Product Arts

SaaS – A Product Perspective

Software-as-a-Service (SaaS) is quickly gaining credibility and market share against traditional packaged software. This presents new opportunities for product groups and also new challenges to teams used to developing packaged software. This article provides an overview of SaaS, how it differs from packaged software and specific new areas of focus from an end-to-end product perspective required to ensure a successful service.

What is SaaS?

SaaS has a broad definition and can mean different things to different people and is one subset of Cloud services. We'll start with a definition compared to traditional packaged software. Most applications today are purchased and installed on computers resident in the buyer's location, otherwise known as "on-premise". For enterprise applications, these are typically server-based products and enabled to operate across the organization to many users, e.g. Email, CRM, ERP, Collaboration, etc. The business model is usually to buy a server license upfront, and probably access licenses for every user who will want to access the server. This can be a significant upfront cost plus annual maintenance and support fees. This is addition to the cost of the server hardware and the ongoing cost of IT resources to install, update and maintain the system. The high price tag usually precluded small and medium businesses from owning these enterprise applications.

In the late 1990's, Application Service Providers (ASPs) introduced the concept of moving the software and servers out of the company and into a 3rd party who paid for the hardware, software and hosting administration. They then "rented" access to the software on a per-user or usage basis to customers who saw value in not having to deal with the upfront expense and hassle of buying it themselves. The concept was one of the big casualties of the DotCom meltdown in 2000, primarily due the high cost of running the service for each customer, many of whom needed some form of customization for their use of the application. It turned out to be difficult for the ASPs to make money by just offering the hosting service with traditional software.

About this same time, some enterprising companies were looking at ways to build the software in a different way to lower the cost of hosting for others. One of the most successful has been salesforce.com. They helped establish the definition of SaaS as follows:

- The service is hosted for the companies using the service (it is not on-premise software)
- The same software is shared by many users (and is standardized with few customizations)
- Access is over the internet (from the "cloud"), usually with a browser or thin client
- The business model is subscription or pay-per-use

SaaS today is increasingly available for many applications which are competing directly against traditional on-premise versions. While they are primarily in the B2B market, SaaS is also coming on strong in the B2C space for products like Personal Finance Management and Online Backup. Gartner estimates an 18% growth in the SaaS market in 2010 to $7.5B worldwide.

Author: Don Vendetti

SaaS Architectures

There are a few different architectures that are important to understand for SaaS offerings, as the implementations directly drive the cost model, but also merit consideration for individual strengths and weaknesses in various situations. The three architectures presented are Dedicated Single Tenant, Virtualized Single Tenant and Multitenant. Hybrids are also possible.

Figure 1 represents a Dedicated Single Tenant architecture. Each customer has a dedicated set of hardware for the three layers of web interface, application and database. The software running on each set of hardware can be the same, or there can be customizations at each layer. As new customers are added, a new set of hardware must be deployed. The benefits of this architecture is the relative ease to deploy a legacy application, the ability to provide some customizations to each customer, and the isolation of data in security or privacy conscious environments. The downside is much higher cost. This is the configuration used by the original ASPs, so the extra hardware and administration costs can be significant. Deploying and upgrading requires each customer to be done individually.

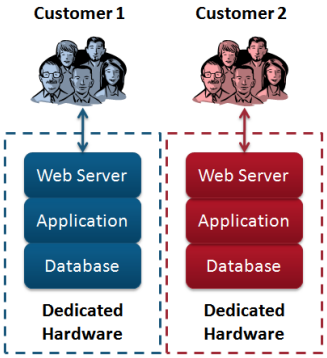

Figure 1 - Dedicated Single Tenant Architecture

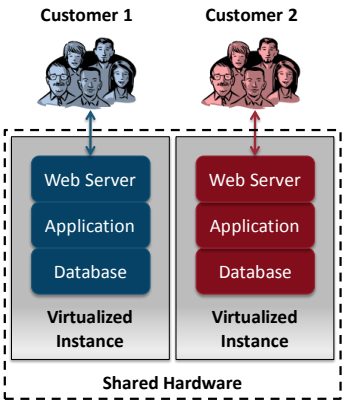

Figure 2 - Virtualized Single Tenant Architecture

Figure 2 depicts a Virtualized Single Tenant architecture. It is similar to Figure 1 in that each customers has their own dedicated software instance, but in this case they are running on a shared set of hardware. This is implemented by using a more powerful hardware platform than required for the individual Single Tenant implementations, plus the addition of virtualization software (such as VMWare) that enables the target software to think it has its own hardware. The advantage of this is much lower setup cost than the Dedicated scenario while preserving the customization and isolation capabilities. The disadvantage is scalability, as there is processing cost to the VM software that makes the system less efficient and scaling the architecture is more complex. Deploying and upgrading customers remains an individual customer task.

Figure 3 depicts a Multitenant architecture. All customers are using the same set of software deployed on scalable sets of hardware. A load balancer allocates users across the hardware sets. The advantage to this architecture is cost and massive scalability, plus the ease of adding a new customer. During an upgrade, all customers are upgraded at once and can be done in minutes. The downside is the application must be written to work in a multitenant environment for administering multiple customer sets and maintaining the data integrity across all customers. In addition, there are few, if any, customization options per customer, and each is limited to a predetermined set of options through configuration methods. Multitenancy is considered by some as the only "true" SaaS configuration due to cost and scalability, however, different customer needs, their willingness to pay, time to market and lifecycle stage of the market could enable any of the architectures.

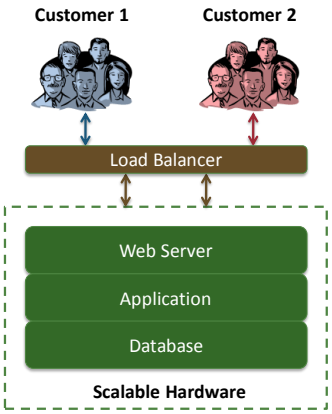

Figure 3 - Multitenant Architecture

How SaaS Differs

As mentioned, product teams used to delivering products for on-premise applications have several new issues to contend with for SaaS applications. These can affect the product requirements, the operational systems and processes, the business case, or some of each. The following highlights some of those challenges.

Issue	Discussion
Security & Privacy	Along with hosting the application, you have your customer's data. Your system and all operational processes need to be robust to protect it. The perception of security is considered the biggest hindrance to the adoption of SaaS for many companies. Questions your customers will have are: How is data transferred/transmitted, how is it stored, where is it stored, how is it backed up and archived, who has access, who controls access, etc. Especially in a multitenant environment, all customer data is likely comingled and this raises questions of privacy. You customers will want to know: How do you keep customers separate, how do you know access my data is protected, who has access, who controls access, etc. Besides designing a robust system, you'll also benefit from transparency with customers about answers to their questions. These can be in the form of whitepapers or specifically in your sales presentation backup material.
Purchasing & Billing	For an on-premise solution, purchasing is usually done in one-time purchased blocks of licenses paid in advance. This can be accomplished via a shopping cart approach, or even manual invoices through direct sales. Maintenance is usually an annual invoice. For SaaS, purchasing can be on-demand for any number of users, and can also be cancelled for any number on-demand. This requires a tighter coupling between the purchasing and operational systems to ensure users are activated and deactivated from the service in a timely manner. Billing is typically monthly.
Admin & Configuration	On-premise solutions provide full administrative control of the application and users. Once the application is moved to SaaS, administrators still expect full control; however, the standardized product will likely be much more restricted as to the controls that will be available. Instead, a limited set of configurations of the system are typically provided in addition to managing users, at least to some rudimentary level. Choosing the right controls and configurations to satisfy the majority of customers will be required to minimize the development and operational costs of the system. There will also need to be some process for handling unique customer requests not addressed by the standard platform.
Upgrades	Upgrades are conceptually one of the benefits of SasS, as you just do it en masse with the same release to all customers. It could also be one of the larger pains, both for you and for your customer. If your product is tied into their major business processes, advance notifications and potentially even training customers will be required before upgrading, and they will have to be ready with process changes on their end before the flip of the switch. If you've made large changes in the update, especially in database structures, you may also need to do a database migration of potentially a large amount of data. The planning and logistics of this can encompass a project nearly as large as the feature development itself, especially if minimal downtime is a requirement for customers.
Service Level Agreements (SLAs)	For on-premise software, the system is kept running by the IT department who responds to each issue that arises, with varying degrees of response. For SaaS, your customers will expect your service to always be available, on demand. Defining "always available" for your service is accomplished through your SLA, usually provided as part of your Terms & Conditions or contractually. It usually contains some level of

Issue	Discussion
	availability of the system (i.e. 99.9%) over some timeframe with a definition of how it is calculated and makes available exemptions for normal system maintenance and updates. You will probably also have a Support SLA that defines how responsive you'll be to fixing various classes of issues. Both of these may provide a financial penalty in the form of a credit back to your customers if you fall below the stated levels. These agreements are a party for lawyers to create. They also require your operations team to have the ability to monitor and quickly respond to issues, as the clock is always ticking during downtime. Because of the potential financial hit to revenue, expect your execs to have a high degree of interest in a real-time dashboard of your availability and support response.
Reporting	For on-premise systems, your customers usually have the ability to generate some level of reports, or potentially look into logs, to extract information out of the system. The more the system is tied into financial or operational systems of your customers, the more important this is and may even be required by SAS70. Your SaaS system will need to provide similar capabilities and this can often be a challenge. You either need to provide a set number of canned reports that addresses everyone's need (and which is unlikely for all, so plan for custom reports too) or you can provide access to the data and use reporting tools to let customers do their own thing. For a large amount of data, providing these capabilities and ongoing processing will effectively turn into a separate product by itself, and a costly one as the data set grows. The other interesting set of data accessible to you in SaaS is customer usage information. For on-premise systems, getting information out about how your customers use your product is difficult, if not impossible, as the system is sitting behind their firewall. In SaaS, you can get whatever you want to track, but like the customer reports above, comes at a cost to processing, storage and possibly real-time performance.
External Integrations	Some customers may have need to interface with data or applications still resident on-premise. This could be for a different application, or perhaps it could be to the same app split between on-premise and in-the-cloud for different sets of users. Providing APIs and documentation to customers could enable them to provide their own custom integrations to extend the functionality of your system. Additionally, some applications may benefit by the ability to have 3rd party integrations to other cloud applications, with customers able to create their own mash-ups.
Trials	Providing a free trial for on-premise software is possible, however, the implementation is usually messy if it isn't hosted for the customer. They need to download and install the software, configure it, and then enable it for specific users to test. For SaaS, adding trial customers into the system is usually pretty easy, but you do have to account for the trial mode (or a dedicated platform) and trial expiration and may also want to disable specific features, limit number of users, etc. Depending on your architecture, you may also have a customer migration event to contend with off the trial platform and onto a commercial platform if they continue purchasing.
Offline Access	Almost by definition, SaaS involves a browser or thin client, so if you are not connected to the internet, then you have no access to the application. This does not have to be a rule and some applications require operation while disconnected, especially in mobile environments. This can be enabled with a hybrid scenario of SaaS in the cloud and an intelligent client on the access device. The most common example of this is email, where the email servers can be in the cloud with a full email client residing on the device. The client maintains a local database that is synced to the cloud when

Issue	Discussion
	connected, but is also available for offline use if the cloud is temporarily unavailable. This can present a much better user experience, but of course, this is much more complex to develop and manage.
Testing Environments	The ease of updating the application, especially in a multitenant environment, makes it especially important to be able to do full testing in an environment that mirrors what is in production. If your upgrade fails for some reason, all customers are affected, and it is very easy to have a failure due to a configuration differences between the testing and production environments. This can be a large cost required to replicate the production environment, but the business case comes from the savings incurred from SLA violations and hard-to-quantify brand reputation if outages are common.
Customer Service	Customer Service and Tech Support for on-premise software can often be handled by VARs or other channel partners as the first tiers to customers. For SaaS, the vendor takes on the brunt of the customer support activities from tier 1 to tier n, and requires a responsive and deep organization. In effect, your organization will need to turn from a primarily development-focused team to an operationally-focused team. This will require multiple levels of support, reaching from front line support reps all the way into the development team, to be able to respond to operational and application issues and questions on a regular basis. This may be one of the biggest cultural changes (and shocks) required in moving from being a traditional software vendor to an SaaS provider.
Solution sale vs. Product sale	Similar to Customer Support, the focus from a sales perspective for SaaS needs to be on the entire solution being offered and over the lifecycle of the product. The biggest effort may no longer be in getting the initial sale for on-premise software that then disappears into an organization. The customer is buying a partner that they will interact with and be dependent on for the life of the product. Sales tools will be required to quantify the ROI, in addition to published documents and product roadmaps to address many of the questions that have been posed in the preceding issues list.

Summary

Software-as-a-Service is quickly becoming major player in both the B2B and B2C markets and can provide some significant benefits to customers. It can also present significant challenges in product design, operations and company culture to deliver them effectively. This article has provided a discussion of many of the different issues that will be encountered by traditional product teams in delivering an end-to-end service. For additional reading:

SaaS Defined & Resources
Gartner Forecast
Multitenant Architectures

About Product Arts

Product Arts specializes in Product Management consulting and training. Our consulting consists of product strategy and planning, market performance improvements and processes for creating and delivering products. Our training includes public and private custom training for product management and associated staff. For more information, go to www.product-arts.com or email info@product-arts.com.

