Cloud computing service models, Part 3: **Software as a Service**

Dan Orlando
CEO
Creative RIA

31 January 2011

Explore the four primary factors that drive return on investment for Software as a Service developers and how those factors are leveraged to make SaaS profitable for stakeholders. In this final article of a three-part series, discover the business model opportunities that SaaS presents and gain insight into the role of user experience design for SaaS applications.
In this three-part series, straightforward, real-world examples of cloud computing help eliminate the confusion around the concepts. Each article in this series covers one of the three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS).

View more content in this series

Software as a Service provides network-based access to commercially available software. Chances are, you've already used SaaS, even if you didn't know it at the time. Examples of SaaS include Netflix, Photoshop.com, Acrobat.com, Intuit QuickBooks Online, Gmail, and Google Docs. SaaS implementations that may be somewhat less obvious include a significant portion of the growing mobile application marketplace.

Frequently used acronyms

- API: Application programming interface
- GUI: Graphical user interface
- IT: Information technology
- LAN: Local area network
- ROI: Return on investment
- UI: User interface

SaaS represents the potential for a lower-cost way for businesses to use software—using it on demand rather than buying a license for every computer, especially when you consider that most computers sit unused almost 70% of the time. Rather than having to buy multiple licenses for a single user, the closer a business can get to putting a license to use 100% of the time, the more money that business will save.

The cross-concept matrix of the three services provided in Part 1 of this series is provided again in Table 1 for your convenience.

Table 1. Cross-concept matrix of the three classifications of cloud computing

	Paradigm shift	Characteristics	Key terms	Advantages	Disadvantages and risks	When not to use
Infrastructure as a Service (IaaS)	Infrastructure as an asset	Usually platform-independent; infrastructure costs are shared and thus reduced; service level agreements (SLAs); pay by usage; self-scaling	Grid computing, utility computing, compute instance, hypervisor, cloudbursting, multi-tenant computing, resource pooling	Avoid capital expenditure on hardware and human resources; reduced ROI risk; low barriers to entry; streamlined and automated scaling	Business efficiency and productivity largely depends on the vendor's capabilities; potentially greater long-term cost; centralization requires new/ different security measures	When capital budget is greater than operating budget
Platform as a Service (PaaS)	License purchasing	Consumes cloud infrastructure; caters to agile project management methods	Solution stack	Streamlined version deployment	Centralization requires new/ different security measures	N/A
SaaS	Software as an asset (business and consumer)	SLAs; UI powered by "thin client" applications; cloud components; communication via APIs; stateless; loosely coupled; modular; semantic interoperability	Thin client; client-server application	Avoid capital expenditure on software and development resources; reduced ROI risk; streamlined and iterative updates	Centralization of data requires new/ different security measures	N/A

The four driving factors of ROI for SaaS

Develop skills on this topic

This content is part of progressive knowledge paths for advancing your skills. See:

- Cloud Computing: Introduction to Software as a Service
- Cloud computing: Fundamentals

SaaS presents new opportunities for software vendors. In particular, four driving factors are cited by SaaS software vendors as leading to increased ROI:

- Increased speed of deployment
- Increased user adoption
- Reduced support requirements
- Lowered cost of implementation and upgrades

Speed of deployment

Traditional desktop applications have historically involved significant deployment hurdles. In fact, I've heard desktop application developers refer to updating their applications as a "deployment

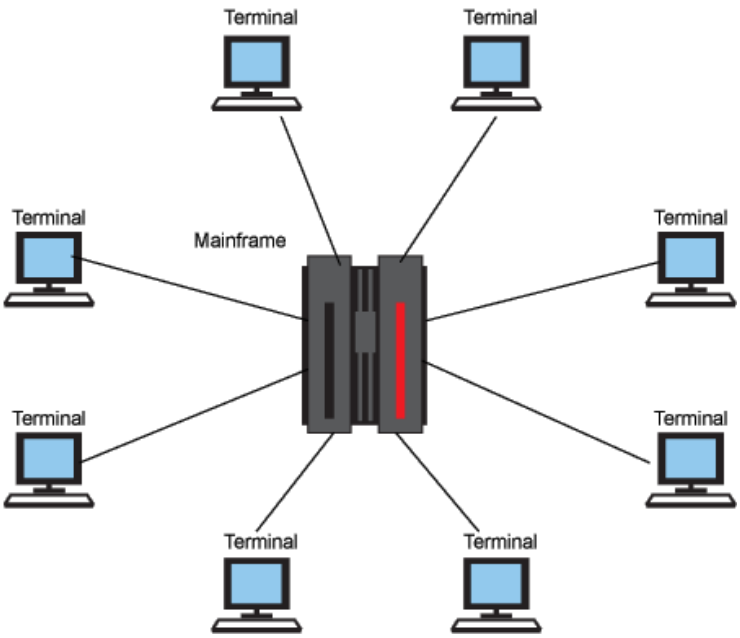
nightmare" on several occasions. As Tariq Ahmed states in the first chapter of *Flex 4 in Action* (Manning Press), "The logistical complications of trying to get thousands, if not hundreds of thousands, of clients to run the precise version of your software at the exact same time are immense."

Ahmed goes on to say that such complications are so immense that most desktop software development companies don't even consider it reasonable or even feasible. Developers who have struggled with this in the past are good candidates for deploying SaaS versions of their software. However, the biggest barrier to entry into the SaaS marketplace that traditional software houses experience is enabling desktop applications to run as SaaS applications. In many cases, doing so involves re-writing the software on some level, which some companies find too cost-prohibitive.

This is one of the primary reasons the movement to cloud computing has been a slow and gradual process. In most cases, the logical solution is to move software to the cloud in phases, beginning with a highly scaled-down version of the original application provided as SaaS. This makes obvious sense when considering the level of control the developer has on version control. It's also where the specific anatomy of SaaS plays a significant role.

You can see many similarities between cloud computing and the "LAN computing" of years past. A typical LAN architecture consisted of an array of on-site workstations, often referred to as *dummy terminals,* that ran applications by connecting to a powerful mainframe, usually provided by IBM, as seen in Figure 1.

Figure 1. A simple diagram showing the relationship of client terminals and mainframe system in a basic LAN

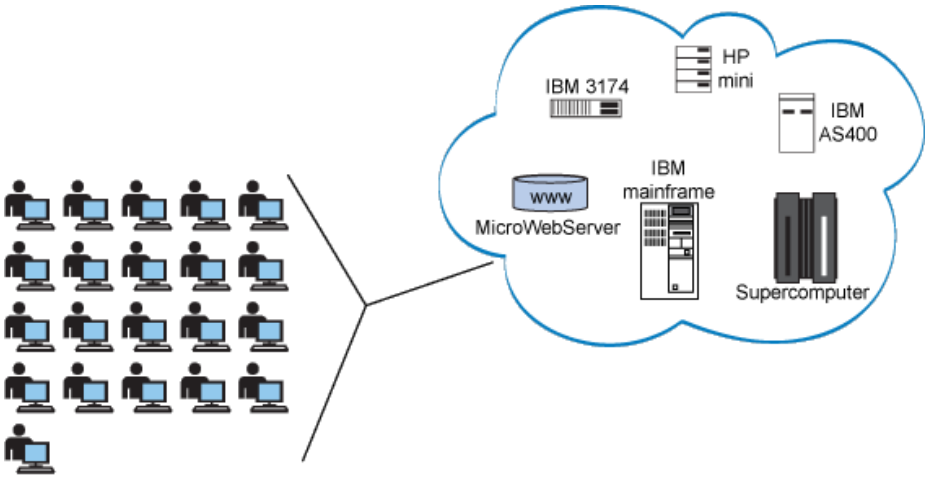


This type of computing worked out nicely for enterprises, because IT departments had the ultimate level of control over versioning and updates could be deployed on an iterative and seamless basis with little or no hassle. Similarly, the logistical barriers that prevented version control for developers

of desktop software applications in the past is nonexistent in the cloud, because the software runs on an infrastructure that the developing company has immediate access to.

The size and scale of an SaaS infrastructure is clearly massive in comparison to that of a LAN in consideration of the number of clients that it must be able to serve. But the underlying concept is the same. Whereas Figure 1 shows a single mainframe that is able to host enough software instances to serve all of the clients connecting to it within the local network, Figure 2 shows a cloud that consists of many different computer resources, all contributing to the total compute power able to run the many software instances required to service clients around the world.

Figure 2. A simple diagram showing the relationship of client devices to the cloud in SaaS



Increased adoption

If you step away from business or a moment and look at what SaaS does for the general consumer, you find that SaaS makes software available that previously may have carried too high a licensing fee to be reasonable for general consumer use. A good example is Adobe's efforts to make Adobe® Photoshop® an SaaS. Although this is still a work in progress and has been an evolving experiment for Adobe, it still shows progress. For instance, I've noticed that a growing number of my friends and family members are starting to prefer the use of Photoshop.com for basic photo editing as opposed to launching the full-blown version when they need to carry out rudimentary photo-editing tasks. The significance of this is in the fact that people who don't require the functionalities contained in the full version will now save money. Meanwhile, others are using Photoshop.com who would not have otherwise tried it, which presents an opportunity for Adobe to capture long-term customers that it would not have otherwise had access to.

The various business models for SaaS are particularly intriguing. For example, Intuit offers QuickBooks Online as an SaaS with a monthly service charge. As a business owner who does a lot of traveling, I've found this to be particularly useful, especially because my business partner lives 400 miles away in a different state. Meanwhile, Adobe leverages the power of SaaS with Photoshop.com and Acrobat.com by offering software as a *freemium* service—a term that was

coined to describe a particular business model based on a scaled-down SaaS offering of a licensed software product.

Freemium SaaS is based on a revenue model where it is anticipated that a certain percentage of free users will eventually find it useful enough to upgrade to either a paid version of the SaaS that has additional features enabled or a licensed copy of a desktop version of the software that includes all the additional features and functionalities available. This tends to be a preferred method of trying out software in "restricted demo" mode compared to having to install an application on your desktop that you may not end up buying. In addition, this model can be further supplemented with advertising if the ratio of free users to upgrades is less than expected. This is a common method that traditional desktop software vendors use as a way of adapting to the changing marketplace as cloud computing continues to evolve.

Reduced support requirements

The cost of large customer service help desks and the problem of increased support issues that occur as a result of having to support multiple platforms is largely mitigated with SaaS. For starters, ease of deployment allows developers to implement fixes shortly after bugs are initially found, which means that most bugs can be fixed before the vast majority of users encounter them, resulting in fewer phone calls for customer support as well as a much higher likelihood for customer satisfaction and retention.

In addition, manufacturers of traditional desktop software applications often have to support more than one platform. For example, a developer that must support the Windows® 7 and Apple Mac OS X version 10.6 operating systems has nearly doubled the cost of development just by adding support for the second operating system—and this is before getting into the problems associated with supporting the many different versions of such operating systems. Supporting multiple versions of an operating system presents limitations, as well.

For example, if you are building a program to run on Windows 7 but it has to be compatible with Windows XP, you have to be very careful that the features and functionalities will be able to run on both versions; otherwise, you'll be forced to branch the project again, leaving you with separate code bases for each, which inevitably decreases productivity and efficiency and increases your time-to-completion estimates. One of the quickest ways to give a business executive a heart attack is to tell him or her that the estimated time lines for the next two years' worth of iterations have been doubled. Add to that the increase in budget for supporting different operating systems and the different versions of those operating systems, and (among other things) it's no wonder that you see such a high rate of failure for software development projects presently.

Lowered cost of implementation and upgrades

The fourth driving factor of ROI for SaaS is somewhat similar to the first. However, *speed of deployment* describes the advantages gained from being able to quickly and painlessly deploy application updates. In contrast, *lowered cost of implementation and upgrades* describes the financial benefits to the development company, which are gained as a result of having control over versioning and the infrastructure that runs the software.

A big savings to the developer comes from not having the added overhead of testing and deploying bug fixes and new features on multiple platforms, because the developer has control over the platform on which the software runs—typically completely transparent to the user. This makes the upgrade path for SaaS applications less cost-prohibitive. The indirect financial result comes from customer satisfaction and retention, because the substantial savings in both time and money provide the developer with the opportunity to have a greater level of responsiveness to feature requests and enhanced usability.

SaaS and user experience design

SaaS applications represent a sort of next-generation approach to application design. Although it might not be technically stated in any of the documents that I have seen to date, it seems that SaaS programs also include a modern approach to UI design that is more consistent with the product design process seen in most other industries. This approach includes a process known as *user experience design* (UXD), where the GUI is designed by a product team rather than the development team.

The primary objective of UXD is to identify what will make the application most useful for the intended customer base and including that knowledge as part of the design. Although it can be logically argued that this should take place in the development of any kind of software, it seems to be most prevalent among SaaS application development. Perhaps the reason for this has to do with the different business models available with SaaS compared to that of traditional software as well as the substantial savings gained from developing SaaS.

SaaS for developers

As you've learned, full-blown cloud computing is a massive transition for both businesses and consumers, and many challenges must be overcome. As a result, the process will take time and go through several periods of gradual change. During this evolution in computing, it is critical for software houses to be able to adapt to the changing environment to continue to fulfill the needs of businesses and consumers alike.

Just as businesses must be able to move with the changing environment as computing in the cloud evolves, software programmers will need to adapt their skill sets and understand SaaS programming models in order to stay sharp and keep themselves in demand. Cloud computing is not just about scalable infrastructures and platform portability through virtualization. It also takes software to an entirely new level and represents what could reasonably be considered the next generation of computer programming. That might be a bold statement, but consider the opportunities that SaaS presents discussed in this article.

For instance, affordability means wider availability, which equates to a larger potential customer base. Add to that the savings that come as a direct result of having control over the platform, infrastructure, and versioning of the software, and it quickly becomes evident that SaaS brings with it a certain level of "democratization," where small and medium-sized development shops can play on the same field as the majors.

Conclusion

In this article, you learned about the anatomy of SaaS, the third classification of cloud computing. More importantly, you learned about the opportunities and challenges presented by the growing popularity of SaaS applications.

My hope is that after having read the three articles in this series, you have a clearer picture of what cloud computing means for the future of your career or business. In addition to the resources provided on IaaS and PaaS in the previous two articles, the Resources section provides links to more information on SaaS for your enjoyment.

Resources

- Watch video on SaaS from InfoClipz.
- Check out the *Baseline Magazine* article on what to expect for SaaS in 2011.
- Check out the Experian survey on SaaS spending trends.
- Sys-Con Media provides a solid article on open source software in the cloud.
- Explore developerWorks Cloud Computing zone, where you will find valuable community discussions and learn about new technical resources related to the cloud.
- In IBM Smart Business Cloud Computing, get valuable business advise to enhance performance and efficiency in the cloud.
- Follow developerWorks on Twitter.
- Watch developerWorks on-demand demos ranging from product installation and setup demos for beginners to advanced functionality for experienced developers.
- Grace Walker's developerWorks article "Cloud computing fundamentals" provides a good introduction to cloud computing.
- Check out "Flex 4 in Action" by Tariq Ahmed, Dan Orlando, John C. Bland II, and Joel Hooks (Manning, November 2010).
- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Dan Orlando

Dan Orlando is a recognized leader in the enterprise development community. As a long-time consultant, Dan's expertise on Adobe technology platforms is often called upon by industry leaders as well as publications such as IBM developerWorks, Adobe Developer Connection, and Amazon Web Services. Dan can also be found blogging regularly at DanOrlando.com.