

Singapore Management University Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

7-2007


The business model of "Software-as-a-Service"

Dan MA

Singapore Management University, madan@smu.edu.sg

DOI: <https://doi.org/10.1109/SCC.2007.118>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Computer Sciences Commons](#), and the [Management Information Systems Commons](#)

Citation

MA, Dan. The business model of "Software-as-a-Service". (2007). *IEEE International Conference on Services Computing SCC 2007: Proceedings: Salt Lake City, UT, 9-13 July 2007*. 701-702. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/572

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

The Business Model of “Software-As-A-Service”

Dan Ma

*School of Information Systems
Singapore Management University
Singapore 178902
madan@smu.edu.sg*

Abstract

In the recent years, the emergence of the Software-as-a-Service (SaaS) business model has attracted great attentions from both researchers and practitioners. Under the SaaS business model, vendors deliver on-demand information processing services to user firms, and thus offering computing utility rather than the standalone software itself. The SaaS has become an attractive alternative to the traditional software delivery model, which typically requires users to purchase, install, and maintain software systems by themselves. In this work, we propose an analytical model to study the competition between the SaaS and the traditional COTS (Commercial off-the-shelf) solution for software applications. The competitive model considers heterogeneous users who differ in terms of their transaction volume, while the SaaS and COTS vendors differ in terms of their pricing structure, setup cost, and system customization. We conclude that when commercial software becomes more open, modulated, and standardized, the SaaS business model will take a significant market share. In the extreme case, it may dominate the whole software industry and drives the traditional software out of the market. We also show that it is never optimal for the SaaS vendors to exert their full lock-in power through harsh software contracts. Under certain conditions, we suggest SaaS vendors to offer their existing users an easy exit option rather than to establish switching barriers to lock them in.

1. Introduction

In the past few years, the Internet has given rise to the Software-As-A-Service (SaaS) business model. The SaaS vendors offer a bundle of software applications, an IT infrastructure, and all necessary support services to users across a network. Under the SaaS business model, the software system and users' data are stored off-site in a central location run by the vendor. The vendor is in charge of all IT support services, including daily software maintenance, data backups, software upgrades, and security. Therefore, it is delivering computing utility, rather than the software only.

In a recent memo, Bill Gates proclaimed that the emergence and rise of the SaaS will be the “next sea-change” in computing [14]. According to AMR Research, the SaaS market is growing more than 20% a year, compared with single-digit growth in traditional software [13]. An increasing number of software vendors, including industry giants such as IBM, SAP, Oracle, and Microsoft, are moving to such a service-oriented business model. For example, IBM is offering “IBM-on-demand,” which allows corporate users to acquire IBM's computing power and software applications as a service.¹ IBM is also providing a package of services and incentives to help other software companies deploy their products as hosted applications [15]. Another good example is Oracle. Oracle is ranked as one of the top ten SaaS providers. In January 2006, it acquired Seibel, and now is delivering its fledgling CRM on-demand software through the Internet [16].

To many corporate users, SaaS has become an attractive alternative to the traditional software solution. Traditionally, most software has been delivered in the form of commercial off-the-shelf (COTS) products.² The vendor sells the software application to users and helps to install it on users' sites. The users possess the full ownership of the software, and must provide IT infrastructure, hardware, and support services in order to enable continuous use of the software. However, the SaaS deliver the software in a novel way. Under the SaaS model, unlike the software perpetual licensing, the software is priced as a service, and typically users pay a fee per transaction. Users' payments are closely tied to the actual utility obtained -- they pay only when they have demand for the software. In many cases, the SaaS may prove cheaper than owning and maintaining an in-house IT system. Users expect to save money on support and upgrade costs, IT infrastructure, IT personnel, and implementation [13]. In a web survey by ThinkStrategies fully one-third of 118 respondents were already using

¹ See <http://www-1.ibm.com/services/ondemand/success.html>.

² A COTS product is a commercial software application that “is designed to be easily installed and to interoperate with existing system components.” Almost all software bought by the average computer user fits into the COTS category: operating systems, office product suites, word processing, and e-mail programs are among the myriad examples. See <http://whatis.techtarget.com> for more information.

SaaS, and another third were considering using one within the following 12 months [7]. A research report from Summit Strategies indicates that small and medium businesses with limited IT resources and constrained budgets are more likely to use such fee-per-transaction applications [5]. The SaaS is a powerful trend, which is becoming an important disruptive force in the software industry: "It is not the end of software. It is just another way of deploying it" (Shai Agassi, president of SAP's product and technology group).

Despite of numerous advantages, the SaaS market is still in its formative stage. Data security and reliability as well as application control are among users' top concerns [1]. In addition, most SaaS vendors offer a one-to-many solution, with limited customization.³ The vendors keep the software in their own sites, and users access and run the software remotely. The software therefore is not tailored to fit an individual user's specific requirements or unique business environment. As a result, the user may need to pay extra to make the standard software application work smoothly with its existing IT systems. Software pricing / contracting is another issue. In reality, many vendors demand users' long-term commitments through contracts. For example, a typical contract offered by the SaaS vendor in the radiology industry requires client hospitals to use the product for at least five years. An early exit incurs cancellation fees that can be as high as a full year's payments. Hence, the potential of being locked in by an outside provider becomes a barrier for users to use the SaaS. All these factors cast doubts on the future of the SaaS business model. So far, little research work has been done in analyzing the long run viability of the SaaS from the economic perspective. Specifically, it is unclear how such a new business model can compete with the well-established software solution (i.e., the COTS in-house system), what factors play important roles in software users' choices, and what is the optimal contracting strategy for the SaaS vendors. This motivates our work.

In this study, we propose a model to study the competition between a SaaS and COTS software. We focus on a software market in which corporate users have these two IT options for external sourcing. The competing vendors are differentiated in the following ways. First, they deliver different products: an easy-to-be-customized software application (from the COTS) versus a bundle of standard software and services (from the SaaS). Second, they adopt distinct pricing modes: an outright purchase (the COTS) versus a "per transaction" fee structure (the SaaS). Third, they employ different delivery methods: software installed on a user's in-house server (the COTS)

versus an interface delivered over the Internet remotely (the SaaS). Users have different cost structures and bear different risks in dealing with each business arrangement. In addition, our model considers software quality uncertainty and investigates the two-stage competition with users' switching costs. Although, in practice, the comparison between the SaaS and COTS involves multiple criteria, such as pricing structure, data integration ability, service level arrangements, as well as technological and security issues, we do not attempt to include all the related factors. Instead, we are particularly interested in estimating the relative economic advantages and disadvantages of the SaaS business model.

Our findings show that the emergence of the SaaS, at the expense of the traditional COTS vendor, benefits every user firm. It offers small and medium firms cost-saving access to software, and competitively reduces large firms' implementation costs even if they will still stick to the traditional COTS software. In specific, the SaaS model is superior to the COTS model when the user firms face a low transaction volume, expensive in-house IT services, or when the user's unfit costs from using a standard product are low. Importantly, we show that the long-run viability of the SaaS business model largely depends on the magnitude of the unfit costs imposed on the users. As such costs decrease, the SaaS's competitive advantage monotonically increases, and eventually will dominate the whole software market. In addition, we find a non-monotonic relation between the users' switching costs and the SaaS vendor's profit. In a matured software market, when users' switching costs decrease below a certain level, the market share of the SaaS vendor increases and its profit improves. In other words, the SaaS vendor may find it profitable to allow users to exit the contract easily. This finding challenges many existing SaaS software contracting strategies.

The rest of the paper is organized as follows. In Section 2, we describe our model. We analyze the competition and identify the key determinants of the future of the SaaS model in Section 3. Section 4 discusses the business implications of our findings, and concludes the paper.

2. The Model

Consider the software market with three parties: software users --- firms in need of software applications; the COTS vendor --- the software provider delivering the traditional COTS software; and the SaaS vendor --- the software provider delivering software as a service. The two software vendors are competing on prices. They set their respective prices, and then the users will choose one of them or just stay out of the market.

Software users have different IT needs. Users' IT needs are measured by the expected volume of their use

³ SAP believes that this lack of integration will eventually lead most large corporations that currently are users of online CRM move back to an in-house system [6].

of the software. Some firms may use the software application more frequently than others, and these firms have larger IT needs. In order to capture this heterogeneity, we assume users are uniformly distributed on a unit-length line normalized from 0 to 1. The location of a user on this line represents its transaction volume. Hence, a user at the location d_i has a demand d_i for the software use (in terms of the number of transactions).

The COTS vendor operates in the traditional way. It sells the packaged software application to users and charges a one time upfront fee. The source code of the application can be modified to fit the user’s specific business needs, which assures a good integration with the user’s existing IT system. The vendor bears an operating cost C to serve one user and receives a one-time payment P from the user. The user must install hardware and IT infrastructures, hire IT staff, and organize an internal IT group to provide software maintenance, data backups, and security and capacity management. The service costs associated with each use of the software is denoted by c (i.e., the service costs per transaction). Each transaction creates a value of u to the user.

The SaaS vendor sells the bundle of software and services, rather than the software only. In reality, most SaaS vendors have established large data centers and strong network infrastructures. To start doing business with a user, the vendor first moves the user’s data to its own server, which costs the vendor a one-time setup cost S . The vendor provides all necessary IT services to enable the software use, which imposes a service cost c per transaction on the vendor. Users “pay as they go”. Hence, they bear no initial setup costs, but need to pay a price p_a per transaction. The software application is installed on the vendor’s site and all users can access and run it remotely via the Internet. Thus, the SaaS vendor is using the “one to many” business structure since one software is serving many users. To any individual user, the application is not well-customized, and each transaction gives the user a total value of $u-t$. The parameter t measures the user’s disutility from not using its ideal product. In many cases, it also represents the cost of extra effort to make the outside application work with the user’s existing IT components smoothly. In the rest of the paper, we call t a user’s “unfit costs.” We assume that the exact value of the unfit costs is unknown by the user *ex ante*, the reason being that software applications are experience goods. Before a user firm uses the software application, it does not know in advance how the application fits its specific business environment or how much effort it must exert in order to run this new application with its existing applications smoothly. Such information will be fully revealed only after the user runs the application in the real settings. We assume that the

unfit costs can be low ($t = t_L$) or high ($t = t_H$), each equal probability.

Figures 1 and 2 demonstrate the COTS and SaaS solutions respectively. The black dot in the figures indicates the location of the software system.

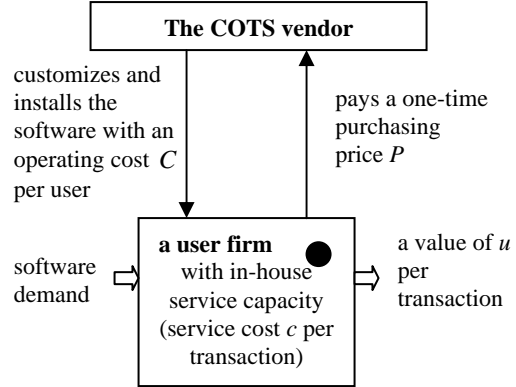


Figure 1. The COTS Software Solution

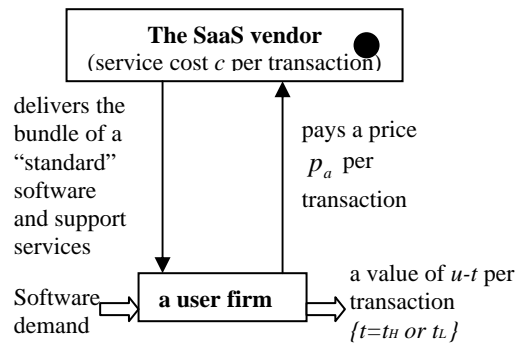


Figure 2. The SaaS business model

Under the SaaS business model, software users are outsourcing their IT services to the vendor. They form a close partnership instead of the simple buyer-seller relationship, which makes users highly dependent on the outside provider. If a user wants to stop using the SaaS software, it bears the costs of getting the data back and recovering the data, which constitute non-negligible exiting costs. In addition, many SaaS vendors will require users to pay a high cancellation fee to stop the business relation. We denote the total exiting costs by the parameter E .

Users need make two-stage decisions. In the first stage, they face incomplete information, i.e., they do not know the exact value of unfit costs t . Users choose the SaaS or COTS software by comparing their expected payoffs. After that, the SaaS users learn their exact unfit costs. With this new information, some of them may exit the market or switch to the COTS, at a cost of E . The decision timeline is shown in Figure 3.

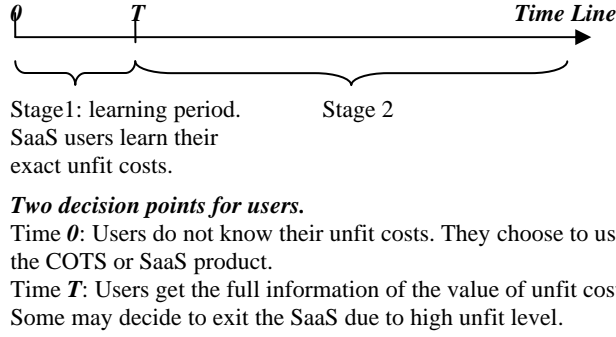


Figure 3. The two-stage decisions

3. The Analysis

We first formally formulate users' decisions as well as vendors' pricing competition in section 3.1. We then study and compare two scenarios – the software market with low switching costs and with high switching cost in section 3.2. Finally, section 3.3 uses numerical examples to demonstrate how the SaaS vendor's profit varies as switching costs gradually increase. To simplify the analysis, we assume that both software providers and user firms care long-run profit (utility) only. In other words, the learning stage $[0, T]$ is short and considered a "transient" stage.

3.1. Formulate the general problems

The COTS-SaaS competition game takes three steps.

Step 1. The COTS and SaaS vendors determine their optimal prices, P and p_a respectively.

Step 2. Given the prices, users, without knowing their exact unfit costs, decide which software to take. It happens at *Time 0*, as in Figure 3.

Step 3. Given their initial choices, users, now obtaining the exact unfit costs, decide whether to exit the SaaS contract (given that they have chosen SaaS initially). It happens at *Time T*, as in Figure 3.

We need solve the problem backward, step by step.

We first analyze the Step 3. At time T , the market outcome must be the following: there is a user with transaction volume d_1 such that all users with transaction volume smaller than d_1 have chosen the SaaS software while those larger than d_1 have chosen the COTS. Hence, users, who have transaction volume $d_i \in [0, d_1]$ and turn out to have high unfit costs $t = t_H$, need take one of the following actions: keep using SaaS, switch to the COTS, or exit the market. The corresponding payoffs

from each action are $(u - p_a - t_H)d_i$, $(u - c)d_i - P - E$ and $-E$ respectively.

We define "the marginal SaaS user" with transaction volume d_{SaaS} . This marginal user is the "last switcher," given that it has chosen the SaaS initially. Hence, all high unfit costs type users ($t = t_H$) with transaction volume smaller than d_{SaaS} will choose to stay with the SaaS, while larger than d_{SaaS} will exit the SaaS. The value of d_{SaaS} is given by

$$(u - p_a - t_H)d_{SaaS} = \text{Max}\{(u - c)d_{SaaS} - P - E, -E\}. \quad (1)$$

Similarly, we define the "the marginal COTS user" with transaction volume d_{COTS} :

$$(u - c)d_{COTS} - P - E = \text{Max}\{(u - p_a - t_H)d_{COTS}, -E\}. \quad (2)$$

After users' switching, the final equilibrium market outcome is depicted in Figure 4. We need analyze two cases: the market is fully served (case a), and the market is partially served (case b). In case a, all users who exit the SaaS choose to switch to the COTS; i.e.,

$$d_{SaaS} = d_{COTS} = \frac{P + E}{p_a + t_H - c}. \quad \text{In case b, users in } [d_{SaaS}, d_{COTS}] \text{ choose to exit the SaaS and stay out of the market, i.e., } d_{SaaS} = \frac{P + E}{p_a + t_H - u} < d_{COTS} = \frac{P}{u - c}.$$

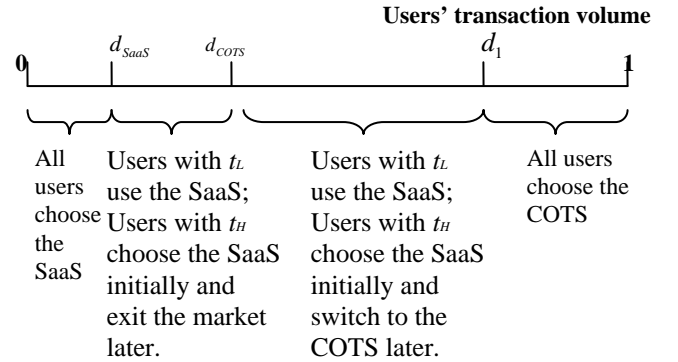


Figure 4. Equilibrium market segmentations

Now we move to the Step 2 to find the expression for d_1 . At *Time 0*, the user with transaction volume d_1 is indifferent between the two options, given that its unfit costs are unknown yet. So, we have

$$\frac{1}{2}[(u - c)d_1 - P - E] + \frac{1}{2}(u - p_a - t_L)d_1 = (u - c)d_1 - P. \quad (3)$$

The left-hand side of the equation is this user's expected utility from choosing the SaaS at *Time 0*. With half probability, the user will be with high unfit costs and switch to the COTS later; and with half probability, the user will be with high unfit costs and keep using the

SaaS. The right-hand side is this user's utility from choosing the COTS at *Time 0*. Solving it gives out

$$d_1 = \frac{P - E}{p_a + t_L - c}.$$

Finally, we analyze the Step 1. Knowing all the consumers' behaviors discussed above, the two software vendors choose their optimal prices simultaneously to maximize their profits.

$$\text{Max}_P \Pi_{COTS} = (P - C) \left[1 - d_1 + \frac{1}{2}(d_1 - d_{COTS}) \right].$$

$$\text{Max}_{p_a} \Pi_{SaaS} = (p_a - c) \left[\int_0^{d_{SaaS}} x dx + \frac{1}{2} \int_{d_{SaaS}}^{d_1} x dx \right] - S d_1.$$

The COTS vendor serves all users in $[d_1, 1]$ and users with high unfit costs in $[d_{COTS}, d_1]$. The SaaS vendor serves all users in $[0, d_{SaaS}]$ and users with high unfit costs in $[d_{SaaS}, d_1]$, but it bears the initial setup costs for all users in $[0, d_1]$. Note that the COTS payment is per user, since it is an upfront one time payment for each user, while the SaaS payment is per transaction.

3.2 Software market with absolute lock-in power versus partial lock-in power

It seems mathematically too complicated to solve the general optimal prices analytically. To get useful insights, in this subsection, we study and compare two special scenarios:

(1) Users face significantly high switching costs. No user, once it chooses the SaaS, is able to exit due to the high switching costs. The SaaS vendor has *absolute lock-in power*. We believe that this type of market is close to the reality given that we do observe many SaaS users are highly dependent on their outside provider and are typically facing high penalty fee if they want to stop the software use.

(2) Users face small switching costs. There are always some users who choose the SaaS initially and switch in the later stage. The SaaS vendor has *partial lock-in power*. We are interested in examining whether, and if so, when a loose bond between the users and the SaaS vendor will benefit the vendor. It will provide useful managerial implications to the SaaS vendor in their contracting strategy.

The following propositions state our main findings in these two types of markets.

Proposition 1. **In the market where users face high switching costs, as users' unfit costs from using a not fully customized system decrease, the relative economic advantage of the SaaS business model increases monotonically.**

Due to the space limit, we are not providing the details of the proof, which is available upon requests. The above proposition shows an interesting finding. To some extent, our model resembles the duopoly competition model with vertically differentiated products, with the SaaS as the low-quality provider and the COTS as the high-quality provider. The unfit cost t measures the quality difference between their products. In the traditional vertical differentiation model, when the quality difference decreases, the low-quality provider initially gets better off because its product becomes more attractive, but it gets worse off as the quality difference decreases further because the competition becomes more intensive and eventually hurts both vendors. However, in the setting we study, we find that reducing the value of t benefits the low-quality provider (i.e., the SaaS vendor) monotonically. Moreover, when the two vendors' products are close enough, the high-quality provider (i.e., the COTS vendor) could be squeezed out of the market. We believe that our findings deviate from traditional vertical differentiation literature because the two vendors here are using different business models.

This result is very important. It reveals the future evolution of the software market. As Figure 5 shows, as unfit costs t decreases, the software market will transit in a way of "the COTS dominates \rightarrow the COTS and SaaS coexist \rightarrow the SaaS dominates." Figure 5 depicts three critical values of unfit costs, $t_1 < t^* < t_2$, which define four different regime. In each regime the market presents distinct structures.

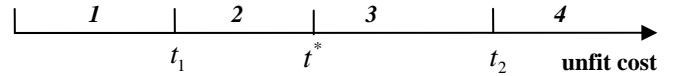


Figure 5. The market structure transition with t

Regime 1. $t \in [0, t_1]$: All users opt for the SaaS model. The COTS solution fails out of the market.

Regime 2. $t \in [t_1, t^*]$: Both the SaaS and COTS models coexist in the software market. The competition benefits each software user (i.e., total consumer surplus increases).

Regime 3. $t \in [t^*, t_2]$: Both the SaaS and COTS models coexist in the software market, but not directly compete with each other. They serve different market segments.

Regime 4. $t > t_2$: The SaaS business model is not able to survive. The COTS model is the only software solution.

Hence, we conclude that users' unfit cost, which represents users' disutility from using a non-ideal, not fully-customized software, is the most important determinant of the success of the SaaS business model. Besides, our analyses also find that when it becomes more expensive to provide on-site IT support services (a large value of c), user firms will find the SaaS solution of outsourcing the IT services to outside experts more

attractive. In such a situation, we can prove that the SaaS model is expected to take a significant market share.

Another important finding of us is that under certain conditions, it is optimal for the SaaS vendor to give up its full lock-in power, which is stated below.

Proposition 2. When the software market is fully covered, namely, when no users are priced out of the market, the profit of the SaaS vendor is higher in the scenario that users can exit the SaaS contract in a less costly way than in the scenario where exiting costs are very high.

The proof is omitted and available upon requests. This result tells us that it is never optimal for the SaaS vendor to assert absolute lock-in power in a competitive marketplace. The reason is: high exiting costs reduce the attractiveness of the SaaS business model. Knowing that it is easy to become dependent on the outside provider, potential users become more conservative when deciding whether to partner with the SaaS vendor in the first stage.

This conclusion, however, is valid only when the market is fully covered (case a). In other words, it is true only when all user firms have access to the software system. We can think of the “matured” software application markets, such as server operating systems (e.g. Linux, Windows NT) and basic software systems (e.g., Microsoft office, email), in which almost all firms are able to afford such applications. We then suggest that the SaaS vendors in such areas should not demand users long term commitments. In contrast, allowing users to free exit will increase the profitability of the SaaS model.

Below, we give an example to show that when some users stay out of the market in equilibrium (case b), a SaaS vendor may gain higher profits by relying on its strong lock-in ability.

Consider a market with $t_L = \frac{u-c}{3}$, $t_H = u-c$, and $S \leq \frac{C_{MOTS}}{6}$. When the SaaS vendor is able to lock in all

existing users, it will charge $p_a^* = \frac{u+2c}{3}$ and keeps all

users in $\left[0, \frac{1}{2} + \frac{C}{2(u-c)}\right]$. However, when users are able

to switch at low costs, in order to keep users, the vendor has to reduce its price to a very low level, close to c . This low price, although giving full market coverage, is likely to make the vendor unprofitable. If so, the vendor would rather give up a part of the users — let them exit in the second stage. For simplicity, we solve the price and profit of the SaaS vendor at the extreme case of $E=0$. It is easy to show that the optimal price $p_a^{E=0} < p_a^*$. On the other hand, all users with high unfit costs will exit the SaaS in the second stage. As a result, the vendor only serves those users with low unfit costs and low transaction volume (

$d_i \in [0, d_1^{E=0}]$). We can further prove that the number of users of the SaaS also reduces. Consequently, the vendor’s profit is lower.

In this example, a SaaS vendor with full lock-in power is able to charge a higher price and still possess a larger market share. In contrast, this vendor will have to set a lower price but still loses market share when users are able to switch costless. Hence, in such scenario, we do suggest the SaaS vendor should only offer long-term contracts which require high penalty costs for early exits to lock in users.

3.3. A numerical example

So far, we have focused on the comparison of two special types of markets: the SaaS provider has absolute or small lock-in powers. It is more interesting if we can draw a whole picture to see how the SaaS vendor’s profit changes as its lock-in power increases. It will help us to identify the optimal exiting costs (in terms of maximizing the SaaS vendor’s profit). To show this, we rely on numerical examples.

Consider the market with following parameters. $u = 15$, $C = 2$, $c = 3$, $t_H = 5$, $t_L = 3$, $S = 0$. Note that in this case, the whole market will be covered, and so the conclusion from Proposition 2 should stand.

Figure 6 reveals a non-monotonic relation between the SaaS vendor’s profit and the users’ exiting cost. As exiting costs increase, the vendor’s optimal profit increases first, peaks at $E = 0.6$, and decreases after that at a fast speed.

Such a similar non-monotonic curve remains after we try different parameter values. This suggests that exerting absolute lock-in power on users is never the ASP’s optimal strategy (as consistent with our Proposition 2); instead, the optimal switching costs for the vendor are likely to be at a middle level.

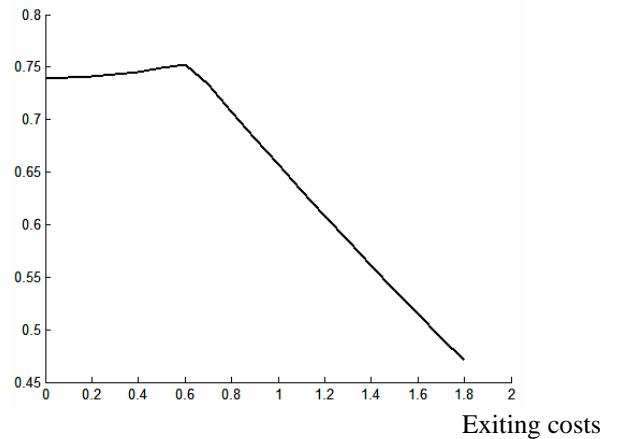


Figure 6. The SaaS’ profit versus exiting costs

4. Business implications

Delivering software as a service, a novel business model, has profound impacts on the software vendors, users and the whole industry. Such a SaaS model has three most important features: First, the vendor offers utility rather than the software. Users therefore are relieved of heavy IT service burdens, which are outsourced to the outside provider. Second, the vendor adopts the usage-based fee structure. It does not require high upfront payments. Users therefore can pay as they go and smooth their payments over a period. Third, the vendor installs and manages the software and users' data on its own site. Users therefore become very dependent on the vendor and face potential lock-in risks. In this work, we identify and model all these important factors, and analyze the competitive advantage / disadvantage of the SaaS business model.

We show that when the users' unfit costs parameter reduces, the SaaS business model will intensively compete and even gradually replace the COTS traditional software. It will result in the transition of the whole software industry to the service-oriented structure. The key question then becomes: how to reduce users' unfit costs in practice? Most user firms have existing IT systems in house, such as the legacy systems. When a new software application is needed, but it is not customized to fit the user's existing IT components, the user has to incur some extra efforts to make them run smoothly together. For example, a hospital that is using a new, un-customized PACS (Picture Archiving and Communications System) has to run an additional software module to convert the output/ input between the hospital's in-house research information system and the new PACS application. Such efforts represent users' extra unfit costs, which may be reduced in multiple ways. On the vendor's side, they should create applications that are most compatible with other systems and programs, by writing the applications using an open language (for example, XML), in a proper modular structure, and with a loosely coupled interface with other applications. All these strategies would help to increase the application's ability of being compatible with other systems. In addition, industry-wide adoption of software standards and protocols, once achieved, will also make the communications and cooperation across different applications easier and thus would reduce the unfit costs. In reality, some efforts have been made in this direction. In January 2006, Salesforce.com developed and launched AppExchange, an online marketplace for on-demand business software. It allows Salesforce.com and other software providers to establish across-application integration and therefore provides users seamless extension of their existing systems [2] [12]. Users expect

to have reduced unfit costs because a uniform platform eases collaboration across software applications.

Our findings suggest that the SaaS vendors can also enhance their competitive advantage through the appropriate contracting strategy. The nature of the SaaS business arrangement endows the vendor significant lock-in power so that the vendor is able to exploit its existing users to reach high profitability. However, unlike some of the classical results on 'lock in' pricing [4] [8] [9] [10] [11], we reach an interesting, and somewhat counterintuitive result: under certain market conditions, when the users' exiting costs decrease below a certain level, the SaaS vendor's market share increases and its profit improves. In other words, the vendor may find it optimal to help users switch out at low cost to them, and therefore it should strategically reduce users' exiting barriers instead of increasing them. In specific, we show that it is never optimal for the SaaS vendor to exert full lock in power in the competitive marketplace. The fear of being locked in by an outside provider will drive users away from the SaaS. This challenges many of the existing contracts in the SaaS market. To negotiate an exit strategy has been an important part of the software contracts [3]. Users are looking for a smooth and quick exit in order to avoid being locked in by one outside provider, but many vendors deliberately increase users' switching barriers in order to obtain lock-in advantages. We suggest that under reasonable market conditions, the interest conflict between the SaaS vendor and its users in fact does not exist. The vendor should cooperate and help their users to have easy exits. Hence, we would suggest the contract designed in the way that the vendor does not charge users high cancellation fees, assures users that they can get their data back intact once they decide to exit the contract, and cooperates and guarantees the data transition done in days or hours. The increased attractiveness of a contract that requires no lengthy commitment and ensures easy exits allows SaaS vendors to draw users who otherwise might opt for the traditional COTS software, and finally increases the SaaS profitability.

To conclude, the future prosperity of the SaaS business model requires both technological and managerial efforts. As the technologies further improve to bring a new set of software applications, and as the software vendors improve their understanding of the market and therefore take economically optimal contracting strategies, we expect to see more software will be delivered as a service.

References

- [1] Bednarz, A., "Manufacturers Eye on On-Demand Software", *Network World*, Apr 24, 2006
- [2] Cowley, S., "Salesforce.com Makes Platform Move with

AppExchange”, *InfoWorld*, Sep 2005

[3] Drummond, M., “The End of Software as We Know It”, *Fortune*, Winter 2002

[4] Farrell, J., and Shapiro, C., “Dynamic Competition with Switching Costs,” *RAND Journal of Economics*, Vol. 19, No.1, Spring 1988

[5] Garner, R., “Software for Hire”, *CRN*, Nov 1, 2004

[6] Hamm, S., “SAP Gets On-Demand Religion”, *Business Week*, Feb 2, 2006

[7] Kaplan, J., “Sorting Through Software As A Service”, *Network World*, Nov 21, 2005

[8] Klemperer, P., “Markets with Consumer Switching Costs”, *The Quarterly Journal of Economics*, Vol. 102, No. 2, 375-394, 1987a

[9] Klemperer, P., “The Competitiveness of Markets with Switching Costs”, *The Rand Journal of Economics*, Vol. 18, No.1, 138-150, 1987b

[10] Klemperer, P., “Price Wars Caused by Switching Costs,” *Review of Economic Studies*, 56, 405-420, 1989

[11] Klemperer, P., “Competition when Consumers Have Switching Costs: An Overview with Applications to Industrial Organization, Microeconomics, and International Trading,” *Review of Economics Studies*, Vol. 62, 515-539, 1995

[12] Kuchinskas, S., “Salesforce Finally Ships AppExchange”, *Ecommerce*, Jan 17, 2006

[13] Lacy, S., “The On-Demand Software Scrum”, *Business Week*, April 17, 2006

[14] Niccolai, J., “Gates Memo Puts Services at the Heart of Microsoft”, *Network World*, Nov 2005

[15] Pallatto, J., “IBM Recruiting ISVs, Partners to SaaS”, *Channel Insider*, Feb 23, 2006

[16] Vara, V., “Web Services Face Reliability Challenges”, *Wall Street Journal*, Feb 23, 2006