

From SaaS to SaaS True “Software as a *Service*” is about CHOICE

Author: Vance F. Brown



Innovative Technology Built Upon Yesterday's Values

What is SaaS?

The term “SaaS” is an acronym for “Software as a Service.” Forrester Research reports that fifty-four percent of all organizations are interested in SaaS deployment. According to Gartner, even though only approximately five percent of the IT Service Management (ITSM) software market today has embraced SaaS, it is estimated that ten percent of this industry will do so by 2012.

So while it certainly appears that many organizations are demanding the benefits offered by SaaS, evidence indicates many organizations are confused about what SaaS truly entails, and whether or not it is the best choice for their software needs. Consider the following question recently posted on a SaaS blog:

“Please tell me just what exactly is SaaS? Is it a ‘Business Model?’ Is it a ‘Delivery/Hosting Model’ ... or something else?”

What a great question! This person is not alone in his or her confusion. Unless we can define the term and understand the components of SaaS, it is difficult to determine whether or not SaaS is the best option for an organization’s specific needs. When I ask this question of others, the most typical response is something like the following: “SaaS is a term that describes software that is delivered over the Internet through a browser and is paid for on a subscription basis.” Wikipedia suggests that SaaS, “sometimes referred to as ‘software on demand,’ is software that is deployed over the Internet... With SaaS, a provider licenses an application to customers as a service on demand, through a subscription or a ‘pay-as-you-go’ model.”

The purpose of this article is to help define and eliminate some of the confusion about SaaS. After looking at a brief history of SaaS, we’ll explore SaaS from its three major components: (1) a financial model; (2) a deployment model; and (3) a user interface model. Some believe that SaaS is limited to subscription pricing, off-site hosting, and browser-only access. We believe SaaS is about customer choice. We’ll examine the “once size fits all” SaaS approach versus SaaS that offers choices in its core components. Finally, I believe so strongly in implementing SaaS with user optimized choices that I propose a new concept that may more accurately describe what most mid-to enterprise-sized companies are demanding - Software as a Choice (SaaC).

The SaaS Evolution.

In his article, On-Demand/SaaS Reality, author Rick Sklarin does a great job of describing the three generations of SaaS evolution. The first generation, which he calls “SaaS 1.0,” became popular in the 1990’s and involved the early Application Service Provider (ASP) model. With SaaS 1.0, an ASP would purchase a limited license from the software vendor and then offer the software functionality to companies on a subscription-pricing basis. This first generation of subscription-based pricing typically was offered via client-server, was mostly accepted by the SMB market, and had very limited ability to customize the applications and integrate with other systems.

The second generation of SaaS, which Sklarin calls SaaS 2.0 or “pure SaaS,” was made popular by the success of Salesforce.com. In the ITSM space, this “pure SaaS” approach was first successfully marketed by Service-now.com. This second generation of SaaS currently defines the SaaS marketplace. It typically requires (1) subscription pricing; (2) deployment and hosting of the application and the data off the customer’s premises (i.e. “off-premise”); and (3) delivery over the Internet through an HTML browser (such as Internet Explorer, Firefox, Safari, etc.). In short, “pure” SaaS, or “modern” SaaS as some vendors label it, offers few choices. This model also has been criticized as having very limited customization and integration capabilities.

¹ Forrester Research article: “SaaS Adoption 2010: Buyers See More Options But Must Balance TCO, Security, and Integration” by Liz Herbert, June 2010.

² Gartner Research article: “The IT Service Desk Market is Ready for SaaS” by David Coyle, April 2009.

As a result of these deficiencies, Sklarin predicts the evolution to SaaS 3.0, which is a hybrid SaaS alternative, offering many more choices for end-users. As he describes, “. . . larger companies are demanding an evolution from the pure-play SaaS model . . .” He goes on to say that larger customers are requiring integration with on-premise applications and sophisticated configurability to meet specific business requirements, together with flexible delivery models. In other words, enterprise companies ultimately will demand more options than “pure SaaS.”

It is quite odd, and unfortunate, that the financial model for “pure SaaS” has been inherently tied to the delivery and hosting models, which are technical issues.

Introducing Software as a Choice (SaaC) – the 3rd generation of SaaS

(1) Financial Model – Lease or Own?

It is quite odd, and unfortunate, that the financial model for “pure SaaS” has been inherently tied to the delivery and hosting models, which are technical issues. The financial model has legal and accounting implications - but no technical connection. From a legal perspective, software generally is licensed from software vendors. However, these licenses can be (1) purchased or (2) leased. By purchasing the software license from the vendor, the customer generally has the right to use the software forever - in perpetuity. When the software is leased, however, the customer only has the right to use the software for as long as the subscription fee is paid. If the vendor goes out of business or becomes insolvent, a leasing model could be problematic. Leasing software, commonly referred to as a “subscription” model, is not much different than leasing a car: Leasing may be cheaper in the short-term (i.e., the first two to three years), but typically is much more expensive over a longer timeframe.

Notably, innovative software vendors are beginning to offer a “lease to own” financial option. They understand some organizations may prefer initially to lease the software because of short-term cash constraints, and then purchase the licenses within the next couple of years when operating conditions change. Some companies offer various discounts on the purchase price based on how long the software has been leased.

From an accounting perspective, when software is leased it is paid out of the company’s “operations’ budget.” In other words, all of the current subscription costs can be expensed on the income statement in the current period. Alternatively, purchased software over a certain dollar amount must be capitalized. The software shows up on the balance sheet as an asset that must be depreciated over time (e.g., three to seven years).

So which financial approach is preferable? It depends. Some companies that are constrained from a cash-flow perspective may be more short-term oriented and choose to lease the software. Other companies might be concerned about utilizing a mission-critical application that they do not own. These are questions primarily for the lawyers and accountants - with some help from the IT professionals. There is no “correct answer.” It will vary given the business circumstances. With respect to the financial model - choice matters. But there is no technical reason that a software product could not be deployed “on premise,” yet paid for using a subscription model.

Most SaaS 2.0 hosted environments require that every client has its software and data upgraded at the same time - offering no choice regarding when that occurs.

(2) Deployment Model – On-Premise or Off-Premise Hosting?

Should the application and data be hosted “on-premise” or remotely “off-premise?” Traditional “pure” SaaS typically requires that the software application and the company’s data be hosted off the company’s premises, which has some real advantages. For example, the software vendor or service provider supplies and administers the server and the operating system, and deploys all software updates from a remote location. Accordingly, all such costs and services are bundled in with the leasing or subscription price. Such a model can offer great efficiencies and economies of scale.

However, this approach is hardly a “no-brainer.” Some organizations, such as government and financial institutions, have greater security requirements for their data. The concept of hosting the data somewhere else and trusting an outside organization to protect their confidential data can be a show-stopper. Hosting data offsite also has implications for organizations that are subject to government requirements, such as the EU Data Privacy Directive, Canada’s PIPEDA, HIPAA and other governmental regulations. Additionally, most SaaS 2.0 hosted environments require that every client has its software and data upgraded at the same time – offering no choice regarding when that occurs. That may be great for “early adopters,” but for companies that would rather wait until all the kinks and bugs are worked out in the applications, it is not a desired approach. Another consideration is that integrating with existing on-premise systems is inherently easier when the application and data are hosted in the same environment and behind the same firewall as other integrated on-premise applications.

From a technical perspective, customers should be able to switch back and forth between hosting on-premise and hosting off-premise as their business needs change. This choice should be driven by the customer and should not matter to the software vendor, with the obvious exception that the software vendor will charge additional fees for the hosting costs if off-premise. The bottom line is that the hosting model should be a choice. Once again, there is no “correct answer.” It depends on the business needs.

Regardless of whether a smart-client or browser-client is utilized, software applications today can deliver the data over the Internet in exactly the same manner.

(3) User Interface Model – Rich-Client or Browser?

Transporting data via the Internet has become the de-facto standard for delivery, and this approach is much cheaper to most companies than deploying large area networks on a global basis. The advent of the Internet has been revolutionary with respect to an inexpensive way to share data from multiple geographic locations.

However, should the data and application be accessed via a browser or smart-client? From a technical perspective, the hosting choice discussed above of “On-Premise” or “Off-Premise” should support either method of accessing the data. However, many “pure SaaS” vendors imply that a remote hosting model requires that the application and the data be accessed only by using a common browser, such as Internet Explorer, Firefox, or Safari. This is not the case with more modern and flexible software applications.

Regardless of whether a smart-client or browser-client is utilized, software applications today can deliver the data over the Internet in exactly the same manner, utilizing protocols such as SOAP over HTTP. In other words, today’s smart-client applications can still be web-enabled. These two types of user interfaces can access the data in the same manner. Similar to the financial choice and the hosting choice, there are advantages and disadvantages to each. With a smart-client, less bandwidth typically is needed and the user experience can be faster and more robust, because the screens and other software overhead do not need to travel over Internet. As I wrote this article, I much preferred using Microsoft’s rich-client Word® application than a browser-based Word® application. The user experience is far superior. An example of a “smart client” on a mobile device is an iPhone® application. Such a user experience typically is far better than accessing an application on the iPhone from its Safari® Internet browser. Normally when someone is working in an application two to eight hours a day, the rich-client user experience is greatly preferred over a browser application.

A browser-based application also has its advantages. For example, all upgrades to the software are done remotely at the server level. This approach is no different than the mainframe computer in the 1980’s that was accessed with a “dumb client.” With a smart-client, updates to the client must be downloaded or “auto-deployed.” However, this automated installation to the client typically requires very little effort on the part of the end-user. Apple’s iTunes® application is a common example of a client update that must be auto-deployed. Windows® Update is another example. Probably the greatest advantage of a browser-only application is that it can be accessed from different computer platforms, such as Windows®, Mac®, or UNIX®. Such cross-platform access is very convenient, and is often a business requirement. Browser-based applications are also ideal for occasional-access applications, such as Amazon.com, or for users that only occasionally access the application.

Once again, the choice in the interface depends on the circumstances. There are good business reasons for each. It seems like the ideal solution would be for the user to be able to choose a richclient interface as the standard, yet be given the capability to switch to a browser interface when required (e.g., when accessing the data from a UNIX system). Once again, the data should be blind to how it is exposed. With both user interface options, the data can be delivered over the Internet.

Conclusion

SaaS 2.0, aka “pure” or “modern” SaaS, has been narrowly defined and deployed. Because of this, its days may be numbered. Enterprise-savvy customers today are demanding more choice in the financial model, the deployment model and the user interface model. An organization should be able to choose, for example, to host the application and data on-premise, yet pay for the software using a subscription model – while accessing the data using either a rich-client or a browser. If any model is really going to offer true “service,” it should offer choices to the customer. SaaS is here to stay, but it will continue to evolve because of customer demands. The 3rd generation of SaaS solutions will continue to emerge, forced by enterprise companies to offer choice. Good-bye “pure” SaaS . . . hello SaaS - *Software as a Choice!*

Vance F. Brown has been involved in the IT Service industry for over 10 years. From 1996 - 2000, Vance was President and CEO of GoldMine Software Corporation (formerly Bendata, Inc. and currently FrontRange Solutions - the makers of HEAT and ITSM Service Desk solutions, and GoldMine contact manager). Under Vance's leadership, the HEAT product went from a rating by the Gartner Group as a “niche” player to the “market leader” in both “ability to execute” and “vision.” Vance currently is CEO of Cherwell Software, the developer of Cherwell Service Management (www.Cherwell.com). Vance graduated from Wake Forest University, summa cum laude, with degrees in Economics and Computer Science. He graduated from law school, with honors, from the University of North Carolina, finishing as a member of the Order of the Coif and the Law Review. Vance would love to get your feedback about this article at Vance.Brown@Cherwell.com.