International Journal of Research in Science And Technology http://www.ijrst.com/
(IJRST) 2012, Vol. No. 2, Issue No. II, Apr-Jun **ISSN: 2249-0604**

HOW IT DEPARTMENTS BENEFITED BY SAAS

Deepak Kumar Kaushik, Dr. Rajesh Pathak

HOD Deptt. of Computer Science, & Engg.
GNIT Institute of Technology,Gautam Budh Nagar, U.P.
Research Scholar, CMJ University, Shillong, Meghalaya

INTRODUCTION

SaaS popularly stands for Software as a Service. Though there are multiple terminologies in use (one of which is Microsoft's way of addressing the same as "Software + Service"), what is ultimately important is that the objective of such a model is met - which is to facilitate the use of the required software functionality as services from the vendor directly without being burdened with the responsibility of hosting the infrastructure and, maintaining the software on the consumer side. Indeed, the concept does look very interesting. Service consumers can now focus on their core business areas by consuming the required software services without having to pitch into their non-core areas like IT infrastructure maintenance, maintenance of an additional IT team etc.

Many ISVs already have a huge list of Enterprise Products in their portfolio. These products are of varying proportions (small, medium and large sized), servicing various domains and business functionalities. A significant challenge that needs to be addressed is with regard to how SaaS based solutions of such products can be rolled out cost effectively (with less effort, optimized infrastructure cost, and effective operational & maintenance cost as primary drivers) and within highly aggressive timelines.

Remember, the core objective of SaaS is to enable a set of business functionalities as hosted services, which can serve multiple customers and enterprises (as also referred to as 'tenants' in the SaaS/Cloud Computing world). It is considered to be most optimum when all tenants are served with the required (and agreed upon) performance and scaling parameters satisfied with same single source code in execution. But what we are talking here is the most optimal or highest maturity level of the SaaS model.

However, it also important for SaaS vendors or implementation experts to understand various options that are available to service tenants to satisfaction, rather than directly jumping onto the highest level of the SaaS model. Let us try to understand the various options that one can look into.

Level 1 - Ad Hoc/Custom:

In the first level of maturity, there will be separate instances of the source code hosted for each customer. In this level, code would have specific customization required for those customers as the multi-tenancy would have been not addressed at architecture level.

As we can understand from the definition above, the scenario is simple. Such products are not expected to undergo architectural changes or rework in order to add the mandatory features of SaaS such as multi-tenancy, data separation, scalability etc. In such products, multi-tenancy is provided by hosting dedicated instances for each customer.

This is still acceptable when the customer base is small and the level customization needed is minimal. Because, with cost effective technologies for optimal infrastructure usage popularly available today (yes, you are right, I am referring to Virtualization), one needn't worry much about infrastructure overheads(though licensing cost is still applicable). Proven Virtualization products available in the market today help achieve optimal infrastructure usage easily. However, one needs to thoroughly analyze as regards how many customers are targeted, what would be the operational and maintenance cost (for those many dedicated instances) etc. (Remember, in such deployments, critical mechanisms such as proper metering (as per various subscriptions required) , automated billing (integrated appropriately), integrated monitoring etc would not be in place to make it cost effective.) But this methodology may be suitable when the customer base is small, the required level customization is considerable and re-architecting the whole product extremely complex and time consuming (remember, turnaround time is important where one wants to take competitive advantage)

Level 2 - Configurable:

The second maturity level provides greater program flexibility through configurable metadata, which allows multiple customers to be served separate instances of the same application code.

This option is very similar to the Level 1 option above. However, in this case, the product architecture and design has been changed in such a way that, personalized customization for different customers (depending on their business needs) are provided through detailed configuration options. The overhead of having to maintain multiple code bases is reduced (as personalized instances of the same code base are hosted) this simplifies maintenance by having a single, common code base.

According to a Gartner Group estimate, SaaS sales in 2010 reached $10 billion, and were projected to increase to $12.1bn in 2011, up 20.7% from 2010. Gartner Group estimates that SaaS revenue will be more than double its 2010 numbers by 2015 and reach a projected $21.3bn. Customer relationship management (CRM) continues to be the largest market for SaaS. SaaS revenue within the CRM market was forecast to reach $3.8bn in 2011, up from $3.2bn in 2010.[6]

The term software as a service (SaaS) is considered to be part of the nomenclature of cloud computing, along with infrastructure as a service (IaaS) and platform as a service (PaaS).

Material and method

Centralized hosting of business applications dates back to the 1960s. Starting in that decade, IBM and other mainframe providers conducted a service bureau business, often referred to as time-sharing or utility computing. Such services included offering computing power and database storage to banks and other large organizations from their worldwide data centers.

The expansion of the Internet during the 1990s brought about a new class of centralized computing, called Application Service Providers (ASP). Application service providers provided businesses with the service of hosting and managing specialized business applications, with the goal of reducing costs through central administration and through the solution provider's specialization in a particular business application.
Software as a service essentially extends the idea of the ASP model. The term *Software as a Service (SaaS)*, however, is commonly used in more specific settings:

- whereas most initial application service providers focused on managing and hosting third-party independent software vendors' software, as of 2012 software-as-a-service vendors typically develop and manage their own software

- whereas many initial application service providers offered more traditional client-server applications, which require installation of software on users' personal computers, contemporary software-as-a-service solutions rely predominantly on the Web and only require an internet browser to use

- whereas the software architecture used by most initial application service providers mandated maintaining a separate instance of the application for each business, as of 2012 software-as-a-service solutions normally utilize a multi-tenant architecture, in which the application servesmultiple businesses and users, and partitions its data accordingly

The *SAAS* acronym allegedly first appeared in an article called "Strategic Backgrounder: Software as a Service", internally published in February 2001 by the Software & Information Industry's (SIIA) eBusiness Division

DbaaS (Database as a Service) has emerged as a sub-variety of SaaS.

Pricing

Unlike traditional software which is conventionally sold as a perpetual license with an up-front cost (and an optional ongoing support fee), SaaS providers generally price applications using a subscription fee, most commonly a monthly fee or an annual fee. Consequently, the initial setup cost for SaaS is typically lower than the equivalent enterprise software. SaaS vendors typically price their applications based on some usage parameters, such as the number of users ("seats") using the application. However, because in a SaaS environment customers' data resides with the SaaS vendor, opportunities also exist to charge per transaction, event, or other unit of value.

The relatively low cost for user provisioning (i.e., setting up a new customer) in a multi-tenant environment enables some SaaS vendors to offer applications using the freemium model. In this model, a free service is made available with limited functionality or scope, and fees are charged for enhanced functionality or larger scope. Some other SaaS applications are completely free to users, with revenue being derived from alternate sources such as advertising.

A key driver of SaaS growth is SaaS vendors' ability to provide a price that is competitive with on-premises software. This is consistent with the traditional rationale for outsourcing IT systems, which involves applying economies of scale to application operation, i.e., an outside service provider may be able offer better, cheaper, more reliable applications.

Architecture

The vast majority of SaaS solutions are based on a multi-tenant architecture. With this model, a single version of the application, with a single configuration (hardware, network, operating system), is used for all customers ("tenants"). To support scalability, the application is installed on multiple machines (called horizontal scaling). In some cases, a second version of the application is set up to offer a select group of customers with access to pre-release versions of the applications (e.g., a beta version) for testing purposes.

This is contrasted with traditional software, where multiple physical copies of the software— each potentially of a different version, with a potentially different configuration, and oftentimes customized—are installed across various customer sites.

While an exception rather the norm, some SaaS solutions do not use multi-tenancy, or use other mechanisms—such as virtualization—to cost-effectively manage a large number of customers in place of multi-tenancy. Whether multi-tenancy is a necessary component for software-as-a-service is a topic of controversy.[11]

Characteristics

While not all software-as-a-service applications share all traits, the characteristics below are common among many SaaS applications:

Configuration and customization

SaaS applications similarly support what is traditionally known as application *customization*. In other words, like traditional enterprise software, a single customer can alter the set of configuration options (a.k.a., parameters) that affect its functionality and look-and-feel. Each customer may have its own settings (or: parameter values) for the configuration options. The application can be customized to the degree it was designed for based on a set of predefined configuration options.

For example: to support customers' common need to change an application's look-and-feel so that the application appears to be having the customer's brand (or—if so desired—co-branded), many SaaS applications let customers provide (through a self service interface or by working with application provider staff) a custom logo and sometimes a set of custom colors. The customer cannot, however, change the page layout unless such an option was designed for.

Accelerated feature delivery

SaaS applications are often updated more frequently than traditional software in many cases on a weekly or monthly basis. This is enabled by several factors:

- The application is hosted centrally, so new releases can be put in place without requiring customers to physically install new software.

- The application only has a single configuration, making development testing faster.
- The application vendor has access to all customer data, expediting design and regression testing.
- The solution provider has access to user behavior within the application (usually via web analytics), making it easier to identify areas worthy of improvement.

Accelerated feature delivery is further enabled by agile software development methodologies such methodologies, which have evolved in the mid-1990s, provide a set of software development tools and practices to support frequent software releases.