The Software-as-a-Service Model for Mobile and Ubiquitous Computing Environments

Nikos Anerousis and Ajay Mohindra
IBM Research
19 Skyline Dr
Hawthorne, NY 10532, USA

Extended Abstract

1 Introduction

In the last few years the biggest online firms have been offering exciting new services to Internet and cellular network users on a variety of devices without any software to buy, install and maintain. This dynamic, online, "pay-as-you-go" service relationship changes the traditional assumptions, relationships and value proposition between software vendors, clients, end-users and 3rd-party service providers. As an evolution of the early Application Service Provider (ASP) model, SaaS is poised to undergo rapid growth and have significant impact in the software and services industry, across all devices.

An area of unique opportunity for SaaS is the mobile and ubiquitous computing space. The SaaS model has a number of advantages that are uniquely suited to a resource-constrained mobile computing environment. This short paper gives a high level overview of the SaaS opportunity and further presents the architectural considerations for SaaS providers and mobile computing environments.

2 Software as a Service

2.1 Main characteristics

Software as a Service (SaaS) describes a trend where customers obtain their application functionality through a network-delivered service. The applications are hosted in a data center and maintained by the service provider. Customers (users) access the application remotely via the Internet (at the back-end), and possibly via a wireless data network in the front end (WiFi, GPRS, etc). The protocol employed to communicate with the application can be anything, but in practice the main method of interaction with SaaS applications is common Web-based access using a browser on the client device.

In the SaaS model applications are designed from the ground up to be delivered as a service. This includes the following characteristics:

- A multi-tenant design where an instance of the application accommodates multiple users (single-level multitenancy), or even multiple resellers of the service with each reseller serving its own pool of users (two-level multitenancy).
- A charging model where customers pay for the services on a metered basis (pay as you go).
- Support for all the functions necessary to provide the application as a service, including subscription, authentication, authorization, metering, monitoring, reporting, billing, and support.
- Minimal level of application customization to avoid major implementation and integration costs.

At a first glance, the additional requirements for providing a SaaS application may seem prohibitively complex. For this reason, service providers are investing in developing

1-4244-0499-1/06/$20.00 ©2006 IEEE

hosting platforms for SaaS that provide all these collateral services. The application designer then is only concerned with providing the appropriate APIs for linking with the platform services. This significantly reduces the complexity of the design.

The key players in the SaaS ecosystem are the following:

- End-users: Market research indicates that a large part of the initial customer segment will be Small and Medium Business (SMB), with consumers following.
- SaaS Providers: the companies that are provide SaaS services to the end-users.
- SaaS Developers: the people who are developing SaaS applications.
- SaaS Hosts: the companies who operate the infrastructure that the SaaS providers use to deliver their services to End-users.
- Hub Provider: an entity who operates a hub to enable the integration of multiple SaaS services and legacy applications.

SaaS represents a potentially disruptive model for software. The entire marketing, development, and business ecosystem for obtaining application functionality is transformed to an on-demand model. Most players in the current application software and applications services industries could be affected. Current Independent Software Vendors (ISVs) will find that their existing customers may prefer to obtain their application as a service, versus the current own and install model. Consequently, ISVs will need to transform their business model to a model based on metrics (typically priced at dollars per user per month) with service and upgrades included, rather than the common licensing and support model.

Current ISVs will need to focus on assuring that service related non-functional requirements (billing, metering, monitoring, QoS, reporting) are considered in the development of their application and providers of hosting will need to assure that they can provide the reliable connectivity required by SaaS providers. If middleware can also be delivered as services, then SaaS will disrupt the providers of Middleware as well.

SaaS providers are focused on selling an annuity service measured, typically, in users-per-month-per-application. Their goals are customer retention, creating/maintaining a single common instance of the application that is shared by all customers, and delivering a highly reliable service. The SaaS provider is typically concurrently working on many simultaneous releases and deploying them on a periodic basis. In addition, SaaS Providers only need to assure that their application is tested and operates in production in one data center (which today they typically own). The SaaS provider's focus is on adding function and minimizing disruption to their current customer base so they can retain existing customers and acquire new ones.

2.2 Advantages of the SaaS Model

The primary reasons SaaS customers select SaaS over purchased software or open source software are:

- Time to market: customers who select a SaaS solution will typically be utilizing the solution faster, in some cases immediately, but usually in not more than a few weeks, compared to the purchased option.
- Reduced I/T skills: the customer of a SaaS application does not have to invest in skills to learn how to install, maintain, and operate the application. The SaaS provider retains all I/T skills for the application.
- Direct accountability: the customer selects the application and holds the SaaS provider accountable for operations and QoS of the SaaS service.

- Reduced up front costs: there are minimal up front costs to a customer of a SaaS application. Compared to a standard application where the customer would need to purchase the software and procure hardware on which to operate the application, in SaaS, the customer typically only pays for the amount of service they utilize on a contractual basis.
- No cost for upgrades and fixes: the SaaS provider is responsible for maintaining the application. The SaaS provider performs all enhancements and patches that might be required to maintain the application.
- Managed security: security used to be an issue for users having customer data maintained at a SaaS provider. Now, however, security is a benefit because the SaaS provider retains records and assures compliance with current and emerging guidelines such as Sarbanes-Oxley.

SaaS providers focus on reducing the fixed cost of operations and minimizing the variable cost to achieve profitability. Consequently, open source technology and low cost hardware are becoming an important part of the SaaS model. A standard platform that provides functions that all SaaS developers need could significantly accelerate the market. Such a platform could also assist ISVs in addressing the SaaS marketplace by reducing the time to market.

2.3 The SaaS Delivery platform

A SaaS datacenter platform is designed to host SaaS applications and the platform services they require. SaaS applications can exist standalone (running on top of an operating system), or as part of an application container, such as a J2EE container. Application containers are central building to the SaaS model. They provide critical lifecycle services for hosted applications, and in some cases, extended availability and load-balancing. In addition, SaaS platform services such as metering and reporting run themselves on application containers.

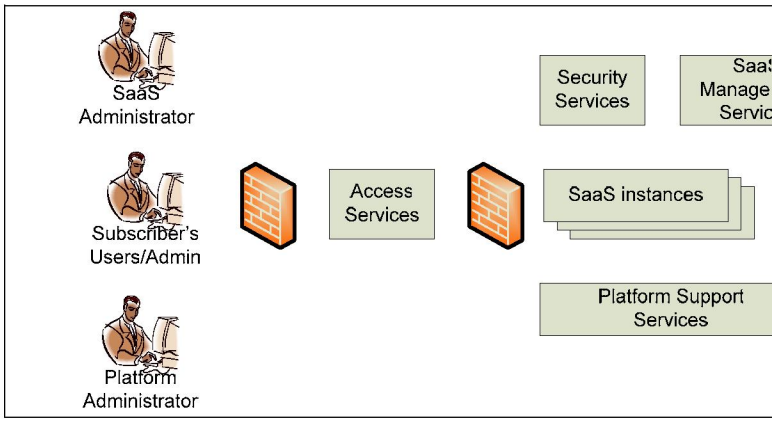

Figure 1 SaaS Delivery Platform

Figure 1 shows the logical topology of a SaaS delivery platform. A typical SaaS delivery platform deployment involves a server farm equipped with the appropriate application containers. The containers that are dedicated to providing platform services are separate from the ones that host SaaS Provider applications. The first set is provisioned with very high availability and security requirements. The second set (which includes the applications, application servers and databases necessary for running a SaaS application), can be configured with a variety of availability and security options, typically specified by the SaaS provider. The deployment model in this case is not different from common industry practice in outsourced and managed services.

The users are the consumers of the SaaS delivery platform. They can be categorized into three subgroups:

1. Platform Administrators manage the platform that hosts the SaaS application. They are responsible for managing the network, hardware, and software that comprises of the SaaS platform
2. SaaS Administrators manage the lifecycle of SaaS applications
3. Subscriber's users and administrators are the consumers of the SaaS.

The SaaS administrator are responsible for managing set of users who have access to the SaaS. The Subscriber's users consume and use the SaaS.

The services of the SaaS platform can be categorized into the following groups:

2.3.1 Access Services

The Access services provide a common entry point for users of the SaaS platform. It consists of services that provide user authentication, and authorization. The Access services are typically located in the DMZ of the SaaS Providers. Further for users with mobile and ubiquitous devices, the Access service also provides ability to

2.3.2 Platform Support Services

The Platform support services consists of a set of services that support the delivery of SaaS applications. The Platform support services comprise of the following two subgroups

1. Business Support Services (BSS)
BSS provides the required interaction with the customers of the SaaS application. Customers can be either end users or resellers of the application. The BSS platform is a portal, data model and set of processes and tools for ordering SaaS applications, tracking contracts, defining SLAs, subscribing to services offered, providing self-help to the customers of the application (reporting and end-user management).

2. **Operation Support Services (OSS)**
The Operation Support Services (OSS) Platform is a collection of systems, OS and middleware software, and IT Automation features into a replicable standard platform. It includes a set of management agents, instrumentation, consoles/portlets, event analyzers, and deployment workflows for defining, configuring, deploying, operating and monitoring the SaaS applications.

2.3.3 Security Services

The Security Services provide and manage the set of users and roles associated with the subscribers of the SaaS. The Access services uses the data managed by the Security services to authenticate and authorize users to access the SaaS instances.

2.3.4 SaaS Management Services

The SaaS Management services are a collection of services that provide lifecycle management of applications that are being delivered as services.

2.3.5 SaaS instances

The SaaS instances are the actual applications that are being delivered as SaaS. An instance of the application is created and deployed after a subscriber selects an application for subscription. As mentioned earlier, these applications typically are J2EE applications that run on a collections of servers. Based on the SLAs supported by each of the applications, the servers could be configured to provide high availablility.

2.3.6 Data Services

The Data services manages and persist the data used by SaaS instances.

2.3.7 Integration Services

The Integration services manage access to enterprise data that the SaaS application uses from data providers that are located outside of the SaaS provider's platform.

The design of the SaaS delivery platform is key to the success of the SaaS model. A closed platform that requires extensive setup for publishing and hosting a software application is bound to fail as very few software developers would be willing to invest the resources. In contrast, an open SaaS platform that makes it very easy to publish and host software applications would permit large number of software developers to experiment with delivering their applications as a service. We believe that a Service Oriented Architecture (SOA) approach based

on the web services paradigm is the correct approach to designing a SaaS platform. In this approach, each of the logical components shown in Figure 1 would comprise of SOA components that publish well known interfaces for software developers to write to. When designed and developed properly, the components would offer reusable services (similar to shared libraries) that application developers can reuse during application development. For example, a software developer need not develop and implement application specific components for monitoring, logging, authentication, and authorization, but instead can reuse the components that have been provided by the underlying platform. This model of software development promotes reuse and allows the software developers to focus primarily on the core logic for the application.

3 SaaS in Mobile and Ubiquitous computing

For several years, mobile computing has been evolving around the network-centric model. Because of size and power limitations, the complexity of applications that run natively on mobile devices has been limited. SaaS does not fundamentally change the model for network-centric computing. Applications will continue to be hosted in the periphery and accessed from a thin client running on the mobile device. What is fundamentally changing however, is the complexity and the logistics for developing, installing and operating such applications. The SaaS Delivery platform promotes an open service delivery model. The service provider, usually the provider of mobile network access acts as a service integrator. Applications are developed by third parties and hosted on the SaaS delivery platform. The service provider maintains the client relationship, performs all metering and billing functions and is responsible for all operational issues of the platform including capacity planning.

The promise of SaaS is that functionally rich application software is delivered as a service to its consumers. Underlying this promise is the expectation that the consumers of the application software have continuous network connectivity. A web browser is the preferred software client for consuming an application. The SaaS model does not mandate any client-specific installation of software to access the software service. However, to support mobile and ubiquitous devices, the SaaS application developers have to be aware of and need to design for the characteristics of the devices that they intend to support.

3.1 Challenges to support mobile and ubiquitous devices

Some of the differentiating characteristics of mobile and ubiquitous devices than traditional desktop computers are as follows.

- Small form factor – The mobile and ubiquitous devices come in a variety of fomr factors ranging from notebook computers to handheld organizers and cell phones. The small form factor of the devices translates to small screen size, limited memory and disk space, and a limited user interface.
- Low bandwidth – The state-of-art wireless networks today offer a very limited bandwidth to a majority of mobile and ubiquitous devices. Typically bandwidth ranges from 9.6 Kbps for GSM and CDMA devices. Higher bandwidth, mostly for downloads, of upto 2.4 Mbps is available for EVDO protocol.
- Periodic disconnection – Due to inherent mobility, the mobile and ubiquitous devices are susceptible to periodically disconnecting from the network either as a result of poor wireless network coverage or lack of power.

These differentiators warrant that the SaaS application developers design their solutions such that the applications are usable on small form factor devices. By utilizing a transcoding proxy in the SaaS delivery platform, SaaS application developers can limit the number of application features that are available to users on mobile and ubiquitous devices. This calls for design of "adaptive" SaaS application that adapts to the devices that it is being used from. For example, a SaaS application developer may choose to expose only a subset of a application features that can be easily performed on small-form factor devices using a text-driven menu. In contrast, on fully functional devices, the same set of features could be available using a graphical menu. Similarly, to handle low bandwidth for mobile and ubiquitous devices to is for the application developers to ensure that high bandwidth items such as rich graphics and images, are made optional on the small devices, and key application function does not rely on the availability of rich user interface.

The most fundamental change in the design of SaaS applications is to handle periodic disconnection of the devices. By design, the SaaS application either requires only a web browser to operate, or a minimal client side application code. However, to handle disconnection of mobile devices, the client side application code becomes a requirement. The client-side application is responsible for caching data on the device, and enabling "offline" operation of the application when the device is disconnected. The application code is also responsible to periodically "synchronize" modifications to the data with the master copy whenever connectivity is restored. The need for client-side code introduces additional complexity for the life-cycle management of a SaaS application, as the SaaS management services need to install, and manage the correct version of the client code as the SaaS application evolves.

4 Conclusions

The software industry is in the midst of an important transformation to a services-based model. It is already progressing rapidly in the online space and is expected to affect the entire software and application development ecosystem. This paper presented an overview of the SaaS model with special attention to mobile and ubiquitous devices. We believe that SaaS brings a significant opportunity in this space, as it gives additional leverage to traditional application developers to create and extend offerings on mobile platforms.

5 References

[1] K. Bennett, P. Layzell, D. Budgen, P. Brereton, L. Macaulay, M. Munro, "Service-based software: the future for flexible software", in Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC'00), p. 214.

[2] SIIA, "Software as a Service: Changing the Paradigm in the Software Industry", SIIA and Tripletree Industry Analysis Series, 2004.

[3] Mark Turner, David Budgen, Pearl Brereton, "Turning Software into a Service", IEEE Computer Magazine, October 2003 (Vol. 36, No. 10) pp. 38-44.

[4] IBM Corporation, "Common Services Delivery Platform Architecture Document", Unpublished Manuscript, 2005.

[5] George H. Forman and John Zahorjan, "The Challenges of Mobile Computing", IEEE Computer, April 1994. pp 38-47.