Divyakant Agrawal
K. Selçuk Candan
Wen-Syan Li (Eds.)

LNBIP 74

New Frontiers in Information and Software as Services

Service and Application Design Challenges in the Cloud

Springer

Divyakant Agrawal K. Selcuk Candan
Wen-Syan Li (Eds.)

New Frontiers in Information and Software as Services

Service and Application Design Challenges in the Cloud

Springer

Volume Editors

Divyakant Agrawal
University of California at Santa Barbara
Department of Computer Science, Santa Barbara, CA 93106, USA
E-mail: agrawal@cs.ucsb.edu

K. Selçuk Candan
Arizona State University
School of Computing, Informatics
and Decision Systems Engineering
Tempe, AZ 85287-8809, USA
E-mail: candan@asu.edu

Wen-Syan Li
SAP China
Shanghai, 201203, China
E-mail: wen-syan.li@sap.com

ISSN 1865-1348
ISBN 978-3-642-19293-7
DOI 10.1007/978-3-642-19294-4
Springer Heidelberg Dordrecht London New York

e-ISSN 1865-1356
e-ISBN 978-3-642-19294-4

Library of Congress Control Number: 2011920956

ACM Computing Classification (1998): H.3.5, J.1, H.4.1, K.4.4, C.4

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Lecture Notes in Business Information Processing 74

Preface

The need for a book focusing on the challenges associated with the design, deployment, and management of information and software as services materialized in our minds after the success of the two consecutive workshops (WISS 2009 and WISS 2010) we organized on this topic in conjunction with the IEEE International Conference on Data Engineering (ICDE).

Over the recent years, the increasing costs of creating and maintaining infrastructures for delivering services to consumers have led to the emergence of cloud-based third-party service providers that rent out network presence, computation power, storage, as well as entire software suites, including database and application server capabilities. These service providers reduce the overall infrastructure burden of small and medium (and increasingly even large) businesses by enabling rapid Web-native deployment, lower hardware/software management costs, virtualization and automation, and instant scalability. The emergence in the last decade of various enabling technologies, such as J2EE, .Net, XML, virtual machines, Web services, new data management techniques (including column databases and MapReduce), and large data centers contributed to this trend. Today grid computing, on-line e-commerce and business (including CRM, accounting, collaboration, and workforce management) services, large-scale data integration and analytics, IT virtualization, and private and public data and application clouds are typical examples exploiting this database and software as service paradigm.

While the financial incentives for the database and software as service deployments are obvious, convincing potential customers that outsourcing their data is a viable alternative is still challenging. Today, major customer demands from these third-party services include competitive pricing (including pay-per-use), performance-level and service-level assurances, and the flexibility to move services across third-party infrastructures or maybe to in-house private clouds maintained on-premise. Behind these demands lie serious concerns, including the security, availability, and (semantic and performance) isolation provided by the third-party infrastructures, whether these will work in accordance with in-house components, whether they will provide sufficiently complete solutions that eliminate the need of having to create complex hybrids, whether they will work with other clouds if needed, and whether they will be sufficiently configurable but still cost less.

Note that, while tackling these demands and concerns, the service provider also needs to find ways to optimize the utilization of its internal resources so as to ensure the viability of its own operations. Therefore, the immediate technical challenges faced by providers of information and software as service infrastructures are manifold and include, among others, security and information assurance, service level agreements and service class guarantees, workflow modeling,

design patterns, and dynamic service composition, resource optimization and multi-tenancy, and compressed domain processing, replication, and high-degree parallelization.

The chapters in this book, contributed by leaders in academia and industry, and reviewed and supervised by an expert editorial board, describe approaches for tackling these cutting-edge challenges. We hope that you will find the chapters included here as indicative and informative about the nature of the coming age of information and software as services as we do.

November 2010

Divyakant Agrawal
K. Selçuk Candan
Wen-Syan Li

Editorial Advisory Board

Table of Contents

Service Design

Service Security

Service Optimization

Multi-tenancy and Service Migration

What Next?

Study of Software as a Service Support Platform for Small and Medium Businesses

Chang-Jie Guo, Wei Sun, Zhong-Bo Jiang, Ying Huang,
Bo Gao, and Zhi-Hu Wang

IBM China Research Lab, ZGC Software Park No. 19, Beijing, China
{guocj,weisun,jiangzb,yinghy,bocrlgao,zhwang}@cn.ibm.com

Abstract. Software as a Serivce (SaaS) provides software application vendors a Web based delivery model to serve big amount of clients with multi-tenancy based infrastructure and application sharing architecture so as to get great benefit from the economy of scale. In this paper, we describe the evolution of the small and medium businesses (SMB) oriented SaaS ecosystem and its key challenges. On particular problem we focus on is how to leverage massive multi-tenancy to balance the cost-effectiveness achieved via high shared efficiency, and the consequent security, performance and availability isolation issues among tenants. Base on this foundation, we further study the concepts, competency model and enablement framework of customization and configuration in SaaS context to satisfy as may tenants' requirements as possible. We also explore the topics on service lifecycle and the subscription management design of SaaS.

Keywords: Software as a Service, Multi-tenancy, Customization, Service Lifecycle Management, Subscription, Small and Medium Businesses (SMB).

1 Introduction

Software as a Service (SaaS) is gaining momentum with the significant increased number of vendors moving into this space and the recent success of a bunch of leading players on the market [1]. Designed to leverage the benefits brought by economy of scale, SaaS is about delivering software functionalities to a big group of clients over Web with one single instance of software application running on top of a multi-tenancy platform [2]. Clients usually don't need to purchase the software license and install the software package in their local computing environment. They use the credentials issued by the SaaS vendor to log onto and consume the SaaS service over Web through an Internet browser at any time and from any where with Internet connections.

Today's economic crisis makes it imperative that organizations of all sizes find new ways to perform their operations in a more cost-effective fashion. This is particularly true for small and medium size businesses (SMBs) which often operate with thinner margins than their larger enterprise counterparts [3]. From this point of view, SaaS is a delivery model that has everything to lure SMBs -- easy installation, low cost. As a consequence, SMBs can afford those more powerful enterprise applications, such as Customer Relationship Management (CRM), Enterprise Resource Planning (ERP) and

D. Agrawal et al. (Eds.): Information and Software as Services, LNBIP 74, pp. 1–30, 2011.

Supply Chain Management (SCM), via SaaS alternatives to the traditional, on-premise software packages of the past. It thus has achieved a prosperous development and covered most of the well-known application areas during the past several years. According to a recent survey [4], 86% of the 600+ SMBs that participated in the survey said they expected to deploy SaaS in their organization over the next year. Further, 55% of the survey participants indicated that they were planning to spend the same or even more on IT over the next year.

As more people are diving deep into the market, the SMBs oriented SaaS evolves gradually into a complex ecosystem. In the ecosystem, the service provider hosts many applications recruiting from different software vendors in the centrally-managed data centers, and operates them as the web delivered services for a huge number of SMBs simultaneously. Furthermore, some value-added service resellers (VARs), also appeared to help distribute, customize and compose the services to end customers more efficiently. All of these roles collaborate together to create a healthy and scalable SaaS value chain for the SMB market. The SMBs greatly reduce their operating costs and improve the effectiveness of their operations. Other stakeholders of the ecosystem share the revenues by providing support in the different stages of the service lifecycle, such as service creation, delivery, operation, composition and distribution. To enable the ecosystem and value chain, several technical challenges are inevitable introduced, especially compared with the traditional license software based business model.

- *Massive multi-tenancy* [2] refers to the principle of cost saving by effectively sharing infrastructure and application resources among tenants and offerings to achieve economy of scale. However, the tenant would naturally desire to access the service as if there is a dedicated one, which inevitably results to the security, performance and availability isolation issues among tenants with diverse SLA (service level agreement) and customization requirements.
- *Self-serve customization* [14] Many clients, although sharing the highly standardized software functionalities, still ask for function variants according to their unique business needs. Due to the subscription based model, SaaS vendors need take a well designed strategy to enable the self-serve and configuration based customization by their customers without changing the SaaS application source code for any individual customer.
- *Low-cost application migration* may help service providers recruit more offerings in a short time, since the service vendors needn't pay too much efforts in transformation. However, one issue is most of existing applications are on-premise or with very initial multi-tenancy features enabled. Another challenge is the extremely heterogenous programming models.
- *Streamlined service lifecycle* [15] refers to the management processes of services and tenants in many aspects like promotion, subscription, on-boarding, billing and operation etc. It focuses on optimizing the delivery efficiency and improving the process flexibility to satisfy the dynamically changing requirements.
- *On-demand scalability* refers to effectively deliver the application and computational resources to exactly match the real demands of clients. Today, this study is mostly located in cloud computing [5]. One key challenge is the on-demand allocation/de-allocation of resources in a dynamic, optimized and transparent way.

The existing IT infrastructure, middleware and programming models are difficult to satisfy the requirements and challenges described above, which show in many aspects. For example, software vendors should pay significant development efforts to enable their applications with the capabilities to be run as SaaS services. Meanwhile, the service providers are assailed by seeking a secure, flexible and cost-effective infrastructure and platform to effectively deliver and operate the SaaS services in the massive multi-tenancy scenarios. Furthermore, they also need well-designed management processes and programming toolkits to attract more software vendors, value added service builders and resellers to join the ecosystem, and recruit, promote and refine the services in a more efficient way.

This paper will introduce our experiences and insights accumulated in the real industry practices, to set up an effective and profitable SaaS ecosystem for the large-scale service provider to deliver internet-based services to a huge amount of SMB clients by working with plenty of service vendors. This paper focuses on exploring the key customer requirements and challenges of the ecosystem from three important aspects, e.g. massive multi-tenancy, flexibility and service lifecycle management, and the corresponding technologies having the potential to resolve them in practice.

The rest of this paper is organized as follows: Section 2 illustrates the evolution of SaaS ecosystem for SMB market. The next three sections are the main body of this paper. Section 3 focuses on massive multi-tenancy, which is the most important characteristic of SaaS. It explores how to achieve the cost effectiveness via effective resource sharing mechanisms, and the consequent security, performance and availability isolation issues among tenants. Section 4 will explore the configuration and customization issues and challenges to SaaS vendors, clarify the difference between configuration and customization. A competency model and a methodology framework have been developed to help SaaS vendors to plan and evaluate their capabilities and strategies for service configuration and customization. Section 5 targets to the service lifecycle and the subscription management design of SaaS. A subscription model is introduced first to capture the different entities and their relationships involved in SaaS subscription. Then a method supported with service structure patterns and business interaction patterns analysis is presented to empower a SaaS provider to design an appropriate subscription model for its service offering. A case study is also made to demonstrate the effectiveness of the method at the end of this paper. Section 6 introduces related works, and finally Section 7 concludes this paper and discusses future works.

2 SMBs Oriented SaaS Ecosystem

In the primary SaaS model, service (software) vendors are responsible for all stages of the complete service lifecycle. As illustrated in Fig. 1, they have to develop and host the SaaS applications by themselves. To promote the businesses, they should define a suitable go-to-market strategy to connect the potential SMB customers to persuade them to subscribe the services. Furthermore, service vendors need also pay significant efforts in the daily operations of the services, like charging the monthly "hosting" or "subscription" fees from customers directly.

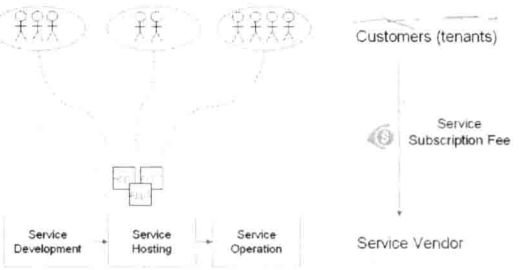

Fig. 1. The Primary SaaS Eco-system

SaaS customers, e.g. tenants, wish to get cost-effective services that address their critical business needs via a continuous expense rather than a single expense at time of purchase to reduce the Total Cost of Ownership (TCO). Meanwhile, the quality of the services, such as the security, performance, availability and flexibility, should be ensured at an acceptable level, considering the web based multi-tenant environments. Furthermore, the capability of highly on-demand scalability is also strongly required to adapt the dynamic and diverse requirements of their rapid business developments.

In this case, service vendors should accumulate enough knowledge in SaaS domain and have an insight into the architecture design of the SaaS applications. They need pay great efforts to enable those SaaS specific features, such as multi-tenant resources sharing patterns, isolation mechanisms, SLA management, service subscription and billing etc. This demands that developers own more strong technical skills, and inevitably increases the development cost and cycle.

Things could be a lot worse if the service vendors want to build more than one SaaS applications. They have to repeat the effort one by one since the implementation of the SaaS specific features have already been closely tangled with the business logics of each application. It may produce multiple independent and silo SaaS applications, which is not scalable to both development and management.

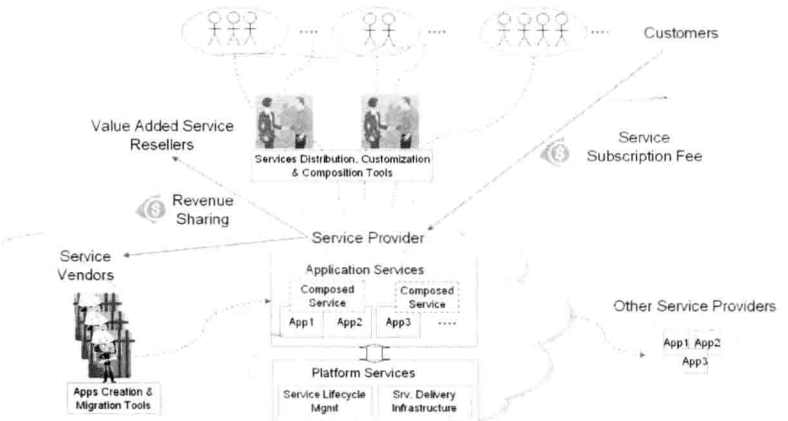

Fig. 2. The Advanced SaaS Ecosystem

Fig. 2 shows a more complex ecosystem, in which a new role, e.g. the service provider instead of the service vendor, will take full responsibility for hosting and operating the SaaS applications, and focus on providing better service quality to customers. In general, service providers may not develop applications by themselves, but tend to recruit from the service vendors and share the revenue with them in a certain way. In this case, service vendor becomes the pure SaaS content provider, while the value propositions of service provider in the ecosystem are as follows.

First, service provider sets up a cost-effective, secure and scalable runtime environment to deliver the applications of service vendors as services. It includes the hosted infrastructure, middleware and common services for SaaS enablement as well as the management and operation support, such as subscription, metering, billing, monitoring etc. To be noted, all of these features are provided in a standard "platform as service" way, independent of the applications running above. It's somehow similar to the BSS/OSS (Business/Operation Support System) platform of a telecom carrier, but targets to the SaaS domain.

Secondly, service provider also provides a suite of programming packages, sandbox, migration and offering lifecycle management tools for service vendors to quickly develop, test, transform and on-board their SaaS applications. In this case, service vendors need only focus on the user interfaces, business logics and data access models of their applications, without concerning with the detailed implementation of the SaaS enablement features. Those applications following the given programming specifications can easily run as services inside the hosted environment of service provider. Obviously, it can greatly reduce the development or migration costs of SaaS applications, and has potential to attracting more service vendors for a short time.

According to the definition of Wikipedia [6], a value-added reseller (VAR) is a company that adds some feature(s) to an existing product(s), and then resells it (usually to end-users) as an integrated product or complete "turn-key" solution. In the SaaS ecosystem, the value proposition of VAR mainly comes from two aspects:

- *Service Distribution*: The economics of scales demands the service provider to recruit a large volume of subscribed customers. In general, the VARs are geographically closer to end customers, and more familiar with the businesses and requirements of SMBs. By building a well-designed distribution network with resellers, service provider may recruit more SMB customers with least costs of go-to-market. In practice, service resellers may have pre-negotiated pricing that enables them to discount more than a customer would see going direct, or share the revenue with service provider.
- *Service Engagement*: The value propositions of VARs can also be added by providing some specific features for the customer's needs which don't exist in the standard service offerings, such as services customization, composition and integration with the on-premise applications or processes. Customers would purchase these value added services from the resellers if they lack the time or experiences to satisfy these requierments by themselves.

3 Massive Multi-tenancy

3.1 Overview of Multi-tenancy Patterns

Multi-tenancy is one of the key characteristics of SaaS. As illustrated in Fig. 3, in a multi-tenant enabled service environment, the user requests from different organizations and companies (referred as tenants) are served concurrently by one or more hosted application instances and databases based on a scalable, shared hardware and software infrastructure. The multi-tenancy approach can bring in a number of benefits including the improved profit margin for service providers through reduced delivery costs and decreased service subscription costs for clients. It makes the service offerings attractive to their potential clients, especially the SMB clients within very limited IT investment budget.

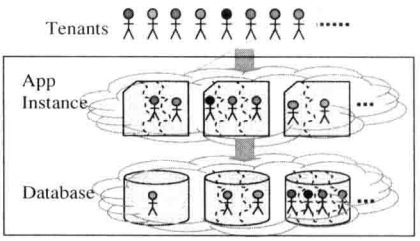

Fig. 3. A Multi-Tenant Enabled Service Environment

To achieve the economies of scale, the multi-tenant approach wishes to increase revenues by recruiting large number of clients and reduce the average service delivery costs per client by serving these clients with highly sharing infrastructure and application resources. Although higher resources sharing level can effectively drive down the total costs for both service consumers and providers, there are essential conflicts between cost-effectiveness and isolation among tenants. From the user experience, QoS (Quality of Service) and administration perspectives, the tenants would naturally desire to access and use the services as if there were dedicated ones. Therefore, isolation should be carefully considered in almost all parts of architecture design, from both non-functional and functional level, such as security, performance, availability, administration etc.

Generally, there are two kinds of multi-tenancy patterns: multiple instances and native (or massive) multi-tenancy, as illustrated in Fig. 4. The former supports each tenant with its dedicated application instance over a shared hardware, Operating System (OS) or a middleware server in a hosting environment whereas the latter can support all tenants by a single shared application instance over various hosting resources.

The two kinds of multi-tenancy patterns scale quite differently in terms of the number of tenants that they can support. The multi-instances pattern is adopted to support several up to dozens of tenants. While the native multi-tenancy pattern is used to support a much larger number of tenants, usually in the hundreds or even thousands. It is interesting to note that the isolation level among tenants decreases as the

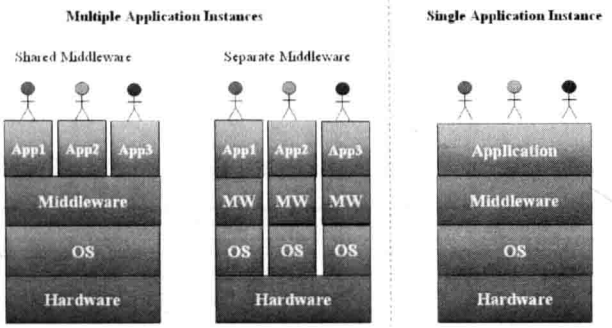

Fig. 4. Multi-tenancy Patterns: Multiple Instances Vs. Native (Massive) Multi-tenancy

scalability level increases. By using native multi-tenancy to support more tenants, we should put more efforts to prevent the QoS of one tenant from being affected by other tenants in the same sharing multi-tenancy environment.

The selection of multi-tenancy technology depends on the specific application scenarios and the target clients' requirements. For example, a large enterprise may prefer to pay a premium for multiple instances to prevent the potential risks associated with resource sharing. While most SMB companies would prefer services with a reasonable quality at lower costs, and care less about particular kinds of multi-tenancy patterns that the service providers use. Therefore, in this paper, we will focus on the native multi-tenancy pattern to explore how to achieve cost-effectiveness with acceptable isolation level for the SMB oriented SaaS ecosystem.

3.2 Cost-Effectiveness

Typically, most SaaS offerings target to SMB clients with very limited IT budgets. It's well known that low price is one of the most important reasons that SaaS can attract the attention of SMB customers. Therefore, the success of SMB oriented SaaS extremely depends on cost effectiveness and scalability, e.g. economics of scale. This trend is quite obvious, especially in the emerging markets.

For example, in China, one SMB with 3 concurrent users need only pay about $300 per year to subscribe the online accounting and SCM applications [7]. Meanwhile, the service provider, which is the biggest ERP software vendor of China SMB market, wishes to recruit several hundred thousands or even one million subscribed SMB customers in future several years. The scale of tenant number is predictable since China owns over 40 million SMBs in 2009. The key challenge is that how to make it profitable within such low pricing model.

First, from the view of service provider, it should extremely reduce the expense of service delivery infrastructure including the hardware, software and utility of hosting center, e.g. bandwidth, power, space etc., and save the costs of human resources to maintain the service operation lifecycle.

In this case, the multiple instances pattern in Fig. 4 is not practical as the small revenue generated from each tenant won't justify the allocation of dedicated hardware/software resources for the tenant. Actually, many resource types can be shared

among tenants in a more fine-granular and cost effective way if we take some kind of suitable resource management mechanism. These resources are located in different tiers and artifacts of SaaS applications, like user interface, business logic and data model. Table 1 gives some of these resources in a J2EE based SaaS application.

Table 1. Sharable multi-tenant resources in the J2EE based SaaS application

Layer	Components	Sharable Resources
Persistent Data Model	Database Access (JDBC, SDO, Hibernate, JPA etc.)	✧ Data Source & Connection Pool ✧ DB Account ✧ Database/Schema/Table/Buffer Pool etc.
	File / IO	✧ Directory ✧ File
	Directory Server (LDAP)	✧ Directory Tree ✧ Schema
Business Logic	Authentication & Authorization	✧ Organization Structure / Privileges Repository ✧ Login / authorization modules
	Global Java Object	✧ Static variable ✧ Variable of Singleton Class
	Remote Access (Socket/Http, RMI, CORABA, JNDI, Web Service etc.)	✧ Remote Services ✧ Connection Parameters like URI, port, username, password etc.
	EJB	✧ Stateful EJB instance ✧ Data source connection, table of Entity Bean ✧ Queue, sender's identity of MDB
	Logs	✧ Log file location, content, format and configuration
	Cache	✧ Cache Container
Process/Workflow	BPEL Process Template	✧ Template Level Attribute, Activity, Link Condition etc.
	Human Task	✧ Verb, Task UI etc
	Business Rule	✧ Rules
User Interface	JSP	✧ Application Scope Variable (tag, usebean, applicationContext etc.) ✧ Declaration variable ✧ Logo,Style,Layout etc.
	Servlet	✧ Single-thread servlet ✧ servletContext

For each kind of sharable resource type, we start from identifying all the potential sharing and isolation mechanisms, and evaluate them according to the estimation of the degree of cost saving. The additional management costs introduced by different level of resources sharing should be considered carefully. Since current administration tools for application, middleware and database are totally unaware of the concept of tenants, service providers have to pay more efforts and human resources to execute those multi-tenancy related operations manually because of lacking tenant-aware toolkits and automation processes.

For people to understand better, we take the database access as an example [8], and identified at least three kinds of resources sharing patterns in Fig. 5.

- *E1: Totally isolated*: each tenant owns a separate database
- *E2: Partially shared*: multiple tenants share a database, but each tenant owns a separate set of tables and schema