Master Thesis
Software Engineering
Thesis no: MSE-2004:04
June 2004

Alternative methods of financing software

- Replacing illegal copying with free copying

Patrik Mjörner and Erik Bosrup

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author: Patrik Mjörner
E-mail: pt99pmj@student.bth.se

Author: Erik Bosrup
E-mail: erik@bosrup.com

University advisor: Anders Hederstierna
Department of MAM, management section

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/ipd
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

ABSTRACT

The software industry's largest and most severe problem is software piracy. Most attempts to stop illegal copying of software have failed. Alternative ways to remove software piracy must therefore be discovered and some are already being used. These ways include alternative methods of financing software products instead of trying to stop illegal copying. The alternative methods use the low cost of software duplication as an advantage to increase sales instead of seeing it as a problem. This requires a change in the business model to focus on other revenue sources than per-copy fees.

Many different alternative financing methods exist. These are suitable under different conditions and there is no universal solution. The methods all have one thing in common. The user can legally copy and use the software product without purchasing a license. All have different positive and negative effects on the user and this can create the opportunity to use several business models to cater different market segments.

Keywords: Software piracy, Adware, Spyware, Private provisioning, Support seller, Bundling, Content provider.

CONTENTS

1 INTRODUCTION

1.1 Background

Software has the characteristics of information goods and that makes it possible to copy them easily without degrading the copies. This has created a problem with illegal copying that affects the entire software industry. Solving the software piracy problem is not an easy task. Numerous attempts to hinder piracy have been tried, e.g. the establishment of anti piracy bureaus, the introduction of copy protection etc. So far the piracy problem is still an unsolved issue for most software publishers as the countermeasures tried so far have not been able to stop illegal copying.

There exits many threats to the traditional way of financing software products by selling licenses. Seen from historical perspective software products were considered public goods and anyone who wanted the software product could get it for free. The market of selling software products have gone through dramatic changes since then and is now a multi billion industry. The problem is that consumers still consider software products as public goods why software piracy is generally accepted. Software piracy is actually theft, but convincing the consumers that software piracy is wrong is not an easy task as the duplication of information goods does not reduce the value of the original product.

The solutions presented in this thesis do not see copying as a problem. Instead of fearing copying because of lost revenues software developers will be favored by the copying phenomenon. Instead of fighting the hard battle against the copiers the software developers utilizing the solutions presented in this thesis will actually encourage copying. Instead of trying to stop software's intrinsic characteristics of easy and costless duplication, we try to look at software from the other perspective. We examine a number of business models that use copying to the software's benefit and make use of the near zero duplication cost. This means that instead of charging for the software license, the revenues have to come from a different source, which may be the end user, but also from someone else who can use the contact with the user.

We begin our thesis by looking at the problems and opportunities surrounding software and its special characteristics as information goods. Understanding why software is easy to copy and why that is hard to stop is important to understand as the business models we describe act as alternatives to fighting illegal copying.

The reason why alternative ways of financing software got our attention was because one of the authors had attended Economic Driven Software Development at Blekinge Institute of Technology located in Ronneby, Sweden.

1.2 Structure

The thesis is divided into four main chapters: introduction, problem description and background, description and discussion of alternative methods, and conclusions. In the problem description and background chapter, problems and some information associated with software will be further described. Alternative solutions for the problems found by the authors are described and discussed in the alternative methods chapter. A closing discussion and the conclusions that were drawn from the authors work is found in the final chapter.

1.3 Audience

This thesis is written to be accessible to anyone interested in alternative methods of financing software. The audience may include software engineers, business managers, students, and other professionals in the software or information goods industry. Basic knowledge of software development is assumed. Included is an introduction to the economic aspects of software products and information goods where we describe the principles that are fundamentally needed for continued reading.

1.4 Research goals

The purpose of this thesis is to analyze alternative methods of financing software. This is done from a business and economic perspective. The goals are to:

- Find alternative financing methods that are currently used in software products.
- Identify under which circumstances these alternatives methods are usable in an economic context.
- Identify the types of software products that are suitable for alternative financing techniques and which are not.

1.5 Limitations and scope

This thesis is equivalent of two students' full time work during twenty weeks. This limits what is possible to include in the thesis and we have therefore focused on a certain area within this domain.

- There are other business models that aren't harmed by piracy such as the application service provider model that hosts the entire application remotely and loss-leading models that use free software to generate demand for an application that needs to be purchased. These are out of this thesis' scope, but yet interesting.
- It is likely that there are other alternative methods of financing software used today that we have not been able to identify.
- Our descriptions of the business models are brief and could be more detailed. Some of the models also have sub categories with different attributes that possibly could be divided into smaller business models.

1.6 Methodology

This thesis is solely built on a theoretical emphasis. No empirical studies or experiments have been performed. Practical studies have however been performed to some degree. These practical studies consisted of tests of free software utilizing alternative financing methods. No tests regarding the suitability of the alternative financing methods can be performed why the results of these tests will be limited.

The first phase consisted of information search. During the information search further questions were raised as the authors' understanding of the problem was heightened.

The business models described in this thesis uses a, by the software industry accepted and used, special nomenclature. The terms used is therefore briefly described in order for the authors as well as the readers to gain an understanding of the domain.

2 PROBLEM DESCRIPTION AND BACKGROUND

2.1 Problem domain

2.1.1 Information goods

"Anything that can digitalized – encoded as a stream of bits – is information" [Shapiro, 1999]. This broad definition of *information* permits goods such as movies, music, software products and even books to be considered as *information goods*.

2.1.1.1 Production and reproduction price

The major characteristic of information goods is production and reproduction price. Information is very costly to produce but cheap to reproduce [Shapiro, 1999]. Once the product is completed all costs associated with the development are non-refundable. Information goods therefore have very high initial *sunk* costs. The reproduction of an information good such as software has very small variable costs. As shown in figure 2.1.1.1, cost per item drops very fast. Just think of a Hollywood movie. Production of a Hollywood picture is extremely expensive with sky high salaries for actors, special effects etc. Reproducing this movie, once it is made, is however very inexpensive and costs normally just include the cost of a blank VHS cassette. Information goods copied over the Internet have even lower reproduction costs.

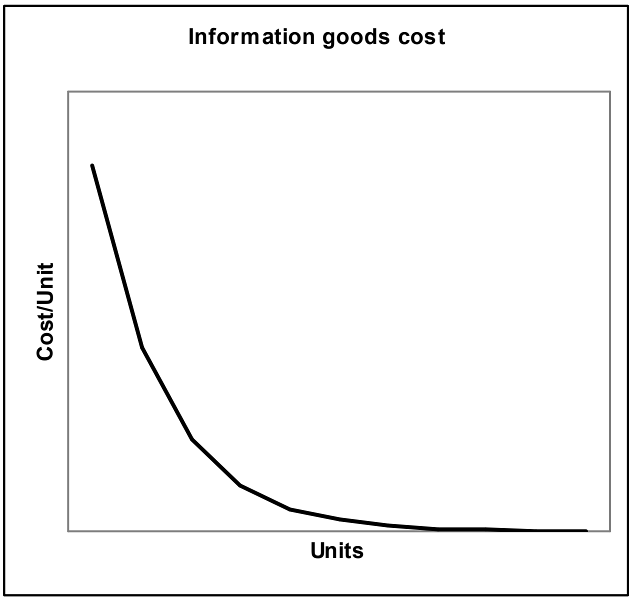

Figure 2.1.1.1

2.1.2 Pricing information goods

From a microeconomic perspective, or the perspective of the individual organization, price is the single most important decision in marketing [Rowley, 1998]. Pricing your information good right is imperative for every software publisher. Pricing an information good is however very hard. Old conventional economics advocates that *cost-based pricing* should be used whenever possible. When utilizing cost-based pricing a margin which constitutes as developers profit is placed on top of the marginal cost of producing the good e.g. placing a 20% markup on top of production cost.

Marginal production costs for information goods is however near zero why this kind of approach can't be used [Altinkemer, 2002].

Information is not quantifiable in terms of "normal" physics and can therefore not be charged for by using quantitative measures as with mass-produced commodities. Further, information has no intrinsic value, its worth and value is dependent upon its use and context. Pricing information goods is further aggravated because of its special characteristics: information does not depreciate when transmitted to another party, neither does an individual lose by transmitting information, nor does it diminish in value when used. This means, information can be sold and kept at the same time thus adding to possible difficulties regarding pricing [MADAME, 2000].

Time is a very important factor regarding information goods. The lifespan and lifecycle of an information good is usually much shorter than for other types of goods [Shapiro, 1999]. Technological limitations of hardware and technology of software engineering itself is in a state of constant flux, hence making the life span of information goods much shorter. The short life-cycle of information goods affects pricing decisions of information goods.

2.1.2.1 Value based pricing

As data included in an information good can't successfully be measured, nor production cost can be used to guide price settings something else is needed. "Value-based pricing uses buyers' perception of value, not the sellers cost, as the key to pricing" [Kotler, 2003]. Shapiro further elaborates of how an information good should be priced using value-based pricing, he states that: pricing an information good "should be done according to its value, not its cost" [Shapiro, 1999]. *Value-based pricing* tries to adjust the price according to customers perceived value of the product, e.g. price is set in relation to how big internal cost savings customer can obtain by using your software.

Value-based pricing is very rare among companies [Stedman, 2000]. It is rare because value-based pricing might scare customers away from purchases thus resulting in fewer sales. The idea of paying for a product based on its possible and future use is fairly unestablished among customers why many might feel reluctant to pay the requested amount.

2.1.2.2 Know your customer

An information good usually presents a different value for different consumers as customers place different values in different information goods according to their needs and wants. Some customers might request certain business information and therefore place high value on this kind on information goods while others find this information very redundant why their value of this good is much lower. Finding the value that can be made general for all customers regarding information goods is very difficult [Steadman, 2000]. In order for the right price to be set for an information good thorough knowledge of your customers is needed. Retrieving customer information is both difficult and expensive, thus making pricing harder. Retrieving this information is usually done by using surveys and personal interviews of distinction customers.

2.1.2.3 Personalized products and prices

It is easier to charge for a product that is requested. In order to produce requested products a thorough knowledge-base of what is expected and requested is needed. The knowledge needed is also known as *customer knowledge*. When knowing your customers preferences, likes and status producing personalized products (product differentiation) is reachable. Product differentiation includes anything ranging from completely customized products to merely small changes in design of content. Having personalized products admits to having personalized prices (price discrimination) thus solving parts of the information pricing problem [Shapiro, 1999]. Price discrimination is the name for selling your products to different customers using different prices hence maximizing possible profits [Kotler, 2003]. In order for product differentiation accompanied by price discrimination to work you must be able to modify your products (versioning). Modifying an information good is usually much simpler than with a normal commodity why information goods are to be considered very suitable for this approach [Odlyzko, 2003].

2.1.3 Network effect

"The network effect occurs when the perception of a product's value to a consumer depends on the number of other people using the product" [Tse, 2002]. This means, a product with a large user-base will attract even more users merely because of its high number of users (network externalities). It also means that a product, even if it is inferior to competitors, can attain a monopoly and dominate the market provided it has a large installed user-base. The key concept of the network effect is *positive feedback* Positive feedback makes the strong grow stronger and the weak grow weaker thus resulting in a winner-take-all market situation where only one competitor can survive [Spears, 1998].

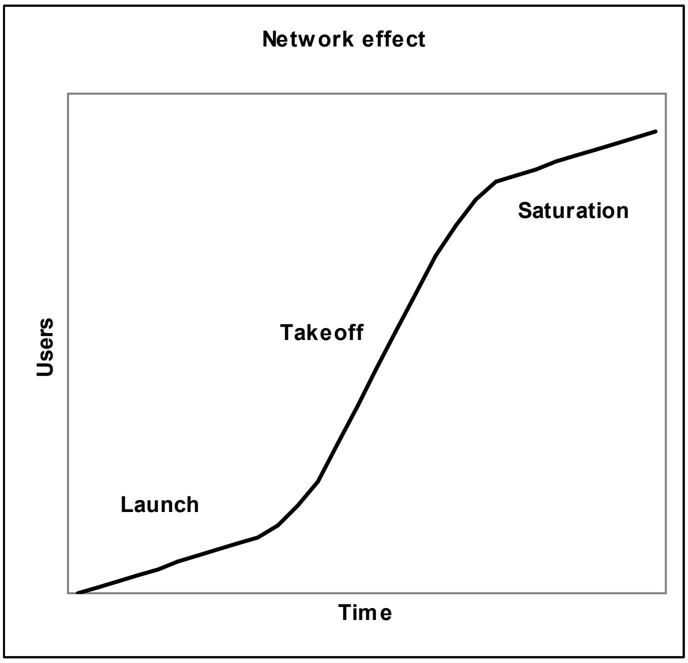

Figure 2.1.3

The users themselves are the reason why network effects exist. The network effect is driven by users' strong desire to select the technology that ultimately will prevail [Shapiro, 1999]. Nobody wants to buy the wrong technology and be left behind why buying the technology with most users comes natural. The network effect follows a very predictable s-pattern described in figure 2.1.3.

Another reason why the network effect exists is because of compatibility issues. Being the only one utilizing a technology has major drawbacks compared with using the conventional technology: if you are one of the few using this technology, you may encounter problems connecting and sharing your work with others. Technologies with fewer users usually attract less attention from third parties, hence reducing the availability of complementary products.

2.1.4 Lock-in

Lock-in occurs when it becomes too expensive or difficult for a customer to change from one brand or technology to another because of *switching costs* [Shapiro, 1999]. Switching costs measure the extent of a customer's lock-in to a given supplier.

Switching costs can be divided into two categories: transaction costs and learning costs [Nilssen, 1992]. Transaction costs are incurred when cost of changing from one supplier to another is substantial, e.g. a company might have purchased a very expensive system and this may hinder them from purchasing a new. Learning costs are very often included with information goods such as software products. In order to use a new system all operators must be instructed and learned, thus very high learning costs are introduced, hence keeping companies from changing.

The factor of lock-in can either be seen as very positive or negative depending on your current position in the buyer-seller relationship. If you are a customer you will most definitely seek not to be locked-in whereas you as a technology provider will seek to lock in as many customers as possible.

2.1.4.1 Lock-in effect on the network effect

The benefit of enjoying the network effect regarding sales of your product is something that most software publishers seek. In order to obtain the benefits of the network effect a large installed user base must be present. Price is generally a very useful tool which can be used in order get the large and needed user base, e.g. by using a market-penetration price. However, once the user base is installed price has only a very limited influence upon the network effect. Keeping users by lowering the price is impossible, as they already have purchased the software.

Generally, software publishers are very keen on keeping their users once the large user base is installed. The network effect is dependent upon the number of users in the installed user base. If the size of the installed user base drops, the number of new users also drops. A visible relation between these two parameters therefore exists. Lock-in is used in order to keep the users in the installed user base which is an advantage for the network effect.

2.1.5 Bundling

Information is often sold in bundles, e.g. a magazine is a perfect example of a bundle. A magazine usually contains information that interests certain customers, but it also contains information that is less interesting. By utilizing bundling consumers are however forced to buy the whole package including the interesting material as well as the less interesting material.

By utilizing bundling sellers combine different software products into one package. Reasons for bundling may vary. Bundling usually makes customers less dispersed thus leading to bigger revenues. Bundling may provide economics of scale by admitting to larger quantities produced [Chuang and Sirbu, 2000]. Bundling's primary use is however to increase sellers' revenue as bundling tries to extract maximum consumer surplus [Shapiro, 1999].

Bundling can be seen as a form of price discrimination [Altinkemer, 2002]. By utilizing bundling the number of products to price drops significantly, thus reducing the need for expensive user-needs research. Bundling also makes the number of different users fewer by delivering several products packaged into one.

One prerequisite which must be fulfilled in order for bundling to work is customers' perceived surplus from buying the products in bundle. Meaning, a purchase discount must be made by buying the products in a bundle instead of buying them separately. This perceived consumer surplus can be reached either by bundling a few requested products and giving customers discount over buying them separate, or by bundling a large number of goods thus making the consumer believe he receives a lot. It has been concluded that the variable reproduction cost of an information good normally is very small, hence the possibility to reproduce many different information goods is made larger. The bundling of hundreds or thousands of even unrelated information goods is therefore possible and could be a profitable strategy [Bakos, 1998]. The overall efficiency of bundling can still be kept high even if bundled information goods are unrelated [Bakos and Brynjolfsson, 2000]. The bundling of large quantities of traditional goods has historically been unprofitable as the increased number of goods that are included in the package results in higher manufacturing and distribution costs [Altinkemer and Jaisingh, 2002]. This is however not the case with information goods.

2.1.6 Open source, free software and freeware

2.1.6.1 History

Free software originates from the early days of computing when the computer industry sold huge computers to large organizations. At that time selling hardware generated the profits and the software was supplied free of charge [Rosenberg, 2000]. This fitted the academic culture of sharing new ideas, and software was copied freely. Eventually the availability of source code and free copying ended and the loss of that freedom urged Richard Stallman to found the Free Software Foundation (FSF) [DiBona et al, Stallman, 1999].

The goal of the Free Software Foundation is to create an entire operating system, and its surrounding tools, freely available to all [DiBona et al, Stallman, 1999]. These tools later made it possible for Linus Thorvalds to create the Linux kernel and start the wide-spread use of open source software [Raymond, 2001].

2.1.6.2 Public domain

The easiest way to make software free is to place it in the public domain, where the author surrenders his copyright rights. Such software can be seen as free to do whatever you wish with. It does not come with a license. Users other than the author can also apply a license of their choice, distribute, and sell the software in that form. [DiBonda et al, Perens, 1999]

2.1.6.3 Free software

Free software is the term used by the FSF for software licenses that fulfill certain criteria. The term free is often misinterpreted as being free of charge. The term free however, is about “the users’ freedom to run, copy, distribute, study, change and improve the software” [Free Software Foundation, 2003].

2.1.6.4 Open source

One reason for using the term open source instead of free software is to avoid the confusion with free as in price. Open source and free software describe the same type of software in almost all cases [DiBona et al, Stallman, 1999]. To call a software open source, its license agreement must fulfill a ten item criteria. Its main parts are free redistribution, source code inclusion, allowing derived works, and no discrimination. At the time of this writing there are 47 approved licenses, with the Free Software Foundation’s GNU GPL license being the most popular [Open Source Initiative, 2003][Lerner and Tirole, 2002].

The main difference between the free software idea and the open source counterpart is that free software intends to make all software using parts of the system free as well. The open source definition does not include such requirements in its definition. Software applying the free software GPL license is open source software, but an open source license may not necessarily fulfill the requirements of free software. Which approach that is the best varies and we will use open source as our term as it allows wider flexibility, is more established, and is not confused with for-free software.

2.1.6.5 Open source licenses and their uses

There are many open source licenses available today and the most common ones after the GPL (GNU General Public License) are the Apache License, the BSD License, the MIT License, the Artistic License, and the Mozilla Public License [Lerner and Tirole, 2002]. These all vary in what is allowed and required from the parties.

There are two major features that set them apart from each other. The first is whether it allows the software to be mixed with non-free software on source code level, or requires these to be distributed under the same terms if they are mixed. That is known as a *viral*, *reciprocal* or *tainting* license. The second is if modifications that are made are required to be made available as source code to others or can be used without releasing the source code of the changes. This is what’s called *copyleft*. [DiBona et al, Perens, 1999][Lerner and Tirole, 2002] [Free Software Foundation, 2003].

As the copyright holder of an open source program there are many licensing options. One of these is to dual-license the software where the same code can be made available under the GPL (a license that taints) and a normal commercial license. This however requires that all outside contributors agree to transfer their copyrights [Fink, 2003].

2.1.6.6 Effects of open source

Many open source program evolve from solving an initial problem the project founder encountered [Lerner and Tirole, 2000]. After releasing the software, bug-fixes created by other users are typically sent back and integrated in the standard version. The reasoning behind this is that it is easier to use a standard version with one’s own fixes and features than maintaining your own patched version. With this interaction, important bugs are fixed and the features the users want are implemented. This creates a network effect where new fixes and features add to the offering’s value, which in turn will likely lead to more users and more development. This change in development model is described by Raymond as a *Bazaar* where people have different goals and methods to reach them and anyone can contribute. With short release cycles defects are less of a problem. The Bazaar model is in contrast with the traditional *Cathedral* model where plans are made beforehand, teams are kept small and releases are few [Raymond, 2001][Fink, 2003]. The open source development model can of course vary widely between projects, but common for them is the possibility for their *career concern* incentives and *ego gratification* incentives to have a larger impact due to the transparency of open source projects. These incentives can also have a network effect on the amount of contributions to the project [Lerner and Tirole, 2000].

Corporations have been reluctant towards open source because of tainting licenses. Such fears can have its place, as it is important to know the implications of a license before an application is used. A tainting license may not necessarily be a problem if the source code used together with it may as well be released as open source. A tainting license is never a problem when the source code is not mixed, so open source application usage is not a problem. Another corporate aspect of open source is the reduction of lock-in. While users still get locked-in to a certain software application, open source makes it possible for companies to make modifications themselves or have several firms compete for contracts and hence possibly reducing costs for new functionality.

2.1.6.7 Freeware, shareware, demoware, shared source and others

There are many other licenses and distribution forms that are similar to open sources licenses such as freeware, shareware, demoware, Microsoft's Shared Source, and the Netscape Public License. These are not open source. As previously described open source is not about using the software for free, or accessing its source code. Open source is about the right to change and distribute the software.

2.1.7 Public goods

There is not one single definition of a public good, but a commonly used definition is something that has the features of non-rivalness and non-exclusion. *Non-rivalness* means that one person's consumption does not reduce the amount available to others, and *non-exclusion* that someone cannot be excluded from consuming the good, even if he does not pay for it (free-riding). Clean air and national defense are common examples used to illustrate public goods [Ver Ecke, 1999].

Information goods that are freely available have the characteristics of public goods [Varian, 2001]. Copy protection may prevent the treatment of software as public goods as it can exclude users. Open source software however are public goods.

2.1.8 Peer-to-peer

Peer-to-peer (P2P) networks are often seen as synonymous to file-sharing networks such as Kazaa and the now closed Napster. Peer-to-peer networks are however more than just file sharing. Peer-to-peer technology is also used for distribution of hardware resources like storage, bandwidth, and processing power. There are many similarities with the client server model. The difference between them is that in the peer-to-peer network, the resources are located at client (peer) computers instead of the server. The transfer of resources between the peers also bypasses the server going directly from the source to the target, peer-to-peer. A P2P network is not necessarily server-less. Three different types of peer-to-peer networks exist: *centralized* where there is a server that helps locating resources among the peers; *distributed* where there is no server and the peers typically connect to a few other peers and pass along each others queries; and *hybrid* networks that use supernodes that are connected to each other in a distributed manner as well as being connected to peers and serving as a centralized server to them [Asvanund et al, 2003]. Shutting down Napster was possible by removing the centralized servers, without them the network does not function. Distributed and hybrid networks, including Kazaa and Gnutella, that do not use centralized servers cannot be shutdown through legal action targeted at one network point.

Peer-to-peer networks are dependant on the network effect. More users will add more value to the network (e.g. a user is more likely to find the file she is searching for on a network with many users). On centralized networks, there is also a negative effect where more users cause increased congestion, hence reducing the value of the network [Asvanund et al, 2003]. Free-riding can also cause problems in P2P networks. Free-riders use the network to access resources, but do not share their own resources. Such users do not add value to the network. The largest network today is Kazaa's, the software has been downloaded 280 million times from Download.com and is by far that site's most downloaded software [Download.com, 2003].

2.1.9 Target marketing

All customers are not alike [Sudjak , 2001]. Companies of today recognize that they cannot appeal to all customers' demands; instead they have to find their *target market*. In order to find a company's target market the whole market must be divided into smaller and more manageable groups (market segmentation). These customer segments are found and divided based on geographical, demographical, psychographical, and behavioral factors [Kotler, 2003]. Target marketing consists of choosing the best customer group from the market segmentation.

The *pareto principle* of money distribution states that 80% of all wealth was concentrated among 20% of the population [Sanders, 1988]. A common modern interpretation of the pareto principle gives that 20% of your most important customers account for 80% of your company's business. Finding the most important segment and marketing towards them is therefore imperative for your business.

Target marketing has now reached a new level with the advent of the Internet. The process of target marketing is very information dependent. Accessing this customer information is however not an easy task. The reason why target marketing has been given so much attention during the late 90's is because of the e-commerce phenomenon and the increased possibilities regarding the storage of customer information [Sudjak, 2001]. With computers comes the ability to store vast amounts of data. Before the time of computers, target marketing was a privilege given only to companies with a very limited number of customers as the collection of customer information needed to be stored was manageable. With help from databases and computers, storing thousands even millions of consumer profiles is possible.

The internet also presents further possible improvement for target marketing. Utilizing the internet consumer information can even be collected without consumers consent [Sudjak, 2001]. This means, information which is not intentionally provided by consumers can be collected; companies can spy on their customers with help from the Internet.

2.1.10 Digital Rights Management

Digital Rights Management (DRM) is a group of varying technologies aimed at monitoring, regulating and pricing the use and distribution of files. Its uses are among others to enforce rules limiting how many times a file is used, how long it can be used, or if it may be copied and then how many times and to which computer. This access control allows producers to stop unauthorized reproduction and also allows versioning and price discrimination of information goods depending on their use [Einhorn, 2003]. Software may then be sold in usage time or number of uses instead of a conventional license.

2.1.11 Privacy

Privacy is something that most consumers take for granted. Consumers should however not be so naïve regarding how companies they trust handle their personal information. Regulations and legislations regarding how specific personal information must be handled do exist in the western parts of the world, e.g. the UK Data Protection Act and the Swedish PUL. No universal law does however apply to all software applications and the Internet why consumers should protect their private information from misuse [Blight, 1997]. A common understanding is that people think they are anonymous on the Internet [Desai et. al., 2003]. This understanding is however very vaguely supported in practice. Actually, the Internet is a large tool for distributing data between peers. If you entrust your personal information to an unethical party, it may easily be distributed to other parties without your knowledge. Your personal information is an important merchandise which possess a certain value [HSBWK, 2003].

To whom consumers entrust their personal information is generally based upon subjective thoughts. An important aspect which generally is regarded when choosing whether to entrust the company personal information or not is company goodwill. During the last couple of years many privacy issues has been raised and questioned. These issues of privacy have been given a lot of attention why the privacy issue has become important to customers.

2.2 Problem description

Software piracy is defined as the unauthorized copying of computer software, which constitutes copyright infringement, for either commercial or personal use [SIIA, 2000]. Software piracy is one of the worst problems facing the software industry today [Givon, 1995]. In 1999, SIIA (Software & Information Industry Association) estimated that global sales of computer software totaled at $21.6 billion. SIIA also calculated that more than $13 billion in revenues was lost because of software piracy and that 36% of all computer software was used illegally [SIIA, 2000].

Software piracy has also established itself as an industry of it own. In third world countries and in countries with developing economies the growing demand for software products is not met by the appropriate publishers but instead by pirates of large scale. Often these software piracy industries attain a larger share of the market than the original publishers thus reducing profits for the rightful owners.

2.2.1 Software licenses

As the characteristics of software products allow for easy duplication and distribution, selling software products can’t be done using normal ways of transaction. Possession of computer software is almost universally transferred from publisher to end-customer using licenses. By selling licenses instead of selling software products the companies can legally restrict further distribution [Andris, 2002]. A typical software license contains restrictions about its use, distribution and duplication. The problem with software licenses is that most people in both the consumer and business user base have a least moderate belief in the sanctity of copyright laws concerning computer software. The general belief among users is that everyone who uses the Internet eventually violates copyright laws and disregard software licenses [SIIA-KPMG, 2001]. This well established belief gives way to the excuse for doing something: “everyone else is doing it, so why shouldn’t I do it?”

A software license can be written in multiple ways and not until recent years have standardized licenses been adopted by software publishers. The lack of standardized licenses has made it very difficult for consumers to handle software licenses which have resulted in their reduced importance. Software programs are also protected by copyright law. Most countries in the civilized parts of the world have different aspects of what the copyright law should protect, hence adding to the possible confusion of software consumers. Something called *fair use* is generally included in copyright laws. Fair use is a doctrine developed in order for some copying to be allowed. It states that some copying of copyrighted material should be considered legal in certain, limited circumstances. The necessary conditions needed to be fulfilled in order for fair use to be considered are however defined very broadly: the purpose and characteristics of the use, including whether such use is of commercial nature or is for nonprofit educational purposes; the nature of the copyrighted work; the amount and substantiality of the portion used in relation to the copyrighted work as a whole; the effect of the use upon the potential market for or value of the copyrighted work. These fuzzy criterions make the rulings of what is fair use somewhat arbitrary thus adding to even more confusion [LOC, 1999]. Changes in copyright law regarding allowed copying also confuse users.

2.2.2 Software piracy and the Internet

The Internet has made software piracy more efficient by supplying pirates with a full grown distribution network and almost unlimited resources in terms of both hardware and manpower. The accessibility of pirated software has also grown and the Internet has made it possible for virtually any novice computer user to gain access to pirated software.

Utilizing the Internet and peer-to-peer technology, software producing companies were given the opportunity to explore new ways of distributing their software. Software distribution over the Internet opened ways to new and uncharted economic models that in theory would mean broader markets thus resulting in larger possible profits.

The actual products being sold by the software producing companies - computer software – is nothing more than pure information (bits) combined in a special way. As the sole purpose of the Internet is to transport streams of bits over large distances the Internet is the perfect distribution medium for these information goods. The problem the software industry has to deal with lays in charging for the information. The reproduction – copying – of information goods is just as easy and cheap for the rightfully owner as it is for any one else [Shapiro, 1999]. Using today’s technology, almost anyone with access to a cd-r can virtually just as easy as the software publisher make an additional copy of the original.

2.2.3 Types of software piracy

Software piracy can be done in multiple ways, the most common forms of software piracy are: *End user piracy* which occurs when a person reproduces copies of software without authorization; *Client-server overuse* which occurs when too many employees on a network are using a central copy of a program at the same time; *Internet piracy* which occurs when software is downloaded from the Internet; *Hard-*

disk loading which occurs when a business who sells new computers loads illegal copies of software onto the hard disks to make the purchase of the machines more attractive; *Software counterfeiting* which occurs when someone duplicates and sells copyrighted software products with the intent of directly imitating the copyrighted product [BSA, 2003].

2.2.4 The extent of software piracy

The software industry is among the fastest growing industries [OECD, 2003]. The actual extent of software piracy is estimated by measuring the amount of software products installed without a legal license [BSA, 2003]. Measures of software piracy is therefore done by calculating the difference in demand for new software products and the actual quantity of software products legally shipped. The demand is calculated by observing new PC shipments, since for every new computer shipped new software is needed. Although accuracy is lacking regarding these figures calculations show large deviations between the estimated demand and actual delivered software products. Estimations also show large differences between countries and regions. The highest piracy rates are found in Asian countries such as Vietnam (94%), China (92%) and Indonesia (88%) [SIIA, 2000]. Piracy in the US does however account for the biggest revenue losses for the software publishers: $1.8 billion in 2000 [SIIA, 2000].

The fact that open source products are gaining a fair market share does however make these calculations less valid. IBM among other are currently delivering new computers with the open source operating system Linux pre-installed [IBM.com, 2003]. Even if new computers delivered with Linux pre-installed are few in numbers, they actually invalidate the figures published by SIIA and BSA.

2.2.5 Contributing factors to software piracy

According to Chung et al., the three most important reasons for software piracy are: “software too expensive”; “want to try out software”; “can’t afford software” [Hsing K, 1997]. Using at least one of those reasons – software being too expensive as the most important – software pirates justify their illegal acts when performing software piracy.

When comparing software goods with normal consumer goods a moral inconsistency can be found, as stealing a package of chewing gum would not be tolerated by society but stealing software goods worth thousands is generally fully accepted by the public [Rasmussen, 2003]. One contributing reason for this inconsistency is that no materialized quantities are stolen, merely copies are made and the original still remains untouched with full functionality - compare this with the characteristics of public goods. Another valid reason is that the software publishers do not loose anything quantifiable from the piracy act, just that they loose a possible sale. This lost sale is merely a possible sale because no evidence proving that the pirate actually would buy or needs the information good exists.

2.2.6 Effect of software piracy

The last twenty five years have seen an explosive growth in the software industry. The global software market now exceeds $370 billion in a rising trend [Commander, 2003]. The Business Software Alliance (BSA) claimed that the software industry lost “$13 billion per year”, “$35 million per day” and “$407 per second” [BSA, 2003]. The official statement from the software industry regarding this is that it poses a serious threat to the industry by constraining further growth, adversely affect investment decisions and limit the development of software products in regions where piracy is prevalent [Poddar, 2002].

2.2.7 Positive effects from software piracy

The software industry is affected by network effects such as positive feedback. In order for software publishers to attract customers a large user-base must be installed. Here within can a dilemma be found: with "help" from software piracy obtaining a large installed user-base is possible, but by doing so short-term revenues are lost. Several peer-to-peer file-sharing software products – such as Kazaa – are being blamed by software publishers for reducing their profits by allowing the distribution of pirated software. What might be forgotten is that these file-sharing software products also help software publishers to attain their desired large or larger installed user-base.

2.2.8 Stopping software piracy technically

Throughout history there have been numerous attempts to stop piracy by implementing software and hardware techniques that verify the existence of a valid license. Some of the past and present methods are described below.

Hardware dongles

A dongle is an extra piece of hardware attached to the computer. One dongle is provided with each license and the software can communicate with the dongle and verify its existence. It is typically used in low quantity, high-priced software.

Delivery media verification and copy prevention

A common protection method in computer games is to require the original media to be present while running the game. This method uses knowledge of the delivery media format and operating system and extends the format in ways that are not normally accessible by copying tools [Ernie, 1997] [DRM Guide, 2003]. Using this hidden information, the original media can be verified and prevented from being copied to an unauthorized copy.

Serial numbers

Serial numbers are usually distributed with the software's license agreement and are required during the installation where they are verified as authentic using checksums or white lists.

Product activation

Product activation is a recent development where software, after installation, connects to a registration service that assigns a customer's serial number with a specific hardware configuration. A serial number can then only be used once on one computer or the product will not continue to function.

Signed code

Code signing is used by the operating system to verify the integrity and authenticity of the software run on a system. [Huang, 2002] By keeping the keys secret, a system that runs only signed code and uses delivery media verification and copy prevention, can prevent copying.

Many forms of technical copy protection may appear effective but the common form of circumventing these copy protections is by a "crack" (change in the software) that by-passes the protection code and responds that the software is licensed. To stop such changes there are software protection tools. They aim to make changes harder. This is done mainly through three methods: *obfuscation* where the purpose of the code is hidden; *watermarking* where a secret message is placed inside; *tamper-proofing* where attempts are made to stop execution if the program has been changed [Collberg and Thomborson, 2003].

The effects of copy protection are limited. They may stop certain forms of piracy (such as end-user piracy of an original) and reduce piracy to some degree but complete protection is impossible. Since the software is available to any kind of attack once delivered to the host machine, copy protection and software protection tools only serve to slow down the process. [Collberg and Thomborson, 2003]

One should also note that copy protection mainly affects the legal users, not the pirates. Once copy protection is removed, a dongle, serial number, original delivery media and so forth is no longer necessary for the pirate while it serves as an annoyance for the legitimate user.

Copyright laws usually contain passages allowing “fair use”. What use that includes fair use may vary between countries but it often includes non-commercial or private use such as backups. By using technical methods to prevent copying, consumers may loose their right to fair use. The law is over-ruled by the actual possibilities.

2.2.9 Stopping software piracy legally

Legal action has always had a central position in anti-piracy actions. Primarily this is done by taking action against distributors, “warez-groups” (organizations of hackers with the purpose of removing copy protection and distributing illegal copies), Internet service providers, individuals, and companies. One part of this is to deter others from making illegal copies if they face the risk of punishment.

Together with copy protection laws such as the Digital Millennium Copyright Act (DMCA) in the USA this could have some effect on piracy. The DMCA addresses the problem in two ways: by prohibiting circumvention of technical protection and by banning devices that are designed to do so. Van Wijk divides the persons involved into two classes. One inner circle consisting of rivals, criminals, professionals and hackers who have the ability to overcome copy protection, and an outer circle of consumers where copy protection is effective. By outlawing the research on, use of, and distribution of decoding knowledge, hence stopping the knowledge from moving from the inner circle to the outer, copy protection will have more effect [Van Wijk, 2002]. By slowing down the diffusion of knowledge in the area piracy can be reduced, but not stopped.

2.2.10 Stopping software piracy through self-help

A third method of fighting piracy is to teach about piracy to try to change the behavior and morale of consumers. Usually this is telling people that piracy is theft, publishing figures of loss-estimates due to piracy, and publicizing court cases where the industry has been successful and won against pirates.

There is also a more hands-on approach to self-help. The BSA freely makes programs available to help determine legal and illegal copies, as well as providing shopping tips to help identify legal software. [BSA, 2003] Copy protection may also be seen as a form of self-help in the manner that consumers are helped not to do something illegal. [Yu, 2004]

2.2.11 Information goods in history

Other information goods industries such as the music and movie industries have previously fought, and are fighting, similar battles to the one in the software industry. The music industry has opposed the music cassette as well as public radio. The movie industry objected to video cassette recorders but in the end, despite their initial objections, lobbying, and legal action; the new technology has succeeded and not ruined their market [Shapiro, 1999]. The difference between this technology change and the previous is that the goods now are digital and that copies made are perfect.

2.2.12 Possible solutions

There are many possible alternatives for dealing with the software piracy issue. We list some of the possible solutions we know of but there may of course be others available.

Trusted computing environment

The hardware and software industries are moving towards trusted computing where the hardware will include cryptography, an embedded public and private key systems, trusted input, output, and storage. Information will be safe from tampering from entering the system until leaving. Together with software making use of this to avoid tampering and signing software specifically for a certain hardware system, it is possible to stricter enforce the rights of the copyright holders. [Microsoft, 2003][TCG, 2003][Naumovich and Memon, 2003]. While this is not the stated purpose of trusted computing, it is likely to be the main use. There are several reasons why this can fail. Consumers may not be willing to pay for systems that will limit their rights to fair use, or limit their possibility to copy freely [Kelsey and Schneir, 1998]. Systems that previously have tried similar approaches have failed. The copy protection of DVDs as well as Microsoft's X-Box (uses many methods to stop copying, including code-signing) have failed [DRM Guide, 2003][Huang, 2002]. Whenever the hardware is available for attack it can be altered to disable features necessary for copy protection and Digital Rights Management (DRM). Software bugs are also at risk of causing a breach in the trusted environment. E.g. the X-Box can, in certain circumstances, run unsigned code due to such a fault. In trusted computing one failed system in the security chain can result in compromise of the information aimed at protecting. Trusted computing may yet reduce piracy just as any other method as it will become more cumbersome to make illegal copies.

Legal action

The industry can continue with, and expand, its legal actions against offenders to forcedly uphold copyright laws and also lobby for further legal protection to increase the deterrence and punishments for offenders.

Changed moral

There is the possibility that the moral of consumers change and the willingness to pay for software increases. As long as software piracy is socially acceptable, piracy is likely to remain [Collberg and Thomborson, 2003]. Considering for how long piracy has been a problem a change in the near future is unlikely.

Fighting back

Senator Hatch, chairman of the Senate Judiciary Committee of the USA, remarked that he favored technology to destroy the computers of those who violate copyright. Congressman Berman also proposed a law with a similar intent regarding peer-to-peer networks [Yu, 2004]. Such actions risk alienating consumers as well as mistakenly targeting innocent users, resulting in bad publicity and reputation.

Removing the software model

One solution is taking advantage of the new high-speed networks and placing all software on servers and selling access to software services instead of software licenses. Having software installed on central and trusted servers is known as the Application Service Provider (ASP) model. With this model there is a risk of loss of performance and scalability [Naumovich and Memon, 2003]. Since it is centralized there is also a risk of data and value loss due to down-time or error in the third-party system.

Continuing, ignoring, accepting and using piracy

There is also the possibility to continue as present by introducing new copy protections and pirates responding with new ways around them, new laws to stop piracy, and new campaigns to stop illegal copying. Piracy can in many ways be ignored as long as there is a sufficient base of paying users. It can also be accepted and used to benefit the product through the network effects of its user base [Osorio, 2002].

Utilizing price discrimination

In order for software publishers to maximize their profits and possibly minimize their losses because of piracy some charge monopoly-high prices for their products. By charging high prices they also encourage piracy among low-value buyers [Harbaugh and Khemka, 2001]. One possible solution to the software piracy problem would be to adopt price discrimination. By utilizing true price discrimination high-value customers can be charged high prices along with letting low-value customers obtain the products more cheaply, thus removing one incentive to software piracy.

Levy

Levy is a possible solution for governments. Posting levy does not solve nor does it aid the software piracy problem in any way. By using levy, governments usually put an additional levy on the distribution-medium used for piracy [Jannuska, 2001]. This means, the cost for distribution mediums such as blank cd-r is increased as a levy is put on top of the regular price. The purpose of this levy is to be refunded to the entertainment and software industry as compensation for lost revenues. Problems arise when levy's are to be divided among the beneficiaries. The full control of the levy lies in the hands of the governments why software producing companies can't implement this solution.

2.3 Traditional licensing value chain

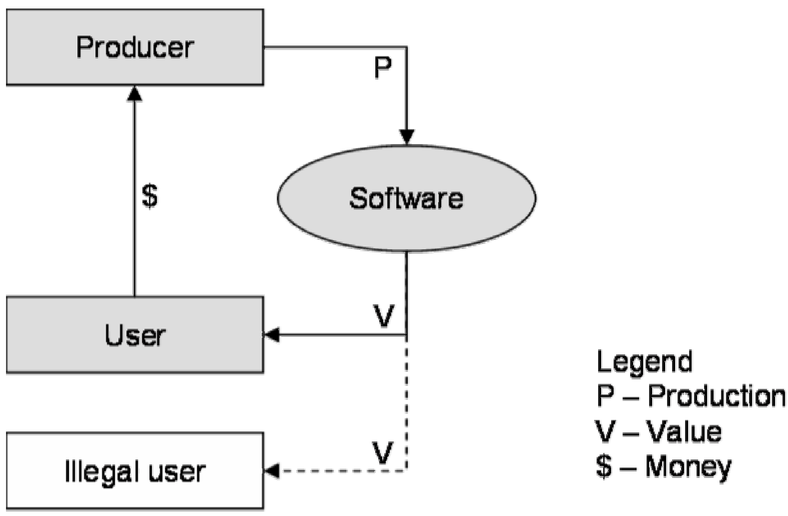

Figure 2.3

All value-chains described in this thesis contain at least two actors (boxes), the producer and the user. Their functions are the same for all value-chains. The producer is the company that produces the software. The user represents any user having access to the software legally. All software, as well as other goods, is represented by ovals.

In the traditional licensing model there are, along with paying users, illegal users. The illegal users are the users who use the software without having the appropriate legal licenses.

2.4 Research question

The selection of which alternative business model that is the most efficient for all kinds of software products is a hard task. All business models vary regarding efficiency. Their overall efficiency is determined in close relation with the attributes of the software product: product type, target market, etc. Therefore, this thesis will only contain descriptions of the business models we find most suiting rather than trying to find the most optimal one.

The following conditions must be fulfilled in order for a business model to be placed in this thesis. The business models:

1. revenues must be closely derived from the chosen software product
2. revenues must not be negatively affected by copying
3. must allow for companies to influence their inflow of revenues

The business models described have different characteristics and where appropriate we discuss how they interact in areas described in the problem domain. Namely we discuss the:

- competitive situation compared to other products
- product life-cycle positioning
- types of software and the target market
- pricing problem and how it is effected
- stability and viability of the revenues
- technical threats that may harm the model
- open sourcing possibilities
- end-user effects and how end-users might react
- lock-in and how it changes

Certain criteria won't be affected by some business models and for them no discussion will be done.

3 ALTERNATIVE METHODS

3.1 Complimentary Tangible Goods

3.1.1 Description

Sales of support, documentation, training, consulting, customizations, hardware, and other complimentary tangible goods is nothing new, it has been around since the early days of computing when hardware was the main income source. The sales of complimentary goods is a very important aspect of many corporate business models. For the open source software industry complementary goods are a primary revenue source and they are often called “support seller”, “widget frosting”, and “accessorizing” [Hecker, 2000][Raymond, 2001].

What we call complimentary tangible goods, are goods that are not required for running the software, but the need for them is likely to increase when the software is used. The focus is shifted from revenues generated from the direct sale to indirect revenues arising from the use of the software. While some of these goods may be possible to treat as information goods (digitalized support information, documentation, and Internet based training courses etc), they are typically requested in physical form or require some kind of human interaction.

By being the author or contributor to the for-free software, the producer will have knowledge to produce the best suited complimentary goods. With this knowledge the producer is also positioned well against competitors trying to do the same. The complimentary goods are often packaged together with the for-free software in a retail box.

3.1.2 Value-chain

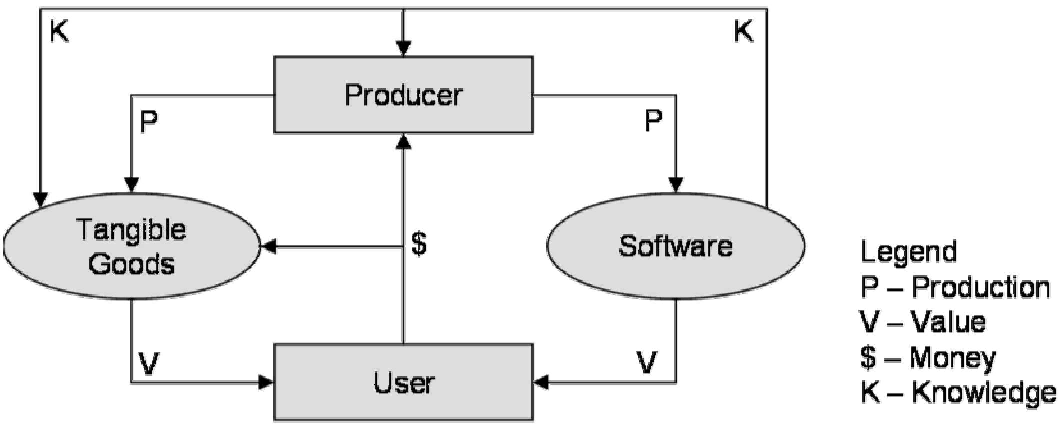

Figure 3.1.2

In the value-chain for complimentary tangible goods, the producer puts resources into producing the software and the complimentary goods. Both result in value returned to the producer, but in different ways. The software development and maintenance returns knowledge which will help the production of complimentary goods. The software creates a demand for the complimentary goods which in return generates the revenue for the producer. For the user, value is delivered by both the software and the complimentary goods, but she only has to pay for the complimentary goods.

3.1.3 Examples

3.1.3.1 Red Hat

Red Hat was founded in 1993 and is almost synonymous with Linux as it is one the most widely used distributions. Red Hat Linux consists of large amounts of packages of open source software. Red Hat bundles these into a system that is easy to install and to use. While Red Hat Linux is available for download at no charge, Red Hat also sells Linux distributions in different forms. The products range from basic subscription services (where Red Hat automatically keeps your system up-to-date) to fully boxed versions that include cd-rom's, printed manuals and multi-language, around the clock support. Red Hat also offers consulting services, training and certification for their products. Red Hat wants users to buy their product for the same reasons as consumers buy bottled water (when there is free tap water available) – Red Hat wants to offer convenience and quality. Although Linux is not their own product, Red Hat is involved in Linux's, and other open source software's, development. For the fiscal year 2003, Red Hat broke-even with $90.9 million in revenue, 47% of these from their services and 53% from their subscription services [Red Hat, 2003][DiBona et al, Young, 1999].

3.1.3.2 O'Reilly & Asssociates

O'Reilly & Associates is a computer books company that has been around since 1978 and published books since 1984. They were early to start using the Internet and today they run several topic focused sites as well as publishing parts or complete versions of their books on homepages. O'Reilly is involved in open source development to support their book sales. The open source language Perl is widely used to serve dynamic pages on the Internet. It was created by Larry Wall, who now is employed by O'Reilly to work on the language. Wall's name adds to the credibility and knowledge of the publisher, which can lead to better books that reach a larger audience [O'Reilly, 2003][DiBona et al, 1999].

3.1.4 Discussion

The most important aspect for the complimentary tangible goods business model is that it puts the focus on competing through knowledge instead of ownership; something that should interest many of today's knowledge focused organizations. With several companies delivering similar services and goods for the same software, the application needs to be developed in a way that the competition can be outpaced. One way to do this is to have frequent releases with new features, not only to stay ahead of other competing applications, but to be the leader in knowledge. New features will add to the product offering, as well as to your company's knowledge, while the competitors have to play catch-up. At the same time, these new features require new training, documentation, services and so forth from the customers which will help maintain the revenue stream.

Having frequent releases is close to the open source development model and this business model can use open source to its benefit. The reasoning for many companies using open source seems to be that since the supporting revenues are shared by several entities, the costs of development may be as well. A part from sharing the costs, open sourcing may expand the user base by its increased availability and reduce customer fears of lock-in due to its openness. One the other hand, open source has drawbacks in terms of knowledge protection. Revealing the source code to competitors will give them access to the same information base making it easier for them to attain the same knowledge level of the software's architecture as the original implementer. Open source also creates the possibility for competitors to use the same tactics you use against them – they can implement features and gain a head start in an area. Controlling the main open source distribution and requiring copyright ownership on all contributions like MySQL does, can help control such problems. Not open sourcing will help protect knowledge but at the same time it is not all that matters. Software like MySQL is technically advanced and requires a lot of skill and knowledge from the developer's wishing to start working with the source code. Complex software requiring domain knowledge like relational databases and operating systems have less to fear from open source than general end-user applications that are easier to understand and enhance for the average software developer.

Complex software is most suitable for open sourcing, and they are also the ones that work best with the complimentary tangible goods business model. Naturally, something that is hard and time consuming to learn is more likely to require assistance than something easy. Applications that are suitable would therefore include business suites, operating systems, databases, programming tools, and similar software. The buyers of these applications and goods tend to be in the business market. They are (or should at least be) more aware of total cost of ownership and lock-in; factors affecting the overall lifespan of the purchase. Compared to existing non-free competitors, a free and superior software product may not receive a large market share because of these factors. The initial purchase price is often a small portion of the overall cost of owning the product. For this model to be successful, the cost of the complimentary goods must also be held down to a competitive level compared to the service level offered. While open sourcing may reduce client fears of being locked-in, lock-in is still an issue. The category of complex software described is usually embedded deeply into the customer's core function and they are typically locked-in to a certain system and vendor. The fast rise of MySQL and Red Hat is largely due to the boom of new Internet start-ups that coincided with the growth of open source software. The Internet start-ups where not constrained by lock-in from old legacy systems, and could choose the most appropriate tools for the specific task. The process of taking clients from traditional vendors has gone much slower.

There are other threats to the complimentary tangible goods business model. As we mentioned in the description, some goods have the potential for digital distribution in the form of digitized books or courseware. However, most people prefer to read printed books, but that may change when "digital paper" arrives. The majority of the revenues come from services and traditional goods that are non-replaceable anyway so piracy is a small threat.

With free software more users can be expected to try the application and that could lead to a faster maturity growth – leading to revenues which typically arrive after the first use, when users recognize the need for support and training. Competing in this business oriented marketplace and with these types of software, a full range of goods is preferred. Buyers like to purchase the software, installation, training, support, documentation, certification, customizations etc in one package. Completeness and quality in providing these goods are just as important as the functionality and quality in the product you will be giving away.

Instead of pricing the delivered software, this model requires pricing of the goods and services. This is not necessarily easier than pricing software. Aspects such as service levels and quality can be hard to define and can incur losses if they are badly defined and not met. E.g. training and support that are expected to reach certain standards are very reliant on the users' previous knowledge, and that has to be assessed prior to any agreements. A delivered package might include training and certification of personnel. Should they have low levels of knowledge prior to the training, either more time has to be spent on costly training, or the requirements for certification have to be lowered. Neither are of course any good.

For the end-users, this business model has little effect. They may provide input to a decision of which product to choose, but other than increasing the availability of the product by making it free, this is not expected to impact significantly on either the end-user or the network effect.

3.2 Advertising

3.2.1 Description

The concept of delivering goods financed by advertisements is nothing new to the market. Commercial television financed by advertisement has been around for many years, is adopted, and generally approved of by the public. Back in the 1950s shows like the “Kraft Television Theater”, “Colgate Comedy Hour” and “Coke Times” filled the television schedules in America [Mashon, 2001]. These programs were sponsored by the advertising companies and free of charge for the viewers, consequently having an alternative source of revenue.

The trend among advertisers during this last decade has been to try to personalize advertisement messages as much as possible using *micromarketing* [Kotler, 2003]. Micromarketing is the practice of tailoring products and marketing programs to suit the tastes and likes of selected individuals. One subpart of micromarketing is known as individual marketing or one-to-one marketing. With the advent of the Internet and large computer databases one-to-one marketing is made possible in a large scale and is being explored by many Internet advertisers [Morris, 2000]. This means that having different advertisement messages shown for different users of a software application is possible. If the advertiser has access to information about the current user - e.g. demographic, geographic or behavioral information - one-to-one marketing can be pursued without too much trouble hence enlarging the possible impact of the ad.

Making your software application financed by advertisements is not complicated. In most cases the software producer leaves a blank space in the main window of the application’s GUI (Graphical User Interface) [Haar, 2000]. This blank space is then filled with an appropriate advertisement message suiting the advertisers’ wants and needs. According to agreements between the software producer and advertiser money is exchanged in return for having advertisers message shown to the software application’s users’.

Advertisement messages can either be static of dynamic. In order to make the advertisements dynamic some kind of internet connection must be available for the user. Utilizing the Internet, advertisement messages can be sent from a centrally located advertising server to any software application wanting to advertise as means for income.

In some cases the advertiser and producer might be one and the same; the advertisement messages displayed might include some other service or product offered by the software producer.

3.2.2 Value-chain

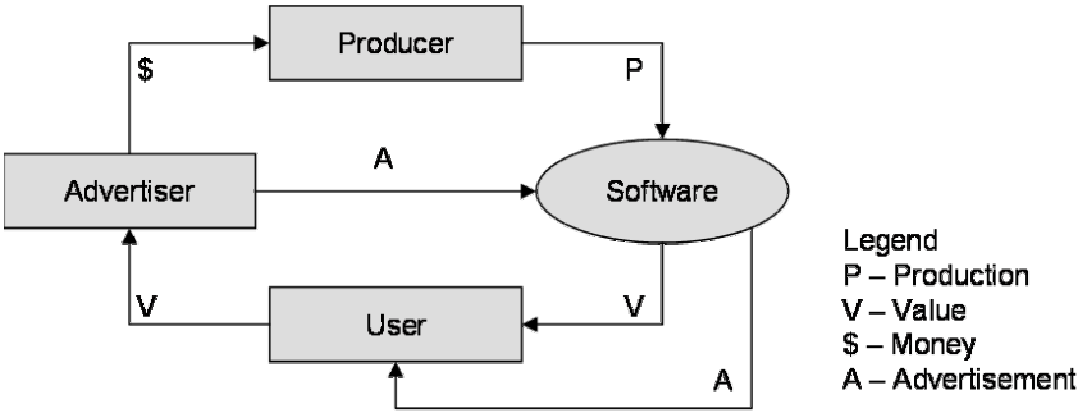

Figure 3.2.2

In the value-chain of advertising, the advertiser introduces advertisement messages into the software application. When the software application is run, the advertisement messages are consequently introduced to the user. The introduction of advertiser's advertisement messages into the producer's software application would then give a possible transfer of value from the user back to the advertiser. The advertiser then pays the producer in exchange for letting him introduce advertisement messages to the software's users.

3.2.3 Practical implementation

The advertisement agency normally functions as an intermediary, using its connections with the business environment to match companies wanting to advertise and software producers wanting to place ads in their software. The advertising agency normally supplies the software producers with software modules that easily can be inserted into their applications. This software module handles the display of ads, measures how many times the ads have been shown, and getting new and updated ads etc. The software producer then leaves out an open spot where the ads can be displayed and inserts the advertisement agency's software module into its own source code. The software module downloads ads from the advertising agency and they are displayed for the user according to specified criteria. The advertiser is then charged according to how many ads that have been displayed. Another commonly used way to charge for advertisement messages shown is by click-through, e.g. how many times the specific ad has been activated by the applications users'. This common solution of advertisement financing requires that the user is connected and uses the Internet.

3.2.4 Examples

3.2.4.1 Qualcomm Eudora mail

One of the large software producers that have adopted the advertisement financing method is Qualcomm. Qualcomm used to offer its customers two versions of its email client: a stripped down version that could be downloaded and used free of charge; a full commercial version without limitations that only could be purchased using traditional licensing. Qualcomm made headlines when it started to offer a third option: the full version almost without limitations for free, but with advertisements. This was called *sponsored mode* and is still a chooseable option. The sponsored mode includes all features of the paid mode excluding their *spamwatch* feature, but has a special ad window, up to three sponsored toolbar links and no person-to-person technical support [Eudora.com, 2003]. The ad-sponsored version was, according to official statements, a big success with over two million downloads [Tan, 2001].

3.2.4.2 ICQ

ICQ is one of the most widely used and downloaded software products, ICQ Pro 2003a has generated 236.902.780 downloads so far at shareware.com [shareware.com, 2003]. What many downloaders fail to comprehend is that ICQ is not freeware or shareware, it is software financed by advertising. With every message you send, an advertisement message is shown.

3.2.5 Discussion

In order for the advertisement business model to work it is not enough that the advertisement messages are shown to the application's users, the messages must also be given attention either implicitly or explicitly. Measuring how many of the advertisement messages that are given attention is a problem for most advertisers. The problem is usually solved by measuring how many ads that are actually somehow activated during user's execution of the software application. According to this, the best suited software applications for the advertisement business model would be software applications that require and have a large portion of their users' attention. In order for advertisers to measure how the advertisement messages are perceived by the users, users should be given the alternative of follow up on the ads. A connection to the Internet is therefore needed for the end-user. This means, when combining these two prerequisites the best suited applications for the advertisement business model are software applications that require much user attention, user input, and that preferably are depending on an internet connection.

Computer games are another type of software product that easily can be financed using advertisements. The computer games having additional financing by advertising are usually sport games filled with static banners, e.g. football and car racing games with banners on hoardings [Economist, 1999]. There is no need for advertisement messages to explicitly be displayed as banners. Advertisement messages can be woven in to the core concept of a software product, e.g. discrete messages can be put in computer games or an application used to find goods can recommend certain sponsored products.

Using this model, instead of valuating users possible gained value of your product and pricing it accordingly, the users themselves must be valued. This means, you must value how much the possible attention of your users is worth to an external advertiser. Valuing and calculating the worth and value of your users attention might be just as hard as valuing your product, maybe even harder. Additionally, all users might not be worth as much, e.g. a user located in the western part of the world might be of much higher value as he possess a larger amount of purchasing power than another user located in a poor region.

The primary source of revenues is not the end-user in the advertisement business model. Instead, most revenues are collected from external third parties such as advertisers. By having the bulk of your revenues collected from external third parties, a very tight relationship with these advertisers must be built apart from the relationship with the users. This means, by using the advertisement model, both end-users and advertisers must be pleased in order for revenues to be received. If the end-users are not pleased they won't use the software application and if no one uses the software application no revenues will be gained from the advertisers. The type of advertisement messages that are introduced is a very important issue for the producer regarding this two way relationship. As the advertiser's message is incorporated into the producers product, all advertisement messages should have an effect on the users perception of the product, e.g. if the software application displays ads for toys that is aimed towards kids the software application can become categorized as childish. The overall quantity of advertisement messages included in the software application is another important aspect. Incorporating too many ads usually leads to lessened perceived customer value whereas having too few can lead to lost revenues. The amount of advertisement messages incorporated into the product can therefore be considered as a competitive aspect. For example producer A delivers approximately the same product as producer B but by having less advertisement messages shown it can be perceived as the better alternative.

Lack of stable revenues can be another problem for the software producers utilizing the advertisement business model. Even if the new-user inflow is steady, this does not constitute for means of stable income. As income and revenues are dependent on the advertising market, it also becomes dependent on the seasonal patterns of the advertisement market. These two widely dispersed businesses also have different trade conditions, e.g. when there is a recession in the advertising market there might exist a time of prosperity regarding the software market. During a recession in the advertising market prices drop for advertising spots and hereby revenues are lessened.

The advertisement financing method should, according to us, be best suited for consumer users. As business users tend to seek professional software applications with which their professional tasks can be made more efficient, software financed by advertisement should not suit them. One of the major points and tasks for advertisement messages is to get the observer's attention. Attention is however not something professional users, or at least not their employers, want to give up. Qualcomm's advertisement-supported version of Eudora mail receives the majority of its clientele from non-business users. Software applications financed by advertisements usually incorporate ways of delivering targeted or customized advertisement messages to its users. In order to deliver a targeted or customized ad, much information about the user must be collected. Companies rarely want this information to be collected and leave the premises of the workplace, why the advertisement financing method is not suited for business-users.

The worst enemies for the advertisement business model are crackers, firewalls and anti-virus applications. Depending on the factor of annoyance regarding the advertisement messages and the application's number of users, hackers can be more or less interested in producing a "crack" which task is to remove the advertisements, thus removing the annoyance. These cracks are regularly distributed on the Internet and by using it its users are relieved of advertisement messages in the software application they have chosen to crack. Releasing the software product with the advertising business model under an open source license is not recommended. By opening up access to the source code you actually ease the task of removing the advertising from your software product. As the majority of all software applications financed by advertisements are Internet related, these advertisement supported applications use the Internet in order to obtain new or updated advertisement messages and the connection to the Internet is vital for the value-chain. Firewalls can however hinder the transfer of information between the advertiser and the end-user. The firewalls can sense that your advertisement supported software application is sending information to the Internet and that uncategorized information is returned. The activity of sending and retrieving information hereby stands the risk of being categorized as malicious by the end-users firewall and accordingly being hindered. Software applications sometimes collect user information in order to better perform target marketing. This fact puts them in risk of becoming categorized as intrusive by the anti-virus companies, hence facing the risk of being removed by end-user's anti-virus software.

Every customer gets affected by advertising in one way or another, but not every customer acts on it. This fact is well known among all advertisers and means that in order to advertise to the customers that actually act on advertisement messages you must be willing to advertise towards the customers that don't. Even if the advertisement messages are targeted towards a specially selected group of people, the fact that only a few observers of the ad act on it remains. Consequently, having a large user base would result in a larger amount of users that act on ads, hence leading to larger possible revenues for the producer. Software products that already have been introduced on the market and already has built a steady and large user-base is therefore preferably.

3.3 Selling computer resources

3.3.1 Description

The core task of the Internet is to connect the computers on which it is built upon with each other. All computers connected to the Internet are owned by different parties, some are owned by companies, some by large institutions while others are privately own. All computers, whether they are company or privately own, contain a certain amount of hardware resources. These interconnected resources are seldom utilized to the max, wherefore a large level of unused hardware resources can be spared without any noticeable drops in performance. The fact that hardware resources are quantifiable and measurable makes it a viable source for payment. This means that instead of paying for a software application using monetary means your computer hardware is used.

As the world has become more and more computerized, more digitalized data has been created and is being created than never before. In order to properly understand this data it must be processed into information. Processing data into information is usually a very time- and resource consuming activity. So, as the amount of data increases so does the need for computer resources used to convert data into usable information. One way for companies to obtain this amount of computing power is to purchase an expensive supercomputer, another way is to use smaller ones built into one. The task of combining several smaller computers into one using a network is called clustering or distributed computing. The obvious benefit from using a clustered system over a supercomputer is cost savings. Using a clustered system, supercomputer power can be achieved at a tenth of a supercomputers price [McGrath, 1998]. The most common usage of distributed networks is to use the combined processor power to perform large calculative jobs. Distributed networks can however be utilized in other ways: combining storage space of several small systems will result in one large and massive hard drive; combining the collective shared bandwidth of numerous small systems can lead to a large amount of shared bandwidth and faster file transfers.

3.3.2 Value-chain

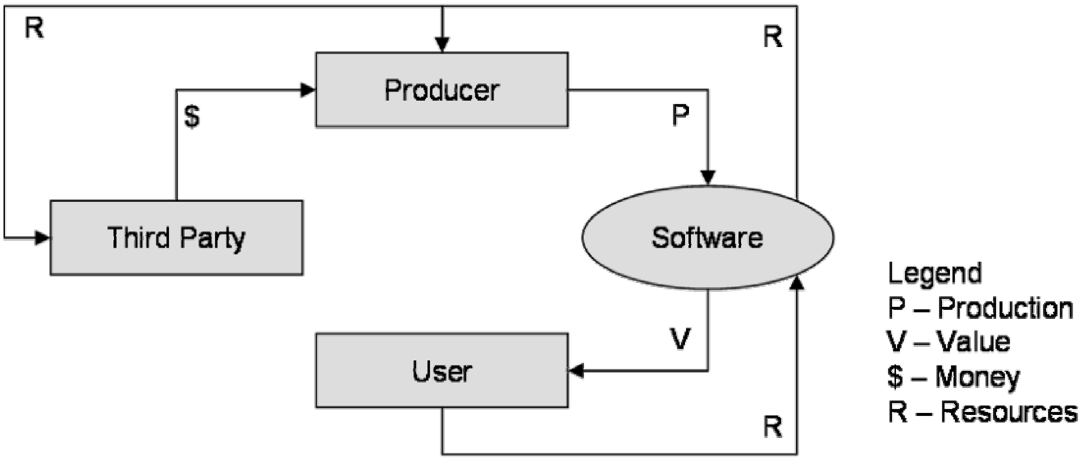

Figure 3.3.2

In the value-chain of selling computer resources, the user pays for the legal use of the software by allowing the use of his computer hardware resources. The users' hardware resources are collected and redistributed back to the producer of the software. The software thereby grants the producer limited access to the user's computer hardware. The access to the users' hardware resources are collectively managed and later sold to third parties in need of large quantities of distributed computer hardware resources.

3.3.3 Practical implementation

Depending on what type of hardware resource that is to be shared and used as means for payment, different levels of internet connections are needed. Computer calculations can be fully distributed why a persistent connection to the Internet is not required at the user's side. On the other hand, if bandwidth or disc space is to be used a fast and persistent connection to the Internet is required.

Included in the software must be some kind of communication software. Using this communication software, data and information or resources is being shipped back and forth between the user and the third-party or producer. At the receiving side must be some form of verification software. The verification software must verify that the information or resource that is being received is valid and not manipulated in any way. If the distributed system requests a user's processing CPU power in order to be used in a distributed supercomputer, the software which enables the utilization of the users' computer is normally placed within a software applications that is run when the computer is not used, e.g. in a screensaver [Goldsborough, 2001].

3.3.4 Examples

3.3.4.1 SETI@Home

The fastest supercomputer in the world is not placed at Berkeley University, it is only operated from there [Kher, 2000]. The computational power of the SETI@Home project can be described as: "The most powerful computer, IBM's ASCI White, is rated at 12 TeraFLOPS and costs $110 million. SETI@home currently gets about 15 TeraFLOPs and has cost $500K so far. " [SETI@Home, 2003]. At Berkley University, analyses of radio signals is done in search of alien activity, Search for Extraterrestrial Intelligence (SETI). Analyzing radio signals in search for anomalities is a very resource consuming activity and up until the deployment of SETI@Home this activity was impossible in any larger scale. To aid this search, the world's largest distributed supercomputer was built in 1999. Even if Berkley University, does not sell their computer power to external actors it clearly displays the possibilities of doing so.

3.3.4.2 Popular Power

Popular power was one of the first pioneers using the distributed computing model to gain revenues. Popular power had a number of Fortune 500 companies as customers for their software [Koman, 2001]. What Popular power hoped for was to become an intermediary between pharmaceutical companies, computer graphics studios, etc. and large companies not fully utilizing their full computer power. Popular power's calculation was that most companies would be interested in renting computer power instead of buying large supercomputers of their own. It also included that most companies would be willing to sell the use of their computers during night hours when no actual computer power is needed. Popular power would charge a fee for making the unused computer power available for the scientific companies etc [Shankland, 2001]. Popular power went out of business March 2001.

3.3.4.3 Gateway

Gateway is one of the largest computer hardware vendors. Gateway sells its hardware mainly through their computer hardware stores. In their computer hardware stores several computers are normally set up for display in order for potential customers to come and try them. Computers that are up for display are seldom used and their resources are only used sporadically as the software applications that are used in display don't require the full amount of resources at hand. Instead, in order to utilize the full potential of the computers' hardware an additional software application is run in the background on these computers. This small application is installed in order to harvest and distribute the remaining potential of the hardware at hand [Musgrove, 2002]. The computer resources are then sold to third parties in need of computer resources, mainly processing power.

3.3.5 Discussion

By using this method, users' already purchased hardware can be utilized more efficiently, not only by him but by others as well. For organizations, this method can reduce vast costs as a company's fixed assets can be taken advantage of in additional ways. Huge costs are introduced for large businesses along with the purchase of computer hardware. By using this model for payment, these costs can be utilized in additional ways as the computer hardware can be used not only for using the software applications but for the payment for these as well. Large organizations seldom utilize the full potential of their computer resources. Most computer hardware resources have a long lifespan. This lifespan is generally only moderately lessened when using the resource at full potential, e.g. a central processing unit (CPU) does not wear down faster when used at full capacity compared to running at a lower. The only real additional cost associated with the running at full capacity compared to running at a resting capacity is power consumption. A computer running at full capacity generally consumes more power then a computer idling. What can bother organizations regarding this method of payment are security concerns. When distributing your computer hardware resources, access to some portion of your systems must be handed out. Businesses strict security polices would generally hinder this kind of distribution, hence making the business model impossible.

The software applications suitable for this business model should not be hungry for computer hardware resources. A computer game in need of massive computer hardware resources usually can’t allocate sufficient computer resources to distribute why it shouldn’t be used. There is no need for user involvement regarding these software applications as chances are high that computer hardware resources are only moderately used when a computer is unattended. Consequently, software applications most suitable for the selling computer hardware business model would be software applications only performing few calculations. Software applications utilizing the selling computer hardware business model is not suitable for making open source as these features could be removed or the resources directed elsewhere.

Deciding the quantity of computer hardware resources to charge for your product however might be just as hard as pricing your product using money. What can be done is to measure how much money you would charge for your product if you weren’t utilizing the selling computer hardware resources business model. This amount is then translated into the corresponding market value of computer hardware resources and charged. End-users or companies might not want to distribute the full potential of their computer hardware resources. They usually want to keep some in spare for unplanned computer activities or because of power consumption, e.g. a company in possession of a large supercomputer might not want to grant full access to its supercomputer but only a portion of it. If the software producer’s customers would have been willing to spare the full potential of their hardware, pricing would not be a problem. But as consumers seldom want to give up the full potential of their computer hardware resources, the software producer must decide how much to demand from its customers. Therefore, the amount of computer hardware resources different competitors’ demand for their product becomes a main competitive aspect regarding this model, for example producer A delivers approximately the same product as producer B but by demanding less computer resources it should be perceived as the better alternative.

Another problem related to the pricing problem is storage of hardware resources used for payment. Computer hardware resources can’t be stored and therefore must be considered perishable. This means that sometimes the producer has access to a large amount of computer hardware resources but another time he might only be able to access a small amount. Nor can the demand for computer hardware resources be seen as stable. One day there might be a large demand while the demand can be rather low on other days. On days when the demand is high the software producer might not be able to obtain satisfactory amounts of computer hardware resources from its users and therefore the revenues may not be entirely stable unless the supply and demand are managed successfully.

The most serious threats to the selling computer resources business model are technical threats. The model is dependent upon a connection to the Internet. If the connection between the producer and the software application’s users were to be permanently hindered the model is rendered unusable. This means, the most noticeable threat to the model is firewall applications installed on users computers. A common implementation of the model consists of the users’ computer hardware which is combined into a large distributed supercomputer. When a large calculative job is to be performed using a distributed supercomputer, the calculative job is divided into smaller sub parts and distributed among the peers. When the calculative job is distributed, raw data is sent to peers. If this data is intercepted or hindered by a permanent firewall the calculative job can’t be performed by the peer and thereby making the user’s computer hardware resources unusable. If the application that grants the producer access to the user’s computer hardware is delivered in an additional software application, e.g. a screensaver, this external application can also become considered as malicious by user’s anti-virus software. A software application that becomes considered as malicious stands the risk of being disabled or removed automatically by user’s anti-virus software. As the utilization of users’ unused computer hardware resources seldom affect the user it does not pose as a factor of annoyance for the end-user. “Cracks” might remove the part of the software application that enables the use of user’s computer hardware resources. But, if this business model is implemented correctly and not bothers users, few will get annoyed and only a few will feel the need to “crack” the software application.

A large installed user-base does not constitute as having access to a large amount of computer hardware resources regarding this business model. A set of few stable users possessing a large amount of computer hardware at your dispense is typically much better than a large amount of unstable and dispersed users. This means, the total amount of active users is irrelevant to this financing model, your users amount of computer hardware resources at your dispense is however. Actually, having too many peers can make distributed jobs difficult to manage.

3.4 Selling market information

3.4.1 Description

Information about the market has become valid merchandise in today’s market society. Market information is very hard to come by and usually requires thorough market research [Kotler, 2003]. Market information can be gathered in several ways: having interviews with the potential market, handing out questionnaires to the potential market or simply by watching the market. Watching the market as it performs certain activities is known as *ethnographical studies*. Ethnographic studies is the most newly adopted discipline of the market information gathering techniques but is very expensive as it is time consuming and difficult to manage [Masten and Plowman, 2003]. Ethnographic studies are however very truthful as no personal censorship or any tampering can be done to the results. The best results from ethnographic studies can be found when the observed party is unaware of it is being observed [Horovitz, 2002].

As market information is very hard to obtain, the value of it is high. Competition between companies should, in theory, have gotten tougher during the last previous decades. During this time, trade between countries has become easier, e.g. the European Union and its free trade regulations. These drastic changes have made every market actor a part of a larger market. Generally, if you want to survive in a tougher market with more competitors you must be able to deliver goods suiting consumer needs much better [Kotler, 2003]. Because of this change in the market, companies have started to value market information and a larger demand for market information has been created. Some actors have noticed this change in demand for market information and now sell this kind of information profitably [Martin, 2000].

With the advent of fast information systems, businesses have been given the capability of storing large amounts of market information in their databases. Using the Internet, huge amounts of consumer information can easily be collected and recorded. This information might include customer data such as name, age, and location. In some cases the information also includes users’ interests, hence being very attractive for marketers. In other cases the market information only includes data about certain patterns of behavior, e.g. patterns found about consumers purchase behavior. On the Internet, a possible customer’s every move can be recorded and analyzed. Recording and storing this kind of market information for future sales is therefore an option for the software developers wanting to receive revenue from alternative sources.

3.4.2 Value-chain

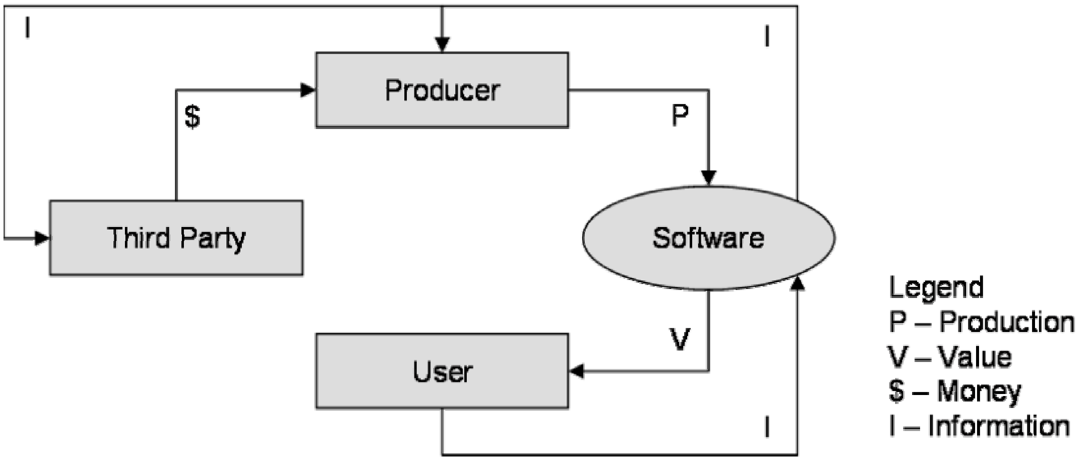

Figure 3.4.2

In the value-chain of selling market information, the software collects information about and from its users. This information is sent from the software back to the producerwho collects and manages it. Once collected and organized the producer can sell and redistribute the collected information to third parties wanting to further expand their knowledge of the market. In exchange for this market information money is transferred from the third party to the producer.

3.4.3 Practical implementation

The software product that is to be financed using the selling market information business model is equipped with a data retrieval software module. This software module gathers data according to what it is programmed to collect and sends it back to a designated location. The retrieved data must be managed and processed in order for it to be transformed into usable information. Once managed and processed information can be mined and sold to interested parties.

3.4.4 Examples

3.4.4.1 Claria

Claria is the new name for the previously widely discussed Gator software. Claria's own vision is to become the world's leader in what they call "Online Behavioral Marketing". The official statement regarding the purpose of Claria's software is to gather information about its users in order to display advertisement messages based on the user's online behavior [claria.com, 2003]. The core of their business idea consists of the assumption that click-through-rates will be much higher if an advertisement message that should be suited for the user is displayed. What Claria seeks to accomplish is to manage what marketers might call one-to-one marketing [Kotler, 2003]. Claria's official statements include no information whether they sell recorded market information to third parties or not.

3.4.4.2 Amazon.com

Amazon.com is an e-commerce site that primarily sells books. Amazon.com has an enormous customer database with more than 24 million customers stored in 2001 [Krishnamurthy, 2002]. This database includes customer information such as name, birth-date, social security number, address etc. Amazon's database also includes information regarding what purchases their customers have made. The primary use for this additional information is according to official statements for book recommendations [Linden et al., 2003]. However, by using Amazon's customer database, finding a certain user's special interest can be made easier, e.g. chances are high that a consumer buying books about sport fishing is interested in fishing, hence making this information valuable for direct marketing efforts from sport utility manufactures. Official statements from Amazon state that customer information will not be sold to third parties without users consent [Amazon, 2003]. Amazon's privacy policy does however state that this information can be transferred to partners in co-branded business or joint-ventures [Network Security, 2001].

3.4.5 Discussion

Market information is a desirable good for many businesses and this makes the business model of selling market information interesting. The user-base generating the information is of great essence. In most cases it has to be relevant to the companies seeking information, and it has to be large enough to create a useful information base. The larger and more relevant number of users in the target segment, the more valuable information can be derived and charged for from the corporate customers. Since the revenues are based on the value of the user and market information, and to a large extent the number of users, software targeting wide customer areas is suitable for this business model. Targeting niche groups may still be very profitable as the value of a group increases if it is more specialized. While this user base is being built-up, the sales will likely be small and the true value of the users show itself when an adequate number of installations have been reached.

There are major drawbacks with this business model, mainly in privacy and consumer acceptance. Users tend to dislike that information about them gets passed around, especially when they cannot control who may receive the information. Selling such information without user knowledge can violate their right to privacy and consequently backlash on the company. Although it is unknown what Claria does with the information they collect (other than displaying intrusive advertisements) the Gator name had received such a bad reputation that they were effectively force to change their name.

Businesses will definitely not want applications monitoring employees within their corporate environment. Naturally, they do no want it because of fear of sensitive information leaking out that could harm the company. Especially not if the information gathered is sold to their competitors. Generally, selling market information has a negative effect on the software adoption because of the distaste users have for this type of information use. It may also reflect back on other products produced by the same company. Products that do not contain any information gathering parts can get tainted as privacy intruders just because one other product from the same company is using this model. Using this model can severely risk a company's good reputation if the model is not implemented in way that users understand and agree to what being done with the information gathered about them.

Technically there are threats to the revenues. Firewalls can be a problem as they can block information sent back to the producer. Hacking to remove or disable the parts of the software that gathers information is also a risk. Primarily the major technical drawback is that anti-virus software are starting to treat what is referred to as "spyware" as malicious and therefore removing such software from computers. All spyware may not actually sell market information but without user acceptance and communication about what the software does, anti-virus software may remove applications they see as offending. Open source would not be a good choice as it would allow for forks into new applications without the information gathering parts. It can also allow for competitors to create versions of the software that directs the information to their own data warehouse.

The network effect will not directly have an adverse impact because of the attitudes towards privacy harming software. Once initial fears have been over won, the effects can be the same as with credit card usage over the Internet. After initially being reluctant to use their cards online, people started to notice that others used their credit cards online and they figured that they might as well. Perhaps the same effect will happen to privacy concerns. Once people get used to it and feel that the information collected does not harm them, they won't mind sharing the information.

While this business model may be used in all parts of the software's life cycle, it is at its best in the latter parts when the user base is larger. Starting too early may limit the value of the software provided compared to what the user is giving up, making them choose a different software. Waiting for a larger installed base gives the opportunity to keep the users through the software's normal lock-in. Selling market information does not in itself change the lock-in effects.

The pricing problem is moved from the software to pricing the value of the information collected and the problem of trying to maximize the rate when selling it. This is still hard to do but since this business model has been used in the offline world for a long time, there are good benchmarks to use to help the pricing. Considering that much more information can be collected online, it may be harder to value vast amounts of aggregated information, and also verifying the accuracy of the information collected.

3.5 Paid bundling

3.5.1 Description

The bundling of several even unrelated information goods is both fully possible as well as effective. Financing by paid bundling would mean that a fee is charged or commission gathered for including other information goods into your product's package. The most common form of financing by paid bundling involves the including of either software specialized in displaying ads, software specialized in retrieving user information, or software specialized in using users' hardware resources as mentioned earlier

Paid bundling of information goods can be done in multiple ways according to us, the type of paid bundling can be: positive, neutral or negative. Positive paid bundling brings positive effects for the end-user, it adds to the user surplus by including a good which can be useful for the user. Positive paid bundling can e.g. bring additional functionality to the software application. Negative paid bundling reduces the user's value of the package, e.g. if invasive software applications, such as spyware, is included into the package the user's perceived value of the complete package might be lessened. Neutral paid bundling do not add direct user surplus nor adds any direct negative side effects as lost user surplus. What constitutes as neutral bundling differs much between users. While some users might find a bundled information good intrusive, and therefore being negative, others might not.

Paid bundling can either be done implicitly or explicitly. If done implicitly the users are not notified that additional products are being installed whereas if bundling is done explicitly the additional products are delivered as external parts and users are notified about their existence.

3.5.2 Value-chain

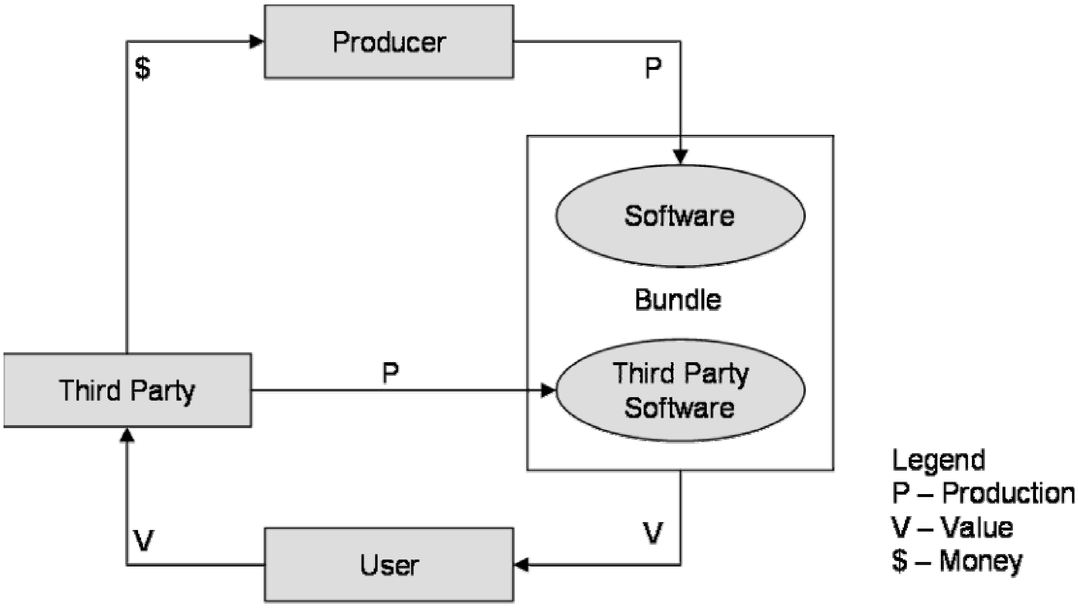

Figure 3.5.2

In the value-chain of paid bundling, the third party develops a software product. The producer delivers its software product to the users in a bundle including both its own product as well as the third-party's product. The third party's software product is consequently delivered to the producer's users and therefore value is transferred from the users to the third party. In exchange for bundling the third party's software money is exchanged

3.5.3 Examples

3.5.3.1 Microsoft Internet Explorer

When Microsoft first introduced their new web browser, Internet Explorer version 1.0, it attracted very few users. A majority of the web surfers preferred Netscape Development Corporation's browser to Microsoft's new product [Schnoll, 1998]. To aid this problem Microsoft used its flagship Windows operating system to help distribute its new product. By including Internet Explorer into the Windows 95 package this new product was automatically introduced to its target market and almost without costs of having to promote their new product. This type of paid bundle would be considered as positive according to our definitions as it adds to user's surplus of the product. This example does not include a transaction of money between the producer and a third-party but does clearly display possible benefits of paid bundling.

3.5.3.2 Kazaa

Kazaa bundles five additional software applications in its package: New.net, Webhancer, Cydoor, OnFlow and EZula [Cave, 2001]. These external software applications pay for the option of being bundled with Kazaa and therefore constitutes as the main source of revenue for Kazaa software. As these added software applications only bring negative effects for the end-user this type of paid bundle is considered as negative according to our definitions. During installation of Kazaa you can choose whether to install what they call plug-ins. Because of lack of information many users choose to install these bundled software applications as they are perceived to be needed.

3.5.4 Discussion

One of the more important reasons why bundling exists and is successfully pursued is because of the believed surplus it gives the customers. Many of the examples which can be found today regarding paid bundling do not bring the customer something that is wanted. Actually, most products bundled today using paid bundling are unwanted by the consumers, e.g. spyware and adware. KaZaa is a perfect example as its bundled software applications do not bring additional surplus for the users

Whether a product should be bundled implicitly or explicitly is dependent upon the nature of it. Negative paid bundling, which commonly consists of the including of malicious software products, should according to us be done implicitly. Negative paid bundling brings the user no additional surplus why he preferable should not be informed of the bundled products. Positive paid bundling brings direct surplus for most users. Therefore, if positive the bundling should be done explicitly. Positive paid bundling is however not common why most implementations of paid bundling should be done implicitly. Neutral paid bundling are however best implemented implicitly as the negative effects are prone to be noticed more easily.

Intangible assets are both important and valuable for most companies. The most common ingredient of a company's intangible assets is company goodwill. A software developer's goodwill consists of the perceived quality of the products delivered. Software publishers with high goodwill can therefore be known on the market for delivering high quality products. When bundling is done, other products are introduced to the users of the product that is bundling. The bundler's goodwill can therefore be severely damaged if the products that are bundled are of lower quality. When bundling implicitly, the quality of your product can directly be affected as the additional products are always installed on users system and normal users can't tell the difference between your product and the bundled products. When done explicitly, bundling the wrong products can lessen your goodwill assets value. For example, a software application directed towards kids must be very careful regarding the products it bundles.

When bundling other's products for payment, you are actually selling them access to your users. In order for a software developer to utilize the bundling business model he should be in possession of a software application with a large installed user-base. This means that if you want to finance your products through paid bundling you preferably should have products that have reached further than the introduction-phase of the life cycle.

The pricing problem is moved from pricing the value of your product to pricing your users. The value of access to your users might even be harder to price than pricing your product. The value of your product varies greatly between different users, hence making pricing very difficult. Your installed user-base can be of variable value for other producers as well. The value of your user-base is dependent upon the nature of the software application. If it only attracts a low number of users but these are very specialized, the value of the user-base is generally low. But, for another software publisher wanting to reach exactly these users, the value of the user-base is very high. This means, a user-base consisting of a very specialized and homogenous set of users can be of higher value than a large user-base consistent of less specialized and heterogeneous users.

Paid bundling can be used to strengthen the competitive level of a product. This is however less common as most products that are used for paid bundling don't bring additional user surplus. Most products that other developers pay you to bundle bring negative effects for you why the amount of bundled software can become a competitive aspect. E.g. product A and B have more or less the same functionality, but B bundles a more negative software application why A becomes the winner. Bundling a software application that is categorized as malicious can be very lucrative from a short-hand economic perspective. Software producers wanting to bundle their “bad” product along with yours are generally more willing to pay a larger sum than producers having a “good” product which they want to bundle.

Revenues can be collected according to different measurements: by number of downloads or by a lump sum. Either way, the revenues can be collected in an early phase of implementation of the business model and can be done relative to growth. Number of downloads can be measured in two ways. It can be measured by using the plain number of downloads or by measuring number of actual software installations of the bundle.

One of the more serious threats to the bundling business model is technical. Many of the bundled software applications used in this business model have technical threats and such threats carry on to this business model. Unless the bundled product has a viable business model, the future of the paid bundling is insecure. Technical obstacles can therefore exist because of the unwanted nature of the bundled products. Anti-virus software can sense that a malicious software application is about to be installed on the user’s machine and may therefore hinder or notify the user. Licensing is another threat to the bundling business model. All software applications are licensed in some form: traditional software licenses, GPL, etc. When bundling, a group of software applications that might utilize different kinds of licenses are brought together. Software bundling can however be implemented in different ways. Only one license agreement is usually displayed during installation of a software application. Problems can therefore arise when bundling is done implicitly. If bundling is done implicitly, using different licenses, the user will only view the license of the main product and hereby leaving the other unviewed.

All software products can be bundled. The application that is bundling other applications should however not be licensed under open source. If negative paid bundling is being done, licensing your product under open source is impossible. Both regarding a negative and neutral bundle, the bundled parts would be removed.

3.6 Content delivery

3.6.1 Description

Content delivery can be described using many names including content providing; information charging; free the software and sell the content [Raymond, 2001]. The basis for content delivery is to charge for the content used within the application – that is transferred to or from it – whether this is from a central server or peers. While the software may allow content creation, that content will not be useful to the user without other users to share it with over the Internet. The value therefore lies in the information distribution network that enables information access. We have divided the networks into two categories depending on who creates the information: *user-based networks* where the users create and use the information and *producer-based networks* where the users use information from the producer.

An analogy can be made to other mediums such as newspapers or television. The value is not in the paper sheets or the television set. Without information their usefulness and value falls drastically. Software using the content delivery model is information carriers and it is the information they provide that is valuable and makes the software valuable.

Time is an important factor in content delivery. The value of the information carried in these applications typically falls with time. This has two aspects to it. It can lessen illegal copying of the information (if it is not allowed) since it will probably propagate slower outside the normal network. Secondly, it can also open up for the opportunity to price discriminate and charge more for immediate updates than for delayed updates, just as watching a movie at the cinema is more expensive than renting and viewing it a month later at home. Content delivery also opens up different forms of payment terms, allowing for per-use charging, per month, per year etc. Having multiple versions of your information goods would actually make it easier to comply with the different demands of different user segments [Varian, 2001].

3.6.2 Value-chain

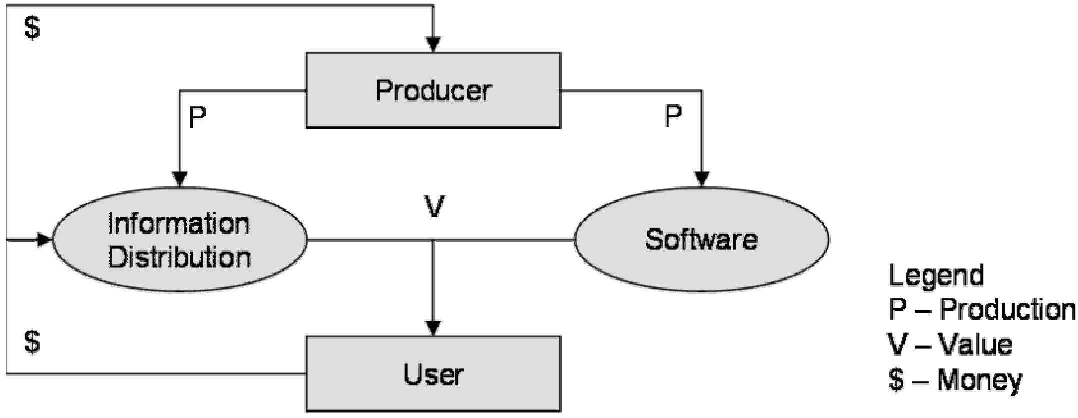

Figure 3.6.2

The value-chain for content delivery consists of the producer making both the software and the information network available to the user. The user derives value from the software and the information together. Payment is made by the user to the producer for access to the information (or access to the distribution network) but not for the software which enables access to it.

3.6.3 Practical implementation

The technical implementation will vary depending on what type of information the software is using and how it is distributed. Generally the software will communicate with a server run by the software producer (or a third party through an agreement) to send and receive information. This communication requires some form of user authentication which, connected to a billing system charges for the information delivered or sent. Software providing access to other information goods that need protection from copying (mainly goods that do not loose its value quickly) may need to implement some kind of Digital Rights Management system. On the server end, a part from billing, the service must monitor and stop dual use of the same subscription.

3.6.4 Examples

3.6.4.1 Anti-virus software

Anti-virus protection requires fast response to virus threats and the distribution to clients can often be time critical. Without an updated virus definition list, anti-virus software has limited use. The major anti-virus software producers have taken advantage of this and usually charge for the anti-virus application as well as for the virus definition updates. The initial application purchase often includes one year's worth of updates and after that the updates can be purchased in yearly increments [McAfee, 2003][Symantec, 2003][Kaspersky, 2003] [Panda Software, 2003]. Without charging for the definition updates old anti-virus software could potentially be used for a long time without creating revenues while still using the resources that make the software useful.

3.6.4.2 Equis MetaStock

Financial analysis applications like Equis' MetaStock (a stock market technical analysis program) require updated and accurate data to be useful. Equis supplies multiple versions of its software, targeting different user groups. The standard version is for end-of-day analysis and the professional version for real-time analysis. The subscription services for these applications also change – the real-time quotes required for the professional version costs more. While neither versions of MetaStock are free, they are a prime example of price discrimination for goods depending on content delivery. [Equis, 2003]

3.6.4.3 Instant messaging

While all the major instant messaging applications are free [ICQ, 2003][MSN, 2003][AOL 2003], they all have the potential for charging for the access to the network that delivers the messages as without it the applications are useless. The value of IM applications depend entirely on the usefulness (friends, family, co-workers etc using it) of the network for each user and are dependant on the network effect.

3.6.4.4 Apple iTunes

Apple's iTunes was the first online music store with wide consumer acceptance. Apple sold over one million songs through iTunes during its first week [Apple, 2003a]. It has since spawned a number of similar services including Napster.com (owned by Roxio and no longer a P2P file-sharing program) and Musicmatch.com. iTunes is an application that integrates a media player, an online store, syncing with the portable iPod player, limited file-sharing, and cd-burning. This software is given away for free – the users only pay for the music they purchase through the online store. The application uses a media format called AAC (Advanced Audio Coding) which allows for DRM. User's are however, to some degree, free to copy and burn the music [Apple, 2003b]. Apple's DRM technology in iTunes has, according to early reports already been circumvented by Jon Lech Johansen, the same person who also broke DVD's CSS protection system [The Register, 2003].

3.6.4.5 Red Hat

Already mentioned, Red Hat offers a software update service on a subscription basis. The difference between this application and the others mentioned, is that Red Hat Linux is fully useful even without the latest packages (an application or component), and that the packages are available to download for free from other sources. It is however a convenience to download them quickly and easily, not having to monitor for important updates.

3.6.5 Discussion

Content delivering is wide spanning business model and it can include just about any type of software, targeting any type of market. As long as the product needs external information in order to work, this is an interesting financing method. With its division in two groups, user-based (e.g. instant messaging) and producer-based networks (e. g. anti-virus software), the aspects affecting the viability of the model are quite different. The effect of lock-in can be quite substantial. User-based networks can achieve strong lock-in due to the amount of users it has and lack of interoperability between networks. Instant messaging applications, for example, do not pass messages between each other's networks. Users are effectively forced to use the same network as their co-workers, friends, and family if they want to use instant messaging at all. Producer-based networks can have the opposite effect. Without paying for the software, a competitor offering a similar product at a lower price may have it easier to attract customers as the user has not sunk any costs into the software. The loss for them is then the time it takes to learn the new tool.

Competition between networks also has different characteristics. User-based networks have to lure key users (e.g. influential users or early adopters) into start using the new network and to bring their contacts with them as the network otherwise has limited value. Producer-based networks can compete more traditionally with product, price, place, and promotion. Established markets based on producer networks would then be easier to enter for newcomers than user-based networks. This also has an effect on revenues from content delivery. User-based networks probably provide more stable revenues as long as they have a positive network effect due to the low likelihood of switching. Producer-based networks may have to "fight" harder against competing networks to maintain the sales as they cannot depend entirely on switching costs to retain their customers.

As the content delivery model is usually based on subscriptions, this also gives that subscription fees are paid in advance which has a positive effect on the liquidity. Content delivery does not necessarily have to use subscriptions however; it could also use per-use charging. The drawback with this is that users will likely think twice before incurring that cost which can limit the usage. This model is reliant on continuous use, especially for user-based networks so that each user continues to add value for the other users.

Many of the content delivery applications can use versioning and price discrimination of the software and the network to attract different customer groups. E. g. companies may want all their computers to have daily updates of their anti-virus definitions and could be charged more than home users who may only need it once every week or month. The same goes for many of the producer-based networks. User-based networks may not allow the same type of versioning mentioned (by delay or access limitations) but may still do it on network features, for example by opening up access to other networks or allowing video conferencing in instant messaging applications. Opening access to other networks would of course have negative implications on the lock-in effect as well.

The pricing problem is moved from the software to the value of the content delivered. While it can be priced from a cost perspective, value-based pricing is possible. Again, user-based networks that have a large user base have an advantage in that they have the opportunity to charge more for it. At the same time user-based networks probably have to wait until the network has grown large before it can begin charging for the network, otherwise it would grow too slowly. A fee introduced too early on can hamper the network effect which should be strong in user-based networks as the users often will try to make their friends join the network to add value to their own usage. Producer-based networks may not enjoy that benefit to the same extent but can on the other hand start charging for the network from the beginning.

End-users should be quite open and willing to pay for content delivery. Consumers have paid for subscriptions and services for a long time and the difference to paying for information in software is small. Many web sites today charge for access by the month or year. This business model does not have any specific privacy concerns tied to the model either. One issue might be that the future costs of the network are unknown. Getting locked-in to a network that was free from the start may very well cost a lot in the long run as the network gains momentum and fees are introduced. Not knowing what the cost of ownership can scare clients away.

Technically, there are few threats to this business model. Some sub-categories are however more vulnerable. Apple must for example protect its DRM technology in iTunes and the same goes for other content delivery applications that use information that do not loose its value quickly. Firewalls may also be in the way for content delivery. It might be tempting to use peer-to-peer technology together with the content delivery model. This can work fine but it is important to note that decentralized peer-to-peer networks have the potential to operate without a central authority charging for the access.

Open sourcing content delivery software is possible, but not always the right thing to do. Apple cannot open source iTunes as that could allow the removal of the DRM technologies. For instant messaging, anti-virus software, and financial applications it may not be a good move either. For these, open source would mean that competitors gains access to information that can be used to create interoperable or competing networks. The data used in Equis MetaStock is also available from other sources, anti-virus definition updates could potentially be delivered from different companies to the same software, and instant messaging application vendors defend their network interface to stop software clients that combine multiple networks into one application. Still though, it can be done. The reason for instant messaging applications to defend their network interfaces is that they now are based on advertisements and competing software would lessen their sales while still creating costs by using the network. By charging for the network access instead they would not have to bother with the client software as much and it could then be suitable for open source. Competing applications connecting to multiple networks may however lessen the lock-in then. By keeping the software closed source, user-based networks open up for the possibility to have, for example, one free version with advertisements and one version that charges for the access instead of containing ads.

3.7 Private provisioning

3.7.1 Description

Public goods have been around for a long time and there have been different suggestions how to "solve" the problem of providing them to the public [Varian 2001]. One way is by private provisioning. This is closely related to suggestions in the software community for the Street Performer Protocol [Kelsey and Schneier, 1998], the Rational Street Performer Protocol [Harrison, 2002], the Wall Street Performer Protocol, Software Completion Bonds [Rasch, 2001] and the Ransom protocol [Ransom, 2003]. While all differ in implementation, the basic principal of them is the same. They focus payment for the development (or performance) instead of the production [Neus, 2002]. Interested parties pay for the goods' development and it is then made freely available to all without any re-production costs. An example can illustrate how private provisioning is done: Assume that a company offers to develop a software program for $12. User A values it at $9 and User B at $8. Since their total valuation exceeds the development cost, it would be rational to produce the software. Should User A offer to pay $7 of the cost, the best response for User B would be to offer \$5 since that is the smallest amount that would provide the software. If instead User A offers to pay \$3, User B might offer \$0 as no matter what B offers, the good will not be provided. In this game, there are different outcomes and some do not provide the good, but this form of provisioning can be effective [Varian, 2001][Bagnoli and Lipman, 1992]. While it is in their collective interest to have the software provided, individually they want to pay as little as possible [Rasch, 2001].

While private provisioning is not entirely different from traditional software licensing in terms of payment, it is still interesting from an alternative viewpoint for two reasons. Although some users will pay for the software, many will not. The payment is made for the software's development, not for every copy. The other reason is from a pricing perspective. No one is willing to pay more for software than one's own valuation of it. By moving that decision to the buyer, the developer does not need to handle the pricing problem.

This form of private provisioning can be done repeatedly from the initial first release of the software and then continued with new payment rounds for versions adding new features. It is not limited to open source although it is often used for that.

3.7.2 Protocol implementation

First of all, a suitable price for the development is set. Contributions towards this sum are then collected (contributions can be of any size) and the public good is released when the target is met. With this simple system there would only be the producer and users involved. To make the system more reliable and trustworthy, both the software itself and the contributions made can be handed over to a third party escrow service that guarantees that the software is released when the target is met, or that the money is returned should that not happen.

3.7.3 Value-chain

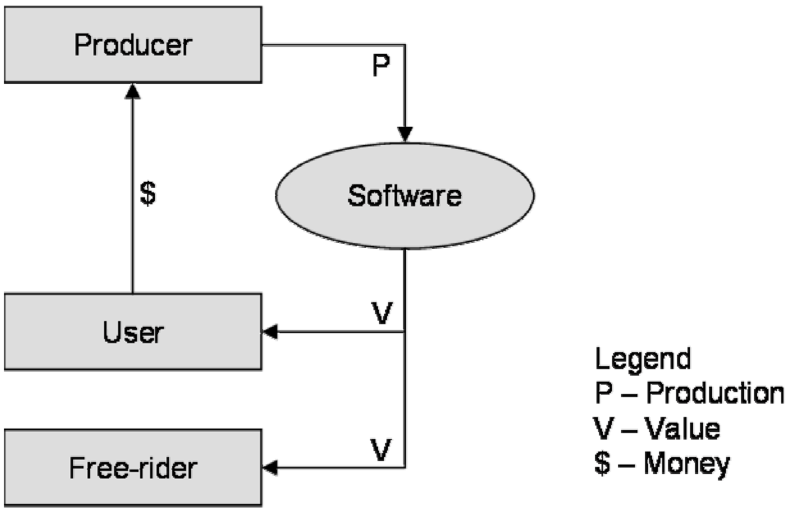

Figure 3.7.3

The private provisioning value-chain is similar to that of traditional software licensing in the sense that some users pay for the software, while others don't. In the private provisioning value-chain, the producer creates the software and the users have already paid for it through the protocol described earlier. When made available, the software is also available to non-paying users known as free-riders and delivers value to them both. The value-chain for traditional licensing may not necessarily contain the free riders (or pirates as they would be called) nonetheless they are still there in the real world. Here they are an accepted actor, although their numbers are preferably as low as possible.

3.7.4 Examples

3.7.4.1 Blender

Blender is a popular freeware 3D program with a relatively large user base. Blender's developer filed for bankruptcy in 2002 and a campaign was then started to release the software as open source. The copyright owner set the price tag of the source code to €100000. Using the Street Performer Protocol, it took seven weeks to raise the capital for Blender's open source release. The estimate for this was 3-6 months. One important conclusion that was noted from that project was that having pledges forced the contributors to make the same purchasing decision twice, while only the last one actually mattered [Neus, 2002].

3.7.4.2 MySQL

MySQL is a popular open source relational database server with approximately four million installations. MySQL powers popular website such as Yahoo and Google. MySQL AB was founded in 1995 and its main sources of revenue are support, subscription services, sales of commercial licenses (MySQL is dual licensed under the GPL as well), and franchising of the MySQL brand and products. MySQL does not contain every feature that its rival commercial databases have. That makes MySQL interesting from a private provisioning viewpoint. MySQL AB offers to implement missing features for a fee. After implementation the new functionality is placed in the normal distribution – made available as open source and to paying users alike. Although this type of pay-for-features model is not necessary paid by multiple parties, it has the potential to use a protocol like the Street Performer [MySQL, 2003].

3.7.5 Discussion

Considering the direct payments for the software in the private provisioning business model, it is closest to the traditional software business model among those described. At the same time it is also very different from it and the others. The pricing problem is solved by moving it to the customers as they have to put a price on the value the software adds to them. The pricing problem is however still present for the producer as the target sum for the software's release has to be set high enough to cover costs and to maximize profits, but it should also not be set too high compared to the number of consumers and their willingness to financially contribute to the release. From that perspective, the price for Blender's release may have been too low. But that price was set by the bankruptcy lawyers, and was not a business decision. Also notable is that the financial effects of free copying arise after the initial release, and therefore this business model cannot fully utilize them. E. g. a new ground breaking application may not be able to attract many contributions from the start, but can create a huge user base that would have been willing to pay much more than what was collected if they would have known about the software. That user base may still be useful when charging for future releases.

Private provisioning adds an interesting aspect to the network effect, not available to the other business models. As the paying users make a financial commitment to the software that is reliant on others doing the same, they have a stronger incentive to convince others to also purchase the software. Not all private provisioning protocols require pre-payment, but it is recommended from the conclusions of the Blender release.

A problem with this model is the uncertainty in how well this works for new products. Both MySQL and Blender were well established on the market when they started utilizing this financing method. Newcomers will probably need to have established trust with its users beforehand for this to work. An alternative would be a marketplace and escrow service that can help provide trusted connections between buyers and producers. For completely new products, private provisioning is unsuitable as it requires revelation of the idea and convincing people to buy it before it is proven to work. Revealing the idea can give competitors a head start to an idea while money still is being raised for the development.

Once a products' release target has been met, private provisioning has some benefits. For starters, the financing has already been secured. Unless the development costs more than expected, the financial outcome of the project will be successful. Payments made to an escrow service or directly to the producer may also be available early as portions of the software are done. The side effect of getting paid for the production instead of per-copy is that you cannot continue to thrive on what you have done in the past. Once the contributions meet the target sum and the software is released, the functionality charged for will not create any additional revenues through this business model, as it can do with traditional licensing. To maintain sales, new functionality must be added constantly. This also means that a larger installed base does not necessarily generate more income from this model than a smaller base would. There may be many free-riders, and only a few paying users. Having many users should however create more a stable income as you have a larger base to attract to pay for some new feature.

Private provisioning is fairly new for software and its long-term feasibility is not yet known. Technically there are no business model related threats to the income. Releasing the application under an open source license is possible, but not a requirement. Even if the application is released as open source, it is possible to charge for new features with a private provisioning protocol. Mainly, that would be appropriate for complex software such as MySQL where domain experience is required from the developer. Not releasing the software as open source can have negative effect on user perception of the long-run sustainability of the investment. Without access to the source code, users have to depend on the producer to set a reasonable price tag for new functionality, and on a reasonably sized user base to share the costs with. These risks are present in traditional licensing as well, but more obvious and risky when there is a game to be played in paying as little as possible while still trying to get the software produced. Open source lessens this risk as it opens up for other producers to compete by implementing features at a lower cost.

This business model can be used on all types of products and both the consumer and business markets. We do however expect that it will be hard to finance unproven software this way. From a software life cycle perspective private provisioning can have an interesting use as an end-of-life solution. It can be used to get the last cash out of a product that is to be discontinued. It will please important customers who wish to continue using the software for a while longer. In that same way it can also be used to leave a market and hurt a competitor that is active in another market common to the producer. These are however special cases and the private provisioning model is not limit to these.

4 CLOSING DISCUSSION AND CONCLUSIONS

4.1 Common to all models

The business models we have described can be divided into two groups based on where the cash comes from for the software. The first group consists of business models where the user pays for the software, although indirectly or through something else. This group includes complimentary tangible goods, content delivery, and private provisioning. The other group consists of advertisements, selling computer resources, selling market information, and paid bundling. Although the user "pays" for the software in some way, the money comes from a third party. This difference between the groups can be seen as a form of degree of alternativeness, where the second group of models deriving incoming from a third party is more alternative than the first.

Similar for these business models except private provisioning and to some degree content delivery is that these models require dual sales. Primarily, the software producer must attract users to use the software in question after which he again must sell the business model specific resource to the party financing the development. This may very well be harder and more resource consuming than the traditional method of selling software through licenses.

4.2 Combining business models

Many of the business models we have described are possible to combine with each other. Perhaps the most obvious example is to combine complimentary tangible goods with any of the other business models as support is always needed. When combining the models, threats to the revenues must be identified and avoided in a way so that the business models do not interfere with each other. The main threats to the business models we have described are technical. Firewalls, anti-virus software, spyware removal tools etc are problems for a few of the models and combining a model with such problems can have widespread consequences if not analyzed properly beforehand. A potential problem is open source, but that can be controlled as it is optional. Complimentary tangible goods, content delivery, and private provisioning support the use of open source to at least some degree. These can be combined freely with each other, but are not suitable for open source if they are combined with any of the other business models. Doing so would be a direct and unnecessary threat to the revenues as described in each model's discussion. Once open sourced however, that decision cannot be reversed and may have a negative effect on future business options.

Paid bundling is in itself a form of business model combination as it is reliant on the bundled software to derive revenues from a different model. Selling market information and advertising are the most bundled software today. The difference between them is seen as small as few people actually know what the software does once it is installed. Such software is often called spyware or adware. We do however not use these terms as these applications are seen as negative and intrusive. These business models do have their uses in advertising and information gathering. It is more a matter of how they are used and how they are explained to the users that decides if they can become accepted or not. When combining a business model that is seen as negative by the user, careful consideration must be taken to the impact that will have on the revenues from the other business model as well on the overall attitude towards the software. It must not necessarily be negative however. E.g. a two-version system with one advertisement supported version and one content delivery version can be very effective as a form of differentiation between the users and can help attract even the users who are not willing to pay for the software access thus resulting in a form of price discrimination.

4.3 Conclusions

This thesis has shown that there exits many alternatives to the traditional way of financing software products by using licenses. Common to all of these alternatives is that software piracy does not constitute as a problem for the business models, but instead utilizes the benefits found in the characteristics of software as they are information goods.

No model can be considered universally best. The different models are suitable under different conditions and for different types of software products. Most business models require an Internet connection to some degree, but they all benefit from the free copying and easy distribution that the Internet enables.

The different alternative business models have different sets of attributes that affect them in positive and negative ways. This can be a problem and disallow certain business decisions when combining models but may also be used to differentiate between the software's users and provide an appropriate offer to each user – something that is more difficult in traditional licensing where the main versioning attributes are features and price. The alternative methods are of course also possible to combine with traditional licensing where companies find that path most appropriate.

Free copying opens up for cost saving as well as new business opportunities. Peer-to-peer technology can cut distribution costs. The business models that allow of open source do not only lower the development costs, but allowing others to add to the software's offering, may also speed up development and foster new ideas.

4.4 Future work

4.4.1 Detailed descriptions and analysises

Our work gives an overview of alternative business models. It is however possible to describe them more in depth and analyze them in more detail.

4.4.2 Case studies

Some companies have already implemented the different alternative business models and have done so successfully. We have also found implementers of the different business models who have failed. Future work including case studies of the successful implementers would be interesting from a learning perspective. These case studies could be used in order to find possible connections between their work practice and their success.

4.4.3 Interest from producers

Many software publishers are somewhat skeptical to the usage of these alternative business models. Future work could therefore include posting surveys to software publishers in order to find reasons why this skepticism exists.

4.4.4 Other business models

We have identified and described the models we found but there may be other business models available. New business models can also arise at any time. Describing these and analyzing their effects are of interest.

5 REFERENCES

Altinkemer, Kemal, Jaisingh, Jeevan. 2002. Pricing Bundled Information Goods. *IEEE Computer Society*. Pp. 88-96

Amazon. 2003.
http://www.amazon.com. (Nov 1)

Andris, Robert P. 2002. Do software manufacturers still have the license to sue? *Intellectual Property & Technology Law Journal*.
http://proquest.umi.com/pdf/cf15580937f4f3afcd653be589e3b767/1063838299/share2/pqimage/vault1/20030917180819223/18936736/out.pdf (Sep 18)

AOL. 2003.
http://www.aol.com/ (Nov 5)

Apple. 2003a. iTunes Music Store Sells Over One Million Songs in First Week
http://www.apple.com/pr/library/2003/may/05musicstore.html (Nov 5)

Apple. 2003b.
http://www.apple.com/itunes (Nov 5)

Asvanund, Atip, Clay, Karen, Krishnan, Ramayya, Smith, Michael. 2003. An Empirical Analysis of Network Externalities in Peer-To-Peer Music Sharing Networks. Draft.

Bagnoli, Mark, Lipman, L. Barton. Private provision of public goods can be efficient. *Public Choice* 74, July 1992 (59-78)

Bakos, Yannis, Brynjolfsson, Erik. 1998. Bundling Information Goods: Pricing, Profits and Efficiency.

Blight, Jim. 1997. Customer Privacy Versus Customer Service. *Information Security Technical Report* Vol. 2 No. 1 pp 43-46.

Business Software Alliance. 2003.
http://www.bsa.org/usa/ (Sep 24)

Cave, Damien. 2001. Salon.com.
http://archive.salon.com/tech/feature/2001/08/02/parasite_capital/print.html (Sep17)

Chung-I Chuang, John, Sirbu, Marvin A. 2000. Network Delivery of Information Goods: Optimal Pricing of Articles and Subscriptions.
http://www.ksg.harvard.edu/iip/econ/chuang.html (Oct 8)

Claria. Claria Corporation.
http://www.claria.com. (Nov 1)

Commander, Simon. 2003. What explains the growth of a Software Industry in some Emerging Markets. *London Business School and European Bank for Reconstruction and Development*. No. 22.

Collberg, Christian, Thomborson, Clark. 2003. Watermarking, Tamper-Proofing, and Obfuscation – Tools for Software Protection. *IEEE Transactions on Software Engineering* Vol 28, No 8, Aug (p 735-746). IEEE.

Desai , Mayur S. Richards, Thomas C. Desai , Kiran J. 2003. E-commerce policies and customer privacy.
http://masetto.emeraldinsight.com/vl=3489170/cl=41/nw=1/rpsv/cw/mcb/09685227/v11n1/s3/p19_s.pdf (Dec 16)

DiBona, Chris, Ockman, Sam, Stone, Mark (editors). Behlendorf, Brian, Bradner, Scott, Hamerly, Jim, McKusick, Kirk, O'Reilly, Tim, Paquin, Tom, Perens, Bruce, Raymond, Eric, Stallman, Richard, Tiemann, Michael, Torvalds, Linus, Vixie, Paul, Wall, Larry, Young, Bob. 1999. *Open Sources: Voices from the Open Source Revolution.* O'Reilly & Associates.

Download.com. 2003.
http://download.com.com/3101-2001-0-1.html?tag=dir (Oct 8)

DRM Guide. 2003.
http://www.goldenpi.no-ip.org/drm/ (Sep 24)

Economist. 1999. Plug and play.
http://elin.lub.lu.se/cgi-bin/ftxt/ebsco/00130613_1999_352_8129/2057971 (Nov 1)

Einhorn, Michael A.. 2003. Digital Rights Management, Licensing, and Privacy
http://ssrn.com/abstract=332720 (Oct 30)

Ernie. 1997.
http://groups.google.se/groups?selm=33256CC1.7EE0%40mitre.org&oe=UTF-8&output=gplain (Oct 9)

Eudora.com. 2003. QUALCOMM Incorporated.
http://www.eudora.com/ (Oct 25)

Equis. 2003.
http://www.equis.com/ (Nov 5)

Fink, Martin. 2003. *The Business and Economics of Linux and Open Sources.* Prentice Hall.

Free Software Foundation. 2003.
http://www.gnu.org/ (May 22)

Givon, Moshe, Mahajan, Vijaj, Muller, Eitan. 1995. Software piracy: Estimation of Lost Sales and the Impact on Software Diffusion. *Journal of Marketing* Vol. 59 (January 1995), 29-37.

Goldsborough, Reid. 2001. Supercomputing for everybody.

Haar, Steven Vonder. 2000. Soft sell. *Brandweek*. Pp. 54-56. Proquest.

Harbaugh, Rick, Khemka, Rauhl. 2001. Does copyright enforcement encourage piracy? Claremount Colleges.

Harrison, Paul. 2002. The Rational Street Performer Protocol.

http://yoyo.its.monash.edu.au/~pfh/rational.html (May 18)

Hecker, Frank. 2000. Setting Up Shop: The Business of Open-Source Software. http://www.hecker.org/writings/setting-up-shop.html (Jun 11)

Horovitz, Bruce. 2002. Shop, you're on candid camera. *USA TODAY*. http://www.usatoday.com/money/industries/retail/2002-11-05-spy-shop_x.htm (Nov 9)

Huang, Andrew. 2002. Keeping Secrets in Hardware: the Microsoft Xbox Case Study. AI Memo 2002-008. ftp://publications.ai.mit.edu/ai-publications/2002/AIM-2002-008.pdf (Sep 18)

HBS Working Knowledge. 2003. Selling your personal data.

Hsing K, Cheng, Sims, Ronald R., Tegen, Hildy. 1997. *Journal of Management Information Systems*, Spring 1997, Vol 13, No 4, pp 49-60.

Ibm.com. 2003. International Business Machines Corporation. http://www.ibm.com/ (Oct 30)

ICQ. 2003. http://www.icq.com/ (Nov 5)

Jannuska, Jennifer. 2001. Federal Government to Levy Blank CD-Rewritable Media. http://www.dww.com/articles/billc32article.htm (Nov 5)

Kaspersky. 2003. http://www.kaspersky.com/ (Nov 5)

Kelsey, John, Schneier, Bruce. 1998. Electronic Commerce and the Street Performer Protocol. *The Third USENIX Workshop on Electronic Commerce Proceedings*, USENIX Press, November 1998. http://www.counterpane.com/street_performer.html (May 7)

Kher, Unmesh. 2000. Science by Screensaver. http://elin.lub.lu.se/cgi-bin/ftxt/ebsco/08180628_2000__41/3659301 (Nov 10)

Krishnamurthy, Sandeep. 2002. CASE #1- AMAZON.COM- A BUSINESS HISTORY TO APPEAR IN-“E-COMMERCE MANAGEMENT: TEXT AND CASES”. http://www.swlearning.com/marketing/krishnamurthy/first_edition/case_updates/amazon_final.pdf (Oct 22)

Koman, Richard. 2001. Popular Power Turns Off the Lights. http://www.openp2p.com/pub/a/p2p/2001/03/19/poppower.html (Nov 12)

Kotler, Philip, Armstrong, Gary. 2003. *Marketing An Introduction* 6th ed. Prentice Hall

Lerner, Josh, Tirole, Jean. 2000. The Simple Economics of Open Source. http://www.hbs.edu/research/facpubs/workingpapers/papers9900.html (Oct 2)

Lerner, Josh, Tirole, Jean. 2002. The Scope of Open Source Licensing. http://opensource.mit.edu/online_papers.php (May 7)

Linden, G,a Smith, B, York, J. 2003. Amazon.com recommendations item-to-item collaborative filtering.
http://ieeexplore.ieee.org/iel5/4236/26323/01167344.pdf (Oct 21)

LOC. LIBRARY OF CONGRESS. FAIR USE. 1999.
http://www.loc.gov/copyright/fls/fl102.pdf (Sep 29)

McGrath, Steve. 1998. Supercomputing gets a new hero. *Communications News*. Vol. 35 Issue 8 pp 78-79. Proquest

MADAME. 2000. Methods for Access to Data and Metadata in Europe. University of Sheffield.
http://www.shef.ac.uk/~scgisa/MADAMENew/Deliverables/MADAMER4final.pdf (Sep 30)

Martin, Patricia. 2000. Lee & Associates to start selling market information. *LAS VEGAS BUSINESS PRESS*.

Mashon, Mike. Television Advertising: A Brief History.
http://memory.loc.gov/ammem/ccmphtml/tvhist.html (Nov 9)

Masten Davis, Plowman Tim. 2003. Digital ethnography: The next wave in understanding the consumer experience.
http://www.dmi.org/dmi/html/interests/research/03142MAS75.pdf (Oct 28)

McAfee. 2003.
http://www.mcafeestore.com/ (Nov 5)

Microsoft. 2003. The Next-Generation Secure Computing Base.
http://www.microsoft.com/resources/ngscb/ (Sep 24)

Morris, Lou. 2000. Monster in the Basement. *Pharmaceutical Executive.* Vol. 20. issue 6 pp 160-161. Advanstar Communications Inc.

MSN. 2003. http://www.msn.se/ (Nov 5)

Musgrove , Mike. 2002. Computers' Shelf Life Gets Livelier.
http://www.washingtonpost.com/ac2/wp-dyn?pagename=article&node=&contentId=A32701-2002Dec9&notFound=true (Nov 15)

MySQL. 2003.
http://www.mysql.com/ (Nov 26)

Naumovich, Gleb, Memon, Nasir. 2003. Preventing Piracy, Reverse Engineering, and Tampering. *IEEE Computer* Vol 36, Issue 7 (p 64-71). IEEE Computer Society.

Network Security. 2001. Amazon cleared of customer privacy violations. http://www.sciencedirect.com/science?_ob=MImg&_imagekey=B6VJG-43FWBMG-7-1&_cdi=6094&_orig=search&_coverDate=06%2F01%2F2001&_qd=1&_sk=979989993&view=c&wchp=dGLbVzb-zSkzS&_acct=C000041498&_version=1&_userid=745831&md5=479b3c92670e488f055018a9a818c36b&ie=f.pdf (Nov 5)

Neus, Andreas. 2002. Blender and the Street Performer Protocol.
http://www.blender.org/modules/bc2002/Neus-Blender-SPP.pdf (May 18)

Nilssen, Tore. 1992. Two Kinds of Consumer Switching Costs. *The Rand Journal of Economics*. Vol 23 Issue 4 pp 579-590.

Odlyzko , Andrew. 2003. Privacy, Economics, and Price Discrimination on the Internet. Digital Technology Center, University of Minnesota.

OECD. OECD. 2003.
www.oecd.org (Sep 1)

Open Source Initiative. 2003.
http://opensource.org/ (May 22)

Osorio, Carlos. 2002. A contribution to the Understanding of Illegal Copying of Software. Working paper.

O'Reilly. 2003.
http://www.oreilly.com/ (Oct 17)

Panda Software. 2003.
http://www.pandasoftware.com/ (Nov 5)

Poddar, Sougata. 2002. Economics of Software Piracy and It's Global Impact. Department of Economics National University of Singapore.

Ransom. 2003.
http://www.theoretic.com/Ransom/Home (May 18)

Rasch, Chris. 2001. The Wall Street Performer Protocol: Using Software Completion Bonds To Fund Open Source Software Development. *First Monday* Vol 6, Num 6.

Rasmussen, Henrik B. 2003. Explaining Software Piracy. University of Copenhagen.

Raymond, Eric. 2001. *Katedralen och basaren.* Nya Doxa.

Red Hat. 2003.
http://www.redhat.com/ (Oct 16)

Rosenberg, Donald. 2000. *Open Source: The Unauthorized White Papers.* M&T Books.

Rowley, Jennifer. 1998. INFORMATION PRICING POLICY: FACTORS AND CONTEXTS. *Information Services & Use*.

Sanders, Robert. The Pareto Principle Its Use And Abuse. 1988. *The Journal of Business & Industrial Marketing* Vol 3 Issue 2 pp 37-41.

Schnoll, Scott. The History of Internet Explorer.
http://www.nwnetworks.com/iehistory.htm (May 15)

Shapiro, Carl, Varian, Hal R. 1999. *Information Rules*. Harvard Business School Press

Shareware.com, CNET Networks, 2003,

http://www.shareware.com/ (Oct 31)

SETI@Home. 2003.
http://setiathome.ssl.berkeley.edu/ (Nov 12)

Shankland, Stephen. 2001. Distributed computing loses luster.
http://news.com.com/2100-1001_3-254314.html (Nov 12)

SIIA. SIIA's Report on Global Software Piracy 2000. 2000.
http://www.siia.net/piracy/pubs/piracy2000.pdf (Sep 16)

SIIA-KPMG. Doesn't Everybody Do It?. 2001. SIIA.
http://www.siia.net/divisions/content/pubs/kmpg.pdf (Sep 16)

Spears, Mit. The DoJ and the 'network effect'. 1998. *Upside*. Vol. 10 Issue 10.

Stedman, Craig. 2000. VALUE-BASED PRICING. 2000. *Computerworld*, March 13, 2000

Sujdak, E.J. Ethical issues with target marketing on the Internet. 2001. *Technology and Society*. IEEE Comput. Soc pp. 105-113.

Symantec. 2003.
http://www.symantecstore.com/ (Nov 5)

Tan, Hwee Hwee. 2001. Sponsored Software. *MC Technology Marketing Intelligence*. Proquest.

The Register. 2003. DVD Jon unlocks iTunes' locked music
http://www.theregister.co.uk/content/4/34141.html (Nov 23)

Trusted Computing Group. 2003. Backgrounder.
https://www.trustedcomputinggroup.org/downloads/TCG_Backgrounder.pdf (Sep 17)

Tse, Edison. 2002. Grabber–holder dynamics and network effects in technology innovation. *Journal of Economic Dynamics & Control* 26 (2002) 1721– 1738. North-Holland.

Van Wijk, Jeroen. 2002. Dealing With Piracy: Intellectual Asset Management in Music and Software. ERIM Report Series Reference No. ERS-2002-86-ORG.

Varian, Hal. 2001. E-mail to Bruce Schneier regarding software completion bonds.
http://www.livejournal.com/talkread.bml?journal=crasch&itemid=68315 (May 18)

Ver Ecke, Wilfried. 1999. Public Goods: An Ideal Concept. 25 *Journal of Socio-Economics*.

Yu, Peter. 2004. The Escalating Copyright Wars. *Hofstra Law Review* Vol 32.