the multitenant configuration as much as possible. A conceptual architecture of SaaS platform that enables executing of configurable and multi-tenant SaaS applications is proposed in [7]. Craig D. Weissman and Steve Bobrowski [8] give an insight on how multi-tenant is being handled in their application framework - the Force.Com multi-tenant internet application development platform. Frederick Chong, Gianpaolo Carraro, and Roger Wolter [9] describe the approaches to design SaaS data structure for multitenant in order to take advantage of the benefits of SaaS and to ensure security of tenant’s business data. A novel meta data driven schema sharing data storage architecture for multi-level customization in SaaS applications is designed in [10]. However, System integrator, Panorama Consulting Solutions, conducted the research survey during the four-month period of September, 2012 to January, 2013. The results are based on data from 172 respondents who completed a survey on the Panorama website. Seventy-one percent reported revenues of $300 million or less and 21 percent of respondent companies had revenues of $1 billion or higher [11].

Cost, duration, and benefit summary. Although project duration and cost fluctuate from one year to the next, three points stand out about the current data:

- Over 50 percent of projects experienced cost overruns
- Over 60 percent experienced schedule overruns
- Fully 60 percent of respondents received under half of the expected benefit from their SaaS implementation

The following table reports Panorama data for the last four years:

<table>
<thead>
<tr>
<th>YEAR</th>
<th>COST</th>
<th>% OF COST OVERRUNS</th>
<th>DURATION</th>
<th>% OF DURATION OVERRUNS</th>
<th>% RECEIVING 50% OR LESS BENEFITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012</td>
<td>$7.1MM</td>
<td>53%</td>
<td>17.8 months</td>
<td>61%</td>
<td>60%</td>
</tr>
<tr>
<td>2011</td>
<td>$10.2MM</td>
<td>56%</td>
<td>16 months</td>
<td>54%</td>
<td>48%</td>
</tr>
<tr>
<td>2010</td>
<td>$5.5MM</td>
<td>74%</td>
<td>14.3 months</td>
<td>61%</td>
<td>48%</td>
</tr>
<tr>
<td>2009</td>
<td>$6.2MM</td>
<td>51%</td>
<td>18.4 months</td>
<td>36%</td>
<td>67%</td>
</tr>
</tbody>
</table>

3. Cloud Computing

3.1 Definition

Since the Cloud computing paradigm has been conceived several definitions have been given. Some of them focus on on-demand dynamic provisioning of processing and storage resources, others emphasize the service-oriented interface and the exploitation of virtualization techniques. The National Institute of Standards and Technology (NIST) have given a complete reference definition [12]. NIST defined Clouds as follows: “Cloud computing is a pay-per-use model for enabling available, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” Moreover, according to NIST: “Cloud model promotes availability and is comprised of five key characteristics, three delivery models, and four deployment models.”

The key characteristics of Clouds are: On-demand self service, ubiquitous network access, location independent resource pooling, rapid elasticity, and pay per use. Fig. 1 summarizes the main aspects of Cloud computing system both from the technical side and the business side [13].

![Figure 1. Main aspects of Clouds (from [13]).](image)

The delivery models of Clouds are very important because they define three different types of Cloud computing systems:

- **Infrastructure as a Service (IaaS).**
  The capability provided to the user is to rent computing, storage, networks, and other computing resources where the user is able to deploy and run software, which can include operating systems and/or applications. The user does not manage or control the hardware Cloud infrastructure but has control over operating environments, storage, deployed applications, and possibly select networking components. Examples for commercial Cloud infrastructures are Amazon EC2 and Rackspace.

- **Platform as a Service (PaaS).**
  The functionality provided to the user is to deploy onto the Cloud infrastructure consumer-created applications using programming languages, compilers and toolkits supported by the provider (e.g., Java, .Net). The consumer does not manage or control the underlying cloud infrastructure, network, servers, operating systems, or storage, but the consumer can control the deployed applications and possibly the application hosting environment configurations.

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

### 3.2 Cloud Deployment Models

About five years ago, when the first Cloud infrastructure has been deployed by Amazon, the online bookseller company that took the decision to start a new business selling computing resources to companies and private users, the only deployment model was the **Public Cloud** one. It is a pay-per-use IaaS Cloud infrastructure that is owned by an organization selling Cloud services to the general public or to enterprises. Thus, it is public because it can be rent by anyone for developing and/or running any kind of applications. To use Amazon services, users must provide a credit card account and can spend from few cents to thousands or millions of dollars depending on the number of used resources and the usage time. After this early Cloud version, other deployment models different from **Public Clouds** have been designed and implemented (see Fig. 2):

- **Private Cloud.** The Cloud infrastructure is owned or leased by a single organization and is operated only for that organization. No public access to it is permitted. This model can be used in case of strict data privacy and/or security requirements.

- **Community Cloud.** The Cloud infrastructure is shared by a limited number of organizations and supports a specific community that has shared concerns (e.g., goals, security requirements, policy, and compliance issues).

- **Hybrid Cloud.** This fourth class of Cloud infrastructure is a composition of two or more Clouds (private, community, or public) that although they are unique entities, are combined together by standardized or proprietary technology that enables data and application portability (e.g., Cloud federation).

![Figure 2](image)

Deployment models for Clouds. Cloud computing is the most recent results of the advancement of several computer technologies both from the hardware side, such as virtualization and multi-core architectures, and from the software side like cluster computing, Grid computing, Web services, service-oriented architectures, autonomic computing, and large-scale data storage.

**Virtualization** is an essential technological characteristic of clouds which hides the technological complexity from the user and enables enhanced flexibility (through aggregation, routing and translation). A study shows that the average CPU utilization per physical server is less than 10%. The main aim of virtualization is to maximize the CPU utilization by using virtualization softwares. So Virtualization distracts the Operating System and Applications from Hardware.

More concretely, virtualization supports the following features:

- **Ease of use:** through hiding the complexity of the infrastructure (including management, configuration etc.) virtualization can make it easier for the user to develop new applications, as well as reduces the overhead for controlling the system.

- **Infrastructure independency:** in principle, virtualization allows for higher interoperability by making the code platform independent.

- **Flexibility and Adaptability:** by exposing a virtual execution environment, the underlying infrastructure can change more flexible according to different conditions and requirements (assigning more resources, etc.).

- **Location independence:** services can be accessed independent of the physical location of the user and the resource.

### 3.3 Examples of Cloud System

As we can infer from the previous descriptions, the Cloud computing paradigm represents an advancement of the existing computing services available over the Internet. In particular, Cloud infrastructures adopted the Web services paradigm for delivering new capabilities beyond the Web capability. Several companies set up large Cloud facilities and built programming environments where developers can program applications as Cloud software services. Just to mention some example, Amazon on his EC2 and S3 Cloud platforms implemented Elastic BeanStalk, Microsoft implemented .Net technology on Azure, Google provides the AppEngine, and VMware has Cloud Foundry. On the other side, the research community developed open source software that can be deployed and configured on servers, computer farms or data centers for implementing private, public, community or hybrid Cloud infrastructures or for inter-Cloud computing facilities. Examples of these systems are OpenNebula, Eucaliptus, OpenQRM, Puppet, and OpenStack. These open source software projects are also working to systems and services that allow Cloud-to-Cloud interoperability and federation.

### 3.4 Comparison

SaaS is on-demand software and has significant differences with on-premise software in aspects of delivery mode, apply scope, maintenance, price etc. Table II shows the comparison between SaaS and traditional software.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>SaaS</th>
<th>Traditional Software</th>
</tr>
</thead>
</table>

---

*Mala Dutta* IJECS Volume 3 Issue 4 April, 2014 Page No.5447-5452
<table>
<thead>
<tr>
<th>Total Cost of Ownership (TCO)</th>
<th>low</th>
<th>high</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Mode</td>
<td>subscribe, plug in</td>
<td>separate installation</td>
</tr>
<tr>
<td>Apply Scope</td>
<td>similar customers</td>
<td>specific customer</td>
</tr>
<tr>
<td>Specific Demand</td>
<td>flexible configuration or extension</td>
<td>specific re-development or upgrade</td>
</tr>
<tr>
<td>Maintenance</td>
<td>fixing a problem for one customer fixing it for everyone</td>
<td>fixing problem for every customer respectively</td>
</tr>
</tbody>
</table>

3.5 Multi-Tenancy

Multi-tenancy is a highly essential issue in cloud systems, where the location of code and/or data is principally unknown and the same resource may be assigned to multiple users (potentially at the same time). This affects infrastructure resources as well as data/applications/services that are hosted on shared resources but need to be made available in multiple isolated instances. Classically, all information is maintained in separate databases or tables, yet in more complicated cases information may be concurrently altered, even though maintained for isolated tenants.

5. Handling Customizations for Separate Tenants

This section describes how customization can be handled by using XML.

In terms of scalability, the ability to add more customers is extremely high in a multi-tenant model as compare to a single tenant model.

4. Customization Issues

Developer centric workflow engines and business rule engines are quite popular, very complex business rules and workflows can be created by developers and incorporated in Java applications but if we want it in a multitenant and also want some of the business rules and workflows to be customized by a non-IT person let’s say the Leave Management workflow has to be customized by an HR Manager or a Performance Appraisal System workflow being customized by the administrator etc., then it becomes little more challenging to be able to provide the flexibility for the end users to customize the workflow or business rules. So application has to remember that information in the context of which tenant customized what? And then during runtime we need to apply that customization of the business rules or workflow, so that the application behaves differently for those particular tenant users. As all along we have been using single code base, so this is the challenging business requirement that comes up in many BPO applications, KPO applications that requires multi tenancy because they deal with multiple customers and they have to configure some of these aspects without having to go to the IT for training the code.
The figure 4 shows that a tenant has many clients, and each client has a different profile. The tenant ID (TID) column in both tables is used to identify the tenant. Rows 1 and 3 in the left table belong to one tenant with a TID of 1; rows 2, 4, and 5 belong to another tenant with TID of 2. Suppose tenant No. 2 (TID = 2) has a business rule that indicates that users cannot input telephone information, so storing information about telephone numbers would not be applicable to tenant 2. However, in a multi-tenant environment where tables are shared (case 1), the SaaS company needs to consider that other tenants do want to include telephone information. Using a traditional SQL database (left table), the SaaS vendor can create one fixed-column table, which includes columns for all possible cases of telephone numbers (cellphone number, home phone). Even if tenant No. 2 doesn't allow phone numbers in customer profiles, the column is still included. Therefore, there will be a lot of "holes" and dispersed data, as highlighted by the circles. In addition, say tenant No. 1 (TID = 1) wants to change its requirements so it doesn't store only cell and work numbers but also home numbers. In this situation, you may have to alter the table. However, if we follow the normalization rules for database design, we actually need to create a separate PHONE table. Then we'll have to move the data and change your applications so your SQL queries point to the new table PHONE and use join operations. So this method is not flexible. Right side table is the suggested method of handling customizations. The table in this case has only two columns, in the second column the customer information is stored using XML tags. Using XML, it is a lot more flexible to handle changes in the database schema. So each tenant can use separate XML schemas.

**6. Conclusion and Future Work**

In this paper we discussed about Cloud Computing, its delivery models, Virtualization, Multi-tenancy, Customization issues in developing Multi Tenant single database single schema SaaS applications and how to handle customizations in SaaS application development using xml data type in the database. So small enterprise SaaS developers need not be dependent on private framework. This way of SaaS application development considers performance while giving services to a large number of clients at the same time. Our future work will be to implement and develop one innovative application which can fulfill the demand of many customers.

**References**


Author Profile

Asst Prof Mala Dutta has completed her BE in Electrical Engineering from Govt. College of Engineering & Technology, Ravishankar University Raipur (C.G), India in 1981 and Master of Technology in Computer Science in 2000 from Institute of Computer Science & Electronics, Devi Ahilya Vishwavidyalaya, Indore, MP India. She has more than 12 years experience in industry/teaching and presently working as Assistant Professor in Computer Engineering Department, Institute of Engineering and Technology, DAVV University, Indore, MP, India. Her research interests include Object Oriented and Web Technology.

Pritesh Gupta has completed his BE in Computer Science from Truba College of Engineering and Technology Indore in 2011. He has 1 year of working experience in application development and presently pursuing his ME in Computer Engineering with specialization in Software Engineering from Institute of Engineering and Technology, DAVV University, Indore, MP, India. His research interests are Cloud Computing, Object Oriented and Web Technology.