

Platform Leadership in Software as a Service: How Platforms Facilitate Innovation

Abstract

Software as a Service (SaaS) platforms are used for developing and delivering SaaS applications, i.e. software programs that are accessible over the Internet via a web browser. They are part of IT systems, known as cloud computing, which consist of utility computing and SaaS applications (Armbrust, et al. 2009). The former is used by SaaS providers to create and run their software and thus can be associated with SaaS platforms.

The concept of a 'platform' is related to industries where the end customer offering consists of a bundle of components provided by various parties (Gawer and Cusumano 2002). The core product is often of very little value if not complemented with other products and services. Platforms emerge when products are complex and many independent producers make different pieces of them. Forming an ecosystem of innovation, in which a platform provider works with companies supplying complements, can greatly raise the total value of both parties' innovations (Gawer and Cusumano 2008).

SaaS platform firms provide infrastructure and software systems for complementors, i.e. application developers, and facilitate them in innovating applications. The design of SaaS platforms differs. *Amazon Web Services*, an *Infrastructure as a Service* (IaaS) type platform, provides low-level access to computing resources, which gives flexibility, but requires technical expertise. *Google App Engine* and *Force.com* move the abstraction level higher; they are examples of the *Platform as a Service* (PaaS). Complementors do not need to manage the hardware, but there is some loss of flexibility, and, as is the case with *Force.com*, they may need to use proprietary programming languages.

Not all of the conditions for a platform leader to emerge, stated by Eisenmann et al. (2006), are satisfied in the SaaS platform market currently. Namely, it is possible to differentiate among the PaaS-type platforms and there are not strong enough network effects for IaaS-type platforms. Uncertainty, related to the infancy of this industry and no clear leaders, requires complementors to select SaaS platforms with caution.

Following the analysis of the situation in the market, this paper aims at identifying strategic activities of platform firms that help application developers to innovate, and potential threats they should consider. SaaS platforms facilitate complementors, because they lower financial and technical barriers to entry and mitigate the consequences of business failure. They provide infrastructure on a pay-per-use basis, and therefore release from expenditure costs and hardware maintenance. Development time is reduced thanks to re-usable software components and various Application Programming Interfaces. Moreover third parties can reach the market more easily with platform marketplaces and by inheriting the established reputation of the platform firm. The major risk in the use of SaaS platforms for complementors is related to platform lock-in. This is highest for *Force.com*, and lowest for *Amazon Web Services*, in the examples examined here. Security, privacy and reliability issues also need to be considered. In fact, these constrain innovation in mission-critical applications, such as financial services. Nevertheless, the SaaS platform barriers can also be seen as an opportunity for third parties. They can offer platform complements to help application developers overcome the risks, such as common applications that work across different platforms, tools for migrating applications from one platform to another, and software components guaranteeing security.

Key words: Software as a Service, platform, cloud computing

1 Introduction

1.1 Background and Objectives

The Software as a Service (SaaS) principle is to deliver programs over the Internet in a pay-per-use model. The advantages of SaaS are that users can access it from any computer, they do not need to worry about compatibility with the operating system and are released from expenditure costs and software maintenance. SaaS is gaining popularity among a number of customers and has attracted many software vendors as a new business model through which to offer their products. A well known example of SaaS is *Google Docs*, a suite of office applications, such as a word processor and a spreadsheet, accessible with a web browser.

The next move for big software vendors was to provide platforms to build and run programs in the SaaS manner, known as cloud computing. Such platforms consist of hardware infrastructure and software systems that are a base for SaaS development. They are of no use to the end users without applications, so they need ‘complementors’, i.e. software developers, to create these applications. SaaS platform firms attract many complementors in order to build momentum and make them embrace the platform standards (either proprietary or open). They also often offer SaaS applications themselves, and benefit from external developers complementing them.

A business structure, in which a firm needs complementors to create the end product, is the subject of the research on platforms. Gawer and Cusumano (2002), among others, explain many aspects of the dynamics accompanying complex ecosystems, in which a core product, i.e. the platform, is used as a base for complementary products and services. Numerous industries and sectors (including software, personal computer, networking equipment, and banking) were identified and the strategies of successful companies, being able to harness the platform dynamics, were analysed.

Platforms for building and running SaaS applications open new opportunities for small companies, which make complementary products. They are supported with low-level infrastructure and Application Programming Interfaces (APIs), so they can put more resources into innovation in applications. Innovating becomes less risky, as a company does not need to make expenditure on infrastructure, but can outsource it and pay per usage. However, there are also drawbacks, which complementors need to consider, such as platform lock-in and privacy issues.

Existing platform research focuses on the platform firm’s interest and how it can harness the innovation of others for its own good. The aim of this paper is to analyse platform offerings and compare them with the interests of complementors. Identification of platform strategic activities and choices that can facilitate complementors in innovating is one goal. Another is to try to recognise potential drawbacks and risks related to working with platforms, as viewed from the perspective of third parties. The paper is based on final dissertation by Kolakowski (2009) at the University of Cambridge.

1.2 The Platform Concept

The concept of a ‘platform’ is associated with industries where the end customer offering consists of a bundle of components provided by various parties (Gawer and Cusumano 2002). The core product is often of very little value if not complemented with other products and services. Platforms emerge when products are complex and many independent producers make different pieces of them. Forming an ecosystem of innovation, in which a platform provider works with companies supplying complements, can greatly raise the total value of both companies’ innovations (Gawer and Cusumano 2008). The platform phenomenon is closely related to Open Innovation (Chesbrough 2003). Companies need to leverage the work of others in a world of rapid technological change. As the pace of innovation increases, a single company is not able to create all the modules of the product on its own. Building a platform can facilitate the introduction of innovation through components developed by external parties.

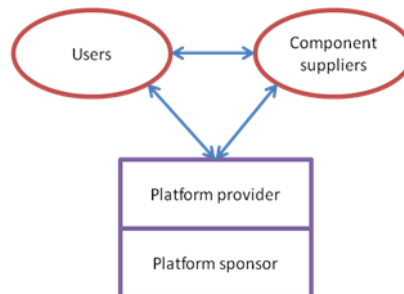


Figure 1. Relationships between parties in the platform ecosystem (source: Eisenmann, et al. 2007)

The platform business model has a triangular structure, consisting of users, component suppliers and platform firms, as depicted in Figure 1 (Eisenmann, et al. 2007). Platform maintenance and development is performed by three parties, namely the sponsor, provider and component suppliers (also known as co-developers

or ‘complementors’). Platform sponsors hold the Intellectual Property (IP) rights and control the distribution and development of the platform. Platform providers are in charge of mediating with users and co-developers. Component suppliers make products and services that are not delivered by the platform provider. End users influence co-developers and vice versa. Transactions can occur not only within the platform firm, but also directly between users and co-developers, the platform facilitates these transactions.

1.3 Platform Strategies

The value of complements increases the value of the platform, so there are opportunities to benefit from the innovations of others. Companies seek to make their products become the foundation, on which others build their offerings. That means they seek to become the platform leaders and to drive innovation in the industry. This goal requires a specific strategy, which is different from the product strategy.

The aspects of the platform strategy can be grouped into business and technology challenges (Gawer and Cusumano 2008), as presented in Table 1. The key technology challenges include designing the right architecture and interfaces, and disclosing IP selectively. The choice of technologies has consequences for the switching costs to alternative platforms. The main business issues are related to the scope of the firm, whether to make the key components itself, or to incentivise complementors to make them. Relationships with complementors are important, particularly in regard to what extent they are collaborative and how the platform firm facilitates the third parties in reaching the market. Pricing mechanisms, i.e. how to charge both sides of the network, users and complementors, is relevant as well. All of these problems require balancing between what is good for the company versus what is good for the end users, complementors, and the industry as a whole.

Table 1. The challenges of platform strategies

Technology Strategy Challenges	Business Strategy Challenges
Designing platform architecture	Defining the scope of the firm
Specifying interfaces	Managing relationships with external complementors
Selectively disclosing IP	Setting pricing strategy
Maintaining switching costs	

2 Defining SaaS Platforms

SaaS platforms are used to develop and deploy SaaS applications; they are a type of software platform. Software platforms provide infrastructure for the development and delivery of software applications in general (Evans, et al. 2006). According to Andreessen (2007), a software system is a platform, when it can be programmed by external developers. In practice this is achieved by exposing capabilities through APIs. A software platform is a software program that makes its services available to other programs via APIs. That feature allows third party developers to provide customised products and address the needs of market niches.

Traditionally software platforms are associated with products such as operating systems for PCs (e.g. *Windows*), for game consoles (e.g. *Sony PlayStation*), or for mobile devices (e.g. *Symbian*). They run on individually owned, dedicated, computing devices. The development of networking technologies has caused the emergence of software platforms that run on web servers and are distributed over the network; they are called web-based or Internet platforms. Their capabilities are accessed programmatically thanks to distributed computing technologies through APIs, using standards such as W3C Web Services.

2.1 Cloud Computing

Platforms distributed over the network have evolved rapidly in recent years. The emergence of the programmable web in the early 2000s was followed by the recent interest and growing popularity of cloud computing. Nevertheless, this concept is still vague and is often used purely for marketing purposes. Recently, however, the research into cloud computing has increased, aimed especially to characterize the phenomenon.

Vaquero et al. (2009) define cloud computing as follows:

“Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs.”

It is important to grasp three key features of cloud computing based on this definition, namely:

- **Scalability:** Infrastructure capabilities are elastic; the capacity can be dynamically scaled up or down.
- **Pay-per-use model:** The cloud computing business model allows buyers to pay only for the resources they need. They do not incur any expenditure costs, only variable costs.
- **Virtualisation:** Virtualisation hides the physical characteristics of the underlying layer. In cloud computing hardware resources are virtualised, as well as some software components. Virtualisation enables sharing and securing isolated resources.

Depending on the capabilities provided there are the following types of cloud computing:

Infrastructure as a Service (IaaS) is aimed at outsourcing computing resources, such as storage and processing capacity. Virtualisation allows for elastic allocation and provision of those resources over the network. In principle IaaS provides a generic runtime environment, not restricting the type of application. An example of an IaaS-type platform is *Amazon Web Services (AWS)*.

Platform as a Service (PaaS) moves the abstraction a layer higher. Computing resources are not directly exposed, but are hidden behind higher-level APIs. PaaS provides software systems instead of virtualised infrastructure. PaaS are easier to use by developers, but are less flexible than IaaS. Usually PaaS-type platforms are supplemented with various development tools. *Google App Engine* is currently one of the most popular PaaS.

Software as a Service (SaaS) are applications delivered to the end user over the Internet. SaaS applications, instead of being run on a desktop computer, are accessed online via a web browser.

Similar classification to this mentioned above, but without the XaaS terms, was given by Armbrust, et al. (2009), who described cloud computing as utility computing and applications delivered over the network, i.e. SaaS. The former are services provided in a pay-per-use model by hardware and systems software data-centres. In utility computing distinctions are made depending on the abstraction (virtualisation) level presented to the developer. At the lowest level of abstraction there are services, such as *Amazon EC2*, in which a hardware virtual machine is controlled by the developer. This option provides the greatest flexibility (as any kind of application is supported), but it is difficult to scale and to secure platform reliability. At the highest level of abstraction there are application-specific platforms, for example *Google App Engine*. These are suitable for a particular domain; for instance *Google App Engine* is dedicated to web applications.

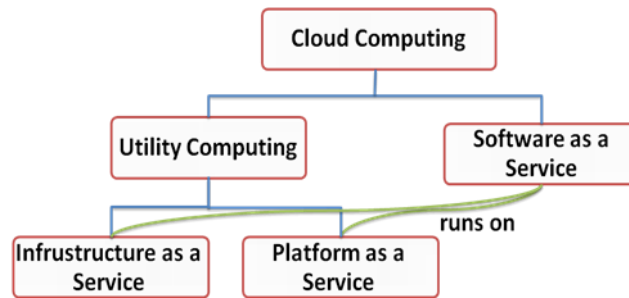


Figure 2. Cloud computing – classification

Utility computing solutions differ depending on the level of abstraction. Higher abstraction-level utility computing platforms can be associated with PaaS, whereas lower level with IaaS. Therefore, PaaS and IaaS are different types of platforms providing utility computing (see Figure 2). All classes of utility computing are used to run SaaS applications. Therefore, SaaS providers are utility computing users. They can sell software without provisioning a data-centre, and without any upfront investment, as computing resources are offered on a pay-per-use basis. Moreover, utility computing permits scaling up and down resources, accordingly to demand.

2.2 SaaS Platforms

Software platforms are used to make applications, SaaS platforms are, thus, systems that enable the development and delivery of SaaS applications. We can distinguish SaaS platforms with low level APIs which configure virtualised hardware (IaaS) and platforms with higher level APIs (PaaS), which are often application-specific. Apart from providing utility computing, SaaS platforms can also support development tools, access to common applications (e.g. billing, authentication) and APIs to other SaaS applications.

SaaS applications can also be delivered by on-premises systems, which are not considered to be cloud computing (Armbrust, et al. 2009). The internal data centres of an organisation can be used to run SaaS. This paper focuses on publicly-available platforms and the case of hosting SaaS on private infrastructures is beyond its scope.

Complementors and users make long-term platform-specific investments. Once a company decides either to use or to co-develop a platform, it acquires a stake in the platform standards (Greenstein 1998). A platform needs a critical mass of complementary products, co-developers and users; positive feedback forces favour the adoption of platform standards. SaaS platform providers try to establish their position as the platform leader, because of the positive feedbacks that accompany it. The more complementors work with the platform, the more users will be attracted to it and vice versa. Next sections present case studies of companies that are seeking to become platform leaders and to drive innovation in the SaaS industry.

3 Case Study 1 – Amazon

Amazon is best known for e-commerce activities. Expansion in this sector required building extensive IT infrastructure. In 2002 the company realised that a new revenue opportunity could be seized by exposing its infrastructure capabilities to external developers. This was the rationale for offering utility computing with the *Amazon Web Services (AWS)* platform. It provides mainly low-level access to computing resources and is an example of an IaaS-type platform.

There are various domains in which users can take advantage of *AWS*. The most common ones are application hosting and backup or general-purpose storage. Researchers can use *AWS* for running high performance computations that require a large amount of data processing. Potential *AWS* usage also includes web hosting, particularly e-commerce, in which implementation can be facilitated thanks to the payment solution from *Amazon*.

3.1 Amazon Web Services – Platform Technology Strategy

AWS is a set of services that are accessed over the Internet, and paid for according to usage (e.g. for CPU hours, data storage or data transfer). The major ones, providing infrastructure, are *Elastic Compute Cloud (EC2)* and *Simple Storage Service (S3)*. *EC2*, in principle, allows renting of computing resources. Technically, users set up virtual server instances, called *Amazon Machine Images*, and define their own configuration. Those virtual machines can be scaled up and down dynamically by adjusting CPU and storage capacity. The server instance operating system can be chosen from *Linux*, *Windows Server* and *Solaris*. In practice, the *EC2* environment allows to run applications written in any programming language. It is up to the user how to specify the server instance and what kind of software packages to include. In that sense *Amazon* provides the most open and flexible platform amongst those analysed in this paper. *AWS*, being flexible, also requires more technical expertise in system maintenance. Some features automated in platforms from *Google* and *Salesforce.com* need to be manually configured when used in *AWS*.

EC2 works in conjunction with the *S3* service, which is used for storing objects and accessing them over the Internet. It is exposed through standard protocol interfaces *REST (REpresentational State Transfer)* and *SOAP (Simple Object Access Protocol)* and therefore can easily work with external development tools. *Amazon* also offers a database service – *SimpleDB*, which provides richer functionality than *S3*, including data indexing and querying.

Moreover, there are application-level services in the *AWS*, such as billing and payments. *Amazon Flexible Payments Service (FPS)* allows third parties to take advantage of the same payment solution as that used on *Amazon's* retail websites. Existing *Amazon* customers can use their login credentials, which are associated with their shipping address and credit cards, for paying on external websites. Hence, co-developers can leverage the potential of *Amazon's* customer base. Another billing service is *Amazon DevPay*, which permits collection of payments for on-demand applications that run on top of *AWS*. It deals with subscriptions management, billing and usage metering. *Amazon.com* accounts can be reused in *DevPay*, as well. Through these services, *Amazon* tries to differentiate its platform offering.

3.2 Amazon Web Services – Platform Business Strategy

Having expertise in efficient infrastructure provision the company set its boundary so that it offers only very low level access to the computing resources. Higher level services, facilitating configuration, management and monitoring are instead provided by complementors. As for example by *Rightscale*, which offers tools for automation, control and portability for applications deployed on *EC2*. *Amazon* needs to build trust among such complementors, as they may fear that it would incorporate their products into the platform. To do this the company announces new features before they are released and discusses the roadmap with complementing firms. As the *Amazon* CTO, Werner Vogels, said: “*We wanted to make sure people had a look at our roadmap, our goal is to be very respectful and recognize the value of the ecosystem*”.

Providers of tools are only one type of component suppliers for *Amazon*, the most important are SaaS application developers, as they create value for the end users. The technology is new, so *Amazon* tries to show its advantages by marketing and evangelising start-ups that succeeded in building services running on *AWS*. Activities in this area include the *Co-Marketing* program and partnership arrangements allowing to use the *AWS* trademark, as well as organising conferences for investors and entrepreneurs. Two companies that have been advertised as successfully taking advantage of *AWS* are *Animoto* and *SmugMug*. *Animoto* is used to create slide-show type videos made from pictures. *SmugMug* is a photo-sharing service. Both require extensive computation and storage resources and their user base, which was reliably supported by the *Amazon* platform. Such real-life examples help to convince others to adopt the technology.

4 Case Study 2 - Google

Google needs very powerful data-centres to support its vast number of users. The amount of *Google* servers is estimated at 450,000. Rapid growth in the early twenty-first century, made *Google* build a scalable software infrastructure. Recent areas of expansion for *Google* are SaaS and utility computing. An example of the former is *Google Apps*, a suite of business productivity applications delivered as a service, accessible through a web browser. Utility computing is offered from May 2008 with the *Google App Engine*. This service may be regarded as an example of a PaaS-type platform. It provides a runtime environment and development tools for building web applications.

4.1 Google App Engine – Platform Technology Strategy

The *Google App Engine* offering consists of processing capacity, storage and networking. There are two supported runtime environments: *Python* and *Java*, which are well established languages among the developer community. *App Engine* is dedicated to web applications working in the request-response manner (a program is executed after receiving a request and returns a response). However, it is also possible to run scheduled programs, using the *Cron* service. Application data are stored in the *BigTable* database, a proprietary *Google* solution, which is different from most popular relational databases. In addition, *App Engine* supports *Google Accounts*, a user-authentication mechanism. Users can sign in to third party applications with their *Google* registration. *Google* also provides developers with a set of tools, known as an SDK (Software Development Kit), which is open source.

In theory, an application written for *App Engine* can be moved to another environment. However, there are some features that make the portability of applications difficult. For example, data storage is proprietary; *BigTable* is not a relational database, unlike most of the databases available nowadays. Therefore, if a service were to be moved from *App Engine* to another environment, data migration would cause problems. This could be overcome with tools to extract data, which have not yet been provided. However, *Google* announced in April, 2009 that it will include them in a future edition. For now, the only solution for third parties in data migration is to make a dedicated application for this.

Another potential platform lock-in is related to *Google Accounts*. Once a service starts using *Google's* authentication mechanism, it needs to run on the *App Engine* platform, as this API would not be accessible from outside. The use of *Google Accounts* by an external application is valuable for *Google*, because it creates an opportunity for more registrations on the system and hence can increase the adoption of *Google's* applications. Third parties also benefit, as their customers can start using the service immediately, without registering. However, they also need to consider the very close ties with *Google* that are created in such situation.

Infrastructure scalability and ease of use are the rationale for the proprietary nature of some of the standards. *App Engine* sets the abstraction level high and takes care of seamless scaling and load balancing. Thus, third parties are released from maintenance issues. This comes at the cost of openness and a potential platform lock-in. The lock-in risk is mostly related to the proprietary data store, and it would be reduced if *Google* made *BigTable* open source. However, decisions taken so far indicate that *Google* prefers to retain its IP in this area. On the other hand *Google* was willing to open the SDK code, which effectively leverages the work of others in improving the development tools.

4.2 Google App Engine – Platform Business Strategy

Google creates some applications for *App Engine* itself to prove the technology. *App Engine* plays an important role in *Google's* internal development process. It is used in the *Google Labs* project, which aims to demonstrate innovative services not necessarily ready to enter the market. One of the largest applications (in terms of usage) built on *App Engine* is *Google Moderator*, a tool for management of Q&A sessions. The system has been shown to work under heavy load conditions, while facilitating President Barack Obama's Town Hall meetings. It allows citizens to ask questions and rank them by casting votes.

Creation of some platform applications is related to the scope of the firm, which is one of the business strategy challenges. However, the whole SaaS ecosystem needs to be considered for the overall view on *Google's* platform strategy. The key SaaS product is *Google Apps* (the suite of productivity applications). Individual customers get this free, but business customers, who need extended features, pay for it. *Google* needs to have a superior offering in terms of functionality to compete for the latter with traditional office software suppliers, such as *Microsoft*. Enterprises often require customisation and extensions to the basic applications. Therefore, *Google* exposes *Apps* APIs, so that third parties can provide complementary products.

There are two types of complementors to *Apps* applications: System Integrators (SIs) and Independent Software Vendors (ISVs). SIs configure, customise and integrate with internal applications. *Google Apps* ISVs develop new products built on top of *Google's* infrastructure. Both of these groups use *Apps* API to gain access to the application data. Although programs using *Apps* API can run in any environment, *Google* promotes *App Engine* for that purpose. Incentives for its use are common compliance and authorisation rules (a single sign-on mechanism). Moreover, *Google* provides ISVs with a sales channel to the existing *Apps* customers through *Solution Marketplace*. It permits third parties to list and sell applications that build on *Google* products.

5 Case Study 3 - *Salesforce.com*

Salesforce.com, founded in 1999, was one of the first companies to enter the SaaS market. It is entirely focused on on-demand applications and services, its main product is a Customer Relationship Management system (CRM). *Salesforce.com* commenced selling the CRM in 2000. It was recognised later that the system needed complementary applications and services. Hence, in 2007, *Force.com* was launched, providing utility computing and supporting third parties with development tools and a runtime environment for applications integrated with the *Salesforce.com* CRM. In the SaaS ecosystem, *Salesforce.com* delivers the core application (CRM); customisation and development of complements is left to users and complementors.

5.1 *Force.com* – Platform Technology Strategy

The abstraction level in the *Force.com* platform is higher than for both AWS and *Google App Engine*. Customisation can be performed by non-IT specialists with a graphical tool, *Force.com Builder*, which permits modification of the data model, business logic and application processes. This is enabled technologically by metadata associated with the application configuration specific to a particular customer. Metadata describe the behaviour of a virtual instance of the application.

More advanced customisation and development of new applications for *Force.com* is done using the Apex programming language. Apex is *Salesforce.com*'s proprietary language, similar to Java. It is dedicated to on-demand applications and its scope is limited to the *Force.com* framework. The *Force.com* environment only supports programs written in Apex. Conversely Apex applications cannot run in other environments, so platform lock-in is clearly associated with *Force.com*. Developers wanting to use other languages need to run them externally and access the application data through Web Services APIs.

According to *Salesforce.com*, the proprietary language needed to be applied due to the specific nature of the *Force.com* applications. They are multi-tenant, meaning that all customers share a single physical instance and version of the application. This results in the potential risk, that if a piece of code behaves abnormally (or maliciously) monopolising shared resources, it may affect other virtual instances and the whole platform. Therefore, security and control measures are required. While other programming languages allow unconstrained usage of resources, Apex includes an additional abstraction level that monitors resource consumption and can limit it. This virtualisation is introduced to separate the platform from applications running on it.

There are many SaaS applications that are multi-tenant, such as *Google Apps*, mentioned earlier. However, there are few platforms for developers to easily create third party multi-tenant applications. Low-level SaaS platforms, such as *Amazon Web Services*, do not provide application-level APIs, so creating a multi-tenant application would require the developers' efforts to program that. In the case of the *Google App Engine* multi-tenancy is yet not supported, although *Google* is working on this feature. *Salesforce.com* has a unique multi-tenancy offering within the big vendors; the technology, enabling multi-tenancy, is well protected with patents.

Apex language hides from developers underlying multi-tenancy and provides them with building blocks of business applications. High-level abstraction in *Force.com* facilitates development, but also limits flexibility. *Salesforce.com* provides third parties with the possibility of using other programming languages with APIs, but such applications are no longer hosted on the *Force.com* platform. *SalesForce.com* has recently created a developers' tool for accessing its API from *Google App Engine* and AWS. It makes it possible for third parties to deliver applications developed with languages other than Apex.

5.2 *Force.com* – Platform Business Strategy

Integration with competitors' platforms, from *Google* and *Amazon*, is one of *Salesforce.com*'s platform business strategy choices. This indicates how the company sets its scope. Namely, it limits the offering to supporting Apex applications, the area of its expertise. It is aware that some third parties would require other environments, in which *Google* and *Amazon* have better capabilities. *Salesforce.com* focuses on providing a high-level abstraction platform dedicated to business applications, in which it possesses its core competencies and has competitive advantage. This collaboration creates a wider ecosystem, in which *Google* and *Amazon* provide platforms for products complementary to *Salesforce.com*. The rationale for collaboration is such that it broadens the market for all the parties, by extending the product features.

Another example of a business strategy activity is *Force.com*'s *ISV Partner Program*, which tries to encourage complementors to develop applications for the platform. In addition to technical expertise and tools, this provides third parties with a sales channel to existing *Salesforce.com* customers. Third party applications can be sold to existing *Salesforce.com* customers or to new ones. In the latter case, ISVs incur the additional cost of the embedded license fee, which is US\$15 per month per user.

External applications that are built for, and hosted on *Force.com* can be listed and sold through the *AppExchange* website. Listing on *AppExchange* is free only for native applications, i.e. applications that are entirely built using *Force.com Builder*. *AppExchange*, used for more complex commercial applications, that can utilise external services, requires an annual fee of US\$5000 per partner. *Salesforce.com* facilitates ISVs communicate with end users through the *IdeaExchange* website. It allows suggestions from clients to be obtained, together with feedback to applications proposed by ISVs.

6 The SaaS Platform Ecosystem

The general platform ecosystem, outlined in section 1.2, has been applied specifically to SaaS. Figure 3 depicts the SaaS ecosystem with its three major parties.

- **The platform provider** is, at the same time, the sponsor for all the platforms in the case studies, and for most existing SaaS platforms. This is because sponsors do not distribute platform instances and do not allow other parties to provide the platform. There is a significant difference between traditional software platforms (e.g. *Windows OS*) and SaaS platforms. Namely, the former require a platform copy to run on a dedicated device, while SaaS platforms run on web servers and are accessible over the Internet. Hence, traditional software sponsors can charge users for their platforms (e.g. *Microsoft* for *Windows*). On the other hand, SaaS end users physically connect with the platform through the network, but do not pay for this access, only for applications.
- **Component suppliers** in the SaaS ecosystem provide SaaS applications. They buy utility computing services from the platform provider. They use the platform environment, tools and common applications to build, sell, run and maintain SaaS applications. Their role is crucial in the innovation process, since most of the software innovation is in the applications. Component suppliers can also serve each other, as is the case for complementary tools providers for *AWS*. For instance, *Rightscale*'s products assist in managing low-level *AWS* platform. Another example of a relationship between component suppliers is when a complementary product exposes APIs to be used by other parties.
- **End users** are interested in the working system as a whole. They are less concerned where the application runs¹ and more in the value that they derive from it. End users purchase SaaS applications from the component suppliers. However, platform firms can also provide them with applications, as is the case for *Salesforce.com*'s CRM and *Google Apps*.

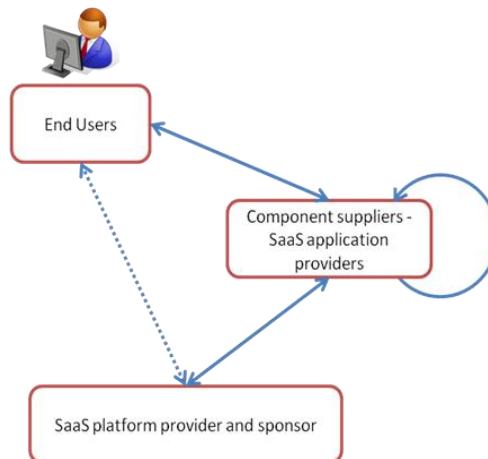


Figure 3. SaaS platform ecosystem

7 The Emergence of SaaS Platform Leaders

According to Eisenmann et al. (2006) platform leadership can emerge under specific conditions. Not every industry is suitable to be served by a single platform, a market is likely to have a leader when all of the following conditions are satisfied:

- There are positive and strong network effects.
- There is little room for differentiation.
- It is not easy for users or complementors to own more than one platform and to switch between them.

Network Effects: *Force.com* exhibits network effects between the end users and the complementors. The more users who use *Salesforce.com* and buy the *Force.com* license, the more complementors will be willing to develop applications. More offerings on *AppExchange* will make the whole platform more valuable to the end users. For the *Google App Engine*, network effects are weaker, but they are present in the whole *Google* SaaS ecosystem with *Apps*. More users of *Apps* will attract more SIs to build expertise in customisation and more ISVs to create complementary applications, and vice versa. In the case of *AWS*, there are no network effects between the end users and the complementors, because there is no core application, such as *Salesforce.com* CRM or *Google Apps*. However, as for the two other platforms, there are network effects within the complementors' group. Developers who embrace a particular standard promote this among the whole community (Greenstein 1998). Also the more complementary development and management tools and services that are provided, the more application developers will be willing to use the platform.

¹ Actually, end users may care about the servers their applications are running on, due to security considerations. This issue is discussed in section 9.2.

Differentiation: the condition concerning little room for differentiation is not satisfied for high-abstraction level platforms like PaaS. In fact they can be distinguished, *Google App Engine* offers special features for web applications and *Force.com* for enterprise software. There are also opportunities to work on improvements in tools facilitating development processes. As, for example, *Force.com* was able to differentiate offering multi-tenancy and the metadata-driven configuration. On the other hand, low-level abstraction platforms are rather commoditised, because of the nature of the computing resources, such as CPU and storage (O'Reilly 2008).

Cost of Owning More Than One Platform: the cost of using SaaS applications on different platforms is, in general, low for the end users. They do not need to install any software, as these applications run through a web browser. For the component suppliers those costs are higher, because using a particular platform requires investment in building capabilities and technical expertise.

Table 2. Platform leadership conditions applied to the platforms studied

	Network effects	Differentiation	Costs of owning more than one platform
<i>Force.com</i>	Strong	Important	Moderate for end users High for complementors
<i>Google App Engine</i>	Moderate	Moderately important	Low for end users High for complementors
<i>Amazon Web Services</i>	Weak to Moderate	Not important	Very low for end users High for complementors

As Table 2 indicates, it is unlikely that there will be a clear winner in the PaaS type platforms. The main reason for this is that they can be differentiated. On the other hand, in the case of the low-level platforms (IaaS), such as AWS, the network effects are not strong enough for a platform leader to emerge. However, as the industry evolves all the conditions for this may be satisfied in the future. Platforms may become more similar to each other, losing distinguished features. Infrastructure providers may be able to harness network effects and raise switching costs as well. Therefore, complementors face a high level of uncertainty when selecting SaaS platforms. In particular, there is a risk that a platform will be marginalised, or will even go out of business. So complementors should be cautious not to become locked-in to any one platform.

8 How SaaS Platforms Facilitate Complementors in Innovating

Having considered a wide spectrum of SaaS platform offerings and the ecosystem in which they operate, this section identifies the platform features that can facilitate complementors in innovating. There are various reasons why component suppliers would choose a SaaS platform for application development and hosting, as opposed to maintaining their own one. The main advantages are related to lowering financial and technical barriers to entry (Andreessen 2007). Moreover, case studies show that SaaS platforms help in reaching the market. These factors allow complementors to focus on innovating in applications and also decrease the risk of failure.

8.1 Providing Scalable Infrastructure On a Pay-per-use Basis

The breakthrough in the industry delivered by SaaS platforms is eliminating the infrastructure expenditure costs for component suppliers, particularly for start-ups. Providing SaaS applications with their own infrastructure would require up-front investment and, because of the uncertainty related to a new product, it would be difficult to scale that infrastructure appropriately. While a start-up product adoption is not easy to define, a pay-per-use model has a significant advantage over maintaining an in-house platform.

High up-front costs make an investment risky; if a product is not successful the losses are substantial. When infrastructure is delivered as service, companies pay only for what they actually use, so, in the case of a failure, running the application would cost nothing. SaaS platforms lower barriers to entry and mitigate the consequences of innovation failure, which can make component suppliers more willing to take risks in innovating in SaaS applications.

Moreover SaaS platform providers take advantage of economies of scale. It is estimated that very large data centres can have price advantages of a factor of 3 to 7 over those of medium size² (Armbrust, et al. 2009). This is reinforced by the growing consolidation of data centres. The cost of infrastructure consists of server costs, cooling and power distribution systems, and the power itself. Currently server costs dominate, but as computer prices fall and power becomes more expensive, the latter may become dominant. In such situation it is beneficial to locate large data-centres to pay less for electricity; *Google*, for instance, has facilities in Quincy, Washington. SaaS platform firms also achieve high server utilisation. All of these factors allow SaaS platform providers to be able to deliver cheaper computing than on-premises systems. Cheaper computing resources available for small companies enable innovation in data and computationally-intensive applications, as the example of *SmugMug* (Web 2.0 photo-sharing service) shows.

² In this comparison a medium data-centre is assumed to have around 1,000 servers; a very large one, around 50,000 servers.

8.2 Providing Software Components, Common Applications, and Tools

The software industry is characterised by high costs in the development process. For this reason, the first unit is very expensive in traditional software, but the marginal cost of producing subsequent units is very low. One can argue that, despite the elimination of infrastructure capital expenses, SaaS providers still have high fixed costs of development. However, SaaS platforms facilitate component suppliers in building applications as well, by reducing time and lowering the technical expertise required to do this.

SaaS platforms hide common software capabilities behind APIs. Developers can simply re-use them and focus on the application logic itself. This is enabled by virtualisation, one of the main cloud computing features. The *Force.com* platform has the highest level of virtualisation. It therefore makes development the easiest; possible even for non-IT specialists, without coding. Regarding more advanced capabilities, multi-tenancy available on the platform releases complementors from implementing it themselves. Balakrishna Narasimhan from *Appirio* (a *Salesforce.com*'s partnering company) acknowledged the facilitating role of the *Force.com* platform in reducing application development time: “*Development savings depends on a number of factors, but in our experience ranges from 2X to 10X faster than traditional development.*”

Google App Engine, on the other hand, provides ready-to-use components for web applications. *Amazon* offers low-level access platforms, but has services that facilitate development as well, such as *SimpleDB*, for data storing. The trade-off for high abstraction is limited flexibility, causing the application spectrum to be constrained to a single domain; for *Salesforce* it is enterprise software, for *Google* web applications; only *Amazon* supports various types of applications.

Another important facilitator in the development process are the common applications, called system services. All three of the studied platform firms allow third parties to use their solutions for payments and billing. *Google* also exposes the *Accounts* service for authentication and authorisation purposes. Third parties do not need to implement these functionalities themselves. They can also take advantage of the platform firm's brand. As indicated by Chris MacAskill, a co-founder of *SmugMug*, the service adoption rate after introducing the payment solution from *Amazon* was surprising. He explained that: “*My guess is their brand is so strong and it saves people from having to worry about storing their billing info at yet another site.*”

Platform firms enable the implementation of their technologies with SDKs consisting of programming tools, sample code and documentation. They also set up developer forums, from which complementors may seek advice. All these elements are present in the platforms studied. SDKs are common throughout the whole computer industry and are not specific to SaaS. Nevertheless, they are important in facilitating innovation.

8.3 Exposing Application Data

Google and *Salesforce.com* offer platform applications themselves, namely *Google Apps* and *Salesforce.com*'s CRM. Both products are rich in user data and exposing it through interfaces enables re-usability. The *Force.com* Web Services API allows external applications to operate on an organisation's data (of course access to these data is granted after successful authentication and authorisation). *Google Apps* APIs makes information available, such as users' calendar events, documents and contacts. An example of an application using the calendar API is *TimeBridge's Personal Scheduling Assistant*, which integrates calendar systems from various organisations. It retrieves users' events from the *Google Calendar* in order to help them schedule meetings. Not only do platform firms expose their application data but component suppliers sometimes do this as well. For instance, the *SmugMug* service exposes API, making available stored pictures and albums to the outside world to enable new applications on top of it.

Exposing application data to third parties facilitates innovation, because it allows developers to find new ways for combining these data. Moreover, component suppliers are freed from building their own data sets, which assists them with launching a software business.

8.4 Lowering Transaction Costs

Platforms reduce transaction costs that different parties in the ecosystem would need to incur to get together (Evans, et al. 2006). SaaS platforms make it easier for component suppliers to reach their customers, and for end users to search for applications they need. *Google* and *Salesforce.com* each provide a single marketplace for their complementors: *Solutions Marketplace* and *AppExchange*. Users can easily find and purchase applications; on the other hand, component suppliers get an effective sales channel. Moreover, *Salesforce.com*'s *IdeaExchange* (the website for customer application suggestions and feedbacks) facilitates complementors in doing the market research. *Amazon* offers a less featured website, *Solution Catalog*, permitting developers to list their products.

Assisting in marketing and evangelising selected complementors also facilitates end users to choose trusted and reliable application providers. Thanks to this, small companies, without a good brand, may gain credibility. The examples of *SmugMug* and *Animoto*, evangelised by *Amazon*, and *Appirio*, promoted by *Salesforce.com*, are relevant here. Better opportunities in reaching the market increase incentives for start-ups, without established reputations and existing customer base, to innovate.

9 SaaS Platforms' Barriers to Innovation

Despite the advantages of using SaaS platforms, component suppliers need to consider threats related to using them. Platform firms facilitate innovation, but at the same time they need to care for their own interests. It is important that third parties are aware of the drawbacks and consciously make decisions to take the best usage of the platform technologies.

9.1 Platform Lock-in

Platform firms will have more power if switching costs for complementors are high. However, too much control over the ecosystem, causes developers to worry about price increase. To avoid that situation component suppliers need to consider whether a particular platform allows to move their applications. There are two levels of portability:

- Application portability – whether application code can be moved to another environment
- Data portability – whether application data can be migrated to another server

Table 3. The levels of lock-in for the studied platforms

Platform	Application portability	Data portability	Platform lock-in
<i>Force.com</i>	Proprietary Apex language Not portable	Proprietary database Not portable	High
<i>Google App Engine</i>	Python and Java languages Portable (with the exception of API specific code)	Proprietary database BigTable Not portable	Moderate
<i>Amazon Web Services</i>	Any programming language Portable (with the exception of API specific code)	Running standard database server (e.g. MySQL) - Portable S3, SimpleDB - Not portable	Low

The levels of lock-in for the studied platforms are presented in Table 3. The most proprietary one is *Force.com*; it is not possible to move Apex applications, nor data, to another platform, unless the code is rewritten. *Google App Engine* and *Amazon Web Services* permit developers to run standard programming language applications, which can theoretically be ported out. This is, however, more difficult when those applications use platform-specific APIs. In fact platform APIs comply with standards in terms of the protocol (e.g. SOAP), however, there are no industry standards, as to how they define operations related to SaaS platform utilisation. There are no standard APIs either, for data portability, so the two platforms have their own proprietary storage interfaces. Nevertheless, *Amazon* permits any database server to run on EC2, which guarantees portability. Summarising, moving applications from *App Engine* and *AWS* is possible but, since it depends on the use of specific APIs, it is not always easy.

Adoption obstacles associated with proprietary APIs would be removed if there were industry standards. In March, 2009 there was an Open Cloud Manifesto initiative, led by *IBM*, aimed at commencing standardisation work. However, none of the main SaaS platform providers, *Salesforce.com*, *Amazon* or *Google*, signed it. They explained that they were in favour of inter-operability, but found the manifesto too vague. There were arguments that it was too early to set standards, as the industry was in its infancy. In addition, it is thought that standards would result in price wars, lowering the profits of platform providers.

The web developers' community is very resistant to proprietary solutions. Web technology gurus, such as Richard Stallman and Tim O'Reilly, have warned that controlled SaaS platforms take away developers' freedoms. This puts pressure on platform vendors to ensure that they do not intend to lock developers in. Platform firms need to address such concerns in order not to antagonise the community. *Salesforce.com* convinces that proprietary solutions are necessary to deliver features such as metadata configuration and multi-tenancy. Moreover, complementors have access to *Force.com* APIs from both *Amazon's* and *Google's* platforms, which are far more open. *Google App Engine* is mainly criticised for data lock-in, so the company announced that it was going to introduce an export tool.

There is a platform lock-in risk related to the use of common applications, such as payments and authentication. This facilitates third party entry, as they do not need to implement common functionalities themselves, but they should consider that using platform services can bind them to a particular vendor. Once a company starts to use *Google Accounts* or *Amazon DevPay*, it needs to stay with the platform firm, because these system services would not work, if the application were moved to another platform. There is a potential opportunity for complementors in developing common applications that would work across various platforms. If they offered such solutions, the lock-in problem would be overcome, which could be a source of advantage over the platform-specific services.

Another risk arising from lock-in is when a particular provider will go out of business and discontinue operations. This happened, for example, with *Coghead*, a company offering a PaaS-type platform. In February 2009 it announced a cessation of services and gave two months for its customers to transfer their applications,

which, due to a lack of standards, required rewriting all the programs. Such a situation is more likely for small platform firms than for big players like *Google*, *Amazon* and *Salesforce.com*, which are trusted to have a stable position and, in this regard, have an advantage over SaaS platform start-ups.

9.2 Security and Privacy Issues

Using SaaS platforms inevitably leads to storing customer data outside the company. This raises security and privacy concerns, which need to be addressed by SaaS platform providers. Strong privacy policies, that take into account storing complementor customers’ data, are required.

Despite giving away data to external firms, it is possible to make SaaS platforms as secure as the majority of on-premises systems. One way to achieve this is through encryption. For instance, this enabled a healthcare company, *TC3 Health*, to use *AWS* for an application involving sensitive patient records. The service was designed to be complaint with the U.S. legal regulations. Security and privacy issues limit SaaS innovation in some fields, due to stricter regulations. As noted by Alvin Abraham, who is the CEO of an IT consultancy firm specialising in the financial sector, there are legal constraints in banking that require storage of customer information on premises.

However, security should not only be regarded as a threat, but also as an opportunity for innovation. SaaS platform complementors may specialise in providing additional virtualisation levels for safety measures. One example of such a company is *enStratus*, which delivers management tools to support security.

9.3 Concerns About Reliability

Utility computing is a new way of using technologies, and it is still not mature enough to ensure high reliability. In fact, the most respected SaaS platform companies, *Amazon* and *Google*, have experienced unavailability periods. There were two *Amazon S3* outages in 2008, one lasting for two hours, and the other for eight hours. This affected, among others, the *SmugMug* service, whose users lost access to pictures and videos. *Google App Engine* was partially unavailable in 2008 once for five hours. Those accidents make negative publicity, however, enterprise IT infrastructures are rarely better. In fact, looking at the numbers, *Amazon S3* achieved a yearly availability of 99.89% and *Google* 99.94%.

Amazon offers a Service Level Agreement (SLA), such that when the uptime percentage drops below 99.9% monthly for *S3* and 99.95% for *EC2*, customers pay reduced fees. Neither *Google App Engine* nor *Salesforce.com* include SLAs. Nevertheless, even 99.9% uptime, meaning lack of availability for 40 minutes monthly, is not sufficient for mission critical applications, such as banking services or controlled medical devices. However, it is enough for web applications, such as *SmugMug* or *Animoto*.

A serious concern that component suppliers have is that platform outages are out of their control and they lack information about them. Platform firms recently addressed this issue with websites providing real time information about their systems’ performance. *Amazon* and *Salesforce.com* now show the status of their services on a dashboard. *Google* has a similar website for *Apps*, however for *App Engine* there is only the downtime notification forum. Building reputation and trust by sharing information about system performance can actually be more convincing for developers than guaranteeing SLAs.

10 Summary

Cloud computing offerings from *Amazon*, *Google* and *Salesforce.com* are depicted at three levels in Figure 4. The two bottom levels present SaaS platforms and the top one SaaS applications that are delivered by platform firms.

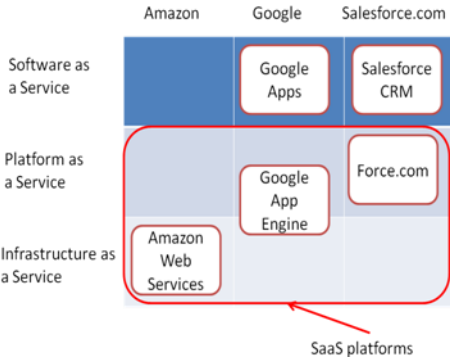


Figure 4. Spectrum of studied cloud computing offerings (adapted from Narasimhan 2009)

This paper covers case studies of SaaS platforms, marked by the big red rectangle in Figure 4: *Amazon Web Services*, *Google App Engine* and *Force.com*. They differ in terms of capabilities, but all of them enable component suppliers to develop and deliver SaaS applications. Since it is possible to differentiate among the PaaS-type platforms (e.g. *Google App Engine* and *Force.com*) and there are not strong enough network effects

for IaaS-type platforms (e.g. AWS), the SaaS platform market may not have clear leaders. Uncertainty, related to the infancy of this industry, requires complementors to select SaaS platforms with caution.

SaaS platform firms make technology and business strategy decisions that can facilitate complementors in innovating. They lower technical and financial barriers to entry. Complementors do not need to incur upfront infrastructure costs or have expertise in hardware maintenance. Computing delivered in the pay-per-use model lowers the risk of innovation failure and improves the complementors' agility, allowing quick responses to business opportunities and market conditions (both good and bad). SaaS platforms facilitate application development by providing ready-to-use software components, abstraction layers and development tools. Platform firms sometimes deliver applications for their platforms as well, e.g. *Salesforce.com*'s CRM system and the *Google Apps* suite. Moreover, they expose application data through APIs, so that the third parties can combine these with their services. In addition, SaaS platforms help complementors reach the market by providing a common marketplace, cooperating in marketing and promoting them.

The main risk that complementors face when choosing a platform is related to the platform lock-in. Portability of applications is more difficult for the higher abstraction-level platforms, which have more advanced features that require proprietary solutions. It is not possible to move *Force.com* applications to another platform, while it is feasible with *Google App Engine*, but requires some effort. The lowest degree of platform lock-in is associated with low-level virtualisation platforms, such as AWS. Security and reliability issues also need to be considered by complementors. In fact, these constrain innovation in mission-critical applications, such as financial services. Nevertheless, the SaaS platform barriers can also be seen as an opportunity for third parties. They can offer platform complements to help application developers overcome the risks, such as common applications that work across different platforms, tools for migrating applications from one platform to another, and software components guaranteeing security.

References

- Andreessen, Marc (2007). *The three kinds of platforms you meet on the Internet*. Blog PMarca.
- Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica and Zaharia, Matei (2009). *Above the Clouds: A Berkeley View of Cloud Computing*. UC Berkeley Reliable Adaptive Systems Laboratory.
- Chesbrough, Henry (2003). *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business School Press.
- Eisenmann, Thomas, Geoffrey Parker, and Marshall Van Alstyne (2006). *Strategies for Two-Sided Markets*. Harvard Business Review, October 2006.
- Eisenmann, Thomas, Geoffrey Parker, and Marshall Van Alstyne (2007). *Network Platforms - Core Concepts*. The MIT Center for Digital Business, June 2007.
- Evans, David S., Andrei Hagiu, and Richard Schmalensee (2006). *Invisible Engine How Software Platforms Drive Innovation and Transform Industries*. The MIT Press.
- Gawer, Annabelle, and Michael A. Cusumano (2002). *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business Press.
- Gawer, Annabelle, and Michael A. Cusumano (2008). *How Companies Become Platform Leaders*. MIT Sloan Management Review, Winter 2008.
- Greenstein, Shane (1998). *Industrial Economies and Strategy: Computing Platforms*. IEEE Micro 18, no.3.
- Kolakowski, Bartłomiej (2009). *Platform Leadership in Software as a Service: How Platforms Facilitate Innovation*. MPhil in Technology Policy course final dissertation supervised by Dr Michael Barrett at Judge Business School, University of Cambridge.
- Narasimhan, Balakrishna (2009). *Cloud Computing Savings - Real or Imaginary?* Appirio CIO Blog.
- O'Reilly, Tim (2008). *Web 2.0 and cloud Computing*. O'Reilly Radar.
- Vaquero, Luis M., Luis Rodero-Merino, Juan Caceres, and Maik Lindner (2009). *A Break in the Clouds: Towards a Cloud Definition*. Telefonica Investigacion y Desarrollo and SAP Research.