# A Survey of Software as a Service Delivery Paradigm

Peilin Guo
Helsinki University of Technology
pguo@cc.hut.fi

## Abstract

Software as a Service (SaaS) is a novel model of software delivery. It has been gaining momentum in recent years. More and more SaaS adoption succeeds. This paper presents an overview of this emerging model. I compare SaaS with legacy software delivery model, and also with Service-oriented architecture (SOA). I investigate SaaS from different perspectives, from development to business model, analyze the competition between SaaS and legacy software, and discuss possible challenges and subsequences. After the survey, I conclude that SaaS could gradually take over the whole market. However, the integration, configuration and customization capability will decide whether a SaaS solution is successful or not.

KEYWORDS: Software as a service, software business, software development.

## 1 Introduction

Software as a Service (SaaS) is a novel concept of delivering software applications. Some of its early work was carried out as a joint project among the universities of Durham, Keele, and UMIST in 1999 [11, 21]. Traditonally, users purchase and install the software application on their premise. In contrast to the perpetual licensing used by traditional software, SaaS adopts a subscription pricing policy, charges on a per-use basis. Hence, SaaS is also known as "subscription software" [21].

SaaS is rapidly growing. Its market was estimated to grow about 25% per year from 2006 and was expected to reach $ 10 billion in annual revenue by 2009 [17]. Well-known SaaS products include Salsforce.com, Employease.com [5, 1]. The content of these products varies, including Customer Relationship Management (CRM) service, Human Resource Management (HRM) service, desktop functionality, email, and supply chain and inventory control [22].

This paper aims to draw a big picture of the novel SaaS delivery model by analyzing it from different perspectives. The rest of this paper is organized as follows. Section 2 compares SaaS with its counterparts. Section 3 analyzes the development of SaaS applications. Section 4 introduces a business model of SaaS proposed by researchers, and evaluates the competition in software market. The last section summarizes the discussion and reaches the final conclusion.

## 2 Comparisons between SaaS and its counterparts

### 2.1 SaaS and legacy software

There are three primary differences between SaaS and legacy software: delivering different products, adopting distinct pricing modes, and employing different delivery methods [16].

First, they deliver different products. SaaS provides the subscribers with a collection of standard applications and services; legacy software provides customized applications to its users.

Second, they adopt distinct pricing modes. Legacy software users purchase the application; SaaS subscribers pay for each time they use the services.

Third, they employ different delivery methods. Legacy software applications are installed on users' premise; SaaS applications are located in services provider's central server and the services are delivered via the Internet or an intranet.

### 2.2 SaaS and SOA

Many people find it difficult to distinguish between Software as a Service (SaaS) and Service-oriented architecture (SOA). Even IT professionals also often consider them the same concept. The misunderstanding of these two terms might causes various problems, from confusion to poor designs. Laplante et al. (2008) [14] states that "the difference between SaaS and SOA is that the former is a software-delivery model whereas the latter is a software-construction model". Both SaaS and SOA are only conceptual level models. Implementation of their features needs support of concrete technology.

SaaS delivers software as utility services and charges on a per-use basis. It separates software ownership from the user. The SaaS providers own the software, store the software system and users' data in a central server, and provide on-demand service to users via the Internet or an intranet.

In a SOA model, the constituent components of the software system are reusable units or services [23]. Functions are separated into distinct units or services [9]. Developers are allowed to access to the SOA services framework, combine and reuse available components of the system to construct their new software applications [12]. Today's major SOA products include IBM WebSphere, Microsoft .NET, Oracle Web Services Manager, and Sun Java Composite Application Platform Suite.

Despite the differences, SaaS and SOA complement each other in the large-scale information systems. Laplante et al.

(2008) [14] concludes that "SaaS helps to offer components for SOA to use, and SOA helps to quickly realize SaaS".

## 3 Development of SaaS applications

Many different issues have to be considered when developing SaaS applications. This section mainly opens the discussion from the integration, configuration and customization perspectivess.

### 3.1 Integration

A SaaS subscriber usually deploys on-premises legacy software as well. This requires integration between on-premises applications and SaaS solutions. Furthermore, if the user subscribes to more than one SaaS solutions simultaneously, integration between different SaaS solutions is also desired. The integration happens in three major layers of SaaS applications, i.e. user interface, business logic, and data [19].

According to a study on SaaS solution adoption trend by IDC in 2004, more than 50% of the survey respondents emphasized "better integration with in-house applications" [7]. An AMR research report in 2005 pointed out that more than 70% of the total 639 respondents in the survey expect that the SaaS solution can be integrated with their on-premises legacy software and other SaaS solutions they subscribed or plan to subscribe [8].

Some of the industrial products addressing the SaaS integration issues are Salesforce.com AppExchange [5], IBM SaaS showcase [2], Jamcracker [3], and OpenKapow [4].

Sun et al. (2007) [19] proposes a hybid model of SaaS integration, in which SaaS solutions and on-premises legacy software applications integrate on three major layers: user interface, business logic (process), and data. In User Interface Integration, first, Single Sign On enables users to access all the authorized user interfaces after a single unified sign on; second, Mash-up [6] enables users to access one SaaS solution or legacy software application's data from another. In Process Integration, four patterns are proposed: Push, Scheduled Pulling, Receive, and Workflow. In Data Integration, data is categorized into two types (master data and transactional data), proposed patterns are Batch Data Migration and Batch Data Synchronization.

In Sun et al. (2007) paper [19], the authors treate SaaS services as web services and state that most non-functional requirements of web services also exist in SaaS. Most important ones are security and privacy, bill reporting and management, and Quality of Service (QoS) reporting and reconciliation with Service Level Agreement (SLA). The authors define the development process of SaaS integration design and development as business process review, integration design, integration implementation, data migration, testing, and finally production and monitoring. In their paper, the authors also introduced a SaaS integration framework reference architecture as an extension of web service description named SaaS-DL, and built a SaaSia prototype. The authors concluded that in SaaS integration, most funcional requirements can be fulfilled by existing SOA integration technologies, non-functional requirements should be fulfilled by extending existing integration technologies.

### 3.2 Configuration and Customization

Configuration and customization is another crucial issue in SaaS development. SaaS services are usually designed and developed as standardized functionalities. However, users always request function variants to fulfill their own unique business need. This should be done through easy configuration and customization. Sun et al. (2008) [20] clarify the definitions and meanings of configuration and customization of SaaS solutions. In addition to that, the authors develop a competency model of configuration and customization, and build a methodology framework for it.

Configuration and customization is not a new thing. Most enterprise software applications face this problem. They should be tailored so as to fulfill requirements' variance due to customers' significantly different background. However, it is obviously that SaaS providers cannot afford to develop and maintain a version of application code for each individual customer. Therefore, a good or poor solution to this problem influences the success of a SaaS product much. The more complex a SaaS service is, the more potential tailoring efforts is needed. This is why the most successful SaaS solutions are CRM, HRM, email services, etc [18].

Two major approaches to solve the requirements variance problem are configuration and customization, which often make people confused. The major differences between them are that configuration does not involve source code changes, whereas customization does. Configuration uses pre-defined parameters to change software functions. For functionalities beyond the pre-defined scope, customization will then occur. Customization causes source code changes, thus requires more resources, increases development expense, and prolongs the lifecycle. As a result, according to Sun et al. (2008) [20], SaaS development should "avoid customization by using configuration to meet clients' tailoring requirements and enlarge configurable limit as far as possible toward client's unique requirements."

In Sun et al. (2008) paper, the authors introduce a SaaS Configuration Competency Model. In the model, competency of SaaS configuration is devided into 5 levels: Entry, Aware, Capable, Mature, and World Class. The variance level supported are none, low, medium, high, and extremely high, accordingly, which transforms from completely standardized offering to fully tenantized offering. The authors also build a methodology framework to plan and execute configuration and customization strategy. The process of this framework involves five steps: 1)understanding environment, 2)defining strategy, 3)assessing competency, 4)identifying gap and actions, and 5)prioritizing and executing.

In the first step, vendors should investigate the environment of SaaS service configuration and customization, mainly focus on client requirements and market leader competency level. In the second step, vendors make their plan to fulfill targeted customers' configuration and customization requirements. In the third step, vendors assess their own competency level of SaaS configuration and customization. In the fourth step, the competency gaps can be identified to guide the actions' definition. In the final step, vendors prioritize and excute the actions.

## 4 Business Model and Competition Analysis

### 4.1 Business Model

This section introduces a business model of SaaS proposed by Dan Ma in 2007 [15]. In the software market, the three major parties are software users, legacy software providers, and SaaS providers. Software users have different demands in terms of their expected usage volume. This model assumes users are uniformly distributed over [0,1].

### 4.2 Competition Analysis

[10, 15, 16, 13]

## 5 Conclusion

In this survey paper, I describe a novel software delivery model SaaS. This model differs from legacy software application by delivering different products, adopting distinct pricing modes, and employing different delivery methods. SaaS and SOA should not be misunderstood. SaaS is a software delivery model, whereas SOA is a software construction model.

Many different issues have to be taken into account in the process of developing SaaS application. Integration, configuration and customization are among the most important ones. Strong integration capability enables a SaaS service work well in conjunction with other SaaS services as well as legacy software applications. Excellent configuration and customization capabilities increase the product's competitive advantages on the market.

I review business model of SaaS proposed by researchers. By analyzing different competition scenarios, I conclude that SaaS and legacy software would coexist in the market in a long run, but the former one could gradually take over the whole market.

## References

[1] EmployEase Website, http://www.employease.com/.

[2] IBM SaaS Showcase website, http://www-19.lotus.com/wps/portal/showcase/SaaS.

[3] Jamcrack website, http://www.jamcracker.com/.

[4] OpenKapow website, http://www.openkapow.com/.

[5] Salesforce Website, http://www.salesforce.com/.

[6] Website: Mashups and the Web as Platform, http://www.programmableweb.com/.

[7] IDC report: Software as a Service in the Mid-market: Adoption Trends and Customer Preferences, 2004.

[8] AMR Research Report: Software as a Service: Managing Buyer Expectations as We Pass the Tipping Point from Novelty to Necessity, 2005.

[9] Bell, M. *Service-Oriented Modeling: Service Analysis, Design and Architecture*. John Wiley & Sons, 2008.

[10] Choudhary, V. Software as a Service: Implications for Investment in Software Development. In *Proceedings of the 40th Hawaii International Conference on System Sciences - 2007*, pages 209a–209a, January 2007.

[11] Elfatatry, A. and Layzell, P. Software As A Service: A Negotiation Perspective. In *Proceedings of the 26th Annual International Computer Software and Applications Conference*, pages 501–506, August 2002.

[12] Erl, T. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall Professional Technical Reference, 2005.

[13] Fan, M., Kumar, S. and Whinston, A. B. Short-term and long-term competition between providers of shink-wrap software and software as a service. *European Journal of Operational Research*, 196:661–671, 2009.

[14] Laplante, P. A., Zhang, J. and Voas, J. Distinguishing between Software Oriented Architecture and Software as a Service: What's in a Name? *IEEE IT Professional*, 10:46–50, May/June 2008.

[15] Ma, D. The Business Model of "Software-As-A-Service". In *2007 IEEE International Conference on Services Computing (SCC 2007)*, pages 701–702, July 2007.

[16] Ma, D. and Seidmann, A. The Pricing Strategy Analysis for the "Software-as-a-Service" Business Model. *GECON 2008, Lecture Notes in Computer Science*, 5206:103–112, 2008.

[17] Pallatto, J. IBM Recruiting ISVs, Partners to SAAS, February 2006.

[18] Summit Strategy Inc. Software-Powered Services Net-Native Software-as Services Transforms the ISV Business Model.

[19] Sun, W., Zhang, K., Chen, S., Zhang, X., and Liang, H. Software as a Service: An Integration Perspective. *ICSOC 2007, Lecture Notes in Computer Science*, 4749:558–569, 2007.

[20] Sun, W., Zhang, X., Guo, C., Sun, P., and Su, H. Software as a Service: Configuration and Customization Perspectives. In *Congress on Services Part II, 2008. SERVICES-2. IEEE*, pages 18–25, September 2008.

[21] Turner, M., Budgen, D. and Brereton, P. Turning Software into a Service. *Computer*, 36:38–44, 2003.

[22] Weier, M. H. and Smith, L. Businesses Get Serious about Software as a Service. *Information Week*, pages 46–48, 2007.

[23] Zhang, L.-J., Zhang J. and Cai, H. *Services Computing*. Tsinghua University Press and Springer, 2007.