Multi-Tenant Engineering Architecture in SaaS

Sunil Kumar Khatri
Amity Institute of Information Technology, AUUP, Noida, India

Himanshu Singhal
Amity Institute of Information Technology, AUUP, Noida, India

Khushboo Bahri
Amity Institute of Information Technology, AUUP, Noida, India

ABSTRACT
Multi-Tenancy in SaaS (Software as a Service) architecture is the concept leveraging cloud computing and virtualization which incurs cost efficiency. Modularity and customizability enhances the strength of multi-tenancy and business opportunities. With the growing business and competition, there arises a need to introduce an IT based technology to the system. Business process re-engineering and development of Enterprise Resource Planning (ERP) has revolutionized the way an enterprise system is build and executed and even more exponentially revolutionized with the introduction of multi-tenancy integrated with SaaS-based ERP system. The proposed architecture introduces the concept of fully modular system, where different modules can be implemented and configured according to the necessities of the user and further improved based on the requirements avoiding the related concerns.

Keywords
Multi Tenancy, Cloud Computing, Modularity, SaaS Architecture, Customizability, Extensibility, ERP.

1. INTRODUCTION
SaaS came with altogether a new idea in software service industry which transformed the way software is being delivered to customers. With the evolution of SaaS came an acceptance mainstream business model. SaaS turned out to be an on-demand software development platform, in cloud environment. It extended the business by unveiling the fact which eliminated the requirement of purchasing and maintaining severalized Information and Communication Technology infrastructure.

It is realised over a period of time that a fine SaaS vendor is one who makes data reliable and secure enduring customizability and extensibility. Level lower to SaaS, PaaS (Platform as a Service) placed the concept of virtualization into third generation category.

The major challenges that are being encountered while developing SaaS is to provide the customer with security, scalability and reliability. With this comes the major concerns such as highest order of customization and extensibility which provide the access to large business opportunities.

Business process re-engineering, identified many different processes running parallely in a single business such as human resources, manufacturing, supply chain management, finance, management accounting, project management, customer relationship management etc. All these contribute to different modules in an ERP. While establishing SaaS ERP, implementing multi-tenancy with highly modular approach opens a large and promising business market.

This centres the focus on multi-tenancy and demonstrates software architectural concern for implementing module - driven architecture for multi-tenant applications which inculcates few differentiating aspects.

2. CENTRAL IDEA OF SAAS: THE MULTI-TENANT ARCHITECTURE
There are different approaches being used to deploy the concepts of SaaS applications and their models in the cloud environment.

2.1 Multi-Tenant Architecture
Multi-tenant applications introduce 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 concern.

2.2 Maturity Models
SaaS can be explained by emphasizing on few important characteristics of a mature SaaS application.

Maturity is not an all-or-nothing proposition. An application can establish just one or two of these attributes and meet all necessary business requirements; in these cases the architects should not consider other characteristics. SaaS application maturity can be expressed using a model with four distinct levels. Each level is distinguished by enhancing it with the addition of one of the attributes.

2.3 SaaS application Server & Database Model
The main technical trade-offs that proves to be a challenge while designing the underlying definition, which remains transparent to the customer, are options to deploy application servers to serve multiple tenants and distributed customers with data across servers, virtual machine databases, schemas and tables according to a criteria such as security, scalability, performance, high availability and maintainability.

2.4 Server Deployment Models
Server deployment can be broadly categorized into four ways which can be considered by evaluating the customer requirement. Deploying dedicated servers can increase the cost but will incur high end data security. Other options can be shared virtualized application servers which has a dedicated application running on different virtual machines. Shared virtualized server shares the virtual machines as well as the application servers and allows the tenants to share the application and access them through separate session threads.

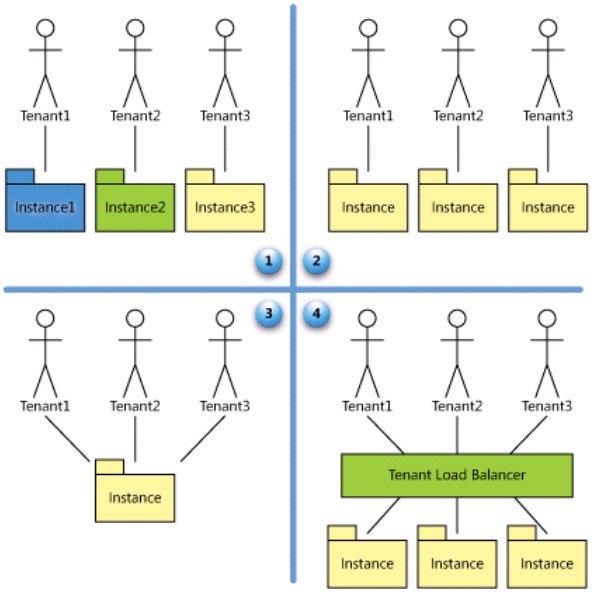

Figure1: Four level SaaS maturity model

2.5 Database Deployment

The different operational modes of deployment, depends on the infrastructure as Servers or hosts, Database and Schema. Data architecture needs an optimal degree of isolation for a SaaS application which depends on technical and business considerations, exponentially.

2.5.1 Separate Database

One of the simplest approaches to data isolation is storing tenant data in separate databases.

Resources and code are generally shared between all the tenants on a server, metadata relates each database with the correct tenant and thus data security is incurred.

2.5.2 Shared Database, Separate Schemas

It involves multiple tenants in the same database, with each tenant having its own set of tables, grouped into a schema, specifically for the every tenant. It moderate degree of logical data isolation for security-conscious tenants simultaneously supports a large number of tenants.

2.5.3 Shared Database, Shared Schema

All tenants share the same set of tables, and a Tenant ID associates each tenant with the rows that it owns. It provides lowest hardware and backup costs, because it allows us to serve the largest number of tenants per database server but security is the main issue.

3. MODULAR CONCEPTUAL APPROACH FOR MULTI-TENANT SAAS ENGINEERING ARCHITECTURE

3.1 Multi-tenancy Modular Design

Multi-tenancy modular design aims at the tenant access layer and database specifications of the user. It clearly identifies and ingrains the functional and non-functional isolation in terms of database specifications and thus the Tenant Access Layer (TAL).

It helps in isolating the functionalities and induces data security between different tenants. It also states the mapping of database with the TAL customizations.

3.2 Modular Modelling

Modelling defines Tenant Level Customization and Configuration (TLCC). It distinguishes and maps the relation between applications and services

With this there arises a need of meta-data driven architecture to be implemented simultaneously. Tenant meta-data can be configured for tenant subscription to the services.

3.3 Modular Injection

Modular injection focuses at adaption of pre-existing services while adding new modules. Changes which need to be made at all the nodes are self-mapped and are also synchronized with deployed modules. . It also induces Inversion of Control (IOC) while modulating the software as a service.

It is not always necessary for the users to make all the business processes IT driven at the very first go. Extensibility thus gives the opportunity of implementing package module wise. Injection thus helps in self-synchronizing different modules with each other.

Modular injection is an essential concept while implementing modular approach. There are various business processes defining each module which user may not be willing to implement at one go. The injection helps in self-synchronization of module with each other, service layer and data base without a need to make changes in technical base code layer.

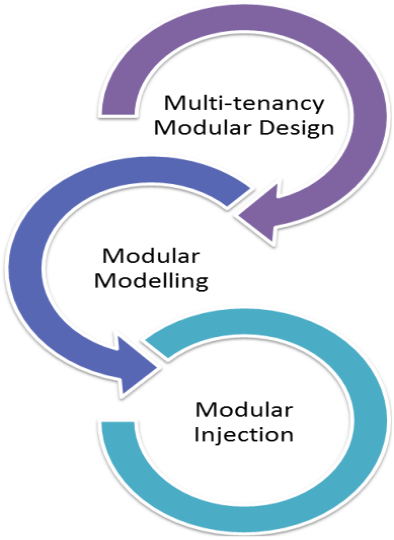

Figure2: Modular Conceptualization

4. PROPOSED ARCHITECTURE ON MULTI-TENANT SAAS ENGINEERING

The proposed architecture explains the requirements to build a truly robust Multi-Tenant SaaS solution on the basis of Modular approach. This architecture identifies the solution as a five layered design namely Tenant Access Layer, Service Layer, Tenant Level Customization and Configuration Layer, Business Functionality Layer and Technical Base Layer. The

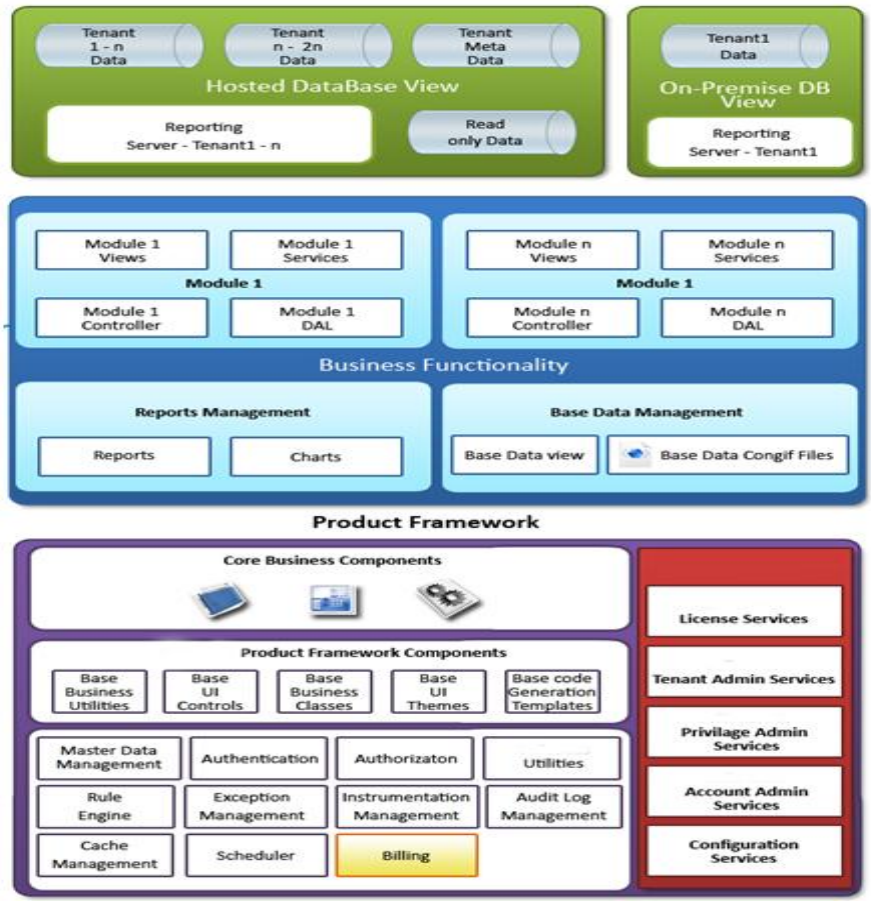

Figure3: N-Tiers and Modularity

major concern while designing SaaS is security, scalability, customization and extensibility.

The first four layers are semi-permeable in nature i.e. user interacts with each layer stepwise. Extensibility is the major issue while considering self-synchronization between all the modules, thus requiring agile development platform. The defined modular conceptual approach gives the platform for agility. Functionality of each layer identifies its implementation with the help of multi-tenancy modular design, modular modelling and modular injection to establish fully modular SaaS application.

Tenant access layer gives the user the additional opportunity to customize the interface, the look and feel of the product. Ajax based browser technology at tenant access layer helps in maximizing the system speed, minimizing the browser refreshes, improves system interactivity and improves the user experience. One of the major characteristic of SaaS is pay per use. While customization is implemented, there arise a need to map them with the services and functions such as billing and metering, configuration, tenant provisioning, authorization and authentication and security.

Tenant Level Customization and Configuration layer enables user to customize the fields according to the requirement of the business logics which may differ for each tenant. These features are customized by the tenant at TAL without interacting with the technical code specifications. It extends the capability of application of implementation of custom business logic, custom work flows, custom reports and validations. As soon as the customization of business logic is designed and thus the interface, fields in database are self-implemented using meta-data driven architecture. An extension and meta-data table stores all the information about every custom field defined by each tenant.

The technical base code layer is one layer which has no access to the tenant requirement and is least affected by any of the operation related to customizability and extensibility.

Designing a fully compatible code layer which supports security, scalability, customizability and extensibility according to tenant requirement and at the same time requires no need of modification is the major concern.

4.1 Security

A SaaS architect is responsible for building adequate data protection as well as defines multiple levels that complement each other to counter both internal and external threats. Data protection can be implemented through filters or firewalls, access control lists and encryption.

4.2 Scalability

For a SaaS application, scalability is important, because one will have to support data belonging to all the customers. Databases can be extended (by moving to a larger server that uses more powerful processors, more memory, and faster disk drives) and deneaned (by partitioning a database onto multiple servers). Different strategies are appropriate when scaling a shared database versus scaling dedicated databases.

The most common techniques to scale database are dynamic provisioning, partitioning and combination of both.

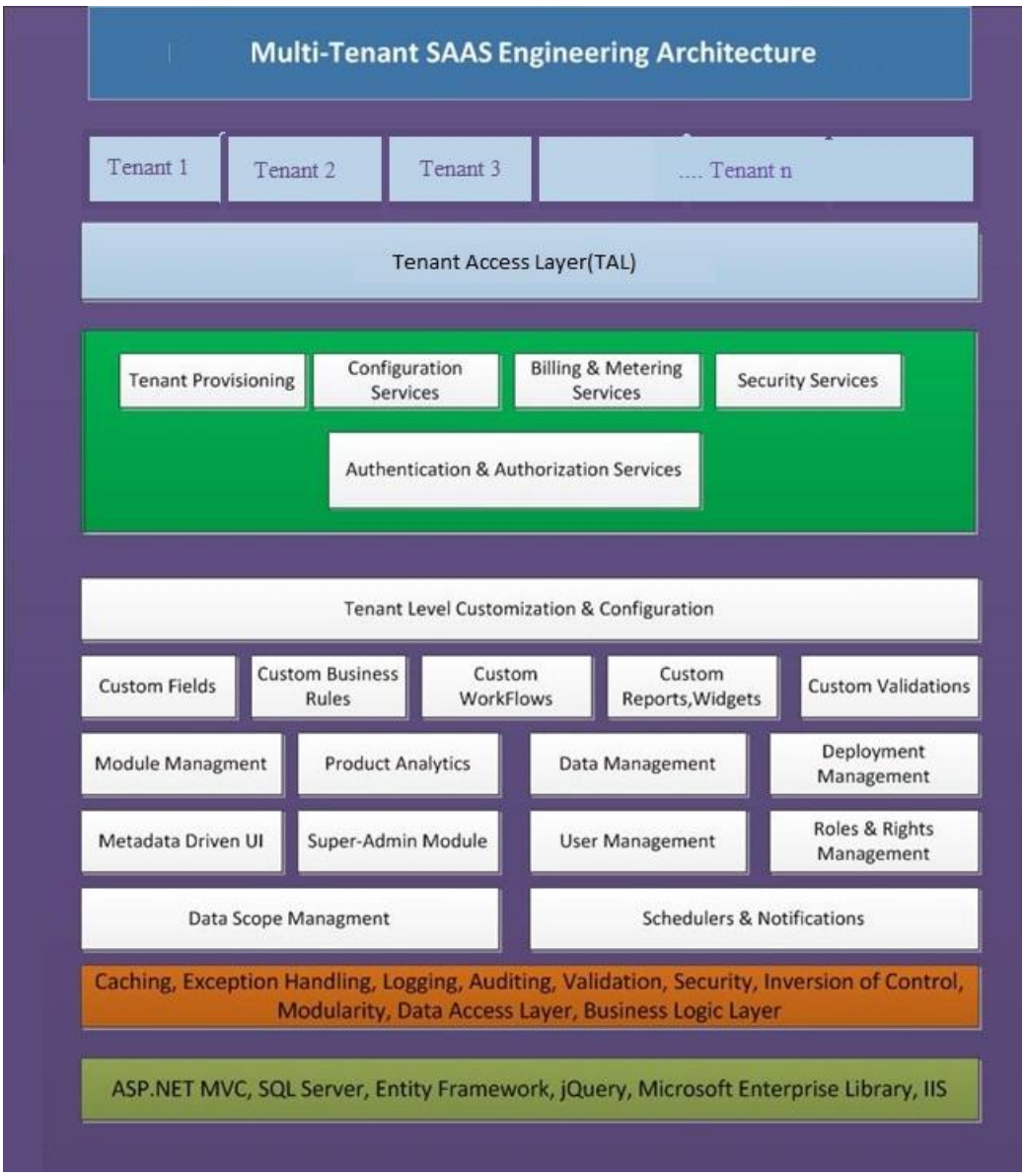

Figure 4 Proposed Architecture on Multi-Tenant SaaS Engineering Architecture

This technique is being deployed specifically to handle load balancing to handle the multi-dimensional access and manage the user traffic to optimize resource utilization, throughput and response time. The motive is to respond all the user requests with minimum response time by routing the request to best available data centres.

4.3 Customization

It may not be wrong to say that each tenant may have a different set of requirement and data structure. One defined template cannot cater all of them. It is critical to deploy database instance and design schema so fields, type and constraints can be created, removed or modified without interrupting the access to the databases.

There are several known techniques to extend existing tables:

4.3.1 Customized predefined fields

When records from different tenants are intermingled within the same set of tables

4.3.2 Customized predefined Tables

Allow the tenant to create new fields and storing specific data into a separate table which has already some predefined labels and data types.

4.3.3 Dynamic fields

It makes sense in the case of the schema is not shared, allow the tenant to add dynamically new columns to an existing table. To discuss this feature in detail requires a distinguished research field and thus all the consideration cannot be introduced.

Each tenant can view the application as per his specified customization of look and feel of the interface as well as logic. Definition of one tenant will not affect the definition specified by other tenants.

4.4 Extensibility

To access the maximum business opportunities and provide the customers with a high end flexibility to design their IT driven business processes inculcating n-tier and modularity while designing a SaaS backbone of Software as a Service.

This feature has been taken care while designing the Modular Conceptual Approach for Multi-Tenant SaaS Engineering Architecture.

N-tier architecture supports the customization for every tenant individually. It brings about isolation of all the details from and among the tenants.

The extensibility induces modularity among applications and services of the tenants and for a particular tenant also. Even the smallest service such as report generation can be modified and modulated according to the needs of isolation and requirement

5. CONCLUSION

The Multi-Tenant SaaS architecture is a contemporary development model which focuses on ingraining IT driven business processes. SaaS is a milestone in software delivery. Modular conceptual approach for SaaS architecture which caters to all the possible features at the customer end, such as security, scalability, reliability, customization and extensibility. The focus is to incur this modular approach to open the services for a wide spectrum of customer.

6. FUTURE SCOPE

Technically, it is difficult to implement such flexible code which hardly needs to be changed while the following functions are being performed: (a) customization according to multi-tenants as well as each individual organization's requirements, (b) selecting applications and modules (c) mapping them with all the services and finally (d) creating such robust data base. Still it is not impossible to achieve the same.

The proposed architecture aims at establishing a completely customizable and extensible SaaS ERP with a fully functional modular application platform.

The future work will focus on its validation.

7. REFERENCES

[1] Nitu, "Configurability in SaaS (software as a service) applications," in Proceedings of the 2nd India software engineering conference Pune, India: ACM, 2009.

[2] S. Merkel, "Parallels Software as a Service (SaaS)," p. 2.

[3] ComputerWeekly.com, "The Computer Weekly guide to Cloud Computing," 2010.

[4] F. Chong and G. Carraro, "Architecture Strategies for Catching the Long Tail," Microsoft Corporation, 2006.

[5] F. Chong, G. Carraro, and R. Wolter, "Multi-Tenant Data Architecture," Microsoft Corporation, 2006.

[6] A. Azeez, S. Perera, D. Gamage, R. Linton, P. Siriwardana, D. Leelaratne, S. Weerawarana, and P. Fremantle, "Multi-tenant SOA Middleware for Cloud Computing," Cloud Computing,

[7] R. Mietzner, T. Unger, R. Titze, and F. Leymann, "Combining Different Multi-tenancy Patterns in Service-Oriented Applications," Enterprise Distributed Object Computing Conference, IEEE

[8] J. Jing and J. Zhang, "Research on Open SaaS Software Architecture based on SOA," in 2010 International Symposium on Computational Intelligence and Design, Hangzhou, 2010, pp. 144

[9] B. Gao, D. C. J. Guo, Z. H. Wang, W. Hao, and D. W. Sun, "Develop and Deploy Multi-Tenant Web-delivered Solutions using IBM middleware: Part 3: Resource sharing, isolation and customization in the single instance multi-tenant application," IBM, 2009.

[10] Amelia Maurizio, James Sager, Peter Jones, Gail Corbitt, Lou Girolami, "Service Oriented Architecture: Challenges for Business and Academia", Proceedings of the 41st Hawaii

[11] John Fontanella, "B2B E-Business in the Supply Chain: New Services and Technologies Require Companies to Re-evaluate their Strategies", AMR Research, May, 2008.

[12] Frederick Chong, Gianpaolo Carraro, and Roger Wolterh http://msdn.microsoft.com/en-us/library/aa479069.aspx, June 2006.

Frederick Chong, Gianpaolo Carraro, and Roger Wolterh http://msdn.microsoft.com/en-us/library/aa479086.aspx, June 2006