Adding Value to the Network: Exploring the Software as a Service and Platform as a Service Models for Mobile Operators

Vânia Gonçalves

IBBT-SMIT, Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussels, Belgium
vania.goncalves@vub.ac.be

Abstract. The environments of software development and software provision are shifting to Web-based platforms supported by Platform/Software as a Service (PaaS/SaaS) models. This paper will make the case that there is equally an opportunity for mobile operators to identify additional sources of revenue by exposing network functionalities through Web-based service platforms. By elaborating on the concepts, benefits and risks of SaaS and PaaS, several factors that should be taken into consideration in applying these models to the telecom world are delineated.

Keywords: Software as a Service, Platform as a Service, Business Models, Web 2.0.

1 Introduction

The Web has evolved from a large group of informational websites to accommodate a whole new range of services and applications encapsulated in the Web 2.0 concept.

Traditional on-premises software solutions in the domains of enterprise resource planning (ERP), customer relationship management (CRM) and storage are now competing with equivalent Web-based versions. An increasing number of companies like Salesforce, NetSuite or Google are offering their software in a web-based environment providing a much more flexible experience in terms of time and location of access. Over the last decade, this new concept, referred to as Software as a Service (SaaS), has gained attention from Independent Software Vendors (ISVs) and general acceptance by end-users. The benefits of this model over the on-premises model will be explained further in the next section.

Therefore, as software provision in several instances has shifted to the Web, so has the software development cycle. Rather than developing in an offline environment and then testing online, a Web-based application can now be developed, tested, deployed and hosted in a single platform. This new approach, called Platform as a Service (PaaS), enables developers to use a Web-native platform to entirely design and deliver applications in the same environment in which they will run. Both SaaS and PaaS offer a template to monetise data services over the Web.

C. Hesselman and C. Giannelli (Eds.): Mobilware 2009 Workshops, LNICST 12, pp. 13–22, 2009.

Telecommunications networks and the Internet have evolved as disjoint worlds with regards to software applications and application development technologies. Mobile operators are now at the crossroads of maintaining 'dumb pipes' on the one hand, and finding other sources of additional revenue on the other hand. As the IT industry is moving into a Web service delivery model, there seems to exist an opportunity for mobile operators to expose network capabilities and combine these with online content and applications, e.g. mobile service platforms [1]. However, operators' attempts to build platforms for software development so far seem to have failed to attract developers' attention, as can be inferred by the small number of available applications. Consequently, few end-users came into contact with these applications and are benefiting from them. The question is thus whether by building upon SaaS and PaaS concepts and lessons learnt from the IT world, mobile operators could create value for end-users as well as developers, and extract revenues from both sides.

Although both SaaS and PaaS definitions are still open to debate and intertwined with the Cloud Computing concept [2] [3], in this paper both concepts are delineated from a business perspective, taking into consideration the current market trends. Secondly, the benefits and risks of both models are analysed in order to better understand the implications for suppliers as well as customers. Finally, mobile operators' experiences in providing a platform for software development are analysed and several factors that should be taken into account for successfully applying the SaaS and PaaS concepts to the mobile world are outlined.

2 Software as a Service

The Software as a Service (SaaS) concept has been presented by ICT market analysts [4] and software industry associations [5] as an improved version of the Application Service Provider (ASP) model, in which providers host and provide access to a software application over a network. This application service is managed centrally and offered in a one-to-many fashion. The SaaS model has evolved to a web-based application interface, which is significantly different from the ASP model, since this model mostly emulated the traditional client-server philosophy. ISVs were then able to shift from delivering on-premise software solutions to deliver a complete software application to end-users over the Web. SaaS is deemed to incorporate other key attributes such as configurability, multi-tenant efficiency and scalability [6].

Although current literature focuses primarily on business-oriented SaaS services, several examples in the market lead to a categorisation that also includes consumer-oriented service offerings. Therefore, two major categories can be identified. Firstly, *business-oriented services* offered to enterprises and organisations of all sizes, but typically focused on SMEs. The offered services are targeted at facilitating business processes, such as accounting, supply-chain management, and customer relationship management (e.g. NetSuite, Salesforce.com, RightNow). Emergent players offer services that are targeted at niche needs that are usually important for enterprises at a given time or for specific projects, such as storage or computing resources (e.g. GoGrid, Amazon S3 and C2, Mosso, NTRglobal). Secondly, *consumer-oriented services* offered to the general public on a wide range of applications like hosting, news feeds, storage, presentations, and so on. (e.g. Blogger, Flickr, Dropbox, SlideRocket).

The following subsections describe current business models as well as the benefits and risks for the main stakeholders.

2.1 Revenue Models

Shifting from offering on-premise software to SaaS obliged ISVs to change the way software was sold. In an on-premise scenario, the customer buys a license to use an application and installs it on its own or controlled hardware. By owning this license the customer perceives unlimited usage of the software. In the SaaS model, instead of owning a lifetime license, the customer pays a fee for software running on a third-party server and loses access when he ceases payment. This fee can be charged as a pre-paid subscription or on a pay-as-you-go basis. In the examples mentioned above, some players employ both revenue models, but contrasting revenue models can also be found amongst competing players. Typically, in the first approach, players offer different subscription fees based on a combination of allocated resources, namely storage, data transfer, session-time, number of users, etc. In the second approach, pay-per-use fees are charged on the basis of actual usage of those resources. But the granularity of pay-per-use fees can be even greater depending on the target end-user – for instance, Amazon Services applies different pay-per-use prices depending on the geographic location where the data is stored and on the data transfers between different geographical zones. Consumer-oriented services are often provided to consumers at no cost, but are supported by advertising or are offered for strategic reasons such as customer lock-in. These free services are often used to up-sell advanced features (e.g. extra storage space) on a subscription or pay-per-use basis.

2.2 Benefits and Risks

The SaaS concept has clearly changed the way software is developed and delivered to the customer. The shift from on-premises to web applications and from one-to-one to one-to-many service provision has impacted on the relationships between ISVs, customers and developers.

ISVs make and sell software to customers and normally deal with software distribution channels and licensing. In the shift to the SaaS model the most important benefits for ISVs include potential economies of scale in both production and distribution costs, more predictable revenues, development of software with lighter operating system and hardware requirements, and shorter sales cycle. The possibilities of spreading the costs of innovative solutions over many customers, lower license compliance management activities, lower need to deal with piracy and maintain expensive distribution channels can potentially lead to economies of scale in the production and distribution processes. In addition, software with lighter operating systems and hardware requirements enables ISVs to shorten the development life cycle, easily provide additional features that customers ask for, rapid updates and support, and ultimately to potentially reduce maintenance costs.

However, in the SaaS model several risks may arise that ISVs are not used to deal with. Moving to the SaaS model initially requires ISVs to develop new skills when planning SaaS offerings, such as management of billing and customer data, usage metering, security, support services and service provisioning. ISVs should also take

up investment in building, and later on, maintaining the IT infrastructure to support SaaS services, or take the approach of partnering with other companies to provide those parts of the system (e.g. DropBox uses Amazon S3 storage services). In this latter case, ISVs now need to manage a network of suppliers that did not exist in the on-premises model. Moreover, with the shift to one-to-many service provision the risk of performance and scalability issues is higher, as many users are using an application running in the same server.

The SaaS concept seems very attractive for business customers such as startups and SMEs which do not see IT-related activities as a core competence and do not have the expertise to develop and maintain applications inside the company. It may enable them to have access to expensive commercial software with little initial investment and low monthly costs, something they could not afford otherwise. Therefore, SaaS offers substantial opportunities to reduce the total cost of ownership of IT resources, alleviating companies from running applications in-house and committing to set-up an IT department to administer software and all related activities involving hardware maintenance, backups, security and user support. For companies with very specific software needs, usually very complex and expensive software used in particular phases of a product development cycle, SaaS can significantly alleviate software costs, as companies would be able to subscribe to services for short periods of time. SaaS providers may also offer customers the possibility to share data with other online or on-premises applications with little effort.

Similarly, the retail customer benefits from access to a wider range of software applications, generally at affordable prices or for free and with little effort of configuration or customization.

For both types of customers, SaaS has the potential to provide a stable, reliable and flexible experience, with access to software regardless of time and location. On the other hand, customers may incur very important risks in exposing or losing business-critical data to platform owners and potentially losing control of data exchanges to third-parties. Likewise, platform owners may lock-in customers by not providing the tools to export (and import) data and therefore making it difficult to switch to other providers. An additional issue significantly different from on-premises software is the interruption of the service due to lack of payment. Consequently, it might also implicate failure to recover stored data.

3 Platform as a Service

The same way the SaaS model is very attractive for companies that cannot afford or do not want to have the burden of investing in hardware and in licensing software, the Platform as a Service (PaaS) model is a novel approach for software suppliers that want to focus primarily on the development cycle and monetisation of new applications, thus bypassing the investment and maintenance of the underlying infrastructure. PaaS platforms facilitate the entire software life cycle by offering the underlying services for application design, development, testing, deployment and hosting [7]. These services enable developers to build on the functionalities of an existing Web platform or SaaS (e.g. MySpace Developer Platform, Force.com) or develop new Web applications (e.g. Google App Engine, Bungee Connect). Although

Amazon Web Services are relevant components for particular stages of the software life cycle, they do not embrace the integral notion of PaaS.

PaaS instantiates the concept of 'Level 3 platform' [8], given the fact that it provides the "runtime environment" to run external applications inside the platform itself. It is likely that PaaS also encompasses Level 1 platform functionality – an access API to allow access to data and services running on the platform – and Level 2 platform functionality – a plug-in API to allow to inject functionality into the platform. For instance, Google App Engine provides a set of APIs to access user-specific data stored in the platform, as well as Google services, such as Calendar events and GMail contacts. In the case of MySpace Developer Platform, applications can run inside or outside the platform, which in the latter case they do via a plug-in API. On the other hand, although Facebook Platform provides the tools for the development and testing of applications, these need to be deployed and hosted outside the platform by the developer, and therefore not constituting a complete PaaS platform according to the given definition.

According to Mitchell [7], a PaaS incorporates six key attributes: integrated environment for development, testing, deployment, hosting and maintenance; delivers a user experience without compromise; built-in scalability, reliability and security; built-in integration with web-services and databases; supports collaboration among developers; and finally, supports deep application instrumentation of application and user activity. Similarly to the previous section, an analysis of the current business models, benefits and risks will follow.

3.1 Revenue Models

Since the software life cycle has many stages, PaaS providers are building their revenue models based on the access to resources in those stages. Some PaaS providers charge a subscription for the access to the development and testing functionalities, while others only charge on the actual time end-users spend interacting with the application, thus after the application is deployed in the market. For instance, the cost for hosting an application with Bungee Connect (currently in beta) is determined by the amount of time users spend interacting with each page of the application. Google App Engine is currently free, but in a future release charging will be based on what the application consumes, up to a budget defined by the application owner [9]. The developer will be able to control daily expenditure by means of adjustable resources, like amount of stored data or incoming and outgoing network bandwidth used.

3.2 Benefits and Risks

In the PaaS model, the most important value element for developers is quite straightforward: develop cloud solutions without having to maintain three environments. In the on-premises software development model, developers work in a development environment, then use a test environment to test it and then move to a third environment for production, which is the only one viewed by the user. This way, a potential faster time-to-market may be expected as developers can test and deploy for production in the same environment. It is also worth highlighting that since the platform owner provides the environments in which the entire software cycle occurs,

developers may considerably reduce provisioning and management of their own IT infrastructure (e.g. servers, storage, databases and so on). Currently, developers may try out different PaaS platforms at no cost for the developing and testing phases, enabling them to assess the functionalities of that platform and whether it facilitates integration with other online services. In this case, developers may consider as main costs the runtime fees based on the actual computing resources used by the applications or the number of users using them. Some PaaS platforms already enclose the same feeling of a developer community, by giving developers the means to communicate, share knowledge and reuse other developers' code.

PaaS platforms can be more or less open and, in the latter case, lock-in developers. Some PaaS platforms are based on standard programming languages like WyaWorks (based on Java), on a combination of standard programming languages and proprietary add-ons like Google App Engine (based on Python), or on proprietary programming languages like Force.com (based on Apex). Therefore, developers may be confronted with a long learning curve to master proprietary programming languages and APIs, thus putting more effort in developing an application in comparison with a development environment of an open platform. A closed PaaS platform may create lock-in by preventing developers to port an application to another platform, which may only be circumvented by building the application from scratch in the new platform. Closed platforms may also prevent developers from integrating data from third-party platforms or services to create, for instance, mashups and therefore hinder innovation. The whole concept of providing developers with all development tools may also create a fundamental gap with traditional development environments – the developer is limited to use the APIs available in the platform, preventing him to give wings to new ideas by extending or incorporating new functionalities in APIs.

Like in SaaS models, platform owners can make platforms available with light operating systems and hardware requirements, both for the development phase and for the runtime. Similarly, the platform owner would avoid dealing with license compliance and piracy issues. Still, the most important value element for platform owners is earning revenues from applications hosted in the platform, which might well be the next killer applications.

Nevertheless, the success of the platform will heavily depend on the platform owner's strategy and the right balance between open and closed, proprietary and non-proprietary, free and paid, and so on, making it captivating enough for developers. It is likely that developers are not keen on relearning everything, that they want to try out how the platform works and responds, are typically eager to use tools to share ideas and knowledge, and ultimately want the option to put the application on the market with little effort. But once the platform is adopted, a platform owner has to deal with a whole set of issues related to performance, scalability, storage and security. The platform owner cannot control the quality and security of the code that will run within the platform, but has to create the technical mechanisms to anticipate the consequences of such problems. Additionally, many of the analysed platforms detail in a terms of service agreement the possible consequences in case of overuse of computing resources or faulty applications that disrupt the platform.

One can argue that in this model the developer can devote more time to the creative process and therefore a whole new range of innovative applications will emerge. The

end-user will benefit from applications that fully realise the Web 2.0 principles making more use of integration and aggregation of data provided by other applications or systems. The enormous offer of this type of applications will likely decrease the price for consumers. The end-user might however incur the risk of paying for a service that is periodically unavailable. Considering Google App Engine's business model, developers are able to set quotas for computing resources in order to prevent the cost of the application from exceeding developer's budget [9]. When the application exceeds the allocated resources, App Engine will return an unavailable status and the user will not be able to use the application.

4 Summary of Benefits and Risks

Although SaaS and PaaS might target different customers – SaaS is focused primarily on business and retail customers and PaaS targets customers and developers – they appear to be based on the same revenue models of subscription or pay-per-use fee, and have some common benefits and risks. The diagram of Fig. 1 summarises the benefits and risks for the three aforementioned stakeholders – platform owner, end-user and developer. The overlapping benefits and risks of the two models are represented in the central section of the diagram.

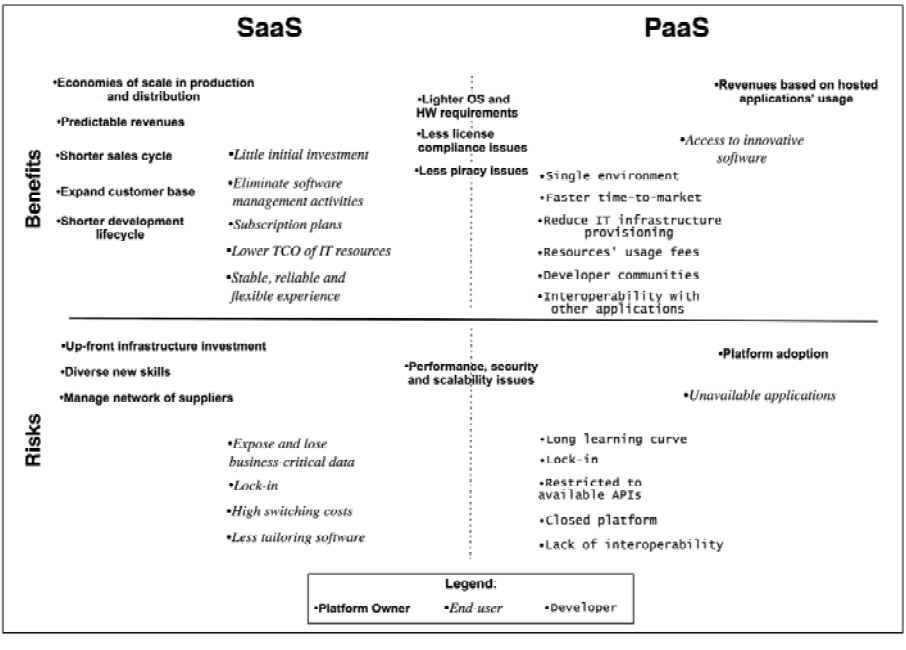

Fig. 1. Summary of benefits and risks for different stakeholders in Software as a Service and Platform as a Service models

5 Analysis of Emerging Experiences of Mobile Operators' Service Platforms

This section analyses the emerging experiences of mobile operators in building platforms for software development targeting end-users and developers. By applying the benefits and risks of SaaS and PaaS concepts to the mobile world, potential drivers for platform adoption are identified.

In case a mobile operator chooses to develop in-house applications targeting retail or business consumers, the SaaS model is applicable. Therefore, a mobile operator would be required to build a team of developers to create and support applications, but it seems rather unlikely that an operator would be willing to take up this task. Mobile operators have always been focused on network operations and few attempts to diversify services through software service provision have been made in the past.

One may argue that mobile operators' thematic portals offering ringtone or game downloads are an instance of a SaaS platform. However, for these portals operators mainly partnered with content and application providers. Moreover, in general these portals did not sustainably attract consumers because they failed to provide a flexible and rich experience and engaging services. For these reasons, a PaaS scenario in which operators rely on third-parties to develop applications seems to be the opportunity to expose telecommunications network functionalities, as long claimed by developers. Some major operators already launched platforms that incorporate the PaaS concept – Ribbit, incorporates former Web21C SDK (BT), Orange Partner (Orange), Litmus (O2), Betavine (Vodafone) and open movilforum (Telefonica).

Blending the Web 2.0 development environment with the network's services and information would result in a richer application environment with personalised services. Mobile operators hold plenty of information about their customers and a large customer base. That means an application could be passing code to be executed on the operator's side and send the resulting information to the end-user, but still keeping user's information private. Thus revenue opportunities for both suppliers and operators are significant.

In order to allow this integration of both worlds, mobile operators need to provide application programming interfaces (APIs) that give the means to developers to efficiently explore their networks' capabilities such as location, rich presence, charging, messaging and so forth. Perry [10] and Warfield [11] analysed current market PaaS and identified several advantages in providing a common and open API. A common API would stimulate innovation, reduce barriers to entry and increase portability between operators. Additionally, it would reach out to more developers, as these would easily identify the advantages of coding once and deploy and monetise in several platforms [12]. Current operator platform's initiatives are using distinct APIs to give access to theirs networks – GSMA OneAPI [13] and OMA IMS [14].

A single environment for the entire application life cycle and the potential of faster market delivery are key advantages of a PaaS platform for developers. Still, a strong developer community around a platform gives developers a feeling of control over the platform and might also be crucial to guarantee platform adoption. More developers

means still more developers, which means more applications, which means more customers, and so on. Quayle [15] studied forty application developer communities and identified six important factors in building a developer community, among which is included a channel to enable developers to earn profits from their applications. In a mobile telecom context, a marketplace would allow a win-win situation between mobile operators that sell and host the applications, developers that write them and customers that want to buy and use them. Additionally, an operator would need to promote this marketplace as it own service, by giving it visibility through the operator's website and marketing campaigns.

Regarding current operator platforms all have succeeded in building up a community providing code examples, forum, blog, and case studies. Almost all provide optional hosting of the application, except for Orange Partner that partnered with Cellmania. Ribbit, Orange Partner and Litmus already gave a commercial component to the platform, enabling the developer to define a price and sell an application; still, only Orange succeeded in incorporating applications into the operator's website via the Application Shop section [16]. Litmus, on the contrary, transmits the idea that the whole concept is still a trial by incentivising end-users to test applications in return for a free application. Ribbit charges developers at production stage based on the resources the application uses. In Orange Partner, the revenue is split between Orange, Cellmania and the developer, but it is not clear whether Cellmania charges developers on a resources usage basis.

To conclude, a PaaS platform brings mobile operators the opportunity to expose their networks to the Web 2.0 developer community. The operator will play a role of intermediary for the distribution of third-party applications to its customer base, earning revenues from both sides. For this scenario to be possible, it is necessary to engage both developers and customers by building developer communities and presenting developers a direct route to the market through a marketplace that effectively reaches the customer. A common API will further engage developers and leverage the potential of reaching customers across operators and countries, just like a regular Web-based application would. However, up to now mobile operators' platforms have been slow to mature and offered APIs lack consistency.

6 Conclusion

This paper analysed the SaaS and PaaS concepts from a business perspective and highlighted the benefits and risks for both suppliers and customers. It then explored the potential of applying these models to the mobile world to allow for additional sources of revenue. The PaaS model seems to best suit a scenario in which an operator exposes telecommunications network functionalities by creating the platform for third-parties to develop applications that offer a flexible and rich experience and engaging services, and acts as an intermediary in the distribution of these applications to its customer base, earning revenues from both sides. Three main factors were identified as drivers for platform adoption: a common API, a developer community and a marketplace. Current mobile operators' platforms have been slow to integrate a marketplace and offered APIs are fragmented among operators.

Acknowledgement

This work was performed within the WTEPlus project (number 10328) funded by the Interdisciplinary Institute for Broadband Technology (IBBT). The author gratefully acknowledges Dr Pieter Ballon for helpful comments on earlier versions of this paper.

References

1. Ballon, P.: Control and Value in Mobile Communications: A Political Economy of the Reconfiguration of Business Models in the European Mobile Industry: SSRN (2009)
2. Armbrust, M., et al.: Above the Clouds: A Berkeley View of Cloud Computing, EECS Department. University of California, Berkeley (2009)
3. Chappell, D.: A Short Introduction to Cloud Platforms. DavidChappell & Associates (2008)
4. Mizoras, A., Goepfert, J.: AppSourcing Taxonomy and Research Guide. In: IDC (ed.) IDC-Industry Developments and Models (2003)
5. SIIA (ed.): SIIA: Software as a Service: Strategic Backgrounder, S.I.I. Association, Washington (2001)
6. Chong, F., Carraro, G.: Building Distributed Applications: Architecture Strategies for Catching the Long Tail (2006), http://msdn.microsoft.com/en-us/library/aa479069.aspx
7. Mitchell, D.: Defining Platform-As-A-Service, or PaaS (2008), http://blogs.bungeeconnect.com/2008/02/18/defining-platform-as-a-service-or-paas/
8. Andreessen, M.: The three kinds of platforms you meet on the Internet (2007), http://blog.pmarca.com/2007/09/the-three-kinds.html
9. Google App. Engine's Quotas (2009), http://code.google.com/appengine/docs/quotas.html#Adjustable_and_Fixed_Quotas
10. Perry, G.: Thoughts on Platform-as-a-Service (2008), http://gevaperry.typepad.com/main/2008/09/the-future-of-p.html
11. Warfield, B.: The Perils of Platform as a Service (It's Not As Bad As All That!) (2007), http://smoothspan.wordpress.com/2007/10/08/the-perils-of-platform-as-a-service-its-not-as-bad-as-all-that/
12. Orange sees API uptake, but operator cooperation needed (2009), http://www.telecoms.com/itmgcontent/tcoms/features/articles/20017613596.html
13. GSMA: 3rd Party Access Project, http://gsma.securespsite.com/access/default.aspx
14. Alliance, O.M.: OMA IP Multimedia Subsystem (IMS in OMA) V1.0 (2005)
15. Quayle, A.: Opening Up the Soft Service Provider: The Telco API (2008)
16. Orange Launches New Widget Experience and Expands Multi-Platform Application Shop (2009), http://mobileworldcongress.mediaroom.com/index.php?s=43&item=525