Software As A Service (SaaS)

Rosnita Binti Abdul Kahar
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia
Email: akrosnita@gmail.com

Abstract

Software as a Service (SaaS, typically pronounced 'sass') is a model of software deployment whereby a provider licenses an application to customers for use as a service on demand.

1. Introduction

When the Internet burst upon the scene in the early 1990s, the concept of software as a service (SaaS) seemed an idea whose time had come. It got hyped along with everything else about the Internet and reached a massive peak of inflated expectations in early 2000 as venture capitalists funded dozens of nearly identical companies that provided various SaaS offerings [1]. As venture funding dried up in mid-2000, the cracks in the SaaS model began to appear.

SaaS is getting a lot of attention these days. The concept of SaaS is not new and has existed for a while. It has been referred to by other names such as Application Service Provider (ASP), Managed service provider (MSP), on-demand services, cloud computing, utility computing etc [2].

Software as a Service (SaaS, typically pronounced 'sass') is a software thats developed and hosted by the vendor and which the end user customer accesses over the Internet. Unlike traditional packaged applications that users install on their computers or servers, the SaaS vendor owns the software and runs it on computers in its data center. The customer does not own the software but effectively rents it, usually for a monthly fee. SaaS is sometimes also known as hosted software or by its more marketing-friendly cousin, on-demand [3].

SaaS software vendors may let the application to download by the customer on their devices and disabling it after use or after the on-demand contract expires. The on-demand function may be handled internally to share licenses within a firm or by a third-party application service provider (ASP) sharing licenses between firms [1].

1.1. SaaS and SOA

Considerable confusion arises in distinguishing between software as a service (SaaS) and service-oriented architecture (SOA). The difference between SaaS and SOA is that the former is a software-delivery model whereas the latter is a software-construction model [4]. SOA basically involves exposing functionality from distributed systems in the form of stateless functions; this is similar to other distributed system architectures such as CORBA and DCOM [2].

SaaS deployments are revenue generating businesses targeted directly at end users, whereas SOA deployments are usually created within IT environments and the services are exposed to other applications as opposed to end users [2]. Author [2] also said that SaaS and SOA are very complementary in nature. In fact, they can't exist without each other. They can be seen in figure 1.

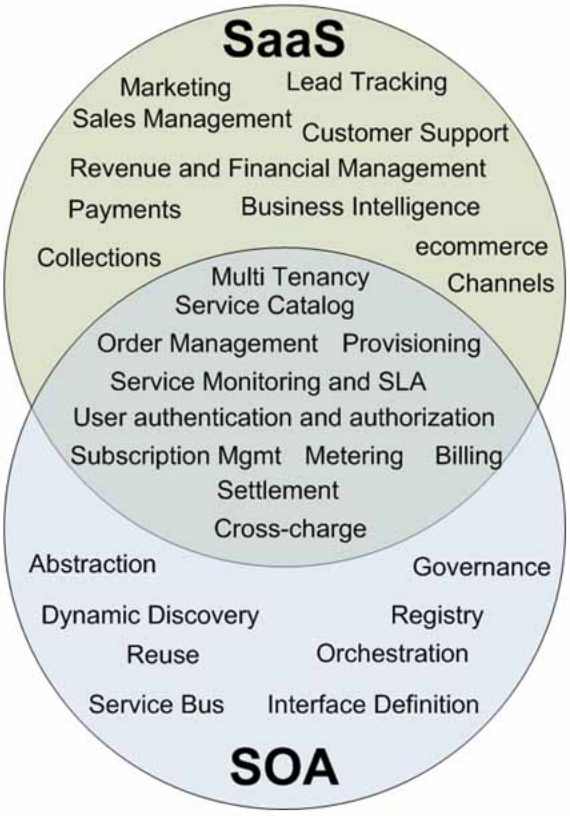

Figure 1. SaaS Vs SOA [2]

What are the key elements of a SaaS platform? Every SaaS platform has to have a few core things in place, these are: multi-tenancy, ordering and provisioning, user

authentication and authorization, service catalog and pricing, service monitoring, SLA management, usage metering, billing, invoicing and payments. Besides these core components, a SaaS platform also needs to support the usual business functions such as marketing, lead tracking, sales, customer support, revenue and financial management, partner settlement, business intelligence etc [2].

Now, let's take a look at the key elements of a SOA platform. A typical SOA platform deployment consists of service producers and consumers from across the enterprise. Service producers publish services via the SOA platform, which get consumed by multiple service consumers. There has been a lot of focus on the technical aspects of a SOA platform e.g. service bus, communication protocols (e.g. SOAP), service interface definitions (e.g. WSDL), service discovery (e.g. UDDI) etc. The importance of service monitoring, management and governance is also well understood, but this is not enough. In a typical large enterprise, the service producers and consumers could be applications or systems belonging to different departments, organizations or even subsidiaries within the enterprise. In such environments, services cannot be produced and consumed informally without proper service management in place since there is a cost associated to hosting and exposing a service by the service producer. In order to derive this cost, the total cost of operations or ownership (TCO) needs to be taken into account besides the cost to create the service. Also, there are security concerns around publishing the services openly. This leads to the need for service catalog management, provisioning, authentication, authorization, usage metering and cross-department charging. As highlighted before, these are also the core elements of a SaaS platform. So as an enterprise SOA deployment matures, it is suddenly in need of the core functions of a SaaS platform [2].

Let's take a look at the flip side of this. Every SaaS platform needs to support the ability to add new service offerings and modify existing offerings in the service catalog with minimal changes to the core platform components. These additions or modifications should not lead to creation of a whole new SaaS platform for every service. Instead, all the basic functionalities of the SaaS platform such as ordering and provisioning, authentication and authorization, service catalog and pricing, metering, billing and invoicing, payments etc should be reused for multiple service offerings. Such reuse necessitates the need for a SOA platform. Further, use of a SOA platform enables other advantages such as a more flexible and plug-n-play architecture leading to lower overall cost of ownership [2].

It is probably quite intuitive that most complex architectures including SaaS architecture will benefit from SOA capabilities, but a SOA platform needing SaaS capabilities is not that intuitive. There has been a lot of hype around SOA for a while but most SOA deployments in large enterprises have either not been successful or have not provided the expected ROI because the SaaS elements are missing in these deployments. In order to realize the full benefits of large-scale SOA deployments, it is essential to have a SaaS like service management functionality in place. This is where SOA and SaaS together can enable the concept of "IT as a service" and help take IT to the next natural step in its evolution [2].

1.2. How is SaaS different from an ASP?

SaaS evolved from the application service provider (ASP) model. When ASPs sprang up in the 1990s, they offered essentially the same thing SaaS vendors offer today: hosted applications delivered over the Internet. The problem ASPs ran into was that they tried to be all things to all people, and they buckled under the weight of their own infrastructure. In trying to serve the unique needs of each of their customers, ASPs lost the economies of scale that were necessary for them to provide their services in a cost-effective manner [3].

Todays successful SaaS vendors, such as Salesforce.com, LeanLogistics and Ketera, have solved the scalability and reliability problems that dogged the ASPs and ultimately led to their downfall. Instead of trying to be all things to all people, they offer one-size-fits-all solutions. That is, all customers of a SaaS vendor use the same software. The underlying code is the same for all customers and cannot be customized. Any features or functionality that the SaaS vendor adds to the software based on a customers feedback becomes available to all customers. This multi-tenancy approach differentiates SaaS vendors from the original ASPs and from other vendors of hosted, on-demand software and gives SaaS vendors the economies of scale they need to offer their software cost-effectivelyand make upgrading their customers to new versions of the software a relative cinch [3].

2. The Benefits

For many companies large and small, SaaS is the best way to roll out new technology. The staying power of SaaS has arisen for several reasons:

2.1. The cost of entry is low.

Instead of paying lots of money to roll out complex technology across the entire company, customers can roll out just one test department of, say, 20 people. The risk is very low if it fails, and the company doesn't have to involve its busy IT staff [1].

SaaS implementations are also cheaper because companies dont have to buy additional hardware or infrastructure to make the software work, so there are no capital expenditures with SaaS [3].

2.2. Less Time Deployment.

SaaS deployments usually take less time than on-premise software implementations simply because the customers not installing software on every users computer. Realistically, implementations can take between three and six months depending on the size and complexity of the implementation. Because SaaS is easier and quicker to implement than traditional software, the customer can achieve ROI faster and provided the users adopt the software [3].

2.3. The onus is on the vendor.

If the vendor's software is broken, the vendor won't be getting money from any customer for long. The vendor is motivated to fix the problem [1].

2.4. The vendor works for the buyer.

Customers don't have to rely on their IT departments to install an application. Everything is running securely at the vendor's location [1].

2.5. Less-risky investment.

Instead of spending $60,000 all at once, for example, customers pay for the software monthly. The monetary risk is lower and less scary [1].

2.6. Vendors must provide a secure data environment, or they're out of a job.

Most vendors understand that data must be backed up religiously, and security is the top priority. Customers' IT departments are typically pulled in many directions and can't be as focused on one technology. Customers can assume their data security is probably safer when it's hosted [1].

3. The Vendors

The example of success companies are Salesforce.com Inc. [1], [3], RightNow Technologies Inc [1], LeanLogistics [3], Cisco WebEx [2] and Ketera [3].

4. The Examples

Some example of the SaaS are [5]:

- Customer Relationship Management(CRM) Software
- Human Resource
- Procurement
- Document Management

The rank of application used as SaaS depicted in Figure 2. It shows that the largest application using SaaS is Customer Relationship Management (about 18 %), followed by Human Resource (15 %) and Procurement Software (12 %).

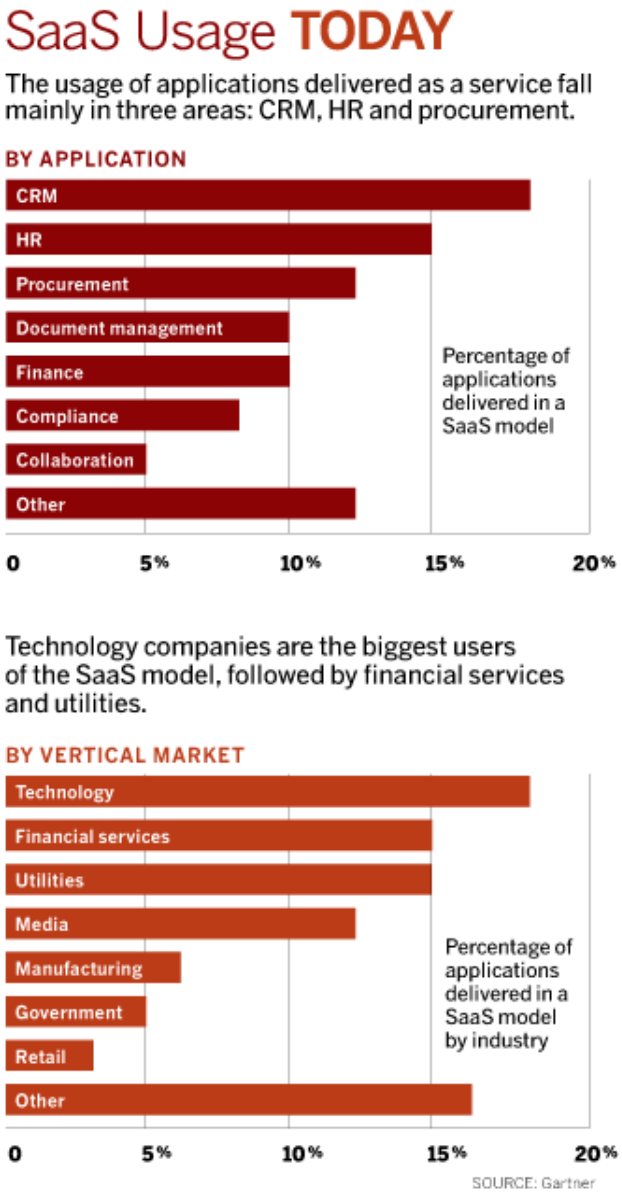

Figure 2. SaaS Today [5]

The figure also shows the industrial sectors that used the application. The highest sector is technology industry (18 %), followed by both financial services and utilities (15 %) and the media (12 %). This figure shows that Government in the sixth place by vertical market.

5. Issues and Challenges

5.1. Security

The customers have to concerns about keeping their data in a SaaS vendor's systems because they have no direct control over those systems - and because of the horror stories that emerged from the ASP implosion of 2000 and 2001. When those ASPs went out of business, many of them left their customers without access to their data, and some ASPs even sold the data they hosted [3].

5.2. Server and software maintenance

blablabla

5.3. Building the Contract

This experience taught customers valuable lessons in negotiating contracts: Always make sure they'll have access to the data and the software or source code in the event the service provider goes out of business.

6. Conclusion

The conclusion is using the SaaS will cut cost to the customer because they do not have to build their own data center and the cost of application development will much more reduced [5].

Acknowledgment

The authors would like to thank to Mr Zamri Murah for this Latex application and his kindness teaching all the superb things.

More thanks to Pn Noraidah be the friend and the study team, working hard hand in hand, sharing the knowledge and the effort to be closer to the destiny.

References

[1] C. Finch, *All Your Money Won't Another Minute Buy: Valuing Time as a Business Resource*. Lulu. com, 2007.

[2] V. Singla. (2009) The Overlaping Worlds of SaaS and SoA. [Online]. Available: http://cloudcomputing.sys-con.com/?q=node/1047073

[3] M. Levinson. (2007) Software as a Service (SaaS) Definition and Solutions. [Online]. Available: http://www.cio.com/article/109704/Software_as_a_Service_SaaS_Definition_and_Solutions

[4] P. Laplante, J. Zhang, and J. Voas, "What's in a Name? Distinguishing between SaaS and SOA," *IT Professional*, vol. 10, no. 3, pp. 46–50, 2008.

[5] G. Gruman. (2007) The Truth About Software as a Service (SaaS). [Online]. Available: http://www.cio.com/article/109706/The_Truth_About_Software_as_a_Service_SaaS_