Software as a service: A look at the customer benefits

Bret Waters
has been the founder and CEO of three Silicon Valley companies: Artmachine, a leading developer of hosted software applications for digital asset management (acquired by eMotion); Metagraphics, a developer of software and services for publishing workflow, and Window City Software, a developer of programmers' toolkits for user interfaces. He has been a featured speaker at many leading industry events, and is the author of several articles focusing on the principles of developing software that better meets the real-world needs of customers. He is a graduate of San Jose State University, and the Stanford University Graduate School of Business, Executive Program. He currently serves as Vice President of the Board of Trustees of the Woodside School Foundation, and as a special consultant to eMotion, Inc.

Keywords: *SaaS, return on investment, ROI, hidden cost, total cost of ownership, licensed software, fast deployment, eMotion*

Abstract This paper introduces readers to the concept of software as a service (SaaS) and helps them evaluate a business case for where to use SaaS in the enterprise. While the underlying concept behind SaaS — renting your business application as a service — is not new, changes in technology and the software landscape now make this model a viable and smart alternative to licensed software. By looking at the history of enterprise computing, the author illustrates why this new wave in computing models represents a natural evolution in technology. More importantly, by examining the hidden costs behind licensed software or the total cost of ownership, the author helps the reader better evaluate the trade-offs between licensed and hosted software.

Bret Waters
eMotion, Inc.,
282 Second Street,
Suite 300,
San Francisco, CA 94105,
USA
Tel: +1 415 546 2110
Fax: +1 415 546 2117
Email: bret@bretwaters.com

INTRODUCTION

Few would dispute that software applications are the driving force behind modern business operations. And yet it is also a fact that surveys consistently show that most business executives view software as one of the major problem areas of their business.

The contrast of these two points can probably be explained by the fact that while the benefits of enterprise software are well-accepted, IT initiatives have a long (and well-deserved) reputation for being the source of frustrations such as cost overruns, implementation delays, and misalignment with real-world business needs.

The reality is that most line-of-business managers do not want to manage IT projects at all. In fact, the technology needs of the line-of-business manager can be distilled down to a single imperative: high-quality IT service, with clearly understandable costs.

This paper will take a look at an emerging trend that addresses this customer imperative: vendors providing enterprise software applications as an on-demand service, in order to relieve the customer of the administrative burden and protecting them against unanticipated costs.

HISTORICAL BACKGROUND

Enterprise computing has gone through several evolutionary waves over the years, and each wave has brought ever-increasing benefits to organizations of every type. Yet, through all these waves of innovation, one common theme recurs — customers' predictable frustrations at being saddled with enterprise software initiatives with unexpectedly high (and sometimes hidden) costs associated with them. These unanticipated high costs can stem from a variety of sources: long and painful implementation, unfactored maintenance and update costs, the inability of the software to adapt to rapidly changing business needs, and/or to the change management, organizational distraction, and internal resource consumption that often comes with a new IT initiative.

Return on investment

As with any business purchase, the initial enterprise software purchase decision needs to be

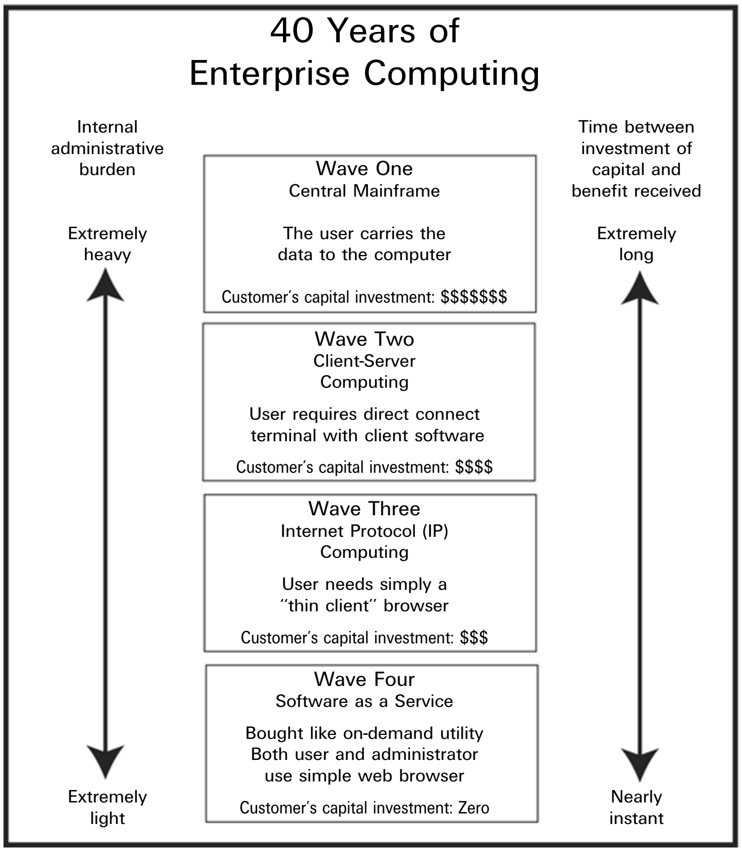
40 Years of Enterprise Computing
Internal administrative burden
Extremely heavy
Extremely light
Time between investment of capital and benefit received
Extremely long
Nearly instant
Wave One
Central Mainframe
The user carries the data to the computer
Customer's capital investment: $$$$$$$
Wave Two
Client-Server Computing
User requires direct connect terminal with client software
Customer's capital investment: $$$$
Wave Three
Internet Protocol (IP) Computing
User needs simply a "thin client" browser
Customer's capital investment: $$$
Wave Four
Software as a Service
Bought like on-demand utility
Both user and administrator use simple web browser
Customer's capital investment: Zero

Figure 1: Evolution of enterprise computing

made with a clear hard look at the expected return on investment (ROI). A look at the history of enterprise computing will show that while the business has often been fraught with customer frustration over costs, generally speaking, vendors have gotten that message, and technology and business models have evolved to slowly give customers improved ROI factors along with ever more powerful applications (see Figure 1).

This paper will examine how the most recent wave — providing software as a service — is the culmination of this evolution and finally provides customers with exactly what they want — powerful software, reliably delivered, with low capital investment and rapid realization of the business benefit.

THE FOURTH WAVE: SOFTWARE AS A SERVICE

The maturity of the enterprise software industry today (specific enablers will be examined below) has made it possible for vendors to deliver effective software applications using a new delivery model, known as *software as a service* (SaaS).[1] With SaaS the software applications run in the vendor's datacenter (although they appear to the users to be part of the customer's internal IT network). The customer has full administrative control, but the vendor assumes all the responsibility for care and nurture of the software and servers, relieving the customer of any of the associated burden. The result is that the customer receives simply the benefits of the software, with clearly understandable costs, at a contractually defined service level.

The utility analogy

This model is sometimes called *utility computing* because it is a service delivery method which is analogous to the way in which utilities such as electricity are delivered. Organizations and individuals today do not build their own electrical power plant — instead they simply pay a monthly fee to a utility provider in order to receive electricity. This electricity they receive is of higher and more reliable quality than the customer could ever produce themselves — and at a much lower cost. Economists would point out that one of the principal benefits of the utility model is that it is a highly effective use of capital because the cost of the capital infrastructure (eg dams, power plants, high-voltage lines, etc) is borne by the vendor who is able to achieve large economies of scale by distributing the costs of these investments across thousands of customers.

The enablers of this model

The principal enablers today for the software utility model — software as a service — are as follows:

- *The relative homogeny and ubiquity of workstations.* With the internet boom now having reached maturity, virtually every business user today has a computer with an internet connection and a web browser, all using exactly the same data communication protocols (HTTP, TCP/IP, etc), regardless of the operating system (Windows, Mac, Unix, etc). To return briefly to the electric utility metaphor, the power company does not need to know your brand of toaster, because all toasters today have the exact same power requirements. Before the age of internet standards, this was not true of computers.
- *Physical location of data no longer matters.* In today's world of high-speed connections, IP-based software and such things as network attached storage (NAS) and storage area networks (SAN), the physical location of data is completely transparent both to applications and to users (and the same security mechanisms can protect access to the data, regardless of where they physically reside).
- *The development of web services protocols* means that applications running in geographically disparate locations can "talk" to one another and exchange data, completely transparently to users, as if the applications were together on the same server.
- *The relative maturity of the software business* makes it much easier today than it may have been a few years ago for average vendors and customers to enter into service agreements with clear understanding of the rights and obligations of both parties. This may be a subtle point, but it is an important one.

WHAT'S WRONG WITH THE TRADITIONAL SOFTWARE MODEL?

Before discussing the benefits of software as a service, it may be worth reviewing in a little more detail what the principle shortcomings and challenges of traditional installed software have historically been.

In the traditional software purchase model the customer buys a license to the software, then installs and manages it on their own hardware. This has been the way that most enterprise software has been purchased for the past 30+ years. Anecdotal evidence and research suggest three main recurring problems with traditional software: unexpected costs, delays in implementation, and ongoing administrative burden. These are examined below.

Unexpected costs

It has become common wisdom that the actual software purchase price is usually only a small part of the total cost of ownership (TCO) of an enterprise software application. In fact, recent Gartner studies have shown that between 50 and 80 per cent of IT budgets are spent not on acquisition of hardware and software, but on system implementation and maintenance (see Figure 2).[2]

The principal shortcomings of this model, from a customer's perspective, is that (1) IT

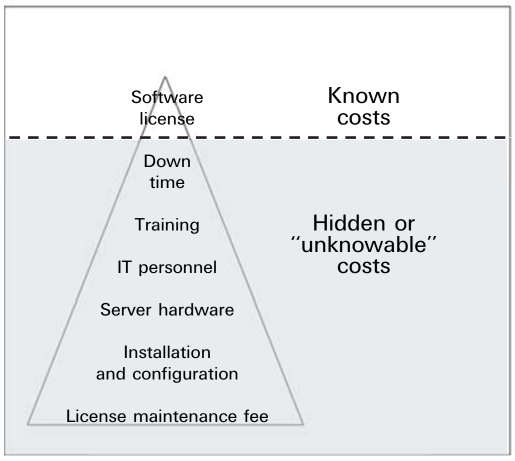

Figure 2: The total cost of ownership of an enterprise software application

expertise is required for implementation, even though the purchaser may be an organization without extensive IT capability (marketing, creative services, field sales, etc); (2) ongoing costs are associated with the care and feeding of the application; and (3) there are often hidden or unanticipated costs which dramatically affect the expected ROI.

The breakdown of costs will obviously vary by company and by type of IT initiative. If all the costs are known, then they can simply be factored into the purchase decision and into the ongoing budget associated with the initiative. What hurts, of course, are unexpected costs and unfactored soft costs — the costs such as organizational distraction and line-of-business resources that often get gobbled up during the implementation and ongoing maintenance of an IT initiative.

Delays in implementation

Historically, software implementations have probably been the most behind-schedule projects in any company — 85 per cent of all software development projects are delivered behind schedule and over budget.[3] The reasons are myriad — scope-creep, misjudging complexity, department A waiting for department B, unexpected bugs and compatibility issues, and so on. Virtually every company has had the experience of a big, long-overdue software project that — when it is finally complete — is no longer relevant because the company and/or marketplace have changed — which is painful money down the drain.

Ongoing administrative burden

This is really a subset of unexpected costs, since administrative burden translates directly or indirectly as additional cost. JP Morgan Chase technology analyst Chuck Phillips cites a recent US Department of Commerce study indicating that software license expenditures account for only 30 per cent of the total cost. The lion's share represents labor costs, with 37 per cent of spending going toward IT staff support and 33 per cent earmarked for external consultants related to software implementations.[4] The administrative challenges that an organization faces today (especially a non-IT organization) in the ongoing management and administration of enterprise software can take many forms, as detailed below.

Managing a heterogeneous environment

Most enterprise environments today are heterogeneous, which means that applications need to be actively managed to maintain compatibility with several different operating systems and platforms. Perhaps the best example of this is within marketing departments, which need to be compatible with the Macs in the creative services department, the Windows machines in the field, and the Unix servers in the IT department.

Capacity planning and utilization

Because every customer wants to allow room for growth in a new IT initiative they tend to purchase more capacity than they actually need today (an inefficient use of capital), and then often end up having to quickly expand the capacity later (at an inefficient panic price).

Security management

For enterprise applications today security is crucial, and yet managing internet-based security requires specialized expertise and nearly constant upgrades. This can be a large and unanticipated cost.

Implementation delays

The time gap between when the capital is invested and when the organizational benefits are realized is a crucial component in the ROI equation. An implementation delay can completely wipe out an organization's anticipated ROI. The economists call this *value delay* and it can add dramatically to the true cost of an investment. Even worse — as discussed above — software implementation delays are sometimes so severe that when the project is finally released the original business needs have changed dramatically, rendering the software obsolete or ineffective.

The snowball effect of upgrade costs

The snowball effect of upgrade costs is a dynamic with which we are too familiar: the new application ends up requiring a new database which ends up requiring a new server which ends up requiring new hardware, and so on.

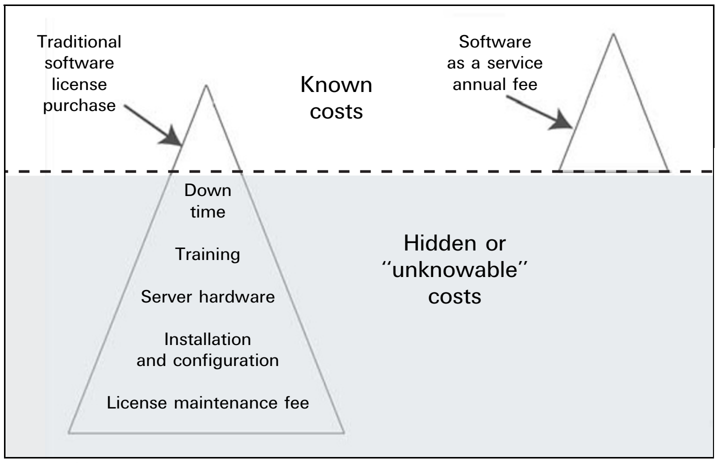
Traditional software license purchase
Known costs
Software as a service annual fee
Down time
Training
Server hardware
Installation and configuration
License maintenance fee
Hidden or "unknowable" costs

Figure 3: Benefits of software as a service

THE BENEFITS OF SOFTWARE AS A SERVICE

The principal benefit of buying software as a service is that every one of the points addressed above (shortcomings of the traditional software model) become the responsibility of the vendor, and the business manager simply receives exactly what he/she wants: high quality IT service with clearly understandable costs (see Figure 3). This delivery model is examined in more detail below.

Total cost of ownership (TCO)

Unlike the traditional installed software model, with software as a service the total costs are knowable, in advance, and contractually defined. And while each implementation needs to be analyzed on its own merits, the total cost of ownership with SaaS is usually dramatically lower because of the efficiencies that the vendor has in managing their own application in their own datacenter, for multiple customers.

Speed of deployment

Because there is no software for the customer to install and configure, an SaaS implementation can be turned on for a customer almost instantly. And even customization, prepopulation, and other professional services are usually implemented much more quickly with SaaS because the vendor is making your customer-specific configurations within their own datacenter, instead of having to fly professional service team members to work on the customer's site.

Reliability

In a global 24/7 world, reliability of an enterprise software application is crucial. A top-tier SaaS vendor can build-in application and datacenter reliability features which would be difficult for an individual customer to match — things such as redundant servers with automatic fail-over, multiple backup power sources, and multiple primary-source internet backbone connections, and 24-hour engineering monitoring. In fact, many SaaS vendors make the reliability promise an integral part of their customer contracts, with an uptime guarantee that can be as high as 99.5 per cent — which, again, would be difficult for individual customers to match internally.

Security, data safety, and disaster recovery

As with reliability, one of the salient benefits of the SaaS model is that all of these important components of the IT infrastructure can be provided by the vendor at a higher level, with lower costs, than a customer could provide themselves.

The myth of buy vs rent

Sometimes the buy vs rent metaphor is used in discussing the relative benefits of traditional installed software as compared with SaaS. This is a misleading analogy for a number of reasons, including the fact that even when traditional software is purchased, in fact, this is only a license to use (rent) it. Table 1 compares the two delivery models and looks at who owns what.

Table 1: The myth of buy vs rent

	Traditional installed software	Software as a service
Who owns the software code?	Vendor	Vendor
Who owns the data stored?	Customer	Customer
Who has administrative control?	Customer	Customer
Who assumes the burden for maintenance, security and reliability?	Customer	Customer

Optimized utilization

With true software as a service the customer can purchase exactly the capacity (data storage, bandwidth, software features) that they need today, and have nearly instant expansion available on-demand, yielding exactly the same capital efficiencies that have led every leading manufacturing firm to employ just-in-time (or on-demand) supply chain management.

Regular updates, with no new software to install

In the old software model, a new update to the application required sending disks out to all the customers to install and configure. With SaaS, updates are seamlessly and instantly performed in the vendor's datacenter, in a process which is completely invisible to customers.

Risk mitigation

Software risk management is a whole specialized field, with entire books on the subject written by authors with doctorates. As such, a detailed discussion of software risk management is well outside the scope of this paper, but even the casual reader will not be surprised to hear that studies suggest that well over 90 per cent of all software projects come in behind schedule, over budget, and misaligned with business needs by the time they are released. Not to mention that post-release operating failures (such as crashes, loss of data, unexpected down time, and so on) are extremely expensive and disruptive to an organization.

This, perhaps, is the most compelling argument in favor of software as a service, as it gives the business executive access to the benefits of powerful software, while at the same time indemnifying themselves against risk by shifting the performance burden to the vendor and minimizing the advance commitment of capital.

EVALUATING YOUR BUSINESS CASE

Despite all the forgoing, software as a service may not be the best choice for every business application. Within the full range of business applications today — finance, manufacturing, marketing, sales — some types of application cases are better candidates than others to realize the full benefits of software as a service. Factors to consider before making the assessment for your particular business case are detailed below (see also Figure 4).

Scope of application reach

If the full reach of your desired application is just one person's desktop, then there is probably very

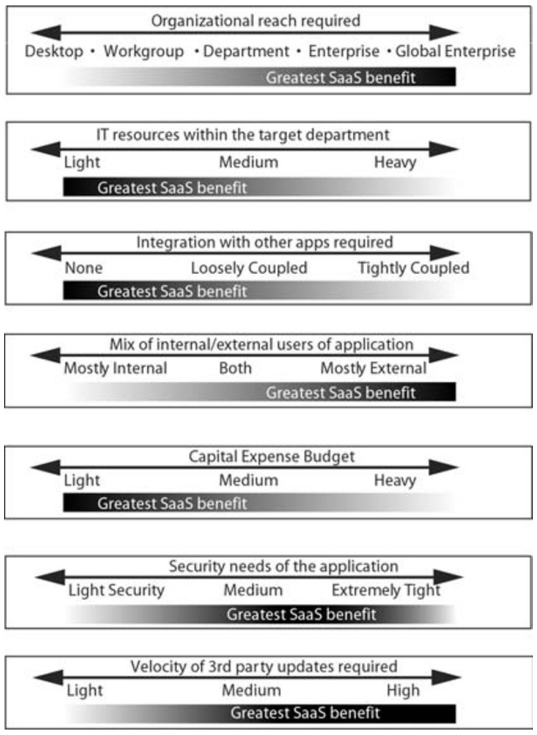

Figure 4: Factors to consider before making the assessment for your business case

little benefit to purchasing software as a service. However, as the number of people to be supported by the application increases, and their geographic distribution widens, the benefits of buying software as a service increase dramatically.

Departmental IT capability

If the department purchasing the application has light internal IT resources, then the benefits of software as a service are tremendous, since the fundamental value proposition of SaaS is providing customers with powerful IT without the need for customer-side IT resources. This makes software as a service especially well-suited to departments such as marketing, creative services, and sales — departments which generally do not have deep internal support staff.

Integration required

Will your new application require integration with incumbent software systems? Older software tends to use what engineers call *tightly coupled* integration — requiring special code to be written around proprietary data exchange protocols and sometimes even requiring that the integration happens within the same operating environment. If you have some of these older applications, and their integration is important to this implementation, then a new SaaS implementation may not be a good fit. On the other hand, newer software is almost always engineered for loosely coupled integration, using open standards for data exchange, and a software architecture where the physical location and operating environment is completely transparent to other applications. This is one of the most important principles of software engineering today.

Mix of users

If you will have both internal and external users of your application, then an SaaS application can have especially profound benefits because your external users will have simple secure access to the application without having to navigate corporate firewall issues. Security is maintained, and yet internal and external users have easy and efficient access. Examples of outside users to be supported might be ad agencies, franchises, joint venture partners, channel partners, etc.

Capital expense budget

If money is no object, and your immediate capital expense (CapEx) budget is unlimited, then the economic benefits of software as a service may be less meaningful to you. If, on the other hand, you and your organization are under pressure to make wise use of both CapEx and operating budgets then, as discussed previously, the economic benefits of SaaS can be significant.

Security needs

Software as a service vendors tend to have extremely robust datacenter infrastructure, and can provide you with application security that meets or exceeds your own internal security standards (in fact, SaaS vendors often provide service to financial institutions and other highly security-conscious customers). For you, this may mean that one of the benefits of choosing an SaaS vendor is that it will give you access to extremely solid application security with lower cost (and much lower security administration burden) than managing security on a traditional licensed installed application. It is worth noting, however, that some organizations with high security needs have cultural and/or policy issues which make them feel more secure if they have physical possession of all their data. In these situations then SaaS may be less of a good business fit, regardless of the fact that the technical reality is that this is a false comfort.

Velocity of update needs

Some applications require regular updates in order to keep current with third-party information and compatibility. An example would be an application that needs to ''know'' about rapidly changing government regulation and tax tables. In these cases software as a service can have unique benefits because the vendor can keep all that information absolutely current through automatic updating of the application and its tables. Similarly, if your application needs to be kept compatible with a whole host of third-party file formats (as in the graphic arts industry) then an SaaS vendor can be making those updates regularly in a way that requires absolutely no effort on the customer's part.

SUMMARY

Any line-of-business manager today who is purchasing new software should look seriously at purchasing the software *not* as a license to something they have to install and manage, but as a service that delivers to them the precise essence of what a business executive needs: powerful software, reliably delivered, with low capital investment and rapid realization of the business benefit.

Not every software implementation may be a perfect fit to realize all the benefits of software as a service, but many business cases will find that the lower internal IT requirements, reduced capital investment, faster implementation, coupled with contractually guaranteed reliability and security, make SaaS a very compelling choice of software delivery models.

The evolution of computing technologies and vendor business models makes this on-demand concept possible today, entirely relieving the customer of the burden associated with installation, implementation, configuration, and maintenance of hardware and software. This "fourth wave" of enterprise computing — software as a service — is changing the way that software is being bought and sold today.

References

1 What's in a name? *Hosted applications*, *On-demand computing*, *Utility computing*, *Hosted service*, *Outsourced application*, *Application service provider (ASP)*, *Software as a service*. While there may be some subtle differences between these terms, they are all names used to describe the same general concept: a business model where the vendor provides software applications not as something for the customer to install and manage, but as a service where the customer derives the benefits of the software without the burden of installation and management.

2 Neela, A. M. and Mein, J. (2003) "How to Cut Your IT Maintenance and Support Costs", Gartner, 23 December, p. 1.

3 The Standish Group Chaos Report (1994), p. 2, http://www.standishgroup.com/sample_research/chaos_1994_1.php

4 Chou, T. (2003) "The Hidden Cost of Software", ASPNews.com, 29 May, http://www.aspnews.com/strategies/asp_basics/article.php/2214031