ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΟΙΚΟΝΟΜΙΚΗΣ ΕΠΙΣΤΗΜΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΑ ΕΦΑΡΜΟΣΜΕΝΑ ΟΙΚΟΝΟΜΙΚΑ ΚΑΙ ΧΡΗΜΑΤΟΟΙΚΟΝΟΜΙΚΑ

ΤΙΤΛΟΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ:

«ANALYSIS OF SOFTWARE PRICING STRATEGIES»

ΦΟΙΤΗΤΡΙΑ:

ΔΑΡΕΙΩΤΑΚΗ ΘΕΟΔΩΡΑ

Διατριβή υποβληθείσα προς μερική εκπλήρωση των απαραίτητων
προϋποθέσεων για την απόκτηση του Μεταπτυχιακού Διπλώματος Ειδίκευσης

Αθήνα
Μάιος, 2009

Εγκρίνουμε τη διατριβή της Δαρειωτάκη Θεοδώρας

Υπεύθυνος Καθηγητής
ΒΕΤΤΑΣ ΝΙΚΟΛΑΟΣ
Οικονομικό Πανεπιστήμιο Αθηνών

Εξεταστής Καθηγητής
ΜΠΑΛΤΑΣ ΝΙΚΟΛΑΟΣ
Οικονομικό Πανεπιστήμιο Αθηνών

Εξεταστής Καθηγητής
ΖΑΧΑΡΙΑΣ ΕΛΕΥΘΕΡΙΟΣ
Οικονομικό Πανεπιστήμιο Αθηνών

13/05/2009

TABLE OF CONTENTS

SUMMARY

More than in many other sectors of the economy, the software industry is constantly under the pressure of emerging technologies and new business models. At the same time, software is becoming more pervasive, entering every aspect of not only businesses and enterprises, but also every day life. The worldwide software industry has been generating huge revenues annually and continues to expand in revenue volume. In a highly innovating area, the key players constantly adapt their strategies and develop viable business models enabling them to generate profits.

The purpose of this MSc. thesis on *Applied Economics and Finance* is to analyze the *strategies* and *practices* that enterprises from the software sector employ in order to price their products and boost their sales. We focus on the business characteristics of software, pointing out the zero reproduction and distribution costs of such an immaterial good along with the high learning and switching costs, the short life cycles and the presence of network effects.

After illustrating the complementary and sometimes misleading role of the cost-based pricing schemes, we concentrate on the value-based schemes, where unit-price is related to the license-duration, the number of users, the capacity of processors or the actual usage of the software product. We describe open source and ad-supported software that have changed the traditional software economics and explain the alternative ways of making profits through the distribution of free software, in which case revenues come from complementary sales or services.

We discuss various price-related practices that are suitable for penetration in the software market or for increasing the market share. We describe practices that help a firm to extract the maximum possible surplus from different customer segments starting from the "crème de la crème" and moving down to the low-value customers. Bundling several products together may also increase under specific conditions the revenues of a firm. We study versioning as a second and third degree price-discrimination technique, and quote on the optimal number of versions that maximizes profits. We examine the switching costs that a firm deliberately imposes on customers and the benefit of a firm that has locked-in customers. Finally, we illustrate various ways of discounting as a means of retrieving cash flows and rewarding loyal customers.

After analyzing the software economics, we study the Greek vendors and examine their position in the international and European software sector. As we expected, Greece is not a key player in the software sector; however, the analysts are optimistic and foresee a growing development that will reduce the loss caused by digital piracy and will bring more revenues provided that the Greek companies will continue to be flexible and keep up with the latest software and pricing trends.

Our study intends to carefully and thoroughly examine the possible pricing options that a software company can employ in order to become established in the software industry and increase its market share. The strategies and practices mentioned here can be adopted to fight competition and to ensure a flexible presence in a quickly changing competition field. Until now, the software industry has proven itself dynamic and lucrative. But it now faces a serious economic threat, exacerbated by today's challenging economic times, focused on the

wild and win-obsessed world of software pricing. The identification and evaluation of the best market segment and the delivery of a superior customer experience is the key to survive in times of economic crisis.

1 Introduction

The worldwide software industry has been generating huge revenues annually and continues to expand in revenue volume. The software industry is increasingly acknowledged as a key driver of both social and economic growth. In a highly innovating area, incumbents had to adapt their strategies and develop viable business models enabling them to generate profits. The fundamental changes in the software industry have subsequently attracted the interest of analysts, business people and researchers of economics and management science. The purpose of this MSc. thesis on *Applied Economics and Finance* is to analyze the strategies that software companies have been employing in order to price their products and the practices that they have been exploiting to increase their sales.

In section 2 we describe how the software market was detached from the hardware market, and then, in section 3, we focus on the special features of software which differentiate its analysis from that of other goods. After describing the goal of a pricing strategy in section 4, we analyse the various pricing strategies that have been adopted by software companies. In section 5 we present the cost-based approach and then, in section 6, we proceed to the main objective of this analysis, the examination of the widely used value-based strategies; we discuss them in detail, mentioning the major differences between them and the most suitable situations where each of them can be applied. In section 7, we continue by presenting the price-related practices that firms are using in order to extract the maximum of the consumer's surplus. For each case, we quote on several corresponding examples of established software firms. Sections 8 and 9 conclude our analysis, by summing up the previous mentioned techniques, and by shortly discussing the future of software pricing.

Hopefully, this analysis will be helpful to people who are interested in software economics and want to be informed about possible pricing schemes that can boost their product sales.

2 Genesis of software market

In the very first years of computer industry, computers were distributed as complete systems; hardware, software, training, maintenance and support, were perceived as a single market. All components were produced by the same manufacturer and they were then bundled together and sold as one unit. At that time, computers were very expensive and only a few government agencies, large corporations and research centers were wealthy enough to lease or buy such electronic equipment. Most companies (among them IBM, which was the dominant vendor of mainframe computers) were developing software that was optimized for their hardware but was incompatible with other companies' hardware. So, the overall production and support of all computer-related components were handled by a single provider, resulting to a very wide computer market.

During the 1960s, a few pure software companies were founded, primarily to provide software to the government. There was little incentive for mainframe users to go to someone other than their software vendor – and indeed, the hardware vendors actively discouraged customers from acquiring third party software (or hardware). Thus, the *Independent*

Software Vendors (ISVs) were limited to niche markets that the mainframe vendors could not or would not service. However, that all changed at June 23, 1969, when IBM, under pressure from pending antitrust litigation by various competitors and the U.S. government, announced that they would unbundle much of its software and would price that software in a separate manner from its hardware and services.

Since then, the production of software is considered to be a separate market in the ICT (Information and Communication Technologies) industry. Later, companies started to develop interfaces and standards, so as to allow the communication of their hardware with any compatible software, whoever the provider was. Thus, the need for common manufacturer was eliminated and software sales were eventually detached from hardware sales.

3 Business characteristics of software

Software products have several characteristics that do not apply in usual material goods. Before making our analysis on pricing, it is necessary to point out the basic characteristics of software; these special features justify why the software market has already attracted special attention.

Software can be obtained and placed in use easily, regardless of time and place. It can be downloaded on demand and set up by using automatic installers that minimize the human effort. Messerschmitt and Szyperski (2003) point out that software is an *immaterial* good and needs no storage costs. In most cases it can be accessed instantly and with very low cost, as for example an Internet or Intranet connection. Software can be updated or enhanced simply by accessing a remote host and downloading the necessary bit stream.

The market of software presents similarities with the market of information goods, as both their products are costly to produce but very cheap to reproduce (Shapiro and Varian, 1999; Shefer, 2007). Until the fist copy of the product is released, all costs are sunk, meaning that they cannot be recovered in the unwished case that the process of development is interrupted. Usually in the software industry, the undertaken costs come up to huge amounts and companies search for sponsors in order to recoup their expenses. After the first copy is released, the marginal cost of producing and distributing the next copies tends to zero, since the production of another copy is simply done by burning of another media or by downloading from an Internet site. So, due to the tremendous *economies of scale* that characterize the software, every single sale ideally leads to pure profit. Moreover, it can lead to unlimited profit since there is no natural capacity limit for additional copies or downloads. *Distribution costs* are also minimum; being digital, software can be delivered electronically, while most of the bandwidth needed can be obtained at close to zero marginal cost. In any case, these minimum distribution costs are paid by the end customer.

In addition to the first-copy sunk costs, *marketing* and *promotion costs* are quite high for most software products. From the moment that a software manager decides to go with on-site visits of sales representatives, the cost of promotion increases enormously. Products that cost less than €5,000 are most efficiently sold over the web or through channels. A purchase of this size is within the decision authority of middle managers, so there is no need for on-site visits to close a deal. More expensive products require sales representatives and on-site visits but have to produce enough profit to support this type of sales effort. That's how the "Death Zone" of $5000 - $20,000 emerged, where software that is priced in this

range is hard to be sold profitably[1] (Shefer, 2007). Also, nowadays software products are not stand-alone isolated installations. They interact with other applications and exchange data in order to produce their output. As a result, the consequential *integration* and *training costs* may reduce the economies of scale.

Zero-marginal cost and unlimited profit appear ideal, but things change when another important characteristic of software is taken into account. That is *short life cycles* (Shefer, 2007). By the time a company has sold enough copies of its product, it has advanced on to a newer version or the market conditions have changed or both. In addition to short life cycles, well-established software companies face another risk related to research and development. Not surprisingly, past experience has proven that innovation may come from emerging companies, which exploit the advances of information technology and gain ground in the battle of software. So, unlimited profit is not actually a feasible anticipation.

The innovative companies should face high acceptance and high sales volumes during the initial period that their products enter the market. However, there is a tradeoff coming from the demand side, which does not allow them to enjoy the lack of competition. Customers are aware of the fast evolution and the strong competition that will follow soon after the introductory period. They also know that sooner or later the newer technology steps will force prices to fall. They exploit their *bargaining power* and as a result, they are willing to hold their orders for a while to avoid paying the full price, and thus they diminish the advantage of innovative companies (Cusumano, 2007).

Software also takes advantage of the *economies of scope* (Cusumano, 2004). In order to produce a greater variety of outputs, a software company needs fewer inputs, such as effort and time. Also, by jointly producing different outputs they can achieve greater business value, since by producing each output independently they fail to leverage commonalities that affect costs. Good architecting and business planning may increase further the economies of scope. Moreover, software companies may exploit the *economies of aggregation*, by bundling software products in order to become more competitive in terms of cost and functionality.

Another characteristic of software is that it is an *experience good* – it needs to be used before it can be evaluated (Shapiro and Varian, 1999). Product characteristics are difficult to observe in advance and they can only be ascertained upon extended use. The metrics that estimate the quality of a software product are bound to be an abstract summary and because of the product differentiation it is very hard to find a common metric system that can be applied to any software product. In addition, as for all experience goods, transaction costs and cost of uncertainty cause significant switching costs.

In the software field, having just a superior technology is not sufficient. Software market is strongly affected by *network externalities* (Katz and Shapiro, 1985). Users that are already familiar with a specific application get more utility by continuing to use the specific application and they avoid the extra training that they would need in order to switch application. Moreover, standardization for attributes such as file-types, helps users communicate more efficiently if the software they are using is compatible to the established standards.

---

[1] See the discussion on http://discuss.joelonsoftware.com/default.asp?biz.5.210074.3 regarding the "Death Zone". Some claim that it ranges between 5.000 and 20.000 while others believe that it ranges between 1.000 and 100.000.

Finally, the high switching costs that are imposed by the installation, customization, interoperability and training may create strong bonds between customers and specific software suites, which sometimes may be so strong that the customers may get *locked in* to specific brands or even to obsolete technologies (Cusumano, 2004; Shapiro and Varian, 1999).

4 Goal of a pricing strategy

Having emphasized on the special characteristics of software, we can now proceed with the analysis of various pricing strategies that can be used in order to price software goods. According to Kortge and Okonkwo (1993) the goal of a pricing strategy is to *"assign a price that is the monetary equivalent of the value the customer perceives in the product while meeting profit and return on investment goals"*. In the following sections we will present and categorize the various strategies and will focus on the situations where each of them is more efficient.

We can distinguish two major pricing categories: the traditional *Cost-based strategies* that aim at pricing at a level that ensures that the company will be able to recoup the costs involved and the more profit-seeking *Value-based strategies* that aim at capturing most of the customer's value for the software product. In the following sections, we will examine these two methods and see what purpose serves each of them.

5 Cost-based pricing strategies

Managers in the software industry have traditionally developed their pricing strategies by overemphasizing cost-related criteria. Cost-based software pricing is historically the most popular method since it relies on more readily available information from the cost-accounting system. Cost-related information is easily measured and handled by accountants, financial, marketing and product managers who are using the price of the software product to yield a desired return on allocated costs (Harmon et al, 2004).

This strategy ignores the value of the customer and focus on quantitative volumes. Because of its computational simplicity, it is generally more suited to businesses that deal with large volumes or which operate in markets dominated by competition on price. The price of software services are constructed as simple functions of the existing cost accounting records of the company. This is a top-down method of constructing prices that starts from the existing situation and attempts to distribute the costs to the products supplied.

In economic theory textbooks (Βέττας and Κατσουλάκος, 2004; Cabral, 2000) a perfectly competitive market will generate marginal cost pricing, but given the high average cost of software production and the near zero marginal cost of software distribution, marginal cost pricing, i.e. the *first-best* pricing solution, is commercially unsustainable. So, the next possible sustainable solution is a price equal to the marginal cost plus the share of the fixed costs, i.e. the *second-best* pricing solution. In some cases however, as for example in the *open source* case, even this solution is not sustainable. Subsequently, according to cost-based pricing, the price should be calculated as:

Price	=	*Variable cost of producing one unit*	+	*Variable cost of selling (distribution and other unit based cost)*	+	*A share of the fixed costs*	+	*A fair profit per unit*

As we have already explained, in the software economy, the variable costs of producing another copy are very low (perhaps negligible). Moreover, the demand curve and the price elasticity are hard to be estimated, because of the constantly advancing technology and the short life cycles (Shefer, 2007). And finally, the intellectual effort and the research and development, which are taken into account in the fixed costs, cannot be calculated based on tangible metrics such as labor hours.

If all costs could be accounted and the product was priced above that level, then it would be guaranteed that this strategy would be profitable. History has proven that basic economics are not sufficient when pricing software. In 1976, Wang Laboratories invented the first commercial electronic word processor. They used cost-based pricing to come up with a price for this revolutionary product. At first, their product was a big hit. The problem arose a few years later when in the early 1980's personal computers started to offer word processing capabilities. As PC-based word processing became more popular, Wang sales slowed. According to cost-based pricing rules, they raised the price of their product even while their competition was reducing the cost of their products. Their pricing eventually drove away all of their customers, because such a strategy causes product over-pricing when there is a weak market[2].

Someone may now wonder why cost-based strategies are still in use if they are so vulnerable. The answer is that they should affect the decision of *whether the product is worth developing* in the first place, and not its actual price. The price determination will involve value-based strategies, which will be analyzed in the following section. In other words, the use only of cost-based strategies will definitely lead to a product failure. On the other hand, if they are not used at all, there is a hazard that the costs may not be recouped.

6 Value-based pricing strategies

According to Harmon et al (2004) the key to *value-based* pricing success is the recognition that the price the customer is willing to pay depends on the customer's value requirements, not the vendor's. The customers make judgments about benefits and prices and choose those products that maximize their perceived value. Revenues come from capturing more of the customer's value, and if they exceed the product costs, then the product's introduction to the market should be pursued.

Several techniques have been proposed that are trying to capture the customer's value. They are classified into the following categories[3]:

[2] The Wang case is further discussed by Dr. Jim Anderson on his article "Why Product Mangers Need To Know That Cost Plus Pricing Is Wrong, Wrong, Wrong!" that can be found at *http://www.theaccidentalpm.com/pricing/why-product-mangers-need-to-know-that-cost-plus-pricing-is-wrong-wrong-wrong*

[3] An early classification of the value-based strategies is provided by Harmon, Raffo and Faulk (2004)

6.1 Duration-based

The pricing techniques that are based on duration may be divided into *flat pricing* and *subscription licensing*.

6.1.1 Flat pricing

In *flat pricing* –also known as *fixed pricing* or *perpetual licensing* – the licenses are paid on a one-time basis, giving the users the right for unlimited use of the software product (Shefer, 2007). The perpetual license is usually restricted to a particular *user* and/or *machine*. It refers only to the product's acquisition and does not imply any right to get the upgrades which will be released in the future; these updates are typically sold separately as part of a maintenance agreement or on a per-upgrade basis.

From the customer's point of view, this pricing method is quite attractive when the licensed software is not highly priced, or when there is a significant discount in the total charge. The reason is that all the fees are paid up-front, so the customers should be rather confident that the selected software will fulfill their expectations, since once bought, the license cannot be resold to another customer without the permission of the producer, and thus the expenses cannot be recouped[4] (Choudhary et al 1998). Apart from the uncertainty regarding the quality of the purchased product, the flat pricing implies some risks that are driven by the definition of the term *"perpetual"*: What does actually mean *perpetual* in a field where the market conditions and the technology advances are rapid? What if the vendor ceases to exist one year after the purchase is done? What if the product is discontinued and a newer, enhanced version is released? We expect similar products to be highly priced when produced by well established companies. In other words, the list price signals the firm's expectations for future presence in the market.

By selecting flat pricing, a customer practically locks himself in a specific vendor. Since there is no way to recover the sunk costs of the product acquisition, the customer has less incentive (of greater switching costs) to replace software from that vendor.

Flat pricing enables customers to predict what they are going to pay for the purchase of a software product. It is thus easier for their managers and accountants to estimate the *Total Cost of Acquisition (TCA)* and compare it with the *TCA* for other competitive software products before making their final decision. However, this approach is not quite flexible when the customer needs change; if the customer's company grows or shrinks frequently, then they will not be able to purchase the optimal number of licenses.

From the vendor's point of view, up-front revenues are generally preferable to future cash flows, since they depreciate immediately a large amount of the sunk, development cost and they have more money available to be invested in research and development. However, there is an uncertainty regarding the future revenues of the company because of existing customers being excluded from any repeated sales (Shapiro and Varian, 1999).

If vendors were to provide a perpetual licensing scheme, they should be willing to undertake any contractual obligation that perpetual licensing creates in order to maintain that product forever, or they would be very careful to make clear what "perpetual" stands for. Nowadays, no licensor wish to be locked into "perpetuity". The support of legacy systems is

[4] See also in this section the discussion on Computer Software Rental Act in 1990.

usually a heavy burden, as it brings fewer revenues than expenses – in time and effort (Bontis and Chung, 2000).

As we already mentioned, flat pricing helps locking-in customers, which in turn helps increase the vendor's market share. Locked-in customers will constitute an installed base open to promotion of the vendor's complementary products (Shapiro and Varian, 1999).

However, this model lacks of flexibility in setting a different, customized price for each customer, based on individual's perceived value. Vendors are better off when it is not easy for their software price to be compared to the price of the competitive products; because of flat pricing being transparent, competition is expected to be strict. The transparency of this scheme is usually exploited when a new firm is about to enter a market. Then, flat pricing, usually along with significant discounts, act as a penetration strategy for new entrants in the software market.

In addition, perpetual licensing is prone to *digital piracy*, leading to significant revenue linkages. Thus, flat pricing is a pricing technique with moderate results and therefore, we would expect the vendors to avoid it. The truth is that this technique, compared to others, is not very profitable by itself. However, when it is combined with separate fees for support, maintenance and upgrades, then it becomes easier for the vendors to make up for lost revenues. For example, IBM offers such type of licensing for software installed in distributed environments; this is called *"One Time Charge (OTC)"* which is combined with the complementary fees of *"Subscription & Support"*. Subscription & Support is an optional[5], annual maintenance charge, that provides customers access to IBM's technical support and enables them to obtain version upgrades at no charge.

A special case of flat pricing is *site licensing*, where customers get unlimited use of a product across their enterprise locations while paying a flat fee. Customers generally like this choice since they don't have to worry about counting seats. However, when the use of the product is limited, a site license may not justify its cost. As far as the vendors are concerned, this scheme causes them to lose their ability to track the number of installations at the customer site, and as a result, if the product is successful, they are loosing money.

6.1.2 Subscription licensing

In *subscription licensing* –named also *term licensing* when the license holds for a specific term, or *Pay As You Go*– the customers do not own a lifetime license, but instead they rent the software for as long as they use it. If the rental fee is not paid, the software stops working (Shapiro and Varian, 1999). Another term that is used for subscription licensing is *renting*. As Choudhary et al (1998) explain, there is a slight difference between these two terms, regarding the duration of the subscription. If there is a time limit shorter than half of the product's life, then the product is considered to be *rented*; otherwise, when there is no time limit or the limit is longer than half of the life of the product, it is considered *licensed*.

Software is generally considered as a *durable* good (Shapiro and Varian, 1999). A very important distinction that has to be made is that unlike other durable goods, software cannot be resold, distributed or rented easily. In fact the passage of the Computer Software Rental Act in 1990[6] made it illegal to rent software without the copyrights' owner specific permission.

[5] except for the first year where it is obligatory and thus it is included in the product's price

[6] See a report on Computer Software Rental Act of 1990 at http://www.copyright.gov/reports/software_ren.html

So, there cannot exist an organized market for used software, and that led to the development of the rental market of software.

Renting software has a number of advantages for both the customer and the vendor. First of all, Choudhary et al (1998) show how renting can be used as a way to price discriminate in a socially desirable way. By rejecting the "all-or-nothing" concept that lies behind perpetual licensing, renting software benefits the consumer by allowing partial use and thus increases the consumer surplus. Consequently, by renting software the vendor's profits are increased.

If both perpetual and subscription licensing are available, the consumers would rent the product rather than buy it if they intend to use it for short term projects and are not going to use it thereafter, or if they want to evaluate the software and, depending on its usability and performance, would later decide whether to buy it or not. Subscription licensing relieves the customer from the up-front investments and reduces the switching costs (Cusumano, 2007). From the vendor's point of view, the low switching costs may be exploited in order to penetrate to the software market. However, as Choudhary (2007) points out, the ability to switch across providers increases the bargaining power of consumers and can also break the stability of a provider's customer base. On the other hand, vendors may be benefited by the fact that subscription licensing is more powerful as far as software piracy is concerned, since hacking the license key of a program may easily be faced by blocking the old key and producing a new legitimate one[7].

The major representative of subscription licensing is *Software as a Service (SaaS)*. In this case the software is running on the premises of the software provider and is accessed through Internet. So, the customer is relieved from the burden and the cost of installing, configuring and maintaining the application. This approach minimizes the switching costs, since the customer has not invested on any infrastructure, and is less reluctant to change software provider (Robinson, 2006). There is however a potential drawback: data security issues emerge since sensitive corporate data are stored and administered remotely.

Another important benefit for the customer has to do with the release of updates (Choudhary, 2007). In perpetual licensing, when a new version of the software becomes available, customers who have previously purchased the software have a choice between continuing to use existing software or spending more money to purchase the new version. Customers do not upgrade unless the new software provides substantial incremental benefits relative to the previous version. For instance, although Microsoft has released newer versions since Windows 98, a large number of users have not yet upgraded. Microsoft tried to end support for Windows 98 in January 2004 but the public outcry following that announcement forced Microsoft to reschedule the cutoff date to June 2006. According to Choudhary (2007), users upgrade to a newer version only if there is sufficient benefit to them that exceeds the cost of upgrading. The problem now is that, because upgrades cannot be released every now and again, the customers may wait longer that they would want in order to buy the release with the enhanced features. It took Microsoft more than five years to develop Windows Vista and the new features in it. However, because all of these features are bundled together in Vista, users had to wait for the release of the completed operating system. So, there may be

[7] See Choudhary (2007) regarding the bargaining power of buyers.

features that Microsoft developers had finished developing in 2002 that were not available to end users until the complete operating system was released in 2007.

In contrast to the perpetual licensing model, which relies on sales of perpetual licenses and upgrades, in the SaaS model there is no competition between the present and future versions of software. As a result, the publisher does not need to hold back new features for the next version and customers may have a higher willingness to pay because they expect to receive further enhancements to software features. SaaS alters a vendor's incentive to invest in software development by allowing the vendor to release new features as soon as they are finished and make them available to all customers. Even if the subscription licensing is quite attractive, the study of PriceWaterHouseCoopers (2007) showed that it has an important side effect as far as the vendors' cash flows are concerned. By moving from up-front license fees to periodic payments, vendors don't have the same level of reserves for research and development, and they will probably need to wait for more until they launch the next generation of their products. Choudhary (2007) contradicted this opinion and claimed that given the convex cost of developing software quality, the publisher will invest more in software development under SaaS than under perpetual licensing –even though he also expected the opposite– resulting to better product quality.

Between the early successful adopters of the SaaS model is Salesforce.com, which offers on-demand customer relationship management software solutions built on its infrastructure and delivered directly to users over the Internet. Salesforce.com does not sell perpetual licenses but charges a monthly subscription fee per user.

Controller's Report (2004) points out that subscription licenses are not the best choice for every situation. Scott Hicar[8], CIO of Maxtor Corp., a hard-drive manufacturer, had experience with subscription licenses from Salesforce and RightNow Technologies and regarded subscriptions as easy, flexible, and low risk. Even so, Hicar's last two software deals had perpetual licenses. The reason is that Hicar believes perpetual licenses provide a higher *Return On Investment (ROI)* for software that a company uses for more than a few years. Many analysts say that there is a major fault in the subscription model: It often requires companies to pay for software upgrades that don't arrive during the life of the contract. This, however, is similar to the well-know problem with perpetual licenses: paying for more software than needed.

According to the Aberdeen Group study (2006) more than half of companies surveyed were either using SaaS or actively exploring its use. The momentum behind adoption of subscription and on-demand purchasing models is driven by *Small and Medium Enterprises (SME)*, who cannot afford large perpetual costs fees paid up-front. The tendency towards Subscription licensing is verified also by the survey held on 857 enterprises, by McKinsey & SandHill (2008).

[8] On *InformationWeek*, "*Software: The Subscription Pricing Model*"

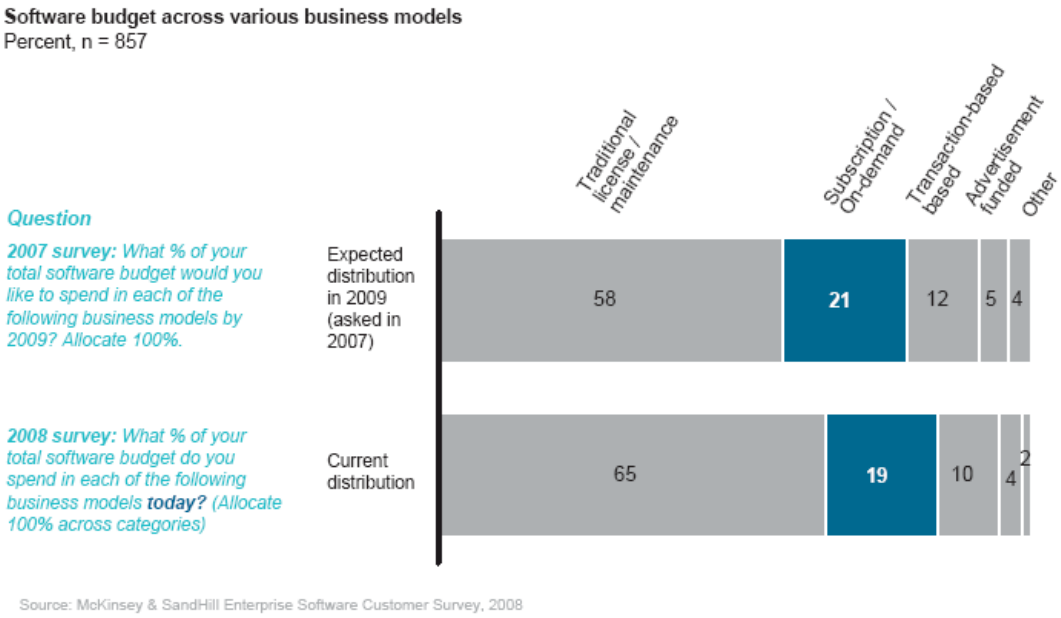

Figure 1: Increased tendency towards subscription licensing

In addition, the Economist (2006) estimates that the market for SaaS is growing at 50 percent each year. What is interesting is that even some large on-premise vendors have introduced SaaS-based licensing policies: Microsoft offers the *Enterprise Subscription Agreement* to business users at an annual rate or a three-year term. Other software vendors like IBM offer a *Fixed Term License* that grants the use of the software over a limited time period (typically 12 months). Sun, Oracle, SAS Institute and Computer Associates also offered their own subscription licenses on their major products. Despite the various forms of business models, all those vendors aim at generating perpetual revenue streams by transforming such a durable good as software into subscription-based services.

6.2 User-based

Apart from the duration, subscription pricing may involve the number of users that are able or actually connect to the system. Software companies consider that the customers' utility increases with the number of users that can work either from the same or from different workstations. This estimation was reflected in the pricing models that were developed; Bontis and Chung (2000) classify user-licenses to *named user licenses, computer-based licenses* and *floating licenses (or concurrent users licenses)*.

Named user licenses permit users with explicitly declared credentials, to login to the software. *Computer-based* license are similar to named user, except that the license is assigned to a specific workstation and there is no limitation on the number of people that may access and use the software through that computer. *Floating* license is a pool of numbered licenses that are given to users according to the time of request. They are not computer-restricted, meaning that the users of all computers with physical access to the software can be assigned a license provided that the number of concurrent licenses is not exhausted. When a user releases the allocated license, then it becomes available for assignment to the next user who raises a request.

This model is attractive for the customers; it is simple in understanding and administration, since the predefined fees do not imply any hidden costs. It also provides a clear method to managers when evaluating different possible licensing options and to accountants when recording the cash outflows.

However, the hazard of user-based pricing is that customers may pay for software more than what would justify the software's value. *Named user* license is the least flexible scheme. It demands purchasing of an individual license even for people that do a minor use of the software. This scheme is suitable for stable environments, where users do not come and go very often, as this would cause also a huge administration cost. *Computer-based* licenses face the same hazard of underutilizing the software, and moreover, it requires dedicated workstations from where the user can work. The most flexible scheme is the *floating* license, which allows access to all computers within a corporate site. Hence, there is a trend to charge a premium (even double) over named or computer licenses. In addition, it requires a management system for the license allotment. Customers must predict the number of desired simultaneous users in order to properly size the licensing agreement. So, all these price structures charge customers based upon their *peak* user predictions (concurrent or not) and not upon their average number of users. As a result, a discrepancy between pricing and value realization may occur.

An advantage for the vendor's point of view is the simplicity of this technique, as they gain well defined, guaranteed cash flows in periodic time intervals[9]. Many companies provide all the previously mentioned licensing options and let the customers evaluate them and choose which can cover their needs. In other words, they apply a second-degree discrimination by offering various schemes and letting the customers to self-classify.

One of the key problems that software vendors are confronting with this technique is that nobody knows how to define what a "user" is anymore. With outsourcing, a user could be someone outside the corporation (Harmon et al, 2004; Sullivan and Schwartz, 2001).

In general, user-based pricing is suitable for organizations with discrete and countable user populations. For uncountable populations, the following scheme based on processor features is preferable.

6.3 Processor-Based

In the very first years of computer industry, the powerful central systems concentrated all computational complexity and the users were accessing data through dumb terminals. The need for computational power was growing constantly: more powerful processing units leaded to more greedy software which then demanded for even more powerful processing units etc. As software architectures evolved, the burden of computation was redistributed between several architectural tiers. When the *n-tier* architectures bloomed, in the late 1990s, the databases were separated from applications and often run in a distributed fashion on much less expensive application servers. With the Internet dominance, the heavy client was replaced by a web client, which –by its nature– can be anonymous. Once companies exposed their ERP[10] systems to the Internet, to do direct sales, to work with suppliers and to tie

[9] Choudhary (2007) is comparing perpetual licensing to software as a service regarding the money investment, the data security and the time of delivery of new features

[10] Enterprise Resource Planning

together their employees around the world, it got much trickier to figure out where software was really running and what a company should be charged to use it.

To respond to the new trend and to better control the pricing procedures, the software vendors started charging their software based on the *number of processors* that the customers had installed on their hardware or on the number of those processors that the customers had been operating. In order to change their pricing strategy, vendors often claimed that powerful systems increase the user's productivity and the *Return On Investment (ROI)* of the corporation. It is commonly believed, though, that vendors had decided to focus on getting revenues primarily from the heavy servers and secondarily from the anonymous and sometimes untraceable clients. Operating systems, middleware programs, and application programs started them being available on per-CPU licenses. This seems more or less fair, until someone realizes that not all CPUs are created equal.

When single core processors approached the physical limits of possible complexity and speed, a major achievement in the hardware area has been made. That was the placement of more than one processor-core to the same chip, resulting to multi-core[11] processors with higher capacity. Thus, a customer could have the same processing power but with fewer processors. This achievement was booming for hardware companies, but disastrous for software companies. Customers were now buying powerful machines with fewer processors than before, causing the profits for software companies to decrease. Hence, changing the pricing policy of software was more a necessity than an option.

As we said before, the main concern of software companies was their inability to count or estimate the number of distinct users who access an application through the web. The per-processor idea was not perfect, but seemed an attractive solution. This led the software companies to develop a new pricing policy based on measurable features such as hardware specifications that would deal with the uncountable population problem. The new pricing technique was based on the number of *cores* that were embedded in the same processor-chip. Still, there were some issues regarding the fairness of this method; not all cores performed equally well, so they could not be considered of equal value. After performing an evaluation of processor-cores developed by various manufacturers, the pricing policy was formed to reflect the processor's capability. This policy is highly used in today's businesses, although it is rather complicated; it has to take into account both the total number of cores in processors and their actual performance.

With this pricing technique, customers are relieved from the burden of counting seats. No matter how many the users are, given a specific hardware, the price of software remains the same. On the other hand, the added benefit to software companies is that they can gain more profit without spending any extra time or effort on software development (Harmon et al, 2004).

Software companies claim that this pricing policy is quite straightforward and that all costs are transparent and easily measured by customers. But customers disagree with them since there is no single list price for a component, there is no floor and ceiling for a price on a

[11] In multi-core processors, two or more processor-cores are attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks. For example, a multiple core set-up is somewhat comparable to having two separate processors installed in the same computer and ideally, is nearly twice as powerful as a single core. In practice however, performance gains are said to be about fifty percent, but this depends on the processor manufacturer.

piece of software, and any hardware update seems to penalize them because it causes the software price to rise (Bontis and Chung, 2000). Customers who want to upgrade the underlying hardware face a hazard of increasing significantly the cost of software without gaining or gaining very little utility compared to what they pay. Moreover, in the extreme case where numerous and demanding applications are installed on the same hardware, the user may not enjoy full capacity because of hardware overloading. Nevertheless, the cost of each application is the same as if the application was the only one running on the specific machine. In another extreme case, there might be a single application running on a very powerful machine, leaving it underutilized.

IBM was among the first software vendors to use a pricing plan for its mainframe computers that was based on the number of processors. With the advent of multi-core processors, IBM adjusted its pricing scheme by taking into account the number of cores and their efficiency (Robinson, 2006). To support this new scheme, IBM introduced the *Processing Value Units (PVU)*[12] metric, where each processor-core is evaluated and assigned with a score. The total score is used in order to calculate the final price of the software[13]. Of course such an approach benefits manufacturers of powerful processors. Indeed, to surpass the inferiority of its core processing capacity, Sun has launched *per-user* (instead of per-processor) annual subscription pricing for its *Java Enterprise System* operating system and middleware stack.

6.4 Usage-based

As we mentioned in the processor-based pricing technique, there might be unsatisfied customers who are reluctant to upgrade their hardware because of the unaffordable extra cost on software and the fear of underutilizing the new hardware. Until recently, 90 percent of servers market consisted of x86 servers, which hosted only one application per server. This practice left up to 80-90 percent of the systems capabilities underutilized.

The advent of *virtualization* gave a solution to hardware underutilization since it allows a partial use of resources. Virtualization techniques create multiple isolated partitions on a single physical server allowing an explicit and more efficient use of resources. With virtualization, the need to develop a *usage-based* pricing technique, compatible to the new architectural trends, emerged[14].

During June and July 2008, Penn, Schoen & Berland Associates performed a study commissioned by Hewlett-Packard, regarding the future adoption of virtualization in Small and Medium Enterprises (Hewlett-Packard, 2008). The study shows that 60 percent of technology decision makers are in the process of implementing and another 26 percent say

[12] For more information on IBM's *PVU metric* please refer to http://www-01.ibm.com/software/lotus/passportadvantage/pvu_licensing_for_customers.html

[13] In July 2006, IBM declared a similar policy for software such as DB2, WebSphere, and Rational tools based on PVUs, to take effect in November 2006. In this model, license fees are based on the specific underlying processor and its perceived power; a dual-core IBM Power5 chip is ranked at 100 PVUs per core. Dual-core x86 chips from AMD and Intel get a PVU rating of 50 per core, Intel Itanium dual-core gets a PVU rating of 100 per core, and Sun T1 (up to eight cores) get a PVU rating of 30 per core. Previously, IBM had been treating dual-core x86 chips with an explicit per socket pricing model. IBM will evolve its pricing scheme over time and presumably use benchmarking tools such SPEC CPU and TPC-C to evaluate processors. (http://developer.amd.com/documentation/articles/pages/9142006142.aspx)

[14] (Robinson, 2006) is quoting on Microsoft's and IBM's point of view regarding virtualization vs. multi-core processors.

they have already successfully implemented virtualization. Even if most of them are only in the beginning stages, global adoption of virtualization is steadily rising. Even if the study results were not as encouraging as they were expected to be, however, it is very likely that in the following years, virtualization will dominate. Respondents understood that virtualization cuts costs and increases efficiency (80 percent and 70 percent, respectively), which strengthen the possibility of adoption.

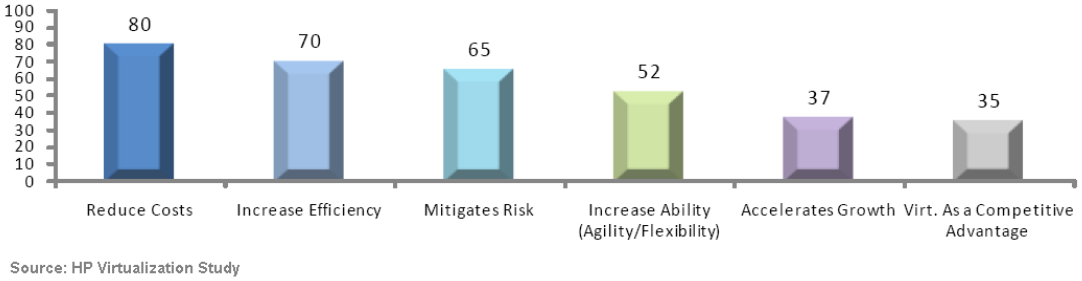

Figure 2: Key Business Outcome of Virtualization

In order to avoid penalizing customers who keep up with technology, a *usage-based* (also *pay-per-use*) technique, named *Sub-capacity pricing*, was developed. The effect of sub-capacity pricing is that customers only pay for licenses based on the number of processors that are assigned to the logical partition where the software is running. So, the product is priced according to the consumed processing power and not the total provided power of the underlying hardware, while hardware upgrades do not cause any price increase.

In addition to hardware issues, the usage-based technique came also to correct the injustice between light and heavy customers. This technique usually applies to web-based applications where the more customers use the application the more they pay. Assume that customers pay a fixed fee (or any non-usage based technique); if they do a moderate use of the provided service, then they pay more than they actually consume. On the other hand, heavy-use customers are taking advantage of the available resources and gain more than they pay. Thus, light users are subsidizing the heavy users (MacKie-Mason and Varian, 1995). In such cases, usage-based pricing is fair for both of them.

From the vendor's perspective, selling software at a fixed price increases the company's revenues by a specific amount. Sullivan and Schwartz (2001) state that with usage-based pricing, the total revenue is dynamic and non-predictable. MacKie-Mason and Varian (1995) argue that usage-based pricing may not be a good idea when they it is poorly designed. This depends on and whether the benefits they provide exceed the accounting and transactions costs. Accounting and billing demand an extra cost because the software vendor needs to have a mechanism to track down some usage metrics per user, so that the billing can be performed. On the other hand, costs may decrease because usage-based pricing increases the efficiency of the software product when it discourages the greedy use.

This technique may rely on various metrics, among which the most usual is the number of transactions performed or the number of users using the application (on average or at most). IBM for example, provides customers with the *Sub-Capacity Reporting Tool (SCRT)* for the generation of Sub-Capacity reports. These reports are based on various workload metrics

and are submitted to IBM each month in order to produce the corresponding *Workload License Charges (WLC)*.

According to Computer Associates, the pricing model would most benefit customers in a situation where they are growing their business or making changes to their infrastructure. For customers who are in a steady state, Stansberry (2005) say that it's not going to have a lot of effect.

Virtualization has been such a rapid market trend that the software vendors are still in react mode in terms of their pricing and licensing for virtualized environments and their support policies. Gartner (2008) predicts that software pricing and licensing will remain problematic for the near future. One potential roadblock according to Thibodeau (2004), is the need to test packaged applications on virtualized systems. Software vendors are often reluctant to troubleshoot their applications on servers running virtualization software and prefer smaller servers with independent applications. Hence, they are not investing in designing a usage-based pricing scheme. In the long run, though, application vendors may have little choice other than to make the adoption of server virtualization software as easy as possible for users.

6.5 Open source software

Open source software is a relative recent[15] trend that resulted by the attempts of the developers' community to build software in a public, collaborative manner. In order to be classified as open source, a software product must permit users to use, change, and improve the source code, and to redistribute it in modified or unmodified form.

Open source gathered so much attention mostly because it comes at no cost – this is the usual though not necessary case[16]. Open source model uses a variant of the *General Public License (GPL)*[17]; developers get access to the source code by paying a low/zero amount, and then they can modify and distribute the new version but without charging more than what they paid to gain access to the source code. This has the effect of freezing prices at a low/zero level, thereby excluding a "second best" pricing solution that would recoup some of the costs involved.

Apart from the pricing scheme, developers are attracted to open source because of the freedom it provides to customize a software product so that it fulfils the customer's needs. Faulty component can by corrected on the spot, eliminating the need of waiting for the next official release of proprietary products. There may be software products that *almost* do what one wants. For example, if the proprietary Microsoft *Office* does *almost* what one wants, then that's all what they get. Closed code cannot be modified immediately and there is no guarantee that the extra feature will be available in the next release. If the open source

[15] The free software movement was launched in 1983 but later in 1998 the term free software was replaced by open source software (OSS) and the Open Source Initiative (OSI) was formed.

[16] See the study *Χρήση Λογισμικού Ανοιχτού Κώδικα (Open Source)* of the Observatory for the Greek Information Society (Παρατηρητήριο για την ΚτΠ, 2007) for more information on the difference between free software and open source software.

[17] The GPL license requires that any subsequent development of the software must itself be distributed under a GPL-type conditional license, which is to say the source code must be made available in some form or other. This has been called the "viral effect" because it carries over to each subsequent generation of software development. For more information please visit http://www.gnu.org/

OpenOffice almost does what one wants, then they can modify it so it does exactly what they want.

There are plenty of reasons why *developers* are in favor of open source software[18]. What interests us in terms of software economics is the incentive for *corporations* to get involved to open source software market. The first to mention is that most open source companies did not have to pay the otherwise huge and sunk cost of development. So, they could take advantage of the extremely *low cost of development* from the contribution of anonymous developers who share the open source idea. There are many examples of companies that undertook no sunk initial development cost: *Red Hat*, for instance, did not develop the operating system *Linux*; *Covalent* did not develop *Apache HTTP Server*. So, since they were relieved from the sunk costs, they could get only revenues from their contribution.

There are also examples of companies that that were actively involved in the development and yet gave away their intellectual property. What seems to be leakage, in fact it helps produce revenues. When Netscape released *Mozilla* as open source in 1998[19] deferring further work to the open source community, it was because they saw an opportunity to lower the cost of developing the browser. By giving away *Mozilla*, Unix system vendors such as IBM, HP and Sun were left without a supported browser, which they needed to sell Internet-connected workstations. Thus, each of them assigned software engineers to work with the Mozilla project, to help keep the project moving forward, but mainly to assure that new releases are compatible with their particular systems.

Another –less common– reason why corporations are moving towards the open source software is to boost the sales in the hardware market. Firms try to commoditize a product so that they can then increase the sales of the complementary product (Shapiro and Varian, 1999). Software and hardware are classic complements. So, by *commoditizing* the software, firms would be able to gain higher profits in hardware sales. The companies that are established in both software and hardware market, are obviously most benefited by such a commoditization.

However, the attempt to commoditize software suffers from the fact that software is not interchangeable. Even when the price is zero, the cost of switching from the proprietary Microsoft *Office* to the open source *StarOffice* is non-zero. Until the switching cost becomes zero, desktop office software is not truly a commodity[20]; people have been accustomed to using a specific interface of an application and may not be very willing to replace it with even a more powerful or a free-of-charge alternative. So, only the act of commoditizing software is not a sufficient reason to choose open source.

Pure software companies exploit another form of complementarities. Giving away the basic product generates *demand for complementary products* that the donor continues to sell. This is a variant of the *"razor and blade"* model (Shapiro and Varian, 1999; Cusumano, 2007); according to this strategy, the basic software platform is given away and then the company charges for other parts of the product systems as well as for updates or more advanced versions. Adobe is following this model with its Acrobat product line: the company gives away Adobe readers but charges for the editing tools and server software. In our

[18] For a complete study on the developers' motivations please check the *Χρήση Λογισμικού Ανοιχτού Κώδικα (Open Source)* of the Observatory for the Greek Information Society (Παρατηρητήριο για την ΚτΠ, 2007).

[19] and terminated all internal development of it in July 2003

[20] For more details, see the discussion at http://www.joelonsoftware.com/articles/StrategyLetterV.html

beloved example, while Netscape open-sourced the web browser *Mozilla* (i.e. the client software), they kept on charging the Web Server (i.e. the server software). Very recently, in January 2009, they announced that the web server, named *JES Web Server 7.0*, would go open source, too. It was a tremendous and anxiously awaited donation to the open source community. However, from a technical point of view, it is not exactly the whole web server product, since the open-sourced code does not include some of the value-adding components such as the administration framework. Those components remain proprietary and continue to be charged.

A second variant of the *"razor and blade"* model is the *services model*; again the basic software product is given away, but the services such as installation, maintenance, training, and customization are charged. Every year IBM spends millions to develop open source software. The reason behind that move is that IBM is becoming an IT consulting company[21]. Most companies that leverage free open source software products, such as Red Hat, follow this strategy. Red Hat's president and CEO argue that *"software should be free, but there should be a charge for services like 24x7 uptime, certifications and support, service level agreements, stability, language customization etc. If the customer gets any value on top of the source code then he should pay for it. That's how open source may be profitable and still free. That's the beauty of open source."* (Whitehurst, 2008)

No doubt, open source software will be the future of the software industry, whether it comes at no cost and no support or it is combined with other services (e.g. customization and maintenance) or with commercial add-in products built upon it. Until now the greater funs of open source software are developers and corporate customers as well. As far as corporate customers are concerned, they are still relying to proprietary software for critical tasks, and for other tasks they experiment with open source applications. What remains to be seen is whether open source software surpasses lock-in and network effects, and moves beyond corporate customers to the broader customer market[22].

6.6 Ad-supported software

In general, the *ad-supported software* is offered for free and provides the same functionality as the paid application. What makes it free is the implied agreement that the user will be watching some advertisements while using the software. So, the product comes at no charge for the user, or at a reduced price, but in fact it is funded by advertisers, who are taking advantage of the large audience that the software product already has, so that they promote their own products (Cusumano, 2007).

Vendors usually choose ad-supported technology as a way to recover development costs or to fund further research and upgrade/maintenance costs. The *ad-supported* software is not a new concept in software pricing. Because of being related to the installation of *Potentially Unwanted Programs (POP)* in the user's pc, it is often not regarded with the respect it deserves. However, nowadays things seem to change, and companies start to launch more applications with the advertising option.

---

[21] For more details, see the discussion at http://www.joelonsoftware.com/articles/StrategyLetterV.html

[22] According to the study *Χρήση Λογισμικού Ανοιχτού Κώδικα (Open Source)* of the Observatory for the Greek Information Society (Παρατηρητήριο για την ΚτΠ, 2007) the Open Source software has not yet completely penetrated in corporations due to concerns related to support and maintenance, legal issues, lack of knowledge and experience.

The Google *Docs and Spreadsheets* software was the first free ad-supported software that competed with the dominant Microsoft *Word and Excel*. In order to confront *Google*, Microsoft proceeded to the launch of the *Office Live*, an ad-supported product competitive to its existing desktop software. Even though these two products differ in the access mode[23] they are both funded by advertisements which run while the software is in use. Microsoft's *Office Live* service is available in two versions, a free, ad-supported version and a monthly subscription without ads. *Sensitive customers* are less tolerant to advertisement and usually select the paid version. These customers are the only ones producing direct revenue for the vendor. The *insensitive* customers bring indirect revenues through the advertisers. Apart from the ad-derived, other potential revenues may also come up. For example, in Microsoft's case the ad-supported *Office Live* can be enhanced with extra add-on services at a charge (Reyes, 2008).

The most frequent case is that ad-supported software products are accompanied by a paid version, and the vendors let the customers to self-select. Yahoo!, for example, is segmenting the market of email, and offers their mail client for free, but with integrated online advertisements. The pricing segment that consists of the insensitive can enjoy an ad-free mail client, the Yahoo! *mail Plus*, by paying an annual subscription fee.

To pay off, advertising must be targeted and relevant enough so that it generates higher revenue and avoids annoying users. If the advertising strategy is designed properly, it can make more money out of advertising revenue than of the royalty based licensing model. Additionally, ad-revenues are ongoing and long-term –however dynamic– while one-time charged software delivers a certain amount of instant profit to the software vendor and nothing more.

Another source of revenue can be maintenance and support. Providing these services on free applications does not mean that they should be free as well. This is the policy of Sun Microsystems: software may be given for free but support is billed. According to a Sun's representative, users who don't pay can be profitable, too (McAllister, 2008). Sun sees each free download as a channel opened to a user; and that channel, can be monetized through co-branding and marketing deals with other companies. The marketing channel to *Java's* audience is already the major revenue source for Sun's *Java* efforts, and currently some co-branding opportunities are examined in the *OpenOffice.org* suite.

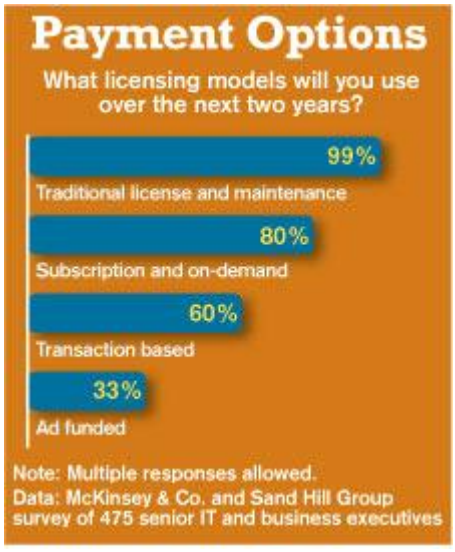

Figure 2: One-third of IT and business executives plan to be using ad-supported software by 2009

Indications are that customers welcome the ad-supported option. The survey of McKinsey & SandHill (2007), found that one-third of 475 surveyed IT and business executives were planning to be using ad-supported software by

[23] Microsoft software is available through an installation on a PC, while *Google* demands an Internet connection so that the applications are accessed.

2009. According to another study from Accenture (2008), the media and entertainment industry will focus on ad-supported content models more and more over the next five years. 62 percent of Accenture's survey respondents said that they expect content creators to focus on an ad-supported model by 2013. By comparison, 25 percent cited subscription-based models, and only 11 percent named pay-per-use services. Not only will ad-supported content dominate the industry, but also executives believe that digital advertising will be bigger than traditional advertising within five years. 52 percent of those surveyed said that online advertising will eclipse traditional advertising by 2013, and 62 percent said that content will be supported by multiple forms of advertising. This could range from sponsorships to branded content to targeted search results. *"Almost every media company is trying to adapt to the reality of digital advertising as a major source of revenue"*, wrote Accenture.

Another issue that should be taken into account is the different viewpoint regarding personal and professional use of ad-supported software. When it comes to personal use, customers are more willing to start using ad-supported software. But for small and medium businesses there are yet some issues to consider before they accept ads with their business applications (Large enterprises are even less interested in ad-supported schemes). Ads may cause a significant wasting of time of employees in a workplace and possible security risks including the potential insertion of malicious code or deceptive ads designed to trick employees into revealing sensitive passwords and other information (Braue, 2008).

There are some concerns on the future of software in case it is driven by advertising revenues. Perhaps the most worrying aspect of the ad-software model, however, is that it tends to consolidate control of software distribution into a few very large companies. To advertise effectively, a large customer base is needed, and dominant vendors tend to have a significant advantage on this.

7 Pricing practices

The main goal of any software company is to make profits. First of all the company has to analyze market information and competition and then decide on a pricing strategy for all its products. Setting the price is not always a straightforward process. Most vendors set the price of a new product with a great deal of apprehension because they have no precedent on which to base their decision. If the new product's price is excessively high, it is in danger of failing because of low sales volume. On the other hand, if the price is too low, the product's sales revenue may not cover costs.

Apart from the actual pricing scheme, some pricing practices can be adopted in order to boost the sales and reduce the risk of wrong estimations or market changes. The pricing practices may specify the target group and differentiate according to the perceived value of each customer segment. Practices should be flexible enough to ensure that revenues can be collected from some (or all) possible customers at different times or through different versions. Marketing techniques can be employed in order to promote the software products and confront possible competition. Except for an attractive price, a product that has penetrated in the market and has created a large customer base is less vulnerable to competition and far more likely to maintain its market share.

We will continue our analysis by examining the supplementary practices that software vendors can use along with their pricing strategy to ensure the success of their products.

7.1 Penetration and Skimming Pricing

Vendors may adopt the *penetration* or *skimming* pricing practice when setting the price of a new product. Later, they may change the price according to any of the pricing schemes we have already mentioned.

7.1.1 Penetration Pricing

In a highly competitive market, a product is usually introduced with a lower than the competition price; this practice is called *penetration pricing*. The reason why companies set a low introductory price is to gain quick acceptance and extensive distribution in the mass market. By such a scheme, the company is expecting to gain a significant market share and to discourage competitors from entering the market. The penetration price has to be low enough as to remove the price of the product from the buying decision but also should be sustainable and higher than the company's variable costs.

The penetration pricing is most appropriate when the product demand is highly *elastic* (Harmon et al, 2004); by a price reduction either new buyers are attracted, or existing buyers buy more of the product. When *economies of scale* are present, this practice is even more profitable and as far as software is concerned, the implied risk of not being able to satisfy the increased demand is eliminated, since the reproduction and distribution mechanisms force no restriction in the production capacity.

Entrepreneurs must recognize that penetration pricing is a *long-run* practice; until a company achieves customer acceptance for the product, profits are likely to be small. Moreover, from the moment that a penetration policy is chosen, it is difficult to eventually raise prices because *long-term pricing expectations* are already set.

Companies that attract customers by offering low introductory prices must wonder what will become of their customer base if they increase their prices or if a competitor undercuts them. The penetration pricing practice attracts customers who are *not loyal* to another brand, and thus there is no indication that they will become loyal to the new product (Farrell and Klemperer, 2007).

So far, the penetration pricing practice is quite attractive but companies should think twice before deciding to adopt such a practice. A low-priced product is running the risk of being perceived as a low quality product. Since price is often used as a quality signaling mechanism, the companies should design carefully the pricing scheme of the introductory period as well as the transition to the future pricing technique (Harmon et al, 2004).

A penetration pricing practice may be adopted in order to promote complementary products. The main product may be introduced and maintain the price to low levels to attract sales, while the add-on services are sold at higher mark-up and hence compensate for lost profits (Harmon et al, 2004).

If the penetration pricing practice works and the product achieves mass-market acceptance, the sales volume increases and the company earns significant profits. That was the objective of Microsoft in 2006 when it introduced the *Windows Live OneCare* bundle –an antivirus and PC care package– at $19.99 through the retailers Amazon and Best Buy, well bellow the list price of $49.95. Microsoft charged *Windows Live OneCare* for 29 percent to 44 percent less than the competitor's prices. According to the NPD Group, this move resulted in a 15.4 percent of security suite sales through retailers (Evers, 2006). At the same time, the

market leader Symantec, faced a drop of 10.1 percentage points, and McAfee lost 3.3 points and Trend Micro dropped 1.3 points. However, even with the penetration pricing of Microsoft, Symantec remained the clear leader in the retail channel, with 59.8 percent of security suite sales. But, Microsoft took the second place, followed by Trend Micro with 8.9 percent and McAfee with 7.1 percent.

7.1.2 Skimming Pricing

The opposite of the penetration pricing is the *skimming pricing* practice: the product is established with a high initial price with a view to "skimming the cream off the market" at the upper end of the demand curve (Harmon et al, 2004). It is often used when a company introduces a new product into a market with little or no competition or in a market with established competition but which contains an elite group that is able and seems willing to pay a premium price.

The firm sets a higher than the normal price in an effort to quickly recoup the huge initial developmental and promotional costs of the software product by extracting the surplus of customers starting with those with high perceived value for the product. The idea is to set a price well above the total unit cost and to promote the product heavily to appeal to the segment of the market that is not sensitive to price. As sales volume increases with the broad acceptance of the new product, the firm can lower its price to generate additional sales and thus to increase the firm's total revenue. This practice is employed only for a limited period of time as a way to recover most of the investment of a product.

According to Köehler (1996), the skimming price practice is a high-price practice which provides a healthy margin but risks a depressed sales volume. Since high prices also attract piracy, protection costs against piracy basically eat up margins. However, there are examples like the case of Apple, where the buyers are not attracted by pirated versions of products because the prestige of the brand is linked to the snobbism of the "members of the Apple family".

In high-tech software industry, at the top of the demand curve, price elasticity is low[24]. Besides, in the absence of any close substitute, cross-elasticity is also low. An innovating product with customers that are anxiously waiting for it is the perfect combination for a skimming pricing. Typical anxious customers are game users, who are willing to pay a premium so that they are among the first to try the new game.

The high price also helps segment the market. Only non price-conscious customers will buy a new product during its initial stage. Later on, the mass market can be tapped by lowering the price. If there are doubts about the shape of the demand curve for a given product and the initial price is found to be too high, price may be slashed. If price elasticity is higher than anticipated, a lower price will be more profitable. The decision regarding how high a skimming price should be depends on two factors: (a) the probability of competitors entering the market and (b) price elasticity at the upper end of the demand curve. If competitors are expected to introduce their own brands quickly, it may not be safe to price rather high. Determining the duration of time for keeping prices high depends entirely on the competition's activities.

[24] For more information please visit http://www.web-articles.info/e/a/title/Skimming-pricing-and-penetration-pricing/

7.2 Bundling

Bundling can be easily considered as grouping together several products or services into a package that offers customers extra value at a special price. Software manufacturers bundle several applications (with Microsoft being the most famous for bundling a word processor, spreadsheet, database and presentation graphics) into "suites" that offer customers a discount over purchasing the same packages separately.

Bundling can be understood as a very efficient way, in practice, to implement *price discrimination* (Βέττας and Κατσουλάκος, 2004). Since it induces consumer self-selection, it is far less information demanding than the theoretical optimal solution where the vendor must collect individual customers' willingness to pay for each good. At the same time, it circumvents regulation forbidding explicit discriminatory pricing[25] (Baranes and Le Blanc, 2006).

In some cases bundling enables savings on *transaction costs*. The sale of a bundle of goods or services automatically reduces the number of market interactions and the attached transaction costs, such as search for the best supplier, comparison of available prices and quality, negotiation and conclusion of the contract (Cusumano, 2007).

The bundling practice is widely utilized by manufacturers as a low-cost marketing tool to introduce a new product into an established large customer base. However, there are several requirements for a successful implementation (Shefer, 2007). First, the products should be *complementary* and not substitutes to each other. Second, even if individual segments have different perceived values for the specific products, they should have similar *overall perceived value* for the bundle. And third, the *profit* of the bundled product should be greater than the total profit of selling the individual products.

Grouping two or more software applications is a common bundling technique. The benefits of this technique were early perceived by Microsoft, which in 1989 started bundling the *Word 2.0c*, *Excel 4.0a*, *Powerpoint 3.0* and *Mail* under the first *Office Suite*, that was launched with the name *"Office 92"*. Until then, all the components were distributed separately. When *Word* was first released in 1983, it was not well received, and sales lagged behind those of the rival product *WordPerfect*. Later, in 1989 the failure of *WordPerfect* to produce a Windows version proved a fatal mistake. The advantage of Microsoft providing a *Word* version compatible with Windows, in combination with the bundling practice in the *Office Suite*, established Microsoft as the market leader in desktop applications.

Manufactures often bundle complementary applications with their base software that has already established a large customer base. They choose bundling as a way of promoting complementary applications, either "add-ons" –which are built upon the base software– or other applications –built from the same manufacturer or from a partner– making thus use of the large acceptance of the base product in order to increase the demand for the promoted product (Shapiro and Varian, 1999). A well-known example is the prevalence of Microsoft's web browser *Internet Explorer* on *Netscape Navigator*. Until late 1997 *Netscape Navigator* held more than two-thirds of the browser market, surpassing Microsoft's *Internet Explorer*, but only two years later the shares were reversed due to IE's bundling with Microsoft Windows.

[25] The Robinson-Patman Act made it illegal for sellers to directly or indirectly discriminate in the price of similar commodities, if the effect hurts competition. Note that the law applies only to products and not to services.

Another common bundling technique is the packaging of software products with hardware, again aiming at boosting sales. This technique is used even more effectively when the company is active in both the software and hardware market. IBM was the first to unbundle software from hardware in 1969, but twenty years later IBM was once again moving towards bundling. For instance, in 2005, IBM along with the pre-installed operating system, started to pre-configure also middleware applications such as *WebSphere Application Server Express* and the database *DB2 Express* on the *Blades Servers*[26] and sold them as a ready-to-go, turnkey system, for e-Business.

Finally, another widespread technique is to bundle the software product with a *maintenance and support* service. In many companies, maintenance and support for the initial term (usually one year) is offered in a pure bundle, and do not allow a product to be purchased unless the one-year support is included.

Bundling seems to be a very attractive method especially for prevailing companies. It can allow a company with monopoly in one market to exercise greater market power in other markets and to strategically disadvantage rivals in those markets (Βέττας and Κατσουλάκος, 2004). Except under unusual conditions of *monopoly power*, courts don't view bundling as a serious antitrust concern, as far as the benefits to consumers offset potential damage to competition.

Microsoft is quite famous for its longstanding practice of tying the *Internet Explorer* browser to the *Windows* operating system. PC users find *Internet Explorer* on their computer desktops without explicitly requesting so. By tying *Internet Explorer* to *Windows*, which runs on over 90 percent of the world's PCs, Microsoft has achieved ubiquity for *Internet Explorer*. This ubiquity distorts competition[27] and favors Microsoft's monopolies in ways completely unrelated to the merits of Microsoft's products.

There is another side effect: *Internet Explorer* is ubiquitous by virtue of its being tied with *Windows*. Web content providers write content using *Internet Explorer*'s proprietary standards that can only be accessed properly in the specific browser. The content will often not work properly on other browsers that comply with open standards. As a result, users are compelled to use *Internet Explorer*, insulating it from any real competition and reducing user choice and innovation. Moreover, Microsoft makes *Internet Explorer* available only for its *Windows* operating system (Farrell and Klemperer, 2007). As a result, users of alternative personal computer operating systems are left using incompatible browsers and will often find that web content appears degraded. This gives an artificial advantage to *Windows* and deprives consumers of the ability to choose alternative operating systems that enable equal access to the Web.

7.3 Versioning

Versioning (also referred as *Quality Discrimination*) is a special form of differential pricing (Varian, 1997). It is based on the fact that different consumers may have radically different values for a particular software product. Vendors provide different qualities/versions

[26] Blade servers are self-contained all-inclusive computer servers with a design optimized to minimize physical space

[27] Similar Microsoft tying practices have already been condemned by the European Commission and European Court of First Instance in 2004 with respect to Windows Media Player.

of a product which then they sell at different prices, and then they induce customers to "self-select" into appropriate categories according to their willingness to pay.

The key element so that the versioning technique is successful is to prevent customers with high willingness to pay from buying a low-priced version of the product. There are cases where the customer's characteristics are observable and others where only the possible types are known, but the type of any given customer is unknown. Typical examples of the first case (*third-degree price discrimination*) are the *academic* and *student* versions, that may contain fewer features than the full version of the product, while for the second case (*second-degree price discrimination*), the most usual distinction is into *standard* and *professional* version, with the professional being enhanced with more features, such as extra functions or computational speed (Shefer, 2007).

Let's examine a little further the case where the customer's characteristics cannot be observed. In this case, the vendor's intention is to get the consumers to self-select into the high- and low-willingness-to-pay groups by setting price and quality appropriately. That is, the vendor wants to choose price/quality packages so that the consumers with high willingness-to-pay choose the high-price/high-quality package, and the consumers with low willingness-to-pay choose the low-price/low-quality package. What is very important for the success of the versioning mechanism is to prevent any possible arbitrage, which may happen if the high willingness-to-pay customer is buying the low-priced version of the product and the opposite. According to Bhargava and Choudhary (2008), if the high willingness-to-pay customer has the greatest surplus by buying the high-priced version, then the arbitrage-possibility is eliminated and versioning is most profitable.

Comparing to other markets, versioning in the software industry has a major difference. From the vendor's point of view, the most expensive version of the product is the least costly. Developers add features until the incremental value of those features to the most demanding customers just equals the incremental development cost. This is the high-end version and is targeting the cream-customers. Then, they deactivate a subset of the available features, resulting to the reduction of the product's quality and creating thus the lower versions (Shapiro and Varian, 1999). Therefore, to create the lower version, the developers need to spend more time and more effort.

Ideally, the number of versions that maximize the vendor's profits equals the number of distinct customer-types. But this would lead to a huge number of versions, which is impractical for two reasons. As we said before, for the developer to create and then to maintain more than one version requires extra time and effort and thus more money spent. On the other hand, customers would be confused if they had to choose between too many versions (Shapiro and Varian, 1999). The search cost would then increase too much, leading them even to not buy at all. So, we understand that both sides are benefitted when only a number of versions are available; but what is the optimal number of versions?

The most common practice is to choose between *two* or *three* versions, with *three* being the most advantageous. The rationale for this suggestion derives from what psychologists call *extremeness aversion*; customers normally try to avoid extreme choices (Varian, H., 1997). If customers have to choose between two options, the odds are that they will choose the low one, but if they have to choose between three options, it is more possible to avoid the two ends and select the middle "safer" choice. So, instead of having only the *standard* and the *professional* version, it is optimal to have one more, the *golden* edition. If

the customer psychology is driving them to choose the *standard* between the two available editions, it is very likely that they will choose the *professional* edition when a *golden*-one is offered. The presence of three versions influences low willingness-to-pay customers to trade up to higher-priced models. The *golden* version may have some privileges such as direct access to technical support. As long as too many people don't choose the *golden* version, the cost of adding this kind of support will be small.

We pointed out why versioning is very useful and desirable for both the customer and the vendor. Not surprisingly, almost all software products are offered in multiple versions. Usually the lowest one is free or becomes unusable after a pre-defined time period (the trial period) in order to attract new customers, while the profits come from the paid versions. Microsoft *Office 2007 Suite* for example comes at the following versions: *Home and Student ($149.95), Standard ($399.95), Small Business ($449.95), Professional ($499.95)* and *Ultimate ($679.95)*.

There are, however, some products that are produced in one single version. *MS-DOS* for example has no versions[28]. The reason is that its purpose was to form the common base for all other applications that would be then developed upon MS-DOS and would be marketed as *MS-DOS* compatible. Having more than one version would be against the exploitation of the network effects of a common operating system.

Versioning does not always have to do with the actual product that the customer buys. It may have to do even with the media, the packaging or the hard-copy documentation. Small and traditional customers are accustomed to having a hard-copy of all possible software related information, and until they keep up with the late trend where everything is downloadable from the internet, they will be willing to pay for the extra cost that vendors charge for physical media and packaging.

7.4 Switching Costs and Lock-in

Switching costs are defined as supplier-specific costs incurred by a customer that must be duplicated should the customer switch to a new supplier (Farrell and Klemperer, 2007). The estimation of average switching costs, assuming perfect competition, are found to be approximately 20 percent of the total cost of ownership of the software package.

The high switching costs affect a business decision regarding the replacement of an insufficient software product with a more adequate one, and thus lock the customers in their existing software providers. There are many factors that increase the imposed switching costs (Shapiro and Varian, 1999; Larkin, 2008; Farrell and Klemperer, 2007), including:

Product learning and training costs: Users should be trained on the system, about the differences with respect to the previous application and to the extra functionalities it includes.

Changes in business processes: Major software installations inevitably involve major changes in business processes across the enterprise. Managers must be trained in how to exploit all new capabilities and perform a re-engineering of business processes.

Equipment and compatibility issues: The new software may need special underlying hardware, and integration with the pre-existing technical architecture. Existing data must be collected and entered with no loss of integrity.

[28] Versioning should not be confused with possible upgrade versions that a product may have. Upgrade versions are to replace the previous versions and not to co-exist with them.

Maintenance costs for the software: The new software may require regular maintenance and support from specialized personnel at a possible charge.

Psychological "brand loyalty": Customers are risk averse when it comes to choosing between investing on a leading company in the software market or an emerging and thus riskier company.

Uncertainty over product quality: The expected utility of the new software is affected by the potential, or the perceived potential, for a customer's business to suffer should it adopt a new provider's product (even if it is from a leading company).

Switching costs do not have to be *"real"*, they can equally be perceptions of costs that might incur should a customer switch. The most famous story here is of Nike Inc's switch to a *Supply chain management (SCM)* package from i2, the leading stand-alone *SCM* provider (Larkin, 2008). In announcing a $100 million sales shortfall in June 2000, Nike blamed problems with its new installation of software bought from i2, while i2 blamed Nike for not investing enough in learning the software. The cost of the package, including software, hardware and services, was reported by Nike CEO Phil Knight to exceed $400 million. On the day of the earnings shortfall announcement, Nike's stock dropped over 20 percent. Customers are unsure how much business pain will come from switching application providers, and this fear represents a *real* switching cost.

Products with high switching costs use to have higher prices (Farrell and Klemperer, 2007). When competing for customers in the initial period, a firm may lower its prices in order to attract customers, but once the customers are *locked-in,* the firm can charge higher prices. Companies try to increase customers' lock in by including complementary products (add-on) in their basic software, but this attempt cannot be successful unless the add-on products increase the switching costs (Shapiro and Varian, 1999).

Finally, switching is affected by the strategic behavior of customers, who realize that the act of *switching* can send a strong signal to suppliers that their switching costs are low (Larkin, 2008). Sending this signal can result in lower prices paid even after the initial period, as the suppliers of choice may be afraid of another switch if their prices become too high.

Switching costs will probably vary from customer to customer. This heterogeneity in combination with the incomplete information about customers' true switching costs may lead the vendors to incorrect estimations about the optimal prices that they should charge.

7.5 Discounts

Discounting is a form of nonlinear price-discrimination technique followed by vendors in order to increase their sales volume and install or increase their customer base. Even if a single list-price is published, each customer is negotiating the potential purchase by taking in exchange the maximum possible discount.

The rationale behind discounts is to exploit the *economies of scale* and pass some (or all) of the savings on to the customer. In some industries, buyer groups and collaborations have been formed to take advantage of these discounts. Discounts are usually based on the sales volume of the specific customer or group with regards to a single or a cumulative purchase. Quantity discounts are, generally, of two types, i.e, cumulative quantity discounts and non-cumulative quantity discounts. The *cumulative quantity discounts*, also known as accumulation discounts, are price reductions based on the quantity purchased over a set

period of time. The expectation is that they will impose an implied switching cost and thereby will bond the purchaser to the seller (Shapiro and Varian, 1999). The *non-cumulative quantity discounts* are price reductions based on the quantity of a single order. The expectation is that they will encourage larger orders, thus reducing billing and order filling, and the seller will get a large present cash flow.

All major enterprise software application vendors have clear-cut discounting rules for their sales; one company's rule is that discounts up to 20 percent are at the discretion of the individual salesperson, discounts between 20 and 30 percent require a district manager's approval, discounts between 30 and 45 percent require a regional manager's approval, discounts between 45 and 60 percent require the Vice President of Sales' approval, and discounts above 60 percent require the CEO's approval (Larkin, 2008). Unless the project involves a competitive bid, which usually means the terms offered by the various vendors leak, offered and realized discounts are guarded with utmost secrecy.

Companies use to offer discounts as a *reward to loyal customers*. Loyalty is often expressed by the sales volume of a company in a specified time interval. Large and loyal customers are aware of their negotiations power and they demand significant discounts as an indication that the vendor is recognizing their faithfulness. The final decision on which vendor's product to buy is frequently based on who is offering the better discount. This behavior is sometimes responsible for very high initial list prices that are later reduced under negotiation, just to fulfill the psychological need of customers for discounting.

Universities and academic institutions are a special discounting band. The discount may reach even the 100 percent in special versions of software products and the reason behind this charity is that manufacturers focus on students for their future revenues. A student familiar with a specific product is a future professional who is going to get the same software at a higher price (or with lower discount).

Discounting is very popular technique for introductory promotional offers; customers are facing high switching costs when they are already using a competitive product and are resistant to trying new products unless they are offered in a friendly price (Harmon et al, 2004).

Apart from the new products promotion, sometimes discounting is just a way to hurt competitors. A company may reduce temporarily a product's price not because this method will help increase the profits, but because it will not let a competitor gain any market share. Temporal discounts are a predatory pricing technique that helps companies to avoid a price war. Failing to respond to competition pricing attacks can cause a competitor to leave the market. This practice is not illegal unless it reduces the customers' utility (Βέττας and Κατσουλάκος, 2004).

There are cases where a company is reducing prices for new programs when they are to replace their own obsolete programs. These discounted sales are called *Trade-ups*; the customers must terminate the use of discontinued programs in order to get the discount. List prices for these "upgrades" are typically only 30-60 percent of the list price for the entire new product.

A software company may be reducing its products price, only to achieve that the competitors already installed products are replaced by its own products. This technique is also known as *Competitive Trade-up*. The customer is already locked-in to a competitor, so the discount should be enough to compensate the switching cost that will incur if the

customer chooses to change software provider. IBM for instance offers licenses for certain programs at a reduced charge if the customers are to replace qualifying Non-IBM Programs. Customers have to agree to terminate their use of the replaced Non-IBM Programs when they install the replacement programs.

Another special case of discounts is the price reduction on products sold to *Value-Added Resellers (VARs)*. A VAR is a company that adds some features to an existing product and then resells it (usually to end-users) as an integrated product or complete turnkey solution. Many companies reduce their prices about 30 to 60 percent in order to motivate other firms to produce complementary products that add value to their existing products.

8 The role of Greece in the Software Sector

More than in many other sectors of the economy, the software industry is constantly under the pressure of change, innovation, growth, emerging technologies and new business models. Even if European software vendors have recently been impacted by an extraordinary concentration, Europe represents a growing, fast moving and innovative market totaling approximately 30 percent of the worldwide software spending, generating revenues of approximately €56,000 million.

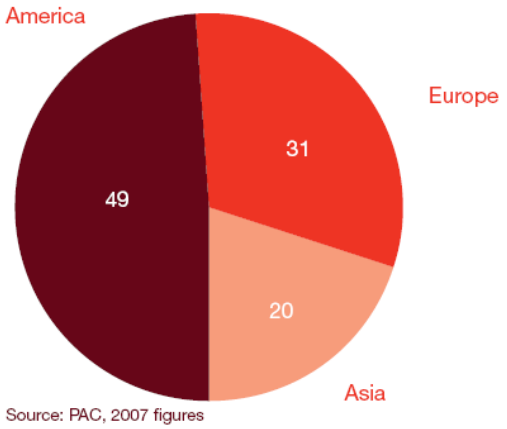

Figure3: Share of the worldwide software market (%)

According to the study of EuroSoftware100 (PriceWaterHouseCoopers, 2008), the US software vendors feature highly in the ranking and have considerably strengthened their position in the European market mainly through acquisitions.

Rank	European leaders	Company		Software Europe	Total Europe	Total Wordwide
1		Microsoft	US	9 355	11 135	39 192
2		IBM	US	4 200	23 664	72 202
3	1	SAP (incl. Business Objects)	DE	3 807	5 105	11 346
4		Oracle (incl. Hyperion in April 07)	US	2 889	3 673	13 860
5		Symantec	US	1 270	1 319	4 155
6		HP	US	1 100	27 215	77 673
7		EMC	US	1 050	2 579	9 670
8		CA	US	765	868	3 025
9	2	Sage	UK	715	835	1 714
10		Adobe Systems	US	645	681	2 327

Table 1: EuroSoftware100 – top 10 vendors ranked by software revenue

The top three software vendors listed (who are also ranked the top three in the world) generate revenues of more than €55,000 million worldwide, the equivalent of the entire European software market.

Rank	Company	Nat	Total revenue	Software revenue
1	Microsoft	US	39 192	31 000
2	IBM	US	72202	13 320
3	Oracle (inc. Hyperion)	US	13 860	11 075
4	SAP (incl. BO)	DE	11 346	8 256
15	Sage	UK	1 714	1 304
16	Dassault Systemes	FR	1 259	1 160

Source: PAC, 2007 figures

Table 2: EuroSoftware100 – European software vendors in the top 20 worldwide

Greece in not a key player in the European software market, and probably is not in position to influence the future trends of the software sector. As we can see in the following list of the top-200 software vendors that operate in Europe, the Greek companies do not manage to compete well abroad.

	Top 20	Top 100	Top 200	Total
US	15	33	22	70
UK	2	13	19	34
FR	1	7	12	20
DE	1	4	14	19
SE		2	6	8
FI		2	5	7
IT		3	4	7
NL		3	4	7
NO		2	4	6
JP	1	4		5
BE		1	3	4
CH		1	3	4
ES		2	2	4
CA		2		2
DK		1	1	2
IE			1	1
Total	20	80	100	200

Source: PAC, 2007 figures

Table3: EuroSoftware100 – Number of software vendors in the top-200 European market

According to George Naoum, Partner of PricewaterhouseCoopers Business Advisors[29], there has been a progress in the market conditions in Greece, but the biggest and persistent problem is the software piracy. In 2007, the software piracy has reached a 58 percent, causing losses of €136 millions. Although the percentage of piracy keeps reducing, it is still high comparing to other west European countries.

On the other hand, Greece seems to adopt very quickly the latest software trends. According to the Global Information Technology Report (2008–2009), Greece is doing well in *usage of latest technologies* and is evaluated with 4.71 (in a scale of 1-7). In the field of *licensing of foreign technology*, the Greek score is also high (at 4.91). Therefore, we expect Greek companies to quickly adopt and promote the latest technologies through their products and as a result to implement the licensing and pricing schemes that the dominant companies in the worldwide software sector propose.

It is true that most of the international software vendors mentioned in the previous tables operate also in Greece seeking for more customers, showing that Greece is a fertile ground for future expansion. With a very high academic and professional level and an above-average quality of scientific research institutions, Greece is a promising country in software development. As far as the Greek branches of multinational companies are concerned, they cannot play a significant role in setting the list prices or designing the sales policy of a company. Usually, the headquarters are responsible for defining the pricing strategy as far as the available licensing schemes are concerned, but also for setting each product's list price. The peculiarities of the product's demand in each relevant geographic market may drive the vendors into setting different list prices across different countries.

8.1 Examples from the Greek software sector

Although most Greek companies build applications on other established international products and provide solutions customized on the needs of the client, there are cases where Greek companies develop their own stand alone software products. Below, we proceed by examining the contribution of selected software companies which operate in Greece, and that adopt the aforementioned value-based pricing strategies and practices. The evolution of pricing has ended up in complex pricing schemes that can be classified under more than one of the pricing categories. For example, Microsoft Office should be classified under flat pricing, since it is paid only once. It is though a bundled application, and it comes in more than one versions. Microsoft has achieved to create huge switching costs and therefore is enjoying the lock-in effect.

Following the classification we have already introduced and described before, we will present some representative examples of each category.

- Duration-based → Flat pricing: The most famous product in this category is *Microsoft Windows*[30]. Windows are purchased only once and after installed they can be used forever. Microsoft provides each legitimate user with a license key for a single installation without which the program is useless. This policy resulted from Microsoft's effort to thwart the illegitimate users that were using some leaked "corporate" keys.

[29] More information can be found at http://www.imerisia.gr/article.asp?catid=12319&subid=2&pubid=5177111#

[30] More information can be found at http://www.microsoft.com/hellas/licensing/programmes/explained/default.mspx

- Duration-based → Subscription licensing: Microsoft offers the *Open Subscription License (OSL)*[31] as a cost effective way for small to medium organizations with 5 to 500 or more desktop PCs. The Open Subscription License lasts for three years and payments are made annually. For instance, Microsoft Office XP Professional is a product that is offered under this scheme. During the term of the agreement, the corporate user has the right to use the specific software, but once the agreement expires, the corporation should either re-new the agreement or choose to purchase the perpetual software. If it chooses not to re-new the agreement the software must be removed.
- User-based: IBM offers the *Rational* software under the *Authorized User* license[32] i.e. it is a named user licence. This license allows a single, specific individual to use the product. IBM offers also the *Rational* software under the *Floating* license. It is a license for a single software product that can be shared among multiple team members; however, the total number of concurrent users cannot exceed the number of Floating licenses that are purchased. If other persons in the corporation want to access the product, they must wait until the original user logs off. The company must obtain Floating license keys and install the IBM Rational License Server, which is responding to end-user requests for access to the license keys.
- Processor-Based: IBM charges the *Express Edition* of the *DB2* database using the Processor Value Unit (PVUs)[33] pricing metric. According to the PVU processor model, each processor running the DB2 software is counted and multiplied by the DB2 per-processor price to determine the eventual license costs. This model is targeting to companies that have large numbers of users in relation to database size as well as environments where end users are not easily counted (like a Web-based application), because this licensing paradigm removes the need to manifest or "know" the exact users (from a licensing perspective). Another attractive feature with per-processor pricing is as new employees are hired, each new employee does not require a license entitlement, which also leads to a reduction in license administration costs.
- Usage-based: IBM offers also the *Usage License Charges (ULC)*[34] for the DB2 database, as an attractive pricing mechanism to deploy low-utilization software products across an enterprise, especially when major subsystem usage is less than 25 percent of the processor. For a product with ULC, software charges are based upon the utilization of that product and are calculated with the help of a *Sub-Capacity Reporting Tool (SCRT)*.
- Open source software: IBM is also famous for giving away the *Eclipse project*[35] to the Open Source Community. *Eclipse* is multi-language software development platform written primarily in Java and is used to develop applications in this language. Several plug-ins have been developed for this framework to extend its capabilities such as development toolkits for other programming languages. IBM's donation boosted the

[31] More information can be found at http://www.microsoft.com/hellas/licensing/programmes/openvalue/ovcw/default.mspx

[32] More information can be found at http://www-01.ibm.com/software/rational/howtobuy/licensing/

[33] More information can be found at http://www.ibm.com/developerworks/data/library/techarticle/dm-0611zikopoulos2/ and https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express?P0=E1&part_number=D52BILL,D52BKLL,D55TULL,D55MBLL,D58NJLL,D58NNLL,D588YLL,D5890LL,D58UULL,D58UXLL,D58MSLL,D58CPLL,D5910LL,D60PLLL,D593PLL,D59QTLL&catalogLocale=en_US&locale=en_US&country=USA&PT=html

[34] More information can be found at http://www-03.ibm.com/servers/eserver/zseries/swprice/other/

[35] More information can be found at http://www.research.ibm.com/journal/sj44-2.html

whole open source sector and motivated other companies to do so. According to IBM, the revenue generated by open-source software does not come from the products themselves but from related hardware, software, and services, and now this revenue is in the billions of dollars and growing.

- Ad-supported software: While ad-supported software is still in pre-mature phase, M-Stat, an innovative Greek company, has already launched an application that is free to the customers and is entirely funded by the advertisers. TxtMe[36] is promoted as a social network from which the user can send free sms limited to 120 characters, while the rest 40 characters contain a short advertisement. M-Stat apart from individual users, is targeting also companies with presence in the internet, which may employ TxtMe to attract more visitors.

Studying further the pricing practices we can distinguish the following examples from the enterprises that operate in Greece:

- Penetration Pricing: The Greek Telecoms are offering a new service, the *Conn-x TV*[37] 12-month subscription, at a friendly price until 31/05/2009. The low price will be effective only for a limited introductory period, after which it will increase to a higher level.
- Skimming Pricing: In 2008 Microsoft announced that they would reduce globally the retail price of *Windows Vista*[38]. According to Microsoft's announcement, the reduction was aiming at pushing customers to switch to the latest version of Windows, and would vary depending on the country.
- Bundling: IBM offers *Integrated Configuration Management software* and *Change Management software* as part of the *Rational Suite*[39]. These products can be offered individually or as a bundle, charging the bundled software significantly less that the total price of the two individuals. In the case of *ClearCase* and *ClearQuest* products, the bundled product is priced 24 percent lower than the accumulated price of the two separate sales, giving thus a clear motivation for the products to be purchased in a single sale.
- Versioning: One of the most famous products that Microsoft offers in several versions is the *Office 2007 Suite*[40]. In the Greek market, the most expensive –and thus less demanding in time and effort[41]–, is the *Office Ultimate 2007* that contains more features than any other version and, at the time of writing this thesis, can be purchased at around 890€. The *Office Professional 2007,* the *Office Standard 2007* and the *Office Small Business 2007* have less features enabled and thus cost 690€, 550€ and 230€ respectively. Finally, the least expensive version, the *Office Home and Student 2007*, targets the least demanding user and contains very little features comparing to the *Ultimate* one; thus it is offered at a very attractive price of only 90€.

[36] More information can be found at http://www.m-stat.gr/index.php?page=2

[37] More information can be found at http://www.conn-x.gr/Default.aspx?MenuID=64

[38] More information can be found at http://gadgetvote.com/blog/micosoft_reduce_windows_vista_price.html

[39] More information can be found at http://www-01.ibm.com/software/awdtools/changemgmt/

[40] More information can be found at http://office.microsoft.com/el-gr/suites/FX101677751032.aspx

[41] We have explained in the section regarding Versioning why software is priced inversely proportional to the time and effort required.

- Switching Costs and Lock-in: Microsoft continues with their policy of ensuring the compatibility of Internet Explorer with Microsoft Windows, and the non-compatibility with any other operation system. A table in Wikipedia[42] is proving that Microsoft intentionally left all other operating systems out of the competition game. This management decision is taken in order to lock the customers in Microsoft's products either because web applications that are built for windows tend to be optimized for IE, or because users that are already familiar with Windows are becoming less willing to switch operating system.
- Discounts: Microsoft offers a better deal if a Small Business customer needs more than 4 copies of a software product (volume discount[43]), that can achieve a discount up to 25 percent of list price. Also, when customers select subscription licensing and upgrades from a qualifying version, then they may achieve a discount up to 50 percent on first year.

9 The future of software pricing

During the past several decades, the software industry has proven itself dynamic and lucrative. But it now faces a serious economic threat, exacerbated by today's challenging economic times, focused on the wild and win-obsessed world of software pricing.

The *enterprise* software industry, which provides products for large corporations, has suffered from a frenetic, often irrational price discounting conundrum. Customers have too often been able to negotiate lower prices for software by waiting to buy until the end of the sales quarter, when software sales representatives are fighting to reach their quarterly sales targets. To ensure they close a sale, sales people often sweeten the offer by giving away other free software applications, capabilities and high value technologies to software buyers. Often, millions of dollars of research and development expense have been invested in those freebies that sales people "throw in" to seal deals. These are too often last-minute, under-analyzed decisions that hurt the profitability of software companies by "leaving money on the table".

Another area of growing concern is *packaged* software used for smaller applications. Typically, this software type runs on fewer computers, often PCs, and is less customized than enterprise software. The biggest problem of software companies addressing to this market segment is the sheer enormity, variety and complexity of packaged software. The packaged software is being offered in an array of bundles for different customers, countries and channels so there are literally thousands of SKUs[44] for what is really one product (Hanson et al, 2008). This problem confounds and overwhelms people working in the business of pricing packaged software. There exists too much detailed and disorganized pricing information, such as pricing sheets and price adjustment forms, to sort through, analyze, compute, collate, compare and make decisions about compared with the limitations of time, people and funding.

[42] More information can be found at http://en.wikipedia.org/wiki/Internet_Explorer

[43] More information can be found at http://www.microsoft.com/smallbusiness/buy/software/buy-software.aspx#VolumeLicensing

[44] *Stock Keeping Unit (SKU)* is a unique identifier for each distinct product and service that can be ordered from a supplier

In both cases of *enterprise* and *packaged* software, the question is *"what software companies should do to grow their profits and revenues*". John Hanson, a senior pricing strategy executive of Accenture (Hanson, 2009), recommends slow reaction to the demand decline. Prices should not drop immediately but other ways should be exploited that deliver value to customers. For example, software firms should evaluate and identify their best customers and serve them in ways that they value. Focus should be set on delivering a superior customer experience to diminish the customer's desire and justification for pressuring to lower prices.

The software companies should estimate the demand from different types of customer segments and geographic markets and adjust the price accordingly. They should also estimate whether it is worth to offer a bundle of products or price individually each part.

Whenever a sale is closed, a significant tradeoff should be taken into account. A software firm may win a sale by lowering the price, but by doing so, the business will suffer long-term losses. If a company reduces the price in times of economic crisis, that company exposes itself to the dire reality that when the market returns to a growth phase, it will be difficult to persuade customers that a price increase is justified. During the economic crisis, cost containment and retention of the best customers are paramount to success.

10 Summary and conclusion

This master thesis was established aiming at describing the economics of software and the possible ways for pricing a product. According to its purpose, each company may choose among a variety of pricing strategies and adopt whichever is most suitable for a specific occasion. Software, unlike other goods, has a huge initial production cost and zero reproduction cost, so a cost-based pricing strategy had soon to be abandoned. The pricing policy for software products has evolved to value-based schemes, where unit-price is related to the license-duration, the number of users, the capacity of processors or the actual usage of the software product. Other trends such as open source and ad-supported software have changed the traditional software economics. The software may also be distributed for free, with revenues coming from complementary sales or services, like consulting and advertising.

We discussed various price-related practices that are suitable for penetration in the software market or for increasing the market share. We described practices that help a firm to extract the maximum possible surplus from different customer segments starting from the "crème de la crème" and moving down to the low-value customers. Bundling several products together may also increase under specific conditions the revenues of a firm. We studied versioning as a second and third degree price-discrimination technique, and quoted on the optimal number of versions that maximizes profits. We examined the switching costs that a firm imposes on customers and the benefit of a firm having locked-in customers. Finally, we mentioned various ways of discounting as a means of retrieving cash flows and rewarding loyal customers.

We also performed a study on the Greek vendors and examined their position in the international and European software sector. Although Greece is not a key player in the software sector, the analysts are optimistic and foresee a growing development that will reduce the loss caused by digital piracy and will bring more revenues provided that the Greek

companies will continue to be flexible and keep up with the latest software and pricing trends.

Our study intended to carefully examine the possible pricing options that a software company can employ in order first to become established in the software industry and then to increase its market share. The strategies and practices mentioned here can be adopted to fight competition and to ensure a flexible presence in a quickly changing competition field. The software industry is constantly under the pressure of emerging technologies and new business models. The top management of a software company has to design carefully their pricing policy, select their target group, remain flexible when the market conditions change and be careful not to publish a too straightforward pricing policy as the fuzziness is the key advantage in maintaining and increasing the software sales.

11 Pricing complexity: a funny approach

The fact that software pricing is so obscure and increasingly complex inspires people to joke on it.

What if... License Agreements Ruled the World..? by Gayle-Edwards

CUSTOMER: How much for a gallon of regular-gas?

SALES PERSON: How Many?

CUSTOMER: One gallon.

SALES PERSON: No... how many cylinders in your car? And, how many people will ride in the car?

CUSTOMER: What!!?

SALES PERSON: We don't sell gasoline as a product, but rather, we license it based on the value of the "use" you'll be getting out of the gas. Obviously, you'll be getting a better value if you are using the gas in a higher-performance engine, or are transporting more people, ...and therefore it's only reasonable that you should pay more...

CUSTOMER: You're kidding..?

SALES PERSON: No sir. This is a standard business-practice.

CUSTOMER: But, what if I realize that I need to use some of the gas, that I bought here, ...in another engine, ...say, ...in my lawn-mower?

SALES PERSON: That is not allowed. You would effectively be "pirating" the use of the gasoline for "unauthorized purposes".

CUSTOMER: How is that..? I paid for the gas?

SALES PERSON: Actually, since you only licensed its use, such an action, on the part of a consumer, would clearly be a violation of OUR rights as a service-provider. By using our gasoline you agree to certain reasonable restrictions on the product's-use. These restrictions actually protect both of us. In fact, you could face legal-prosecution if you undertook such a selfish course of action as to use a product, ...that you ONLY licensed, ...in an un-approved manner.

CUSTOMER: How am I supposed to know about these "use restrictions"?

SALES PERSON: They will be briefly displayed on the front of the gas pump before it begins pumping. Actually, you have to agree to these terms before it will begin delivering gas. However, the "license agreement" is quite long, and somewhat full of legal-ese, so most people just accept the terms without really reading them.

CUSTOMER: What if I don't agree to the "terms"?

SALES PERSON: Then you are not allowed to get gasoline.

CUSTOMER: So, I'll get a refund?

SALES PERSON: Well, no. You see, it is also standard policy to refuse to refund a purchase after the pumping process has begun. This is to protect our commercial-interests, ...indeed the very economy depends upon such basic protections for business. After all, you could be trying to cheat us by refusing to obey our license-agreement.

CUSTOMER: What about MY protections?

SALES PERSON: Actually, the EULA[45] also eliminates any liability on our part.

CUSTOMER: So... What if the gas is no good..? What if it destroys my engine..? What if there is a flaw in the gasoline which causes my car to explode, and kill me..?

SALES PERSON: By using our gasoline, you also agree to NOT hold our company responsible for any negative-consequences. Honestly, it has to be this way. No business could survive if they could actually be held responsible for the quality of their products and services. That's just the way business operates today.

CUSTOMER: I think I'll go somewhere else.

SALES PERSON: That is your choice, ...but I must inform you that, due to cross-licensing deals with your automobile's manufacturer, using any other brand of gasoline will "Void Your Warranty", and may constitute a violation of your automobile's "End-User-License-Agreement".

CUSTOMER: ...and thereby make me subject to civil, or even criminal, litigation..?

SALES PERSON: Exactly. It's only reasonable, ...and, ...its in everybody's best interest.

12 Acknowledgments

The author would like to thank her anonymous software engineering colleagues, the conversations with whom, was the inspiration of this thesis.

[45] End-User License Agreement

13 References

Βέττας, Ν., Κατσουλάκος Ι., 2004. Πολιτική ανταγωνισμού και ρυθμιστική πολιτική. Τα οικονομικά των ρυθμιστικών παρεμβάσεων σε αγορές με μονοπωλιακή δύναμη. Τυπωθήτω, Αθήνα

Παρατηρητήριο για την ΚτΠ, Μάρτιος 2007. Χρήση Λογισμικού Ανοιχτού Κώδικα (Open Source)

Accenture's Global Content Study, 2008. The Challenge of Change: Perspectives on the Future for Content Providers

Baranes, E., Le Blanc, G., Jul 2006. Bundling strategies in ICT sector: an introduction, Communications & Strategies

Bhargava H.K, Choudhary V., May 1, 2008. Research Note: When Is Versioning Optimal for Information Goods?, Management Science

Braue, D., Sep 4, 2008. Feature: Ad-supported software, ZDNet.com Australia

Bontis, N., Chung, H., 2000. The evolution of software pricing: A framework form box licenses to application service provider models, Journal of Internet research, Volume 10, Issue 3, Pages 246-255

Cabral, L., 2000. Introduction to Industrial Organization, MIT Press, Cambridge

Choudhary, V., Tomak, K., Chaturvedi, A., 1998. Economic Benefits of Renting Software, Journal of Organizational Computing and Electronic Commerce, Volume 8, Issue 4, Pages 277-305

Choudhary, V., 2007. Comparison of Software Quality Under Perpetual Licensing and Software as a Service, Journal of Management Information Systems, Volume 24 No 2, Pages 141-165

Controller's Report, Jun 2004. Software Pricing, IOMA Publications, Volume 2004, Issue 6, Pages 8-9

Cusumano, M., 2004. The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know to Thrive and Survive in Good Times and Bad, Free Press

Cusumano, M., Jul 2007. The Changing Labyrinth of Software Pricing, Communications of the ACM, Volume 50, No 7

Dutta, S., Mia, I., 2008–2009. The Global Information Technology Report, Mobility in a Networked World. World Economic Forum, Geneva, Switzerland

Economist, April 20, 2006. Universal service?, Pages 61-62

Enslow, B., Aug 2006. Software as a Service Buyer's Guide, Aberdeen Group

Evers, J., Aug 14, 2006. Microsoft's antivirus package makes a splash, CNET News.com

Farrell, J., Klemperer, P., 2007. Coordination and Lock-in: Competition with Switching Costs and Network Effects, Handbook of Industrial Organization, Volume 3, Elsevier B.V.

Gartner, 2008. Gartner Identifies Six Best Practices Companies Should Consider Before They Virtualize Their Servers, Press Releases

Hanson J., Kazi S, Florio R, 2008. Software Pricing: Getting Back to Growth, Accenture

Hanson J., Mar 2009. New Strategies are Needed to Resolve Growing Software Pricing Problems, Software Business Executive Report

Harmon, R., Raffo, D., Faulk, S., 2004. Value-Based Pricing For New Software Products: Strategy Insights for Developers, Portland International Conference on the Management of Engineering and Technology

Hewlett-Packard, Sep 2, 2008. Power of Virtualization Largely Untapped Despite Massive Adoption, California

IT Utility Pipeline, 2004. Software: The Subscription Pricing Model InformationWeek.

Katz, M., Shapiro, C., 1985. Network Externalities, Competition, and Compatibility. The American Economic Review. American Economic Association

Koehler, H.D., 1996. Pricing Considerations for Electronic Products in a Network and Requirements for a Billing System, VCH Publishing Group, Weinheim, Germany

Kortge, G.D., Okonkwo, P.A., 1993. Perceived Value Approach to Pricing, Industrial Marketing Management, Volume 22, Pages 133-140

Larkin, I., 2008. Bargains-then-ripoffs: Innovation, pricing and lock-in in enterprise software, Best Paper Proceedings of the Academy of Management

MacKie-Mason, J.K., Varian, H.R., 1995. Some FAQs about Usage-Based Pricing, Computer Networks and ISDN Systems, Volume 28, Pages 257-65

McAllister, N., Nov 27, 2008. Is an advertising-supported model the way forward for the software industry?, InfoWorld.com

McKinsey & SandHill, 2007. Enterprise Software Customer Survey

McKinsey & SandHill, 2008. Enterprise Software Customer Survey

Messerschmitt D., Szyperski C., 2003. Software Ecosystem: Understanding an Indispensable Technology and Industry, MIT Press

PriceWaterHouseCoopers, 2007. Software pricing trends. How vendors can capitalize on the shift to new revenue models

PriceWaterHouseCoopers, 2008, EuroSoftware100, Key players & market trends.

Reyes E., Jan/Feb 2008. Ad-Supported Nation, Revenue Magazine

Robinson B., Oct 30, 2006. The next wave of software licensing arrives. Trends such as software-as-a-service, virtualisation and multicore processors are changing the way users pay for applications, Computerworld New Zealand

Shapiro C., Varian H. R., 1999. Information Rules: a strategic guide to the network economy, Harvard Business School Press, Boston, Massachusetts

Shefer, D., 2007. Pricing for Software Product Managers, The Pragmatic Marketer, Volume 3, Issue 5

Stansberry, M., Jun 28, 2005. CA extends usage-based pricing to SMBs, SearchDataCenter.com

Sullivan, T., Schwartz, E., 2001. Retooling software pricing: Vendors look to add usage-based models, Infoworld, Volume 23, Issue 2

Thibodeau, P., Dec 6, 2004. Lack of Tests Could Block Virtualization, Computerworld Inc

Varian, H., 1997. Versioning Information Goods, Working Paper, University of California, Berkeley

Whitehurst, J., Sep 25, 2008. People shouldn't pay for software, Economic Times, New Delhi

Web references

Anderson J., Jan 9, 2009. Why Product Mangers Need To Know That Cost Plus Pricing Is Wrong, Wrong, Wrong! http://www.theaccidentalpm.com/pricing/why-product-mangers-need-to-know-that-cost-plus-pricing-is-wrong-wrong-wrong

GNU Project, http://www.gnu.org/

IBM, Apr 1, 2009. Processor Value Unit [PVU] licensing for Distributed Software. http://www-01.ibm.com/software/lotus/passportadvantage/pvu_licensing_for_customers.html

Morales, A., Sep 14, 2006. Making Multi-Cores Count: An ISV Licensing Primer http://developer.amd.com/documentation/articles/pages/9142006142.aspx

Report on Computer Software Rental Act, Dec 1,1990. http://www.copyright.gov/reports/software_ren.html

Smith, J., Jun 6, 2007. Skimming pricing and penetration pricing. http://www.web-articles.info/e/a/title/Skimming-pricing-and-penetration-pricing/

Spolsky J., June 12, 2002. Strategy Letter V. http://www.joelonsoftware.com/articles/StrategyLetterV.html

The Business of Software, Sep 19, 2005. http://discuss.joelonsoftware.com/default.asp?biz.5.210074.3

http://www.microsoft.com/hellas/licensing/programmes/explained/default.mspx

http://www.microsoft.com/hellas/licensing/programmes/openvalue/ovcw/default.mspx

http://www.microsoft.com/smallbusiness/buy/software/buy-software.aspx#VolumeLicensing

http://office.microsoft.com/el-gr/suites/FX101677751032.aspx

http://www.ibm.com/developerworks/data/library/techarticle/dm-0611 zikopoulos2/

http://www-01.ibm.com/software/rational/howtobuy/licensing/

http://www-01.ibm.com/software/awdtools/changemgmt/

http://www-03.ibm.com/servers/eserver/zseries/swprice/other/

https://www-112.ibm.com/software/howtobuy/buyingtools/paexpress/Express?P0=E1& part_number=D52BILL,D52BKLL,D55TULL,D55MBLL,D58NJLL,D58NNLL,D588YLL,D5890LL,D58UULL,D58UXLL,D58MSLL,D58CPLL,D5910LL,D60PLLL,D593PLL,D59QTLL&catalogLocale=en_US&locale=en_US&country=USA&PT=html

http://www.research.ibm.com/journal/sj44-2.html

http://www.conn-x.gr/Default.aspx?MenuID=64

http://www.m-stat.gr/index.php?page=2

http://gadgetvote.com/blog/micosoft_reduce_windows_vista_price.html

http://en.wikipedia.org/wiki/Internet_Explorer

http://www.imerisia.gr/article.asp?catid=12319&subid=2&pubid=5177111#