SmartSaaS: Model and Architecture

Binbin Yang¹, Jiang Wu¹,a, Huan Wang¹ and W. T. Tsai²

¹College of Information Science and Technology, Northwest University, Xi'an, China
²Dept. of Computer Science and Engineering, Arizona State University, Tempe, AZ 85287

Keywords: Software-as-a-Service (SaaS); Cloud Computing; Modeling; Architecture Design; Intelligent Behavior; Maturity Model.

Abstract. This paper proposes a SaaS (Software-as-a-Service) maturity model, and proposes an intelligent SaaS framework. The SaaS maturity has two dimensions. Traditional SaaS for business computing can also be extended to include intelligent behavior by having ontology to describe metadata of those data used in SaaS. Then, this paper proposed a new SaaS architecture SmartSaaS, and illustrate its key features. Finally, a conceivable architecture of implementation and technology roadmap were given.

Introduction

SaaS is a new software delivery model that provides customers access to business functionality remotely as a service on the Web. In the traditional on-premises model for software, a user purchases a set of software and deploys it to his server or data center, and the software belongs to the user. However, SaaS service providers own the software, a user just use the software via a Web browser by paying the service provider certain fees.

Most current SaaS research focus on multi-tenancy architecture, customization, and scalability [1]. Based on this research background, this paper studies the overall structure of SaaS for the platform. In addition, with the development of network technology and SaaS, intelligence is the development direction of SaaS. The SmartSaaS system is designed based on intelligence, and we have done some developing work by using the existing technology.

Several SaaS maturity models have been proposed in the past. For example, Microsoft proposed a maturity consisting of four stages: ad hoc, configurable, multi-tenancy architecture, and scalable computing [2]. Forrester group puts forward a six-level maturity model from traditional ASP to SaaS service model [3]. Multitude of researcher has analyzed definition, principles and other theoretical aspects of SaaS model [4]. Reference [5] provides the architecture for creating market-oriented SaaS by leveraging technologies such as virtual machines (VM), and provided thoughts on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation.

From another perspective, SaaS is a complex model of business system with a number of roles, including SaaS operators, software developers, service developers, tenants, etc. Rules play an important role in the entire system which provides the process execution with flexibility, reliability and completeness. Reference [6] studied the business process and interactions in SaaS and then proposes a business process customization system. In paper [7], authors dissected a multi-tenant, secure, load disseminated SaaS architecture (MSLD). Some researchers have concluded some of the lessons of realizing SaaS system [8].

SaaS is popular as most companies have adopted a SaaS approach including Salesforce.com [9], Workday.com, Rightnow Technology (acquired by Oracle), SuccessFactors (acquired by SAP), Corent Technology, and Fujitsu [10]. These are leading SaaS companies. Many companies focuses on PaaS (Platform-as-a-Service), and well known PaaS systems include Google’s GAE (Google App Engine), Amzaon’s EC2, and Microsoft’s Azure.
This paper designs and analyzes a novel architecture of SaaS in cloud computing, and proposes a new SaaS framework SmartSaaS to incorporate intelligent behavior into SaaS. Firstly, a series of design principles and a motivating scenario were set to guide and drive the subsequent works. According to the part before, we designed overall architecture of cloud-based SaaS architecture. Finally, on the basis of existing technology, we propose an implementation strategy, and describe the challenges of implementation and the corresponding solution.

Model Research of SmartSaaS

SmartSaaS Maturity Models. The common SaaS maturity levels are Ad hoc/custom, Configurable, Multi-Tenant, and Scalable. Paper [11] decides the axis of maturity model as the core criteria for measuring the degrees of evaluation: service component axis and maturity level axis.

First, we extend the service component and join the operation module. SaaS model refers to the common SaaS maturity levels are Ad hoc/custom, Configurable, Multi-Tenant, and Scalable. Paper [11] decides the axis of maturity model as the core criteria for measuring the degrees of evaluation: service component axis and maturity level axis.

According to the part before, we designed overall architecture of cloud-based SaaS architecture. Finally, on the basis of existing technology, we propose an implementation strategy, and describe the challenges of implementation and the corresponding solution.

Model Research of SmartSaaS

SmartSaaS Maturity Models. The common SaaS maturity levels are Ad hoc/custom, Configurable, Multi-Tenant, and Scalable. Paper [11] decides the axis of maturity model as the core criteria for measuring the degrees of evaluation: service component axis and maturity level axis.

First, we extend the service component and join the operation module. SaaS model refers to the common SaaS maturity levels are Ad hoc/custom, Configurable, Multi-Tenant, and Scalable. Paper [11] decides the axis of maturity model as the core criteria for measuring the degrees of evaluation: service component axis and maturity level axis.

According to the part before, we designed overall architecture of cloud-based SaaS architecture. Finally, on the basis of existing technology, we propose an implementation strategy, and describe the challenges of implementation and the corresponding solution.
Concept Model. SaaS is a typical single-instance supporting applications, building a SaaS platform need to solve the four key issues: configurability, scalability, multi-tenant storage structure and security. Therefore, when designing and implementing a SaaS platform, we should hold these four main buckles as the main line. On the other hand, business needs should be also consider, because SaaS platform what we designed is a set of practical business platform of services, and commercial value will determines its future development. As shown in Table 1, from a technical point of view, we analyzed functions of SaaS in four aspects: Configuration, Scalability, Multi-tenant Support and Security. In the points of business, we also studied from four parts: Market, Scalability, Development and Communication.

<table>
<thead>
<tr>
<th>Technical Function</th>
<th>Business Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Configuration</strong></td>
<td><strong>Market</strong></td>
</tr>
<tr>
<td>User Interface</td>
<td>Open Marketplace</td>
</tr>
<tr>
<td>Working Process</td>
<td>Application Selling</td>
</tr>
<tr>
<td>Business Rules</td>
<td>Business Model</td>
</tr>
<tr>
<td>Customizable Data Model</td>
<td>SLA Adaptation</td>
</tr>
<tr>
<td>Metadata Set</td>
<td>Service Billing Policy</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td><strong>Scalability</strong></td>
</tr>
<tr>
<td>Reconstruction of Database</td>
<td>Guaranteed Performance</td>
</tr>
<tr>
<td>Sharing of Resources Pool</td>
<td>Monitoring and Prediction</td>
</tr>
<tr>
<td>Application with Stateless</td>
<td>Development Toolkit Providing</td>
</tr>
<tr>
<td>Scaling The Devices</td>
<td>Application Release Process</td>
</tr>
<tr>
<td>Load Balancer</td>
<td>Integrated Development</td>
</tr>
<tr>
<td>Monitoring and Optimization</td>
<td>Mash-up API</td>
</tr>
<tr>
<td><strong>Multi-tenant Support</strong></td>
<td><strong>Development</strong></td>
</tr>
<tr>
<td>Shared Database and Schema</td>
<td>Transportability</td>
</tr>
<tr>
<td>Data Access Control</td>
<td>Unified Version Management</td>
</tr>
<tr>
<td>Extensible Data Structure</td>
<td>Application Test</td>
</tr>
<tr>
<td>Metadata Definition</td>
<td><strong>Communication</strong></td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td><strong>Communication</strong></td>
</tr>
<tr>
<td>SLA</td>
<td>Supporting Community for Users</td>
</tr>
<tr>
<td>Authentication</td>
<td><strong>Security</strong></td>
</tr>
<tr>
<td>Validation Rule</td>
<td><strong>Security</strong></td>
</tr>
<tr>
<td>Security Policy</td>
<td><strong>Security</strong></td>
</tr>
<tr>
<td>Verification</td>
<td><strong>Security</strong></td>
</tr>
</tbody>
</table>

Based on the above analysis of SaaS functions, we established a conceptual model of SaaS system. As show in Fig. 3, SmartSaaS model is constituted by five parts: Fundamental Subsystem, Operational Subsystem, Manage Subsystem, Storage Subsystem and Optimization Subsystem. Fundamental Subsystem mainly supplies fundamental functions to support customers’ everyday use and some of the underlying operations. Operational Subsystem is responsible for handling customer requests and producing execution workflow of the different needs, so that the entire system works orderly and efficiently. Security and management of authorization belong to Management Subsystem, which guarantees system safety and data security. All of customized information, files and other data are all stored and managed by Storage Subsystem. Performance is also an element of SaaS platform, so we designed the Optimization Subsystem to monitor and promote efficiency of the system. The platform is a unified whole, therefore, there will be some overlap among them although these five parts are separated primarily by function.

The Design and Implementation of SmartSaaS

In this section, we will introduce detailed internal design and our early experience with SmartSaaS implementation. In section A, we will present the prototype of SmartSaaS Platform and give a detailed explanation of it. In section B, we refer the reader to a series of short demos we have produced based on the prototype.

Prototype of SmartSaaS Platform. Fig. 4 has shown the prototype of SmartSaaS Platform. It shows the position of network services, tenant services, and variety applications running on the SmartSaaS platform. As a platform for loading and executing applications to be provided as software as a service, the SmartSaaS platform consists of three components.
SaaS Resource Platform. This platform provides fundamental resources such as virtual resource management, resources and common security environment including access security from the Internet. The servers are equipped with a hyper array of software based on different operation systems and open-source middleware. If some one requires a server, the platform will assess the requirement base on situation and distribution of server load. After performance testing and security checks have been performed, virtual resource management will provide related virtual resources for customers. These virtual resources compose an intact environment that can be used immediately after prospective applications have been laden. The SmartSaaS resource platform support load balancer and security check natively and is superior in terms of quick startup and safety.

SaaS Application Platform. The platform consists of three core components: "User manager" manages the user information and behavior as a interface between end-user and platform; "Customization manager" is the embodiment of configurable function; "Process & responder management" is used to generate processes and responding to requests. These three departments are described as: User manager, Customization manager, Process and Respond management.

"User manager" is surface and a group of functions that connect SmartSaaS systems and end users. "Customization manager" provide users with all-sided configuration. This part consists of three parts: UI Customization, Business Customization and Data Model. "Process and Respond management" receives requests of the customer or system and responds to them, it also produces customized processes depending on the business and users’ information, and manages the processes to make them run efficiently and correctly. “Automatic analyses of requirement” is added to this section, which is responsible for analyzing the customer's demand based on semantics and context.

In addition, SmartSaaS application platform also consists the other three parts: “Service Integrated Management File System” for integrating services on platform, including user-developed external services and applications; “SaaS Utility Group” for providing basic services interface of development and application, and they are provided as Web services prepared in the form of clusters. “Collaboration management” coordinates different users or the various different functions purchased by a user, so that they can run in phase. “Metadata service” provides the basic support for customer, user customization and data storage. “Validation rule” and “Verification engine” is responsible to granting authorization license to them. Only obtain a license, they can continue, or will be hang or killed by system as a illegal procedure.

Global Service and Development Service. “Security and Performance module” is responsible for the security and performance optimization of entire platform, including: Security Services, Performance Monitoring, Performance Prediction, Performance Optimization and State Manager. “Development Service” achieves functions that manage and support service development or deployment, including two aspects: “Version management Service” is a unified management platform for versions of all applications and services, rather than to manage dispersely by authors.
This is one of our forward-looking functions. This function will save development time and lower
development threshold greatly. “Development testing Service” provides test environment and a
series of tests strategies for deploy of the software or services produced by development-oriented
users.

![SmartSaaS Implementation Diagram](image)

**Figure 5. Implementation of SmartSaaS.**

**Implementation and Challenges of SmartSaaS.** We had planned a SmartSaaS implementation
model, as shown in Fig. 5, we use RIA technology, such as C#, PHP and java and other popular
technologies, to implement the platform display and front-office logic. SOA-based platform can be
built according to REST (Representational State Transfer) [12] architecture.

REST become popular as a way to design Web services, and this method is less dependent on
proprietary middleware (such as an application server) than SOAP-based and WSDL-based (Web
Services Description Language) approach [13]. Because of the perfect match between Ajax and
REST, people have increased confidence in the use of REST today. We can also use Web services
based on SOAP and Web WSDL. But we believe that the better approach is to mix REST and
MVC, because this method can be competed to the vast development of Web application.
Developers only need to take REST developing mode for user demands which are more easily
abstracted to resources, while MVC for the other needs.

We believe that providing a suitable base for a multi-tenanted, cloud-scale collaboration platform
is difficult. Many of our concerns pertain to multi-tenancy. Problems with this aspect arise when
trying to apply it to a service that facilitates collaboration between developers or customers. To
overcome this, we argue the platform should use “hyper access method” via mixture of analog and
trusted subsystem approach.

**Summarize and Future Work**

This paper studies the maturity model, interactive model, concept model and integral architecture of
SmartSaaS System. SmartSaaS is designed so that it can make human interface easy and intuitive.
This paper also presents a design of the SmartSaaS and discussed various implementation strategies.
In the following implementation, we encountered many challenges and difficulties described in the
end of the article.

Our plan is, from the platform operator's point, integrating any software and resources, and
providing high quality and reliable service to customers. The services depend on user's comments
and feedback to achieve optimization automatically. While from the developer's point, tenant
developers develop software freely on the platform or offline, and after that release to the platform
so as to acquire good release management and testing service.

**Acknowledgment**

This work is supported by grants from the key project of teaching reform plan in Northwest
University.
References


