Substitution in the software industry

Msc. Management of Technology
Master Thesis

Oscar Gustavo Alva Nieto (1330217)
September 2008

Graduation supervisors:        Dr. Roland Ort
                                  Dr. Harry Bouwman
                                  Dr. Haiko G. van der Voort
Table of Contents

Table of Figures................................................................................................................ IV
Table Index ....................................................................................................................... V
Summary of the research............................................................................................... VI
Acknowledgment .......................................................................................................... VIII

Chapter 1 Introduction ................................................................................................. 1
  1.1 Introduction ........................................................................................................... 1
  1.2 Problem to be investigated ................................................................................ 1
  1.3 Research Question .............................................................................................. 4
  1.4 Perspectives .......................................................................................................... 5
    1.4.1 Managerial relevance .................................................................................. 5
    1.4.2 Scientific relevance ..................................................................................... 6
  1.5 Methodology .......................................................................................................... 7
  1.6 References ............................................................................................................ 8

Chapter 2 Characteristics of the software industry ................................................... 11
  2.1 Introduction .......................................................................................................... 11
  2.2 Production and marginal cost ............................................................................. 11
  2.3 Types of users ...................................................................................................... 12
    2.3.1 Classification of users by usage ................................................................. 12
    2.3.2 Classification of users by knowledge ....................................................... 12
    2.3.3 Distribution of the users in both alternative classifications .................. 13
  2.4 Segments in the software market ....................................................................... 13
  2.5 The relation between types of users and segments in the software market ....... 14
  2.6 Organizational characteristics of the software markets ...................................... 16
  2.7 Software as a service ......................................................................................... 16
  2.8 Conclusions ......................................................................................................... 17
  2.9 References ............................................................................................................ 17

Chapter 3 Theoretical Background .............................................................................. 19
  3.1 Introduction .......................................................................................................... 19
  3.2 Review of the existing literature ........................................................................ 19
    3.2.1 General dominance of technologies ....................................................... 21
    3.2.1.1 Discussion on Suarez’s model ............................................................ 24
    3.2.2 Competition in the software industry ...................................................... 26
    3.2.3 Suarez’s model in the software industry ............................................... 31
  3.3 Description of the research ................................................................................ 34
  3.4 Research Methodology ....................................................................................... 35
  3.5 References ............................................................................................................ 35

Chapter 4 Software development process and differences between open and proprietary software ............................................................................................................. 39
  4.1 Introduction .......................................................................................................... 39
  4.2 The software development .................................................................................. 39
    4.2.1 Concepts of software building .................................................................. 44
    4.2.2 The (proprietary) software companies .................................................... 45
  4.3 Open software ...................................................................................................... 46
    4.3.1 The open software development ............................................................... 47
    4.3.2 The open software suppliers ................................................................. 49
  4.4 Open software vs. proprietary software ............................................................... 49
## Substitution of Software

4.5 The process of the software substitution .................................................. 50
4.6 Battles for dominance in the software industry ........................................ 52
4.7 References ................................................................................................. 53

### Chapter 5  Study of the product competition in the software market.... 55

5.1 Introduction ............................................................................................... 55
5.2 Methodology .............................................................................................. 55
5.3 Software competition ................................................................................ 56
  5.3.1 General application competition .......................................................... 56
     5.3.1.1 Proprietary software: Large company: Internet Explorer ............... 59
     5.3.1.2 Proprietary software: Small company: Opera ............................... 61
     5.3.1.3 Open software: Organized: Firefox .............................................. 63
     5.3.1.4 Open software: Bazaar style: Konqueror .................................. 64
     5.3.1.5 Days between releases compared .................................................. 66
     5.3.1.6 The dominance on the web browser market ................................. 66
5.4 Conclusions ............................................................................................... 68
5.5 References .................................................................................................. 69

### Chapter 6  Proposed strategies ................................................................ 71

6.1 Introduction ............................................................................................... 71
6.2 General Strategies
  6.2.1 Create a network effect ......................................................................... 71
  6.2.2 Keep prices low, but release often ......................................................... 72
  6.2.3 Never stop the development .................................................................. 72
  6.2.4 Listen your potential customers ............................................................. 72
  6.2.5 Facilitate the continual usage of your products ..................................... 73
6.3 Open software suppliers strategies ......................................................... 73
  6.3.1 Create exclusive products ...................................................................... 73
  6.3.2 Create exclusive features ....................................................................... 74
  6.3.3 Help public organizations to adopt open software ......................... 74
  6.3.4 Get vendors to sell OS installed ............................................................. 74
  6.3.5 Facilitate products’ migration ................................................................. 75
6.4 Proprietary software suppliers strategies ............................................... 75
  6.4.1 Keep relationships with schools ............................................................. 75
  6.4.2 Keep relationship with hardware suppliers ......................................... 75
6.5 References .................................................................................................. 75

### Chapter 7  Conclusion ............................................................................. 77

7.1 Introduction ............................................................................................... 77
7.2 Research questions and the problem ...................................................... 77
  7.2.1 Characteristics of the software industry .............................................. 77
     Production and marginal cost .................................................................... 77
  7.2.2 Market dominance in the software industry ........................................ 78
  7.2.3 Substitution of products in the software industry .............................. 78
  7.2.4 Current situation of the market software ........................................... 79
  7.2.5 Strategies for the software industry ...................................................... 80
7.3 Learning points .......................................................................................... 80
  7.3.1 Multi-source research .......................................................................... 80
  7.3.2 From theory to practice ....................................................................... 81
7.4 Further research ....................................................................................... 81
  7.4.1 The importance of network effect in the software industry ............... 81
  7.4.2 The importance of users’ knowledge in the software industry ......... 81
  7.4.3 Hybrid (open /proprietary) software ................................................... 81
Substitution of Software

7.4.4 The success in the bazaar style software ................................................................. 82
7.5 Discussion .................................................................................................................. 82
    7.5.1 Updating competition models ............................................................................. 82
    7.5.2 Comparing results ............................................................................................. 83
7.6 References ................................................................................................................ 84

Table of Figures

Figure 1.1 The market share in the operating systems market ....................................... 2
Figure 3.1 Porter’s 5 forces analysis ................................................................................. 19
Figure 3.2 Firm and environmental level factors influencing the outcome of technology battles ................................................................................................................................. 21
Figure 3.3 Suarez’s model modified by the author .......................................................... 25
Figure 3.4 Factors influencing the outcome of the technology battle in the software industry .............................................................................................................................................. 32
Figure 4.1 The operating system ...................................................................................... 40
Figure 4.2 The “development-version release” conceptualization .................................... 41
Figure 4.3 Software development process simplified ....................................................... 42
Figure 4.4 Software development process incorporating different product versions .......... 42
Figure 4.5 Spiral representation of the software development process ............................ 43
Figure 4.6 The compiling process .................................................................................... 44
Figure 4.7 The kernel ........................................................................................................ 47
Figure 4.8 Substitution of software ................................................................................ 51
Figure 4.9 Substitution of software in detail .................................................................... 51
Figure 4.10 Unsynchronized software releases ............................................................... 52
Figure 5.1 Market share of the web browsers ................................................................... 59
Figure 5.2 Internet Explorer logo .................................................................................... 59
Figure 5.3 Days between releases of Internet Explorer .................................................... 60
Figure 5.4 Opera logo ....................................................................................................... 61
Figure 5.5 Days between releases of Internet Explorer .................................................... 62
Figure 5.6 Firefox logo ..................................................................................................... 63
Figure 5.7 Days between releases of Firefox ..................................................................... 63
Figure 5.8 Days of support of Firefox ............................................................................... 64
Figure 5.9 Konqueror logo .............................................................................................. 64
Figure 5.10 Days between releases of Konqueror .............................................................. 65
Figure 5.11 Competitiveness in the web browser segment ............................................... 66
Figure 5.12 The competition between web browsers ....................................................... 69
Table Index

Table 1.1 The principles of the open innovation compared with the close innovation ............ 6
Table 1.2 The principles of the open innovation compared with the open software .......... 7
Table 2.1 Distribution of users in both users’ classifications ............................................. 13
Table 2.2 Relation between users classification by usage and market segments .............. 15
Table 2.3 Relation between users classification by knowledge and market segments .... 15
Table 3.1 Practical example of the factors of Suarez ......................................................... 24
Table 3.2 Scenarios of the competence between open software and proprietary software .... 28
Table 3.3 The factors of dominance in the software industry ............................................. 34
Table 4.1 Differences between open and proprietary software ....................................... 50
Table 4.2 Framework for battles for dominance in the software industry ...................... 53
Table 5.1 Market share of the web browser ........................................................................ 58
Table 5.2 Release dates of Internet Explorer ..................................................................... 60
Table 5.3 Release dates of Opera .................................................................................... 62
Table 5.4 Release and end of support dates of Firefox ...................................................... 63
Table 5.5 Release dates of Konqueror ............................................................................... 65
Table 5.6 The model of dominance applied on the web browser market ......................... 68
Summary of the research

In order to survive, the open software suppliers must deploy some tasks out of their initial capabilities. They had successfully achieved technical equality versus their proprietary software competitors. Despite this success, the adoption rate of their products is low. Due to their nature, most of the open software suppliers base their development on the enthusiasm of their developers. If their developer communities get discouraged because of the low adoption rate, they could leave the project. A significant reduction on the amount of developers will affect dramatically the project’s possibilities to stay alive.

Although their products are technically equivalent to their proprietary software competitors, their low adoption rate is a consequence of a deficient (or even inexistent) business strategy. The creation of strategies would help the open software suppliers to reach a dominant position in the market, ensuring their survival in the software industry.

The creation of these strategies involves a complex problem. Unlike most industries, the software industry provides intangible products. This products’ intangibility derivates into specific characteristics which make the software industry unique. These characteristics should be explored in order to understand the software industry on detail. The analysis of these characteristics can be found in Chapter 2.

After analyzing the specific characteristics of the software industry, it will be necessary to understand the competition in the software industry. Previous product competencies in the software industry have resulted into dominance of the winning product of the battle, and the extermination of the other products. Based on this, Chapter 3 analyzes how the battles for dominance are deployed within the software industry. Based on this analysis, a framework is created in order to reveal the factors that determine the dominance of a product in the industry.

In order to analyze the characteristics and the dominance within the industry, it is important to understand the production process. The software development process is analyzed in Chapter 4. The analysis of the software development process delivers insights about the organizational structure of the software suppliers. This organizational structure is included to analyze the battle of dominance in a specific software market, and it is described in Chapter 3.

As a complement for these three parts the current situation of the software industry must be analyzed. A battle for dominance between software products is selected as study case. The battle selected is the web-browser market and is described in Chapter 5. The analysis of the web-browser market provided interesting points that drive the competencies within the industry.

Based on the knowledge obtained in previous analysis, and the analysis of the current situation of the software industry, the proposed strategies to be used by software suppliers are described in Chapter 6. These strategies are provided in 3 parts. The first part refers to the strategies that any software supplier must use to compete. The second part refers to the
strategies that open software suppliers must use in their battles. The third and last refers to the strategies that proprietary software suppliers must use as consequence of the raising of open software products. The three parts together deliver the result of this project and represent the solution to the adoption rate problem for open software products.
Acknowledgment

I want to say thank you:

To my parents. It is difficult to let go the people that you love, but it is even harder to show them the way.

To my brother. Because he is also my unconditional friend.

To the Delft University of Technology. It is my second home and the place that made several of my dreams become true.

To the Faculty of Technology Policy and Management. Thank you for everything but especially for made possible my stance here.

To my thesis supervisors. For their advices and time. And especially because they gave me freedom to write what I think.

To the TU Delft lecturers, I expect to learn a lot from you, but the most important thing that I learn from you is how to think by myself.

To all my friends in Netherlands, for each moment of joy that I spent with you. Thank to accept to be my friend and thank you to show me a little piece of your culture.

To the Dutch people for allow to feel at home so far from my birth town.

And of course, thank you to the one that decide what, when, where and how the things will happen. Thank you for let me be here.

Gracias
Chapter 1 Introduction

1.1 Introduction

Originally the software was a complementary product for the hardware systems. A complementary product depends on the sales of a main product, and it enhances the position of the main product. In some cases the complementary products are necessary for the survival of the main product. The Information Technology (IT) companies used to sell hardware equipments that could improve some of the process of their customers. The software was included in the hardware, or it was created specifically for the equipment and/or task.

Later on, the IT companies decided to create standard hardware architectures for their equipments. Although those products were still different, they could share some parts. This standardization also made possible that similar equipments would share software. Some external companies started to produce generic software for similar equipments, selling their products at a low price distributing the development cost through a large base of clients. That was the origin of the software companies. Today, the software industry represents a market capital of 507.8 billion dollar just in the United States stock markets (Yahoo finance, 2007).

For years, the software companies created the software indoors, and sold software products for their customers’ usage. This development model is called proprietary software. It has produced profits to the software development companies for a long period of time. However, some years ago, a different software development’s model has appeared. Based in different characteristics of the industry like the high production cost of the first copy of a product, and the insignificant marginal cost of reproducing it, a large amount of ideas appeared. Software developers’ communities were formed, in which the developers share their knowledge and skills to create products for the benefit of the community.

The new model of software development was called open software. It represented a breakthrough in the industry. After a while, several open software products appeared in the market to challenge the dominance of the proprietary software.

1.2 Problem to be investigated

The open software suppliers confront a problem out of their initial capabilities. Today, they have a product technically equal to their proprietary competitors, but their market share is in some cases insignificant. Although the existence of the software developer communities is not profit oriented, the lack of growth of their market share can discourage their members. This may cause that some of their members would consequently abandon the project. A smaller community will be reduced in technical capabilities, losing the technical parity with the proprietary rivals. At the end, this will lead to the disappearance of the development projects, and eventually the failure of all the open software.
This research will analyze several aspects of the software industry. The research will generate strategies that can help open software suppliers to reach the critical mass and even to achieve dominance. As a collateral effect the research will generate strategies that will help the proprietary software suppliers to maintain their position in the market.

After the appearance of open software, some paradigms have changed. The open software started to offer similar solutions at a reduced price or even for free. The software was intended to be completely able to be customized with technical support offered at competitive prices. A large part of the developer community sympathized with the idea of collaboration and created a technically “beautiful” product without any profit oriented constraints.

Today, the open software technically competes with the proprietary solutions in almost any product. In some occasions it gets a bigger market share than its proprietary rivals. Some experts like IDC, strongly believe that the open source applications will be preferred by firms based in their better features (Computer Fraud & Security, 2007).

Although the utopian ideas behind open software, its figures do not show the open software products as a success. Several open software products failed to increase their adoption rate. This marked some proprietary products as the leaders of their market by far. The failure is apparently independent of the technological equality between the open and the proprietary products. It is not enough to create a superior product in order to be successful in the market.

![Figure 1.1 The market share in the operating systems market](based on data obtained from (Net applications, 2008))
the market. An example of this dominance is present in the operating systems market, which market distribution is illustrated in Figure 1.1.

If the difference between the products is not originated in their technology, then probably it has its origin on the product management. Nowadays the main problem of the open software is not to achieve the same technological level than the proprietary competitors. Today, the open software suppliers must focus their efforts on the market competition. Based in the fact that the proprietary software is dominant in the market, the open software needs to take over a larger part of the market share to become a real competitor in the market.

As a starting point, this research will analyze the competition in the software market. This analysis will be focused to understand how the dominance is achieved on this particular industry. The research will be focused on the dominance and not just on the competition and co-existence based on the historical battles of the industry. During the life of this industry, several products-battles have occurred: the first browser war, the competence between the word processors, the competence between OS/2 and MS-DOS, etc. These competitions have resulted in the complete market dominance of the winner, and leading most of the time to the total extermination of the loser.

Based on the analysis of the dominance in the software industry, a further step will be performed. Based on the fact that software evolves really fast, the author can conclude that in a relative short period of time, new products and new versions of old products are launched to substitute old products. During this substitution usually the proprietary software users remain with proprietary solutions, and the open software users remain with open source products. Eventually some changes happen. Some users change from the proprietary software to the open or vice versa. What are the reasons of these changes? Are these changes only technological based or there are influenced by the strategy of the organizations behind the products? A second phase of the research will be the study of the substitution of products in the software industry. Also it will be studied their relationship with the openness of the software. An analysis about the software substitution will be necessary as part of the research.

The software market is also an interesting part of this research. A study of the current situation of the industry will be performed. Based in the three building blocks described previously, a more complete understanding of the industry will be obtained. This knowledge will lead to the final outcome of the research.

The competition strategies that will help the open software products to achieve dominance will be proposed as the final outcome. However, these strategies can be also used in a different way. Some counter strategies that allow the proprietary software suppliers to stop the growth of the open software market share will also be proposed. The final result of this research will be the proposal of strategies to achieve and maintain the market dominance in the software industry.
1.3 Research Question

The author found important to address the particular characteristics that make the software industry different from other more traditional industries (i.e. automotive or chemical industry). From this approach the first research question formulated is as follows:

1. The software industry is different from other industries. What are the characteristics of the software industry that differentiate it from other industries?

The answer to the first research question will provide an interesting input to the project with important aspects from the software industry. This point will need to be pointed to perform this research.

In order to understand the industry itself, it is also important to understand the competition in the software industry. The second research question is formulated to understand the competition in the software industry:

2. Is the market dominance relevant in the software industry and what is the model to achieve dominance in the software industry?

Based on the answer of the second research question, the model explains the dominance in the software industry. This model is reinforced with the characteristics of the industry and it creates a better understanding of the software industry.

The software products are produced by the software development process. These products substitute versions from themselves or products created by other software development-process. The third research question is formulated to understand the substitution in the software industry:

3. In the software industry the substitution of products is performed regularly. How does this happen and what is the influence of the openness of the software on the substitution?

From the answer of the third research question a better understanding of the substitution of software can be performed. The answers from the first, second and third questions provide a better understanding of how the software industry behaves and their products.

Based on this understanding, an analysis of the current situation of the software industry is made. This analysis has the objective of understanding how the characteristics, the dominance model and the substitution of the software influence the current software market. The fourth research question formulates this analysis:
4. What is the current situation of the software market for the open products and their competition with the proprietary products?

The combined answers of the four research questions will reveal a clear panorama of the software competition. Based on this clear view, it is possible to come back to the original problem and propose a solution for it. The proposed solution will be a set of strategies that will help the open software suppliers to commercially achieve the impact on the industry. The fifth research question is formulated to provide these strategies:

5. Based on the findings of the first four questions, what strategies can be proposed to the open software suppliers to increment their market share against the proprietary software products?

As a parallel outcome of the answer to the fifth question, some strategies will be suggested to the proprietary software suppliers. This outcome will be formulated from the sixth research question:

6. As follow-up of the proposed strategies in the previous question, what strategies can the proprietary software suppliers apply to maintain their market share against the open software products?

1.4 Perspectives

1.4.1 Managerial relevance

The open software represents a different business model for the software industry. A business model is a plan that converts technology into economic value. It moves the source of the revenue of the software companies from the sale of the binaries to the service of technical support and other complementary services.

The analysis of several aspects of the new model will show their strengths and their opportunities to develop. An improvement on the business model of the open software will allow them to reach an equal commercial competitive level as it was reached on the technical part (Krishnamurthy, 2003).

During the development of this research some topics will be addressed to explain the new business model. Some of the topics to be addressed are: the need of training, the importance of the users’ skills, and the network effects on the decision between open and proprietary software (Lin, 2007), the “elegance” of the software (or quality of the solution) (Raymond, 2000), the distributed control of the projects and their legal protection (Van Wendel de Jode, de Bruijn & van Eeten, 2002). These concepts are believed to be the main issues that separate the open software from their success.

Analyzing the differences between substitutions of products on the software industry will allow us to identify the reasons for which the clients stay using certain software. Based on the analysis potential strategies could be developed to reverse this situation.
1.4.2 Scientific relevance

The competition between products and battles for dominance are well-documented concepts. However, little research has been carried out for these particular topics in the software industry. A study about the differences between the competition of products in both models could lead to a new understanding of which could be performed the substitution of products based in the openness of the software.

In the other hand, the open software could represent an interpretation of the open innovation theory under different circumstances. In this theory of innovation, part of the R&D necessary for the product is performed out of the firm. In addition to this, part of the R&D performed inside of the company and not used specifically for its products is made available to other companies. The royalties that other companies pay will represent an extra income to the original company.

The open innovation is based on principles that differentiate it from the closed innovation. These principles are shown in Table 1.1.

<table>
<thead>
<tr>
<th>Closed innovation principles</th>
<th>Open innovation principles</th>
</tr>
</thead>
<tbody>
<tr>
<td>All the smart people in our field work for us</td>
<td>Not all the smart people work for us. We need to work with smart people inside and outside our company</td>
</tr>
<tr>
<td>To profit from R&amp;D, we must discover, develop and ship ourselves</td>
<td>External R&amp;D can create a significant value; internal R&amp;D is needed to claim some portion of that value</td>
</tr>
<tr>
<td>If we discover it ourselves, we will get it to the market first</td>
<td>We don’t have to originate the research to profit from it</td>
</tr>
<tr>
<td>The company that gets an innovation to the market first will win</td>
<td>Building a better business model is better than getting to the market first</td>
</tr>
<tr>
<td>If we create the most and the best ideas in the industry we will win</td>
<td>If we make the best use of internal and external ideas, we will win</td>
</tr>
<tr>
<td>We should control our intellectual property, so that our competitors don’t profit from our ideas</td>
<td>We should profit from others’ use of our intellectual property, and we should buy others intellectual property whenever it advances our own business model</td>
</tr>
</tbody>
</table>

Table 1.1 The principles of the open innovation compared with the close innovation (Chesbrough, 2003)

This model shares some ideas to the implementation of open software in companies like Apple, Mozilla and IBM. Those companies make part of the development of their products (Safari, Firefox and Linux) in house, but also they include development performed outside of the company in their products. Most of them do not pay royalties for this usage, but in exchange the other parties do not pay for the usage of the company’s developments. Contrary from the traditional open innovation theory, the developer is not really the owner of the code. A comparison between the open software and the open innovation is shown in Table 1.2.

Based on this table, we state that the open software is an adaptation of the open innovation theory for the software industry. However the specific characteristics from this industry make it unsuitable to apply the principles of the open innovation. The exploration of these characteristics will represent an interesting study on how the open innovation theory can be adapted under different conditions. These characteristics will be explained in the next chapter.
### Methodology

The methodology of the project will be divided in six parts directly related to the six research questions. The questions will be answered on the proposed specific order. The answer to each question will give support to the next one.

1. **The software industry is different from other industries. What are the characteristics of the software industry that differentiate it from other industries?**

This research question will be answered based on a short analysis of the software industry. It will be based on literature research and professional experience of the author. This research question will be approached during chapter 2.

2. **Is the dominance relevant in the software industry and what is the model to achieve dominance in the software industry?**

This research question will be answered based on the literature review. Reviewing the literature about battles for dominance from the general to the particular will generate a clear image of how domination is achieved in this industry. This research question will be answered in the chapter 3.

3. **In the software industry the substitution of products is performed regularly. How does this happen and what is the influence of the openness of the software on the substitution?**

The process of the development of industry of software will be studied. Understanding the process of software creation and how this particular industry behaves. The research will clarify how the products are substituted. This research question will be answered in the chapter 4.
4. What is the current situation of the software market for the open products and their competition with the proprietary products?

A case of study will be performed to solve this question. The case of study will analyze the competition in a specific software market. Based on their statistical information both products will be compared to identify the reason of dominance. This will give us a snapshot of the industry showing the current situation of the open software. This research question will be answered in the chapter 5.

5. Based on the findings of the first four questions, what strategies the open software suppliers can apply to gain market share against the proprietary software products?

Based on the information obtained to answer the previous research questions some strategies will be formulated. These strategies will help the software suppliers to increment their market share. This research question will be answered in the chapter 6.

6. As follow-up of the proposed strategies in the previous question, what strategies can the proprietary software suppliers apply to maintain their market share against the open software products?

Derivate from the previous question, specific strategies will be formulated to increment the market share of the open software suppliers. This research question will be answered in the chapter 6.

1.6 References

Scientific Papers


- Van Wendel de Jode, de Bruijn & van Eeten (2002) *Protecting the virtual commons* Faculty of Technology, Policy and Management, Delft University of Technology


Practitioner Papers

*Data collected*


Chapter 2  Characteristics of the software industry

2.1  Introduction

The software industry is relatively young (compared to other industries like chemical or electronic industries). However the software industry presents some special characteristics that differentiate it from the others. The present chapter addresses some of those differences.

The first part analyzes the high production cost of the first copy and the insignificant marginal cost of the following copies as the first characteristic of the industry. The second section analyzes the different types of users for the industry products. The third part of the chapter analyzes the different segments of the software market. A fourth part relates to the types of user with the different segments of software market. The fifth part analyzes the organizational characteristics that appear in different segments of the software industry. A sixth part refers to the concept of “Software as a Service”.

2.2  Production and marginal cost

The software products differ to other products in several ways. But their main difference is related to the nature of the software products itself. The software products are linked to the concept of “softness”. They are essentially a not-tangible product. This means that although the existence of the software product is unquestionable to the user and the tasks that the product performs fulfill his needs, he cannot touch the product. This concept of softness could appear to be insignificant. However, it leads to create interesting differences between tangible products of other industries and the software products.

Several software products are sold in boxes as any another product. However this is only a tangible representation of the virtual product. The storage media that contains the product is only part of the distribution channel. It is certainly not the product itself. At the end, the product in the box is the same than the product downloaded from a server, just with a different distribution channel.

But, how this affects the industry? In the proprietary software model, the cost of software development represents a high part of the production cost. This cost represents mainly the cost of the first copy of the software. In the beginning of the industry, when the software was custom-made for the user, the whole cost of the development was charged to the owner of the first (and unique) copy, elevating the cost of this copy.

After the appearance of standard architectures, the software became generic and started to be used by different users to fulfill their needs. This was an incredible change for the industry based in the fact that the cost of product a second copy of the product is insignificant. This cost only involves the cost of the storage media, the bandwidth, or the carton box of its package. The real cost of replication is insignificant.
The softness of the product led to the fact that the cost of producing the first copy of the product differs just a little from producing a million copies. The only extra cost is the further replication of the software.

2.3 Types of users

The software consumers are also named users. There are diverse classifications of users depending of some factors like: usage and knowledge. These two alternative classifications will be explained next. (Van Wendel de Jode, de Bruijn & van Eeten, 2002):

2.3.1 Classification of users by usage

Traditionally the users of software can be divided by their usage in:

- Enterprise users

  IT experts make the selection of the software for these users. The license of these products is normally specific for the enterprise use. The IT experts are also the main responsible on the technical support. The enterprise users perform their task in the applications as part of their job.

- Personal users

  Personal users are the users that have complete control of their software selection. However they do not have an expert IT technical support in case of problems. The task that this user performs is more diverse, ranging from leisure activities to office work.

2.3.2 Classification of users by knowledge

An alternative to this classification based on the usage, the users can also be divided by their level of knowledge (Van Wendel de Jode, de Bruijn & van Eeten, 2002):

- Regular users:

  This group is constituted by users, who perform tasks over their computer applications and some basic computer maintenance tasks (like installation of applications or software update).

- Advance users:

  This group performs also the task that the regular user does, but they have also the knowledge and skills to perform some advanced tasks. Some of the advanced tasks can be: major updates in the system (like the installation of a Service Pack), installations of a operating systems, hardware modifications, management of server applications and
software development. The advance users are normally enthusiastic about technology, and like to experiment new technology products (they are considered to be early adaptors).

2.3.3 Distribution of the users in both alternative classifications

The two alternative users’ classifications explained overlap. They are two different ways to classify software users. However the software users are not equally distributed through both classifications. Table 2.1 shows how in both classifications the distribution of the users is.

<table>
<thead>
<tr>
<th>Usage</th>
<th>Knowledge</th>
<th>Regular users</th>
<th>Advanced users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise users</td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Personal users</td>
<td>High</td>
<td>Low</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1 Distribution of users in both users’ classifications

2.4 Segments in the software market

The software can be classified on their use. Some literature describes three main segments of software

- System software
- Development software
- Application software

Each of these segments is explained in the following section.

System software

It refers to the platforms over which all the software operates. This segment basically is composed by operating systems. It also includes system utilities and hardware drivers for the specific platform. These platforms are used by any kind of user. Most of this software is installed by the computer suppliers and stays unchanged. This issue gives bigger importance to the installed base of each platform. It also contributes to create high switching costs and high cost for retraining.

Application software

It refers to the software created to perform a determinate task. This software is installed over a platform or “system software”. This segment is divided into three sub-segments:
• General applications
These applications perform general tasks, computer or not-computer related. They are used by any kind of user, and their lock-in varies depending of the case. However, based in the fact that this software is installed over a platform it can be replaced easily by a competitor. This creates a competitive sub-segment of the industry in which the market share can change really fast. In this segment the integration, ease-of-use, fit & finish, etc. are key attributes (Vallopillil, 1998). Examples of these applications are: web browsers, mail clients, office suites, security suites and media players.

• Server applications
These applications are installed in servers and offer services to client applications. Although these applications offer services to any kind of users, that part of the process is managed by a client application that belongs to the “general applications” sub segment. These applications are normally used by advanced users. This fact reduces their lock-in and network effect possibilities. This is because an advanced user can easily be re-trained to use different product and he usually shows interest. From the perspective of the service’s user, the switch between different server applications does not interfere with his experience of the service (Vallopillil, 1998). Examples of these applications are: mail server, file server and web server.

• Special applications
These applications perform very specific tasks in organizations and are normally software developed specifically for every case. The users of these applications are professionals from the organization but not necessarily computer experts. The training to use these applications is important but the knowledge is most oriented to the core business of the organization than to the application itself. Usually specialized IT staff supports these applications. These applications are expensive and are planned to be in use during long periods of time. The amount of users per system is small and there is no competition between products like in other segments. Examples of these applications are: ERP, oil forecasters, banking systems and financial systems.

Development software
It refers to the tools to develop the software. This segment contains: text editors, compilers, debuggers and of course the Integrated Development Environments (IDE). They are designed to be used by advanced users, fact that make difficult to lock them in. Each of these products is heavily linked to a specific technology and is adopted more for the technological capabilities that can create than for their own capabilities. This segment tends to be too specialized and it is relatively small compared to the other. (Free On-line Dictionary of Computing, 2008).

2.5 The relation between types of users and segments in the software market
By using the different users’ types and segments of the software market, it is possible to create a table for each of the software users’ classifications. This can be done in combination with each market segments. These tables are shown on Table 2.2 and on Table 2.3.
It is important to state that not all the users have interest on using all kinds of software. The personal and regular users only use software to fulfill some basic needs. The more advanced products are only used by enterprise and advanced users. It is also remarkable to notice the fact that personal and regular users have similar uses of software even if they shown different categories of users.
2.6 Organizational characteristics of the software markets

The nature of the software development and its virtual existence gives another characteristic to the industry. The assets needed for software development could be insignificant. For the development of most of the products, only one equipment (similar to the one that will execute the product) is needed. This equipment only needs to have the development software incorporated. The distribution of the products can be made online reducing the cost of their distribution channel to a minimum (Raymond, 2000).

This lower need for assets leads to a really low entry barrier to the software market. A high amount of advanced users with knowledge in software development can launch their own product with the use of their personal computer. This low entry barrier gives an explanation to the huge amount of general application products in the market and the fierce competition on this segment. However, for other products like system software products the market is smaller. This can be explained based on the complexity that involves developing those products. It is reasonable that developing and managing a project of less than 10,000 code lines can be achieved by one person in one computer, but how a single person will manage a project like a full operating system with more than 10 million code lines on his own? (Valloppillil, 1998). To create products of that size and complexity more assets are required. The entry barriers can also be really high on the software industry under certain circumstances.

This complexity facts leads to a high and fierce competition in the general application segment, and in some part of the server applications with a huge amount of suppliers and products (Van Wendel de Jode, de Bruijn & van Eeten, 2002). However the system software segment shows something else. The competence is low and the number of real competitors is limited.

2.7 Software as a service

During the development of this report the software has been approached as a product. However this is not the only possible perspective. Some years ago, a new approach of software distribution has been deployed. It is generally named “Software As A Service” (SAAS).

A generic definition of what is the software as a service is obtained from the report “Software as a Service: Strategic Backgrounder” of the Software & information Industry Association: “In the software as a service model, the application, or service, is deployed from a centralized data center across a network - Internet, Intranet, LAN, or VPN - providing access and use on a recurring fee basis” (Software & Information Industry Association, 2000). The software as a service can be explained as a new distribution model in which the software is not located in the computer of the user. The software is located in the equipment in which the user has been granted access to use the service.

In the technical part, the software as a service does not represent a big change. It could be described as one or more server applications that are accessed by a remote user through a
local general application. For the user however, the change can be large. The SaaS’ application is usually provided to the users with transparency by the server application. Transparency is a technical concept that, in this case, states that the user does not need to be aware of the internal operation of the SaaS’ application. He only uses the SaaS’ application from the perspective of his user interface (a general application). All the management and maintenance of the SaaS’ application is not responsibility of the user. In the SaaS, the software supplier becomes an application service provider (ASP). They are not only responsible for creating the software, but they also host the application and are responsible for their correct functioning. They provide the application. They also installed it in a server that can be accessed by the user/clients, who use the application through the network.

The main difference from previous software’s business models is that in the SAAS the service could be provided as a rent or subscription and not as a permanent product. Added to this, most of the technical support is transferred to the application service provider. This change could lead to reduce the need of a technical staff. Other consequence of these services is the reduction on the need of infrastructure to implement new software. The new services, most of the times, do not need the installation of new equipment on the client side, reducing the time to implement and the cost of investment. In a broader view, the SAAS represents lower investment in the implementation of new software, but on the long term could represent higher cost. The characteristics of the industry let new business models to be implemented.

2.8 Conclusions

The characteristics found in this chapter answer the research question 1 of the project: “The software industry is different from other industries. What are the characteristics that differentiate it from other industries?”

One of the characteristics of the software industry is the type of users that they have. They could be divided by use: enterprise and personal, and by knowledge: regular user and advanced user. The software can be classified in three segments: System software, Application software and Development software. The application software is also divided into three: General applications, server applications, and special applications.

In synthesis, it can be stated that the industry of software is different from other industries by the “softness” or virtual nature of their main product. That generates a high production cost of the first copy but a low marginal cost that will mainly be the cost of the distribution of the product. This softness also impacts the low assets needed for develop new products lowering the entry barrier and allowing a fierce competition. However, more complex products can increment the amount of assets required, raising the entry barrier so high that the competition is low and dominance of the leaders is strong.

2.9 References

Substitution of Software
Chapter 2: Characteristics of the software industry

Scientific Papers


Practitioner Papers


Chapter 3  Theoretical Background

3.1  Introduction

The software industry has been studied for years. Year after year, several articles are published on different aspects of the industry. Several authors with different perspectives have studied the software competence and dominance between open software and proprietary software. In the first part of this chapter, these studies are addressed as a base for further research. These studies will be addressed from the general to the particular. Later on in the chapter, a model is created based on the literature. This model will explain the dominance in the software industry. In the last part of the chapter, it is described the methodology that will be performed during the development of this research.

3.2  Review of the existing literature

Several articles exist about the competence. One of the most important is: "How competitive forces shape strategy" by Michael Porter. This article published in 1979, proposed a framework known as “Porter’s 5 forces analysis”. The framework has the goal of determining the intensity of the competition and the profitability of the market. The 5 forces that Porter describes on his article are:

- Threat of new entrants
- Threat of substitute of products or services
- Intensity of competitive rivalry
- Bargain power of customers
- Bargain power of suppliers

A graphical representation of the framework can be observed on Figure 3.1:

![Figure 3.1 Porter’s 5 forces analysis](image-url)
Substitution of Software
Chapter 3: Theoretical background

The forces represent several aspects related to the competition within a specific industry. The description of each force is given in the next section:

**Threat of new entrants**
This force refers to the possibility that new competitors have to entry to the market. With the entrance of more competitors, the competence will be incremented, reducing the profitability of the industry.

**Threat of substitute of products or services**
This force refers to the opportunity that customers have to find products that substitute the product of the firm in case of price increase. When the thread is high, the prices are subject to high competence and profitability is reduced.

**Intensity of competitive rivalry**
This force refers to how is the behavior of the competitors in the industry. The fiercer the competence is, the lower the profitability of the industry becomes.

**Bargain power of customers**
This force refers to the influence of the customers in the firm. Buyer price sensitiveness, volume buyers, and other factors are part of this force. The more than the customer can influence in the firm, the more the profitability will be reduced.

**Bargain power of suppliers**
This force refers of the influence of the suppliers in the firm. Factors as the dependence of a supplier and the control of the prices by the supplier are factors are related to this force. The more than a supplier can influence the firm, the more negatively affected the profitability is.

(Porter, 1979)

Although this research is relevant, there has been some important criticism to it. The model of Porter does not include the relations between suppliers and buyers with the firm. Porter implies that suppliers and buyers do not create a relation with the firm throughout the time and previous deals. Added to this, the model of Porter does not incorporate the influence of those relationships in the competition. He does not show the influence of each force on the other four forces. This lack of inter-relation leads to an erroneous idea. This mistake is based on the belief that all the forces are independent instead of being interdependent. These two aspects are certainly present in the software industry. The software industry is clearly a network of actors that interact continuously. This fact limits Porter’s model when it faces the software industry. However some concepts based on Porter’s can be used to explain certain aspects of the software industry.

From a different perspective, it is well known that the competence in the software industry is highly influenced by the network effect (Lin, 2007). Van den Ende and Wijnberg states on the network effect: “The network effect means that the decision makers react to the decision or the expected decisions of dissimilar decision makers because these decisions affect the chance that the first decisions are or will turn out to be advantageous” (Van den Ende & Wijnberg, 2003). This can be explained as each new customer enhances the position of the product. This effect is based on the need that other potential customers have to interact with the new
customer (Chakravarty, Dogan, & Tomlinson, 2006). This effect influences the competition. It leads the dominant product and increments the strength of their position.

Through the years, the competitions for the software market had had a similar end. The winner of the competence had gained not only the dominant position in the market but also has gained most of the market. It has left some occasions the losers completely out of the market. This is the case of the dominance of Internet Explorer after the first browser war and the disappearance of Netscape (Chiaravutthi, 2006), the dominance of Microsoft Word over other word processors like Word Perfect (Chakravarty, Dogan, & Tomlinson, 2006), or the dominance of Microsoft Excel over other spreadsheets like Lotus 1-2-3 (Kaparthi & Power, 2004). In the software, often it is the case that the winners take-all. Based on these facts, it will be interesting to focus the research of competence on the dominance of the market.

3.2.1 General dominance of technologies

Before studying the dominance on the software industry, it is important to study the dominance between technologies in general. Fernando Suarez (Suarez, 2003) from London Business School on his article “Battles for technological dominance: an integrative framework” proposed a framework to understand the dominance process in the technology battlefield. On his research, Suarez relates how the press have track several “battles for dominance” in the technological field. Technologies like high definition audio technology, operating systems, VCR’s, internet web browser, etc. In all the cases there are several topics to be addressed like: the destiny of the actors involved in the battle, the complementary products and services around the main products.
Suarez cites some other works to strengthen his statements. Arthur (Arthur, 1998) identifies the internet boom in the 1990’s as a factor that contributed to the “standard war” and network effect. However, other researchers like Liebowitz (Liebowitz, 2002) do not fully agree with him. Others works state the complex process that lead to the technologic dominance of a particular technology. Several technologies compete, but some of them have been discarded during the process. At the end, one of the technologies is clearly the leader (Clark, 1985).

On his research, Suarez also proposes a model which identifies the factors that influence the technology battle. These factors are divided in two groups: the firm-level factors and the environmental factors. This model is shown in figure 3.2.

In the model, both groups of factors influence the result of the battle for the dominance. There is one indirect influence of the environmental factors to the output of the firm-level factors. This indirect influence is motivated by the fact that Suarez considers that the environmental factors moderate the influence of the firm-level factors.

In his article, Suarez describe each of the factors:

- Firm-level factors
  - Technological superiority
    *It involves the technical skills of the competitors. The technical superiority allows the firms to create technically superior products. However this superiority is not always relevant in the battles for dominance.*
  - Complementary assets and credibility
    *Complementary assets like manufacturing capabilities or financial support had their role in the battle for dominance. The credibility is related to the branding that the firm has. A better set of complementary assets and a better credibility as firm normally leads to the dominance.*
  - Firm’s strategic maneuvering
    *It is divided in:*
      - Timing to entry
        *It refers to the timing that the firm had to enter to the market. The firm can enter really early in the market to capture the early adopters, or entry late once that the technology has been developed. An early entry gives advantage to the firm of early learning, and helps to create an installed base. However, it can lock the firms with certain technology (Dosi, 1982) and it does not lead firms to maximize their survival chances. These chances are maximized by companies that enter the market few years before the emergence of the dominant technology (Christensen et al, 1998).*
      - Price strategy
        *This refers to the price maneuvers that the firm takes in order to attract more customers. An early aggressive pricing can lead to create a large installed base (Katz and Shapiro, 1985).*
      - Licensing policy and relations with their complementary products
        *The establishment of complementary products through licenses of the technology and keeping good relations with them is also important in the*
battle to achieve dominance. An open architecture can help attracting complementary products suppliers (Khasam and Mowery, 1994). However, it can also lead to an increase on the competition and the lost of control of the technology (Garud et al, 2002)

- Marketing and public relationships
Create positive expectations about products before its launch can reduce the sales of the competitor’s products during certain period.

- Size of a firm’s installed base
The installed base refers to the amount of users that a product has. The installed base can influence other costumers’ demand if the network effects are present (Katz and Shapiro, 1985).

- Environmental factors
  - Regulation and institutional intervention
Government decisions also tend to influence the competition. The selection of national standards for certain technologies or the application of certain laws can lead to the success or failure of the technologies in the battle.

- Network effects and switching costs
The network effect is a positive relation between the demand and the number of users of certain technology. This fact added to the possible switching costs can greatly influence the decision of the potential users about certain technology to use.

- Regime of appropriability
It is the ability that the developer of a technology has in order to block the entry of new competitors to the market or emulate their technology results. A strong regime will allow the firms with better technology to keep their advantage. Reducing the other competitors to catch up

- Characteristics of the technological field
It is divided in:
  - Number of actors
The different actors influence the field in different ways and the firms are aware of it. The number of actors involved in the field and the amount of power of each reveals the structure of the technology field.
  - Level of cooperation vs. competition
The cooperation in a field or the competition could determine the dominant technology based on their own features.

Using this model we can determine the reasons of the dominance of certain technology. An example is the Compact Disc, which battle for dominance is shown in Table 3.1.
### Table 3.1 Practical example of the factors of Suarez

<table>
<thead>
<tr>
<th>Group</th>
<th>Factor</th>
<th>Compact Disc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firm level factors</td>
<td>Firm’s technological superiority</td>
<td>Sony and Philips joined their forces; Philips contributed their knowledge on the manufacture of lasersdiscs, and the EFME, technology that allows long playing time and a high resilience against disc defects. Sony contributed with the error-correction method, CIRC.</td>
</tr>
<tr>
<td></td>
<td>Firm’s complementary assets and credibility</td>
<td>With decades in the industry, Sony and Philips represent two strong brands on the electronic equipments. This fact was added to the support of multiple other media suppliers (led by Sony music business).</td>
</tr>
<tr>
<td></td>
<td>Firm’s installed base</td>
<td>This product is a standalone product. It does not need an installed base.</td>
</tr>
<tr>
<td></td>
<td><strong>Entry timing</strong></td>
<td>The product was launched to the market early in the market of digital media.</td>
</tr>
<tr>
<td></td>
<td><strong>Pricing</strong></td>
<td>The price of the product was high at the beginning but with a considerable price drop with time.</td>
</tr>
<tr>
<td></td>
<td>Firm’s strategic maneuvering</td>
<td>Sony and Philips agreed to create a standard. They licensed the manufacture of media and players to any company that wanted to join.</td>
</tr>
<tr>
<td></td>
<td>Licensing and relationships with complementors</td>
<td>The product was introduced around the world, as the new quality product in media.</td>
</tr>
<tr>
<td></td>
<td><strong>Marketing and PR to manage expectations</strong></td>
<td></td>
</tr>
<tr>
<td>Environmental factors</td>
<td>Regulation</td>
<td>In 1979, the Japanese Ministry of Industry and Technology (MITI) organized a conference to discuss the creation of an audio disc. After the conference, the agreement between Sony and Philips was signed. The involvement of the government made the industry standard possible.</td>
</tr>
<tr>
<td></td>
<td>Network effects and switching costs</td>
<td>The old music systems were not able to play CD. At the beginning, it was necessary to buy a new system to play CD. Later on, some complementary products could be attached to the old systems to play CD. However, it was not necessary due the low price of the new systems.</td>
</tr>
<tr>
<td></td>
<td>Regime of appropriability</td>
<td>The technology behind the CD was strongly protected by patents. However, the licensing policy was so open that allowed multiple firms enter in the market.</td>
</tr>
<tr>
<td></td>
<td>Characteristics of the technological field</td>
<td>The number of actors in the market is huge.</td>
</tr>
<tr>
<td></td>
<td>Number of actors</td>
<td>Practically all the industry decided to cooperate with the format.</td>
</tr>
<tr>
<td></td>
<td><strong>Level of cooperation vs. competition</strong></td>
<td></td>
</tr>
</tbody>
</table>

Based on this application of the dominance model of Suarez, it can be observed the reasons for the dominance of the Compact Disc. The model works for the consumer electronics industry. But, how will it work for the software industry?

#### 3.2.1.1 Discussion on Suarez’s model

Regarding the Suarez’s model, the author has identified some relevant concerns that should be addressed.

The first issue refers to the organization of the factors. The main four firm-level factors have a natural relevance on the battles for the dominance in any industry. However, the environmental factors seem to be a heterogeneous mix of external factors that influence the battle. We have in one hand the “Regulation”, which is an external factor that represents the influence of a third-party stakeholder (government) over the battle. We also have other factors like: “Regime of appropriability” and “Characteristics of the Technological Field”, which
are more linked to the technology field. The last of Suarez’s environmental factors: “Network effects and switching costs” is related to the market and other economic issues.

From the author’s perspective, the environmental factors should be renamed as external factors and divided in three different groups: (a) Technology field factors (that should include all the technology related factors, which influences the battle), (b) Third party stakeholders factors (that should include all the influences of external organizations to the battle), and (c) Economic factors (that should refer to economic issues, which influence the battle).

A second concern on the model is the indirect influence that Suarez’s environmental factors have over the output of the firm-level factors. From the author’s point of view, a direct influence of the environmental factors to the firm-level factors should be considered closer to reality than the indirect influence over their output.

The third concern about Suarez’s model is the importance that each of the factors has over the result of the battle for market dominance. Suarez has never established which of the factors is more relevant or whether they are equally important. However, due to the differences among the different markets on which the model can be applied, the lack of “weight measurement” on the factors is understandable. The effect of each of the factors can be enhanced or reduced by others. Consequently the weight of each factor should be addressed on a case-by-case basis. However, some guidelines could be offered if the model is adapted to certain technological field.
A fourth concern is about the interrelation of factors. It can be clearly observed in the software industry that some of the factors that Suarez proposed are inter-related. Example of this issue is the network effect of the environmental factors and the installed base of the firm-level factors. These both factors are clearly interdependent based on the fact that a larger installed base enlarges the network effect of a product. Also the high network effect tends to enlarge the installed base. From the author’s point of view, the influence does not only exist from the environmental factors to the firm-level factors but there is also an influence from the firm-level factors to the environmental factors.

A last concern refers to the communities that develop some of the open software products. They are software suppliers but not firms. That is why the “Firm-level factors” need to be renamed on this modified model as “Supplier-level factors”.

As result of these concerns, a modified Suarez’s model is proposed. This model is shown in Figure 3.3

### 3.2.2 Competition in the software industry

Suarez’s model represents a general view of the battle for domination between technologies. However the software industry features some characteristics that maybe differ from other technology industries. Are these differences influencing the way of competition in the software industry? A literature review focused on competition related to the software industry will follow. Based on this review, the author will be able to create a model, based on Suarez’s model. This model will reflect the competition in the software industry. On the literature review about this topic, three articles were found. These articles are:

- “Impact of user skills and network effects on the competition between open source and proprietary software” by Lihui Lin
- “Two-Sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry” by Nicholas Economides and Evangelos Katsamakas.

These three articles are analyzed in the next section.

In his study “Impact of user skills and network effects on the competition between open source and proprietary software”, Lihui Lin (Lin, 2007) from the Boston University School of Management performs an economic analysis of the software market.

Lin focuses his research on the demand side of the competition between the open software and the proprietary software. He points that the users’ skills are an important issue on the decision making process related to the software adoption for the companies. Lin bases this importance in three empirical facts:
The open software was adopted initially by developers and expert users. Today, the open software has been adopted by less trained people.

The open software is considered more difficult to set up and install. The firms are concern to the high cost to support systems based on open software.

The cost to use open software systems is different for each firm. It varies based on the skills and expertise of their IT staff and organization. In proprietary software, the firms rely on the supplier for upgrades.

Lin also states that most of the firms have a lack of the skills to modify their open software installed. This issue reduces the impact of one of the most important features of the open software: the possibility of customization.

In the literature review of his work, Lin addresses the fact that multiple authors recognize the impact of the network effects in the software market. In the specific case of the operating systems, Lin mentions that when a product achieves a large market share, more developers produce complementary products to it. A wider user base also increments the possibility of collaboration and communication between his users.

In his paper, Lin performs an economical analysis of the competition of two products. In different scenarios, he makes his research modifying the value of factors like the user skills and the network externalities.

- Without networking effects open software and proprietary software will share the market. When users become more skilled, the open software will start to gain market share.

- With networking effects, the proprietary software will become a monopoly. If the skills of the users in general are low, they are not able to use open software. This will allow the supplier of proprietary software to charge high prices for his products. In this case, the open software will capture a small market share in which the more skilled users are. If the skills of the users are generally high, the supplier of the proprietary software will reduce his prices trying to reduce the attractiveness of the open software. In any of these two cases, after a beginning where the open software capture a small market share, the proprietary software will prevail at the end clearly as the dominant player.

These two scenarios can be summarized in Table 3.2:

As general conclusion, Lin state that the only way that the open software could prevail with network effects is whenever it would offer better benefits than the proprietary software. The main advantage of the open software, the customization, will only represent a real advantage when there will be enough skilled users for who customization would represent a good solution. For the specific case of operating systems(where Linux has an important market share in the servers but not in the desktop) Lin predicts that without a big change in the open software skills of the population, the market structure of the desktop OS market will remain the same (Lin, 2007).
### Substitution of Software

**Chapter 3: Theoretical background**

<table>
<thead>
<tr>
<th>Without network effects</th>
<th>In general low users skills</th>
<th>In general high users skills</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OS and PS share the market (50% - 50%)</td>
<td>OS gains market share (+50%)</td>
</tr>
<tr>
<td>With network effects</td>
<td>PS - High prices</td>
<td>At the beginning: OS become attractive – Gains market share</td>
</tr>
<tr>
<td></td>
<td>PS – High market share</td>
<td>PS try to make his products more attractive - PS suppliers reduce prices</td>
</tr>
<tr>
<td></td>
<td>OS – Low market share</td>
<td>Reaction: PS products become attractive after the price reduction</td>
</tr>
<tr>
<td></td>
<td>OS – High skilled users</td>
<td>PS recovers market share</td>
</tr>
</tbody>
</table>

**At the end:**
- PS - High market share
- PS – Low price

**OS:** Open software  
**PS:** Proprietary software

Table 3.2 Scenarios of the competence between open software and proprietary software (Lin, 2007)

Ramon Casadesus-Masanell and Pankaj Gemawat from the Harvard Business School in their article “Dynamic Mixed Duopoly: A Model Motivated by Linux vs. Windows” (Casadesus-Masanell & Ghemawat, 2006) analyzed the competence between the two most important operating systems. They analyzed the competitive dynamics that the non-profit competitor introduced to the market dominated by a profit-oriented competitor. They pointed 4 assertions of open software that are relevant for their research: the flexibility of the open software to adapt to the needs of the users, the incompatibility of the open software with the protection of property rights, the governmental promotion of the open software and the vision of the buyers to move to the open software.

Casadesus-Masanell and Gemawat addressed several authors to solidify their article. Kuan (Kuan, 1999) simplifies the decision between proprietary software and open software in a “make-or-buy” decision. Kuan concludes that the open software has much more advantages for the programmers than for the no programmers. This finding is closely related to the results of Lin about the customization that only represents a real advantage for the skilled users. Bensen (Bessen, 2002) concludes that since in open software every user test the part that has most interest for him, the most important parts will be tested more than the important parts of the proprietary software.

Furthermore based on their literature review, they partially attributed the effort to develop open software to not fully economic motivations from the IT practitioners. Motivations like altruism, status depending in the quality of the work done, and of course dislike to the proprietary software suppliers. Bitzer (Bitzer, 2004) modeled the fact of the closer in functionality that the open and proprietary software products are, the price of the proprietary product will collapse, and the cost of development of it will become unsustainable.
Casadesus-Masanell and Ghemawat also pointed a “virtuous cycle” in the open software, where the suppliers attract users with a quality product and let them collaborate to improve even more the quality of the product. In their model, they prove some important statements.

- The marginal cost of Linux and Windows is almost zero.
- Microsoft has a high fixed cost for the development of Windows. In comparison, Linux fixed costs are almost zero.
- In case that Microsoft uses a monopoly price strategy, Linux will replace Windows based on its “price-zero” strategy.
- The price strategies performed by Microsoft, have prevented the rise of Linux.
- The strategies of Microsoft and not the quality itself of the product are whatever makes Windows prevail as leader.
- Piracy reduces directly the profits of Microsoft. But at the same time adds users to Microsoft installed base. A bigger installed base of Windows makes harder the diffusion of Linux.

Even if the article written by Casadesus-Masanell and Ghemawat uses the case of the operating systems to analyze the competition between profit-oriented against zero-profit organizations, the statements made in the section below reveal important aspects of the software industry. (Casadesus-Masanell & Ghemawat, 2006)

At last but not least important, Nicholas Economides from Stern School of Business at New York University and Evangelos Katsamakas from Graduate School of Business at Fordham University published the article “Two-sided competition of proprietary vs. Open source technology platforms and the implication for the software industry” (Economides & Katsamakas, 2006). In their research, they develop a model based on two sides: the price strategy to the final users and the price strategy for the complementary products suppliers or application developers. The combination of both strategies affected directly the adoption of the platform.

Based on this research, it is stated that even if the open software platform has zero-cost. The users must cover the switching costs, possible high support costs and a small amount of complementary products. The complementary products of the open software also tend to be proprietary software. This creates important network externalities for the main products and profitability for their development. This statement is aligned with the findings of Lin and Casadesus-Masanell and Ghemawat.

Economides and Katsamakas state that the increase on sales of applications increases the demand for the platform itself. That issue derivates in subsidies from the platform supplier to the application developers in an attempt to increment the network of the platform. Similar ideas lead the application developer of the open software platform to subsidize the switching costs of the new platform to their customers. This research also found that the profitability of the industry is higher when the platform is proprietary and the applications are vertically integrated as Microsoft Windows. However, an industry based on an open software platform has a higher social welfare and the variety of applications is larger. It also points that when a
Substitution of Software
Chapter 3: Theoretical background

Proprietary platform competes with an open software platform the proprietary dominates the market share and profitability.

In difference to Casadesus-Masanell and Ghemawat (Casadesus-Masanell & Ghemawat, 2006), Economides and Katsamakas do not ignore the price to the complementary product suppliers. They give special attention to the strategies of Microsoft to the complementary applications. Economides and Katsamakas conclude that the willingness to subsidize the applications providers is less, if the complementary products can be easily substituted (Economides & Katsamakas, 2006).

This literature review has revealed some important issues. Lin, Casadesus-Masanell & Ghemawat and Economides & Katsamakas address the importance of the network effect in the software industry. They also agree on addressing the importance of the price strategy in the industry. Based on this, the author can conclude the importance of both aspects. Lin and Casadesus-Masanell & Ghemawat also agreed about the lack of importance of the customization ability of the open software.

The positions of the three articles are never opposed. However Casadesus-Masanell & Ghemawat have a different position regarding the price strategies of the complementary products than Economides & Katsamakas. This difference can be provoked by the differences of circumstances between different software segments.

Individually, Lin pointed two important issues: The importance of the users' knowledge and the importance of providing better benefits than the competitors. Casadesus-Masanell and Ghemawat also addressed two important issues: the no-economical motivations of the open software developers to contribute to the product and the 4 assertions of the open software (flexibility, incompatibility with property rights, government promotion and vision of buyers to move to open software).

Besides the issues found in these three articles, it is important to go a step further looking for some other issues to complement Suarez's model.

- Not all the factors involved in the competition are related to the product itself. Some of them are the result of social issues. The regulation of the different governments can influence the diffusion of the technology. Some governments have positive influence on the products like the effort of the Dutch government to incentive the use open software and other open standards (NOiV, 2008).

- However, the governments can also have a negative impact for some products. An example of this could be that the trade embargo of the United States government applies to certain countries (Moreno & Moreno, 2002). This embargo blocks the sales of American products to those countries. Since products like Microsoft Windows are not an option to be used in those countries, some other platforms that are not affected by the embargo get a bigger portion of the market.
3.2.3 Suarez’s model in the software industry

Based on the previous literature research made in the software industry of the operating systems, a new modified model can be created. The model will be based on Suarez’s model that the author modified in the section 3.2.1.1. This model had differences from the original model based on the characteristics of the industry. First at all, some factors have been added to the model because their importance in the industry. These factors are:

**Complexity of the product**

Refers to the importance of the complexity of the product has over the competence. As it was stated in chapter 2, a product or a market segment where the complexity of the products is low, tends to allow the entry of smaller firms. A lower entry barrier, as Porter established, raises the competition and reduces the profitability. However a product or a market segment with high complexity tends to allow only large firms to enter with more assets to invest in development.

**Substitution of the product**

This factor refers to the easiness to replace a product for the user. It represents the difficulty that the target user will have to replace the product in their system. This could be related to the skills of the user or to the difference of use between the products in the battle for dominance. The first part refers that a regular user can replace his media player, but not his operating system. The second it is about a product will have better chances to dominate if it offers better features internally but its interface is similar to the dominant. The user will not have the need to learn to use new software. He will only enjoy the new characteristics added to it.

**Target users knowledge**

The target users’ knowledge represents a big difference in the software industry as Lin stated it. The difference between enterprise and personal users and between regular and advanced users is sometimes determinant on the selection of the product to be used.

On the other side, some factors proposed by Suarez on his model were removed due their low impact in the battles for dominance in the industry. The factors removed are:

**Regime of appropriability**

This factor has small relevance on the industry. The ability to replicate the software features is high. The block effect of the secrecy of the source code of software can be removed when a talented programmer copy the usability of the product. Even if the architecture and the source code of the product are kept secret, a talented programmer can create an alternative that give similar results. This fact makes irrelevant the regime of appropriability in the software industry.

**Characteristics of the technological field**

This factor is not eliminated. It is divided into three new factors added for the industry. Having the factors included in a more important position is not necessary anymore to keep this factor on his place.
The resulting model, which represents the battle for dominance for the software industry, is shown on Figure 3.4:

![Diagram](image)

**Figure 3.4 Factors influencing the outcome of the technology battle in the software industry**

This new model concentrates different aspects of the software industry in the model developed by Suarez. Some notes about each of the factors of the new model can be found next.

**Supplier- Level Factors**
- Supplier’s technological superiority
  - The superiority can be temporal. In a relatively short time the superiority can be reversed.
  - Implement better or unique features increases market share.
- Supplier’s complementary assets and credibility
  - The basic assets needed to develop software are low, leading to the decrease of value of this factor.
  - The personal users are mostly interested in the performance of the product itself and not in the complementary capabilities of the firm.
  - Other assets like technical support and legal liability important for enterprise users.
  - Personal users accept products from firms without references.
Enterprise users only accept products with a tested performance and fully supported by a well known software supplier.

- Supplier’s installed base
  - This factor is one of the most important on this industry. It gives an important advantage for the products. It is inter-related with the network effect
- Supplier’s strategic maneuvering
  - Entry timing
    - To be “first to the market” gives almost definitive advantage creating a installed base and network effect
    - It becomes really important due the link with the Firm’s Installed Base.
  - Pricing
    - It involves strategies to lower the price, but also product bundling, trial versions, shareware etc.
  - Licensing and relationships with complementors
    - The differences on licensing can modify the flexibility of the products for enterprise users, enhancing their willingness to adopt.
    - The complementators with more knowledge of the main product can create better-integrated solutions.
  - Marketing and PR to manage expectations
    - The maneuver of the firm and their image can have a big impact on the adoption of the advanced users.

External Factors
Third-party stakeholders Factors
- Regulation
  - The governments have policies that suggest the adoption of open software
  - Products are subjects to others regulations like: U.S. trade embargo.
  - The software products are difficult to control by the government.
  - The regulations are important for enterprise users.

Software field Factors
- Complexity of the product
  - Refers to how complex a product is. This is relevant to the capabilities of the firms involved:
    - If a product is simple, every company can enter the competition.
    - If the product is complex, only large companies can compete
- Substitution of the product
  - Refers to how easy is to replace a product for another one
  - If the product is easily replaceable the network effect is reduced greatly
- Target users knowledge
  - Refers to the level of knowledge that the users need to use the product
    - Regular users tend to prefer the product that they know or the product that is easier to use.
    - Advanced users tend to like new and technically better products.

Economic Factors
- Network effects and switching costs
For the enterprise users, the switching cost is less important than the Total Cost of Ownership (TCO). For them the switching costs are included in the TCO.

- Network effects have a reduced impact for enterprise users.
- Network effects are important for personal users.
- The switching costs of personal users are increased by the need of re-learning and the rejection to do it.
- The network effect is inter-related with the Supplier’s installed base.

Based on this model the battle for the dominance in the software industry can be evaluated. This model will be helpful in further chapters of this project. This model answers the research question 2 of the project. Table 3.3 gives a synthesis of the model.

<table>
<thead>
<tr>
<th>Group</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supplier-level factors</strong></td>
<td>Supplier’s technological superiority</td>
</tr>
<tr>
<td></td>
<td>Supplier’s complementary assets and credibility</td>
</tr>
<tr>
<td></td>
<td>Supplier’s installed base</td>
</tr>
<tr>
<td></td>
<td>Entry timing</td>
</tr>
<tr>
<td></td>
<td>Pricing</td>
</tr>
<tr>
<td></td>
<td>Supplier’s strategic maneuvering</td>
</tr>
<tr>
<td></td>
<td>Licensing and relationships with complementors</td>
</tr>
<tr>
<td></td>
<td>Marketing and PR to manage expectations</td>
</tr>
<tr>
<td><strong>Third-party stakeholders factors</strong></td>
<td>Regulation</td>
</tr>
<tr>
<td><strong>Software field factors</strong></td>
<td>Complexity of the product</td>
</tr>
<tr>
<td></td>
<td>Substitution of the product</td>
</tr>
<tr>
<td><strong>Economic factors</strong></td>
<td>Network effects and switching costs</td>
</tr>
</tbody>
</table>

Table 3.3 The factors of dominance in the software industry (based on Suarez model)

### 3.3 Description of the research

In some point of the life of a company, the computer software installed must be replaced. An analysis must be performed to choose the best way of replacing the software. The software market could offer several options, all of them with individual advantages and disadvantages. One of the most important choices in the last years is the selection between proprietary or open software.

Today, most of the companies use proprietary software in their operations. At the moment of substitution, what makes companies to stay in the proprietary software and what makes some others to switch to open source? The answer to this question will lead to answer the six research questions of this research.

This topic has multiple perspectives to be addressed. The demand, duopolies, network externalities and the two sided price will be interesting for the economists and the friendliness of the systems will be important for the end users. The total cost of ownership
Substitution of Software
Chapter 3: Theoretical background

(TCO) or the overall cost to deploy a computer system, which includes: license, support, training, etc., will represent an important issue for the managers, but the flexibility and transparency will be fundamental for the IT professionals. At the end, the openness of the software will be important in every decision or only in special cases? My perspective refers that the openness of the software influences certain aspects in some cases.

And the last part of the research focuses on observing the future of the industry. Is the open software a real option? Is a real option for any kind of use? Or is only advisable to be used in some cases? This leads us into the third research question: What strategies can the open software suppliers apply to gain market share?

3.4 Research Methodology

For the development of the present research firstly is necessary to have an overview of the software market. A case of study about a representative sector will reveal the actual state of the market, and how the trends of adoption are. This research will involve the most popular software. These products will be seen not from the perspective of his development, they will be seen as products trying to clarify the difference on strategies and whether they are different based on their openness.

The study will be performed to understand the life of the products. It will identify the differences provoked by their openness. The information will be compiled from the supplier’s websites and other specialized sources. This study will be based on the information of the products like the date of the introduction of each version and its replacements, and the date of end of support. The information compiled for each product will be analyzed and compared with other products. This will show us the life-time of the products and if this is different between proprietary software and open software. The last part of the case of study will involve an analysis of the market share of the products. This will show us how the behavior of the market is and how could be influenced by the openness of the products in the competition.

This study will give an overview of the past and the present of the software market and how it is influenced by the openness. But to define what could happen in the future and what the software suppliers need to do and can do, we need a different perspective. In specific we will try to discover the weak points and the advantages that the IT professionals see in the open software. Based on it, we can create strategies that can be used by the proprietary software supplier to keep their market share. Also we will suggest other strategies that the open software industry can use to improve their market share.

3.5 References


Substitution of Software
Chapter 3: Theoretical background

Substitution of Software
Chapter 3: Theoretical background


Chapter 4  Software development process and differences between open and proprietary software

4.1  Introduction

In the actual world, the software is the main product of a large industry. Companies like Microsoft, Oracle, Apple and IBM have earned millions of dollars selling different types of software.

The software is sometimes produced in factories. However, the software production process is not similar to the production of other products. Several factors make the software production process different from other manufacturing processes. In the first part of this chapter, this process is explained. Some other aspects are also reviewed in this part, like suppliers of (proprietary) software and their organizational characteristics. The second part introduces the open software as a revolutionary concept for the industry. This part also introduces open software suppliers and their characteristics. The third part of the chapter compares the open software and the proprietary software from different perspectives. In the last part of the chapter, a model of software substitution based on the concepts of this chapter is proposed. This chapter also relates the new concepts with concepts from previous chapters.

4.2  The software development

In the beginning of the computing industry, the software was introduced in the equipment each time that it was used. A high skilled user needed to load the program in memory before than the program was able to run. The process was slow, not really efficient and only high skilled professionals were involved. After some years, the secondary storage appeared. The software could be stored in the equipment and it just needed to be executed every time that it was used. In that moment, the software at that moment was custom-made for specific equipments. The variety of hardware and the lack of standards made really difficult for the programmers to create programs that could work in several platforms.

A few years later, IBM, the major builder of computer equipment those days, created several lines of enterprise products with similar architectures. This made possible that some of the software written for one model will be able to be executed in similar models. After a while, the first operating system was created and was used in similar equipments. The software developers started to produce software that could be used in multiple equipments without making modifications on the product.

<table>
<thead>
<tr>
<th>Operating System</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is a software product that manages the hardware resources of one specific computer architecture and offers an interface to the user. It is considered the main software of a computer and a platform for most of the software in it</td>
</tr>
</tbody>
</table>
The software started to be a product by itself, and not only a complementary product of the computer equipment. In that moment the software industry was born.

In the software industry the main cost is the software development process. That is the process throughout which the product is created and maintained over its lifecycle. In a classic conception, the software development could be divided in 4 main phases:

- Design
- Building
- Testing
- Maintenance

These four phases are explained in the following section:

**Design**

The phase starts with the definition of the requirements. A functional design on how the product must work is created based on the requirements. Following the functional design, the most experienced developers create a technical design. The selection of the developing team is performed here. The last stage of this phase contains the tasks division and their planning.

---

**Technical design**

It is how the system will work from the developer perspective. It involves all the architecture of the product’s components that will be created. This design is the master plan of a software product and is closely related to the technology chosen for development.

**Functional design**

It is basically the description of how the system will work from the user perspective. The requirements of functionalities will be described here and also the process that the user will perform using the software like: login to the system, save information, etc.
Building

Based on the planning, the functional design and the technical design, the development team builds the product. This could be the longest phase of the process, and could be quite difficult to create a detailed plan in advance. The experience of the team leader and his knowledge about the capabilities of his team can improve the accuracy of the planning and its execution. A beta version is released in the last stages of this phase. At the end of this phase a final version is released to the public.

Testing

This phase is usually overlapped with the last part of the building phase. In this phase a group of testers use the beta version of the product and try to find problems or bugs on it. During the complete phase, some of the developers must be assigned to correct the found bugs. It is methodologically planned for this phase to be significantly faster than the building phase. However depending of the characteristics of the software, it could be shorter or longer. After this phase, the product is released to the public as final version.

Maintenance

This phase involves basically the support of the product. Based on the feedback provided by the users, some corrections are performed during this phase. Depending on the type of software, other modifications could be performed like updates to add compatibility to new products, new legislation or extra features.

These phases could become an iterative process for further versions of a product. In fact, for some products, all the phases can run at the same time. While one version is on the maintenance phase, the development and testing of the next version could be running to be launched in a future. Even the design of a future version can be made. In this point, it is
necessary to differentiate the development of a product from the versioning of products. In one side, the development is normally a continuous process with milestones. The same project is been continuously build adding each time new functionalities to it. This means that a product is technically improved each day. However, for commercial reasons, the product is released in versions. This could make the illusion that the product is developed in big steps, when only the released versions the ones are made that way. This process is clarified with Figure 4.2.

In the traditional model of development, all this process is performed following the managerial decisions of the company.

Based on his conceptual representation, the software development process could appear as a simple linear process as it is observed in Figure 4.3

![Figure 4.3 Software development process simplified](image)

However this perception is not completely real. The real process is driven by the supplier’s decisions. Based on those decisions, this process is adapted to the characteristics of the product, making the process more flexible and almost unique.

![Figure 4.4 Software development process incorporating different product versions](image)

For example, when a product is planned to have different versions, the design of the product can never stop, starting to design the next version (v2.0) of the product just after the beginning of the development of the first version. The previous design phase has an influence on the design phase of the next version. The design phase v2.0 derivate in the phase building
v2.0, which is also influenced by the original building phase. During the development of any of the building phase, the corresponding testing phase also starts for the entire product version. After the testing, each version joins to a continuous maintenance phase. This process, much more complex than the original perception, can be observed in Figure 4.4.

Even when the process explained in Figure 4.4 is a simplification of the real software process, in some cases the design or the building phase can receive input from the testing or the maintenance phase from previous versions. These inputs will influence the following versions of the product. It is also possible to have a unique process in a spiral as is shown in Figure 4.5. On this representation, the process begins on the design phase. After this phase is complete, the building phase starts and is followed by the testing phase. Just after the testing phase, the product is released. Also after the release of the product, the maintenance phase begins in parallel with the second round of the design phase. This cyclical process can also be the representation of the software development process.

![Figure 4.5 Spiral representation of the software development process](image)

The software development process is so flexible that even some of the phases could be excluded. The only phase that is mandatory on the process is the building phase. The rest of the phases could be reduced or eliminated, depending on the strategies and decisions of the supplier.

Many other variations could be explained about the software development process. The real process is flexible and almost with unlimited options to be modified by the developer team. For the present research, only the original simplification of the process will be addressed. This is motivated due to the fact that more complex representations of the process will not have an impact on this research. However it is important to emphasize the real complexity of the real software development process.
To understand perfectly the software industry, it is important to go a little deeper in some of the phases. That is precisely the case of the Building phase, which is explained in the following section.

4.2.1 Concepts of software building

The software building is basically the writing of the software components in a selected programming language.

**Programming language**

It is a code that consists in reserved words and syntaxes that a compiler converts into binaries. Every language is different and present advantages and disadvantages from others. There are a huge number of different languages. However some of them are more popular than others.

The result of this process is called source code.

**Source code**

The source code represents the blueprint of the software. A programmer writes it on a programming language. It must be processed (compiled) by an application that “translates” the instructions in the original programming language into assembler language. The source code is readable by another programmer but it still cannot be executed by the computer.

The source code must be processed by a compiler to create a binary. It is shown on Figure 4.6.

![Figure 4.6 The compiling process](image)

**Binaries**

When a compiler “translates” a source code in binary code, it creates a “binary”. This is the file that the computer can execute. These files contain only binary code and cannot be translated back in the original programming language (unless can be “reassembled” or translated to assembler language which can be hardly read or modified). The binaries are the goal of the software development and are considered the final product. The binaries do not need the source code to operate.
**Binary code**
The binary code is based on discrete mathematics, where every bit has to positions ON (1) and OFF (0). It is only readable by the computer, and is the only language that can be naturally executed.

**Assembler language**
The assembler language is located in the lower level of the programming languages. It is the next step after the binary code and can be read by programmers. The programmer using assembler has access to any hardware resource of the equipment. However the use of assembler represents a higher degree of complexity. The assembler programming requires the indication of every routine. This fact could be as dramatic as write even the most basic operations in every routine. By example, in the assembler language do not exists other arithmetic operation further than ADD or SUB (for subtraction). To perform a multiplication in assembler the programmer must write the routine based on the operator ADD. The assembler could run directly in the firmware of a processor, making not necessary the use of a platform like an operating system. Today, the programming in assembler is reserved only to the program microcontroller or the kernel of an operating system.

The binaries and the source code are propriety of the developer. The final product sold to the costumer was traditionally the binaries. The user buys a license to use the binaries. The license usually includes some occasional updates. The updates are received through different channels available like: storage media (floppy disk, cd, dvd), ftp, the website of the supplier or an automatic update service throughout internet.

In counterpart to the binaries, the companies used to keep under secrecy the source code. The companies tried to ensure their technological advantage was maintained under this secrecy. This was common and practically the only way to make the business in the past. This model is called “Proprietary software”. This model is the most used today. However when the source is not made available to the user, he is also limited to the developments made by the supplier. This makes the products to lack of flexibility and they are not easy to be extended.

As it was stated before, this is a very classical conceptualization of the software development process. Factors like the software type, company type, openness of the software and customer affect the process. These facts lead to the conclusion that every software development process is unique.

### 4.2.2 The (proprietary) software companies

In previous chapters, it is stated that companies develop (proprietary) software as a product. But, are all those companies the same?
Substitution of Software
Chapter 4: Software development process and differences between open and proprietary software

The software companies producing proprietary software are not homogeneous. We find large companies like Microsoft with more than 90,000 employees around the world and with a market capital of more than US $250 billion (Microsoft Corporation, 2008), or Adobe Systems with 7,317 employees (Adobe Systems, 2008) and a market capital of US $21 billion (Google, 2008). However we also find small companies like Opera Software with 510 employees (Opera Software, 2008) and a market capital of US $522 million (Opera Software, 2008), or the private company Kaspersky Lab with 600 employees (Kaspersky Lab - About Us, 2008).

It is clear that the strategy of a small company like Kaspersky lab cannot be compared with a large company like Adobe Systems. The large corporations usually have a multiproduct portfolio, while small companies only offer a limited number of technical-related products. The large companies take advantage of their portfolios complementing their products, creating a network effect. That is the case of Adobe InDesign that uses PDF technology to export documents. The use of this technology, improves the position of Adobe Acrobat in the market (the natural product of the PDF technology). In the other side, Kaspersky only provides general applications dedicated to the computer security, like firewalls and antivirus. Apart from a bundled price and some integrated solutions, the network effect that they can create is limited.

Based on these facts, it makes sense to classify the proprietary software companies by their size.

- Large companies with a multiproduct portfolio of different technologies.
- Small companies with a technology-related portfolio

This classification will be addressed further on this report.

4.3 Open software

UNIX is an operating system. It was developed in the 1970’s and 1980’s by the University of Berkeley, AT&T and some another research programs. They used to share the code freely. In the 1980’s, AT&T tried to ensure their propriety over the software developed with their investments. They asked the developers and researchers to sign a non-disclosure agreement. Richard Stallman, a software developer from MIT, found that situation uncomfortable. His concerns were about the restrictions to the free exchange of knowledge. He created the concept of “Free Software”, which refers more to the freedom of knowledge than the cost of the software. He founded the GNU (“GNU is not UNIX”) project where everybody can create, download, modify, use and even commercialize software with freedom. Years after, Eric Steven Raymond and Bruce Perens recreated the concept to “Open Software”, which is a more accurate concept that implies the visibility of the source code (Wendel de Jode, Bruijn, & Eeten, 2002).

But what exactly is open software?

When a developer decides to make his source code public, he certainly declines to maintain his technical advantage. He is distributing his knowledge and technology. The term “open” means that the public has access not only to use the application (or binaries). The public has
access to use and modify the source code through a license. The license could vary in the way that allows the usage of the source: from an educational use where non-commercial activities are allowed, to the most extreme case, where anybody can create a commercial product based on the code. But it always keeps the constraint of maintain the source code public.

The project of Richard Stallman developed some basic applications. After a while, they decided to create an operating system. This project represented a complexity that they never faced before. In 1991, Linus Torvalds released a Kernel that he integrated with some of the applications already developed for the GNU project. The resulting system became the first complete open software’s operating system. Based on the use of the development made by the GNU project, the operating system was named GNU Linux. However, it is better known as Linux (Raymond, 2000).

![Kernel Diagram](image)

**Kernel**
The kernel is the core of an operating system. It is the component that controls the hardware resources for the operating systems. It is normally written in a low level language to make them faster and low resources consuming.

Linux achieved a good acceptance within the community of open software. Its developer community did grow on a constant rate. In short time, Linux became the flagship of the open software.

### 4.3.1 The open software development

The differences between open and proprietary software go beyond the cost and the availability of the source code. The development of open software differs from proprietary on several aspects:

*The open software is developed by a community.* The developers are enthusiasts about the project and develop the product mostly in their spare time. However, some private companies
also invest resources on the development of open software. This model of development eliminates concerns about the time spent on a particular component of the product. If some of the developers “over engineer” a component, the final product becomes better without any other concern. In counterpart, under the proprietary software model the project leaders try to avoid any kind of “over engineering”. This is because any extra effort is not paid or planned. In the proprietary software the idea is fulfill the requirements. In the open software is to create a “beautiful” (technologically advanced, that makes feel proud the developer) piece of software.

*The open software treats the users as part of the developer team.*
The developers involve the users on the development of the product. The communities of developers pre-release versions (called beta versions) that users can test and give feedback. The community uses the software, performing extensive tests of the functionality and stability of the product. This beta testing is carried sometimes more extensively than the test phase in the proprietary software model. With more eyes over the software and a variety of hardware configurations, the open software is tested closer to the conditions that will operate after its final release. This intensive testing can represent a huge difference when the products are competing in the market.

In the proprietary software, usually only a limited number of users have access to the beta versions. The new capabilities of the proprietary products are kept private until its release. However this limited testing could lead to do not discover problems in the software product before its release. The problems discovered by the user in the final version, can affect the operations of the client, damaging the image of the product and the company.

*The open software releases versions more often.* During the development of the open software they continuously release versions labeled as “unstable”. These versions contain non-tested additions to the product. Any time, a “stable” version is available, which has been tested enough to ensure the quality of the release. The availability of the “unstable” (and in the “state of art”) versions opens the possibility that the product can be tested by a broader group of users and creates a large community around the development as collateral effect. The feedback provided to the unstable versions helps the developer community to improve faster and even to implement new ideas from the community. In contrast in the proprietary software model, a version with bugs is not acceptable. The proprietary software suppliers are aware the huge implications of release a defective product. A product with bugs can represent a huge credibility problem to the supplier.

*The decisions are made in a decentralized way.* The direction of the development is decided by the contributors to the community. This is sometimes performed by voting of the community or by decision of the leaders. After the decision, if some part of the community does not agree, they separate from the original community, forming a new project with different ideas. This division is commonly called “Forking”. In the proprietary software projects, the top of the organization takes the decisions. This focuses the resources of the organization to follow the same line.

(Raymond, 2000)

Substitution of Software
Chapter 4: Software development process and differences between open and proprietary software
Based on the previous aspects, Raymond formulated a law called “Linus’ Law”:

“Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone”
(Raymond, 2000)

This law concentrates the technical essence of the development of open software.

4.3.2 The open software suppliers

In the beginning, the open software suppliers were essentially communities of software developers that invested their free time to create the “technically beautiful” software products. However, after some time, several companies joined the movement and invested on the development, marketing, and distribution of open software products. A few open software products started to be offered as normal software products. The open software products started to compete for the market against their proprietary peers. The development was kept as open software, but with a more organized structure. One or more companies were participating for the development of the product and most of the times they became the main responsible for it. The products were still offered for free, but they also started to offer premium products based on the same technology. They also started to offer expert technical support for their products for a competitive price. But apart from that, the companies involved in the open software also took the liabilities and commercial responsibilities of the products in a similar way that proprietary software suppliers do it. These changes reduced greatly the lack of credibility that the open software had until that moment. The open software was not seen any more just as a group of people with wonderful ideas. Now, the enterprise users had somebody to deposit their trust. An example of this “organized” open software products is MySQL.

However, not all the open software products took that way. Other products maintained the original open software model. This model, called “Bazaar Style” by Eric Raymond (Raymond, 2000), gives to the communities the control of the development of the products. The bazaar style is from an organizational perspective much fuzzier than the “organized open software”. In the bazaar style the products are normally provided “as is” without any responsibility for the supplier. An example of the bazaar style is the Linux’s KDE environment.

4.4 Open software vs. proprietary software

After the basics concepts about software and its development, this project will involve the software as a product. Table 4.1 summarizes the differences, as products, between the proprietary software and the open software in three perspectives: from the technical, supplier, and client side. The Table includes also the specific differences between the large and small companies of the proprietary software and from the organized and bazaar style from the open software.
Table 4.1 Differences between open and proprietary software

Based on this comparison, the differences between the different approaches of the software products can be observed.

4.5 The process of the software substitution

The software substitution is the process of an organization, which decides to totally or partially renew their software assets. This decision must be made after certain period of time during the life of the organization. Each decision taken will influence future decisions. Several
Substitution of Software

Chapter 4: Software development process and differences between open and proprietary software

Factors like price of the license, training, support, features, flexibility, security and openness must be addressed on this process. As it was stated before, most of the software installed around the world is proprietary software. The market share of some software segments are completely dominated by proprietary software. However, in some cases an open software product dominates the segments.

During the process of substitution, the most common track to follow is to select the same product used previously. This fact is because the effect of the switching costs. Following this idea, the organizations will select proprietary software if they had been using proprietary software previously, and they will select open software if they had been using open software previously.

But, is it possible to change the track? Yes, it is possible but not an easy task. Under certain circumstances the users of proprietary software will decide to change to open software. And from the opposite point of view, under different circumstances the users of open software will decide to change to proprietary software. This competence process is illustrated in Figure 4.8.

---

Figure 4.8 Substitution of software

Figure 4.9 Substitution of software in detail
Figure 4.8 represents a specific substitution in a chain. It can be observed the strong track (represented by solid arrows) to maintain the current decision in respect to the openness of software for the next software product. This model also shows a discontinuous track that represents a change in the decision about the openness of the software product.

However the decision between open software and proprietary software is important, it is not the only one. At each type of software there are several options to choose. The decision makers also need to study if they would keep working with the same product, or would choose a different one. This is illustrated in Figure 4.9 as an extension of the previous model.

In this model can also be observed the possible selection of different product with the same level of openness. However this change is not as normal as the decision of remaining with the same product. That is the reason that the track to select a different product with the same level of openness is represented by a discontinuous arrow. However in this point it is also possible to select a different product with a different perspective on its openness. These options are represented in the model by a discontinuous arrow that allows this change of perspective.

The models shown in Figure 4.8 and Figure 4.9 are conceptual models of the substitution of software. However the releases of software products are not synchronized. The different suppliers do release their new products based in their own strategy and development. Based on this, the possible changes can be performed at any moment of the products’ life. This lack of synchronization can be better understood observing Figure 4.10. These three models answer the research question 3 of this project.

4.6 Battles for dominance in the software industry

Based on the understanding of the software industry until this point, an analysis of the battles for dominance in one software segment can be performed. The framework to perform this analysis will be based on incorporating the factors derived from Suarez’s model modified for the software industry (Table 3.3) with the incorporation of the four organizational categories of the software suppliers. Another modification provoked by the four organizational categories is the change between “Firm” to “Supplier”. This is based on the fact that not all the organizations that develop software are firms. The resulting framework is observed in Table 4.2.
Substitution of Software

Chapter 4: Software development process and differences between open and proprietary software

<table>
<thead>
<tr>
<th>Group of factors</th>
<th>Factor</th>
<th>Proprietary Software</th>
<th>Open Software</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Large companies</td>
<td>Small companies</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Company name</td>
<td>Company name</td>
</tr>
<tr>
<td>Supplier-level</td>
<td>Supplier’s technology superiority</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supplier’s complementary assets and credibility</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supplier’s installed base</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Entry timing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pricing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Supplier’s strategic maneuvering</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Licensing and relationships with complementors</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Marketing and PR to manage expectations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Third-party stakeholders</td>
<td>Regulation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software field</td>
<td>Complexity of the product</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Substitution of the product</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Target users’ knowledge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Economic</td>
<td>Network effects and switching costs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2 Framework for battles for dominance in the software industry

The use of this framework will need to take in account the multiple issues related through the first 4 chapters of this report. A use of this framework will be performed in further chapters.

4.7 References

Scientific Papers


Practitioner Papers


Substitution of Software
Chapter 4: Software development process and differences between open and proprietary software

Chapter 5  Study of the product competition in the software market

5.1  Introduction

This chapter analyzes the competition in the software industry between the open and the proprietary software. Based on this study, the actual situation of the open solutions will be revealed. Understanding the competition in the industry, further steps can be made to perform changes in the industry (i.e. redesign of strategies to improve the market share of certain product). The first part of the chapter explains the methodology of the study that is undertaken. In the second part of the chapter, the study is explained in detail. The last part of the study describe the findings of the competition in the software industry.

5.2  Methodology

As it is stated in previous chapters, the software can be divided depending of their use. The segments of the software market are:

- System software
- Development software
- Application software
  - General Applications
  - Server Applications
  - Special Applications

In this project, four software products are selected. The four products compete each other within a specific software market segment. Each product represents one of the four organizational categories. The selection of the segment will be based on the level of competition on the specific segment. The segment with a fiercer competition between the open and proprietary software will be prioritized. Based on the releasing and the end of support dates of the products, some conclusions are made on the product strategy of the companies involved and the difference in front of their competitors. The market share within the segment is also taken into account to try to explain the product strategies.

The sources of data are diverse. The release dates have been obtained mainly from the software manufacturers’ official websites. This information is provided normally on the release announcements of their products. The dates of the ending of support (whenever the support is available) will be also obtained from the software manufacturer official website. This information is normally located on the support section or in the Product Lifecycle section. However for some products, the support period is not publicly available. For some products, there is no support phase.

The market share information of the segments will be obtained from organizations non-related to the suppliers. These organizations keep track of the market share of the software products based on their internet activities. This information will give us some idea of the strategies’ success in the market.
5.3 **Software competition**

The selected segment for the study is the application software on the specific sub-segment of general application software. This sub-segment has revealed to contain a large variety of products based on their low complexity.

5.3.1 **General application competition**

This segment differs from the system software segment in several things. First at all, the products in the segment are smaller and less complex than an operating system. This fact led the prices to be lower and in some cases the products are licensed for free. Most of these products are installed on top of operating system. Added to this, the installation and removal process of these products are usually short and do not require high technical knowledge. Based on these multiple facts, the impact on some factors (like: network effect, switching costs and target users knowledge) should be reduced.

On this segment, several competitions exist: word processors, spreadsheets, instant messengers, FTP clients, etc. All of them offer proprietary and open software competitors. However a specific competition on this sub-segment has attracted the industry attention several times: the web-browsers market.

The web browsers had competed for years to be the main interface of the World Wide Web. The different web browsers are similar in usage, making the user experience differ a little from one to another. This reduces to the minimum the knowledge needed to switch between them. There is a strict standard on how the internet technologies code must be interpreted. This fact standardizes the main capabilities of the products and makes the actual difference between the web browsers to lie on their back-end.

The real differences among web browsers lie on how they fulfill the standards, implement their security, maintain stability, allow customization, allow upgrades and implement improvements in the performance.

The low re-training required after switching between web-browser products and the low license price (normally provided for free) leads to low switching costs on the web-browser market. This particular issue reduces the network effect that other markets suffer. With the reduction of network effect, other factors are revealed. This is the principal reason for which the author has chosen the web-browser market as case study on this research.

However there is a second reason that makes the web browser market interesting. The competition between the web-browsers has involved several situations that are important within the software industry. (i.e. the face to face competition between products, legal confrontations -Opera vs. MSN.com/Microsoft-, changes on pricing strategies of the products -Opera-, new entries -Firefox, Google Chrome-, changes on the licenses of the products -Netscape-, reuse of previous development -Safari, Firefox and Google Chrome- and product bundling -Internet Explorer with Windows, Safari with iTunes and Konqueror with KDE).

The competition to become the most important web-browser seems to be the most active competition within the software industry.
The competition for the web browser market began some years ago, when the first web browser war happened. Internet Explorer developed by Microsoft challenged Netscape, which was then leading the market. After some time, Internet Explorer obtained the dominance and that led to the eradication of Netscape. But before his disappearance, Netscape made their web browser source code available, inspired by the article of Eric Raymond: “The Cathedral and the Bazaar” (Raymond 2000). The code became open software under the name of Gecko. Gecko was the engine behind the last versions of Netscape. It had several interesting properties but one was determinant for the future of the web browsers. As open software, other organizations start using it as the layout engine of their own web browsers. Netscape disappeared after a while, but Firefox, another Gecko-based web browser, was born and started to gain market share. As it was stated by Valloppillil “the binaries may die, but the code would live forever” (Valloppillil 1998).

Today, Microsoft dominates the web browser market. However a second web browser war is happening. In the last three years, Firefox market share has grown and Internet Explorer has been unable to stop his market share loss. With the launching of new versions, good comments from experts and a growing amount of users; Firefox may show the way on how an open software product can succeed against a proprietary product.

In smaller scale, other web browsers have appeared. During the development of first browser war, Telenor, the biggest Norwegian telecommunications company released his own web browser called Opera. One year after, the project was separated from Telenor. A new company was created: Opera Software ASA. As it was stated previously in chapter 4, Opera Software ASA is a small software company. Today, the Opera Web browser has captured a small piece of the desktop market but has reached an important piece of the market on the mobile devices’ web browsers.

In order to complete the set of products belonging to the four organizational categories, a bazaar style web browser is required. However the market share from all these products is really small. The most representative web browser in bazaar style is Konqueror. This web browser is an integral part of the Linux’s K Desktop Environment (KDE). Its name follows the convention of the KDE desktop to begin with “K”. Konqueror appeared during the development of the browser first war. However it was only available for UNIX and Linux platform. Today, it is also possible to be installed on Windows, although its installation requires high skills on software development from the user.

The most remarkable issue about Konqueror is its layout engine: KHTML. Some years ago KHTML suffered a “fork” by Apple Computer that led to the creation of the layout engine called WebKit. Further development over Webkit led to the release of Safari: the included web browser in Mac OS. After a short period of time, Safari was made available for different platforms like: Mac OS, Windows and electronic devices. As result of several factors, Safari has captured the third larger market share in the web browser market. However, based on the fact that Apple Computer qualifies as a large software company, the Safari web browser will not be addressed in this study as an independent product.

After the small introduction of the products involved on the study, the market share of each of them is addressed. On Table 5.1 the market share of the web browsers is shown. The four
organizational categories are included: Internet Explorer (representing the large companies of proprietary software), Firefox (representing the organized open software), Opera (representing the small companies of proprietary software), and Konqueror (representing the bazaar style open software). Safari’s market share information is also shown, only as reference and not as part of the study.

Table 5.1 Market share of the web browser (based on data obtained from (The counter, 2008))
Based on Table 5.1, a graph with the market share of the web browser market has been created. It is shown on Figure 5.1. On the figure, it can be clearly observed the dominant position of Internet Explorer reaching a market share over 90%. After this maximum point in the market share of Internet Explorer, it can be also observed the rise of Firefox. As consequence of Firefox raising a notable reduction of market share of Internet Explorer is observed. On the figure, it is also present a small rising on the market share of Safari, and an even smaller increment on the market share of Opera. Konqueror has not shown a clear rise on its market share (The counter, 2008).

![Market share of the web browsers](image)

Firefox is clearly taking market share from Internet Explorer. Safari and Opera also are gaining market share but in a smaller scale. A closer look to the products can be observed in the next sections

5.3.1.1 *Proprietary software: Large company: Internet Explorer*

![Internet Explorer logo](image)

In 1995, Microsoft launched their web browser called Internet Explorer. In that moment the browser market was dominated by Netscape. The first edition of Internet Explorer was quickly
Substitution of Software
Chapter 5: Study of the product competition in the software market

replaced by a second one without big success. After a while, Internet Explorer was integrated on Windows, and it became an important component of the platform. Not so many years after, Internet Explorer achieved the market dominance and practically eradicated Netscape. After the first web browser war, Microsoft did not release a new version of Internet Explorer for five years. When the rise in the market share of the Mozilla Firefox was imminent, Microsoft finally released a new version: Internet Explorer 7. The information about the release date of the versions of Internet Explorer can be observed on Table 5.2.

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internet Explorer 1</td>
<td>24-08-1995</td>
</tr>
<tr>
<td>Internet Explorer 2</td>
<td>01-11-1995</td>
</tr>
<tr>
<td>Internet Explorer 3</td>
<td>13-08-1996</td>
</tr>
<tr>
<td>Internet Explorer 4</td>
<td>01-09-1997</td>
</tr>
<tr>
<td>Internet Explorer 5</td>
<td>18-03-1999</td>
</tr>
<tr>
<td>Internet Explorer 6</td>
<td>27-08-2001</td>
</tr>
<tr>
<td>Internet Explorer 7</td>
<td>18-10-2006</td>
</tr>
<tr>
<td>Internet Explorer 8</td>
<td>01-07-2008</td>
</tr>
</tbody>
</table>

Table 5.2 Release dates of Internet Explorer (based on data obtained from (Microsoft Corporation, 2008))

Based on the information of Table 5.2 the graphic on Figure 5.3 can be built.

![Figure 5.3 Days between releases of Internet Explorer](image-url)
A clear tendency to “release less often” is shown. The peak is reached between the version 6 and the version 7. This period corresponds to the time between the fall of Netscape and the rise of Firefox. After the version 7, the time between releases was reduced, apparently by the fierce competition with Firefox. It looks that the competition has a direct relation with the increment on the speed of development of new products in this industry. When a product is released more often or the day between releases is shorter it is because it has a fiercer competition strategy with their peers. A software product that dominates the market does not usually invest in new versions unless their competitors threat their dominance.

5.3.1.2 Proprietary software: Small company: Opera

Opera began in 1994 as a research project of Telenor, the largest telecom company in Norway. One year after, Opera Software ASA was created to manage and develop the project. In 1996, Opera 2.0 was released (Opera 1.0 was never released) with shareware and complete versions. A shareware version is similar to the complete version, but it was offered for free but including some limitations. It is intended to be a “try out” for the customer, before he buys the complete version. Shareware is often limited in features and/or time of use. The Opera web browser was offered with complete features but with time limit.

With the release in 2000 of Opera 5.0, Opera changed its pricing strategy. They eliminated the time limit and its free version became ad-sponsored. In the version 7.0 Opera changed their layout engine from the old “Elektra” to the new “Presto” improving their technical characteristics. When Opera 8.5 was released, they changed its pricing strategy again. Opera became completely free and without advertisements. However Opera Software ASA still offered support contracts. In 2006, Opera 9.0 reached the “technical top” of the browser market when it was the first major web browser that passed the Acid2 test. The Acid2 is a test designed to evaluate how the web browsers fulfill the web standards (Opera Software, 2008, A).

Although Opera market share is limited to desktop applications, it has a strong presence on the electronic devices market. Different versions of opera exist for electronic devices like Opera for PDA, Smartphone, mobile (Opera Software, 2008, B). It has been included as the official web browser for Nintendo’s gaming consoles Wii and DS (Opera Software, 2008, C).

Since its release, Opera has been really active towards the web standards. This involvement also led to several controversies against Microsoft-owned websites like MSN.com and Hotmail due to their restrictions to alternative browsers like Opera. Several criticisms have been raised about the strict usage of the standards by Opera. This strictness has led to a lack of compatibility with several web pages that does not comply the standards, reducing the possibilities of Opera users to view that websites (Opera Watch, 2006).
Substitution of Software

Chapter 5: Study of the product competition in the software market

The release story of the Opera web browser can be observed on Table 5.3

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opera 2.0</td>
<td>01-01-1996</td>
</tr>
<tr>
<td>Opera 3.0</td>
<td>31-12-1997</td>
</tr>
<tr>
<td>Opera 3.6</td>
<td>12-05-1999</td>
</tr>
<tr>
<td>Opera 4.0</td>
<td>28-06-2000</td>
</tr>
<tr>
<td>Opera 5.0</td>
<td>06-12-2000</td>
</tr>
<tr>
<td>Opera 6.0</td>
<td>29-11-2001</td>
</tr>
<tr>
<td>Opera 7.0</td>
<td>28-01-2003</td>
</tr>
<tr>
<td>Opera 8.0</td>
<td>29-04-2005</td>
</tr>
<tr>
<td>Opera 8.5</td>
<td>20-09-2005</td>
</tr>
<tr>
<td>Opera 9.0</td>
<td>20-06-2006</td>
</tr>
<tr>
<td>Opera 9.5</td>
<td>12-06-2008</td>
</tr>
</tbody>
</table>

Table 5.3 Release dates of Opera
(based on data obtained from (Opera Software ASA (2008) A.)

Based on Table 5.3 a graph that shows the time between releases of Opera web browser is shown in Figure 5.5

![Figure 5.5 Days between releases of Internet Explorer](image)

The graph 5.5 shows the time between each of the versions. Apparently the strategy of Opera had change several times during the life of the product. An example of this is clearly the changes on its business model, which had changed its pricing strategy (from Shareware to Ad-supported and from Ad-supported to Freeware) several times.
In 1998 Netscape made the source code of their web browser publicly available. Based on this code, Firefox was released some years after. After a slow start, the web browser started rising on his adoption rate around the world, becoming the second larger adopted web browser in the market. This continuous gain on its market share has been deployed even against the strong network effect of Internet Explorer. This network effect consisted in the fact that the users of Windows had already installed Internet Explorer in their desktop computers. In case that they wanted to change to Firefox, they did have to perform the installation themselves. In Table 5.4 the dates of release and dates of end of support of all the versions of Firefox can be observed.

<table>
<thead>
<tr>
<th>Web browser</th>
<th>Version</th>
<th>Release date</th>
<th>Status</th>
<th>End of support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mozilla Firefox</td>
<td>1.0 &quot;Phoenix&quot;</td>
<td>09-11-2004</td>
<td>Unsupported</td>
<td>13-05-2006</td>
</tr>
<tr>
<td>Mozilla Firefox</td>
<td>1.5 &quot;Deer Park&quot;</td>
<td>29-11-2005</td>
<td>Unsupported</td>
<td>29-05-2007</td>
</tr>
<tr>
<td>Mozilla Firefox 2</td>
<td>2.0 &quot;Bon Echo&quot;</td>
<td>24-10-2006</td>
<td>Supported</td>
<td>17-12-2008</td>
</tr>
<tr>
<td>Mozilla Firefox 3</td>
<td>3.0 &quot;Gran Paradiso&quot;</td>
<td>17-06-2008</td>
<td>Not released</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

Table 5.4 Release and end of support dates of Firefox
(based on data obtained from (Mozilla Foundation, 2008 A) and (Mozilla Foundation, 2008 B))

The number of versions of Firefox is limited. However some figures can be built from the data. For the day between the versions releases, it can be observed a rising trend. This graph can be observed on Figure 5.7.
In terms of its support, Firefox shows a strong guideline. It kept the support of their previous version until 6 months after the release of their new product. The support time is observed on Figure 5.8.

An interesting fact appeared when the “day between releases” is compared between the Firefox Version 3 and Internet Explorer Version 8. The days between releases of both versions are almost equal. This is due to the fact that the dates of release of the actual versions (Firefox 2 and Internet Explorer 6) differ for only some days, and that the expected dates of the new versions follow the same pattern. In synthesis, both products had released their last product on the same date, and they plan to do the same for the following version. This clearly indicates that the competition between both products is direct.

The distribution of both applications differs. Microsoft offers their product through Windows Update taking advantage of the installed base of Windows and by direct download, while Mozilla only offers direct download. In response to this, Firefox launched an important marketing campaign named “Download Day 2008” through the website www.spreadfirefox.com. The users that wanted to use Firefox could register on the website to receive an email with the notification of the release. This campaign aimed to bring the download link to the users reducing the knowledge needed to the lowest level possible. They achieved 8 million downloads within 24 hours. However it did not affect greatly the market share.

5.3.1.4 Open software: Bazaar style: Konqueror

![Konqueror logo](image)
K Desktop Environment (KDE) is a graphical user interface for Linux platform. It contains multiple utilities being one of the most important GUI’s in Linux world. In 2000, the K Desktop Environment launched its second version. The KDE 2.0 included the first public release of Konqueror web browser: Konqueror 2.0. The web browser is strongly attached to the KDE and for years was only available for Linux and UNIX environments. Recent developments had opened the possibility to use Konqueror on Windows. However the installation of the application required advanced knowledge from the user, restricting those activities to advanced users.

Konqueror is based in the KHTML layout engine which several years ago was “forked” to create WebKit, which was the layout engine of Safari web browser. It is expected that the performance of Konqueror and Safari would be similar, based on its common background. In general terms this is true, but small differences between the browsers are present.

Although Konqueror has a small market share, it is considered to have a good success. The open software groups intent to create a beautiful piece of software that offers freedom to the users. Konqueror offers complete freedom and Safari offers similar performance based in similar components to all the users. Further than this, to comply with the license of KHTML, the WebKit was also offered as open software by Apple. From this perspective, Safari is not exactly a proprietary software product. Safari is a mix between open software and proprietary software, which has captured the 3.41% of the market.

The release dates of Konqueror can be observed on Table 5.5

<table>
<thead>
<tr>
<th>Version</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Konqueror 2.0</td>
<td>23/10/2000</td>
</tr>
<tr>
<td>Konqueror 3.0</td>
<td>03/04/2002</td>
</tr>
<tr>
<td>Konqueror 3.5</td>
<td>29/11/2005</td>
</tr>
<tr>
<td>Konqueror 4.0</td>
<td>11/01/2008</td>
</tr>
</tbody>
</table>

Table 5.5 Release dates of Konqueror (based on data obtained from (K Desktop Enviroment, 2008))

Based on these dates, a graph can be drawn. This graph can be observed on Figure 5.10
Substitution of Software
Chapter 5: Study of the product competition in the software market

A clear rise on the development time can be observed in the releases of Konqueror. Although the releases of Konqueror are linked to the releases of KDE, it looks that the community of KDE are not trying to compete with the other web browsers for the market. They are more interested in creating a good desktop environment for Linux and as a collateral project they migrated some of their applications to other platforms with reduced impact on the market. However the “beauty” of their code is real and appreciated by the users. A clear result of this fact is the raising market share of Safari.

5.3.1.5 Days between releases compared

Following the logic of the direct relation between the “release often” and the fierce competition of the products it is important to compare the competitiveness of the four products involved in the study. This comparison can be observed in Figure 5.11.

The competitiveness (or time between releases) of the four products is deployed in relation to the time. It is important to acknowledge that a higher value on the x-axis represents less competitiveness. The graph also includes the average of the segment. This graph results on certain conclusions. At the end of 2001 the competitiveness of the whole segment was reduced. This reduction coincides with the end of the first web browser war. From the end of 2004 till the end of 2006, the competitiveness of the segment was increased again in a constant rate. This increment coincides with the first release of Firefox. Based on this empirical evidence we can state the relation between the competitiveness strategy of the companies and their rivals.

5.3.1.6 The dominance on the web browser market

Based on the previous research a deep analysis on the competition in the segment can be made. Using the framework proposed on Table 4.2, the results of the battle for dominance are shown on Table 5.5. To order the results of each firm, the result of each factor for each product is rank in front of their competitors.
# Chapter 5: Study of the Product Competition in the Software Market

## Factor group

<table>
<thead>
<tr>
<th>Supplier-level</th>
<th>Factor</th>
<th>Proprietary Software</th>
<th>Open Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplier’s technology superiority</td>
<td>Far to pass the Acid2 test 3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>Acid2 test passed. Technically superior 1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>Close to pass the Acid2 test. 2&lt;sup&gt;nd&lt;/sup&gt;</td>
</tr>
<tr>
<td>Supplier’s complementary assets and credibility</td>
<td>Huge amount of assets of the firm. The firm has high credibility. However it has been damaged by bad publicity. 1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>The firm assets are limited. The credibility of the firm is high 2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>The firm assets are limited. The credibility of the firm is high. 2&lt;sup&gt;nd&lt;/sup&gt;</td>
</tr>
<tr>
<td>Supplier’s installed base</td>
<td>Large installed base 1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>Tiny installed base 3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>Small installed base 2&lt;sup&gt;nd&lt;/sup&gt;</td>
</tr>
<tr>
<td>Entry timing</td>
<td>1995 1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>1996 2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>2004 3&lt;sup&gt;rd&lt;/sup&gt;</td>
</tr>
<tr>
<td>Pricing</td>
<td>Free (bundled with Windows) 4&lt;sup&gt;th&lt;/sup&gt;</td>
<td>Free 3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>Free(but also offered bundled) 1&lt;sup&gt;st&lt;/sup&gt;</td>
</tr>
<tr>
<td>Supplier’s strategic maneuvering with complementors</td>
<td>The license is given for free but with limitations. There are some add-ins for IE. 3&lt;sup&gt;rd&lt;/sup&gt;</td>
<td>License offered to include Opera with software (like Adobe products) or devices (like Nintendo consoles) 2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>Open software. The openness gives it the opportunity to exist to a big amount of add-ins 1&lt;sup&gt;st&lt;/sup&gt;</td>
</tr>
<tr>
<td>Marketing and PR to manage expectations</td>
<td>Promotion of the product in regular basis. 1&lt;sup&gt;st&lt;/sup&gt;</td>
<td>Promotion of the product in regular basis. But in small scale. 2&lt;sup&gt;nd&lt;/sup&gt;</td>
<td>Starting to use the user community to promote the product like the “Download day”. 2&lt;sup&gt;nd&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

## Third-party stakeholders

| Regulation | No special regulation |
| Web standards are proposed but they are not mandatory |

## Software field

<table>
<thead>
<tr>
<th>Complexity of the product</th>
<th>Not complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Substitution of the product</td>
<td>Easy to substitute 1&lt;sup&gt;st&lt;/sup&gt;</td>
</tr>
<tr>
<td>Target users' knowledge</td>
<td>Any kind of users</td>
</tr>
</tbody>
</table>
Substitution of Software
Chapter 5: Study of the product competition in the software market

<table>
<thead>
<tr>
<th>Economic</th>
<th>Installed by default with Microsoft Windows. Updates provided by direct download and by Windows Update (automatic update system). Switching cost minimum 1st</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Binaries available for download. Only installed by default in some electronic devices. It presents incompatibilities with non standard websites. Switching cost minimum 3rd</td>
</tr>
<tr>
<td></td>
<td>Installed for default in KDE. Available for install in other platforms. Low switching costs. 4th</td>
</tr>
</tbody>
</table>

Table 5.6 The model of dominance applied on the web browser market

Based on the use of the framework on Table 5.5, it is clear why Konqueror did not reach a huge market share. Being the only web browser, which presents difficulties for installing, it achieves only a high ranking on technical superiority.

Opera has a different situation. Their rankings are similar to the ones achieved by Firefox. However Firefox has achieved a market share around 20 times bigger than Opera in a shorter life. Opera is a little bit technically better than Firefox. However the strong strategy of Firefox is superior. The close relation with Google generates a network effect on Internet, leaving behind the old-school operating system’s network effect.

The network effect can explain the rise on the market share of Firefox and the actual dominance of Internet Explorer. Due to the fact that the working environments are switching from the desktop computers to the internet, the network effect has a similar impact. In a short period Firefox will dominate the web browser market.

5.4 Conclusions

The present study gives us some interesting ideas about the competition of software industry regarding the open software. The study findings broadly explain the competition in the general application sub-segment. Based on the findings, the Figure 5.12 can be built. On this Figure, the reasons of the market performance of the two major web browsers can be observed and explained.

The web browser market was captured by Internet Explorer after the first browser war. In that precise moment, although Netscape was the first one on the market, the challenger Internet Explorer achieved the dominance of the market share. Sometime after, Firefox appeared and took advantage of the easy to change property of the general application market. It did not change the user experience dramatically, but included better features. This has started to make impact on the market share division even the presence of the large installed base of Internet Explorer and the fact to be bundled to Windows.
This study gives us a snapshot the software markets. Based on this snapshot we can synthesize ten points about the software competition. Based in this points it is possible to answer the research question 4 of the project.

1. Be first to the market tends to give dominance
2. Dominance leads to network effect
3. Network effect strengths dominance
4. “Reduce-price, release-often” allows gaining market share
5. Dominance is not forever
6. Network effect is reduced by the knowledge of the user
7. Network effect is reduced with low switching costs
8. The high skilled users like the open software
9. Better features are more important than the likeness
10. Unique or better characteristics give market share

5.5 References


Chapter 5: Study of the product competition in the software market


Chapter 6  Proposed strategies

6.1  Introduction

This chapter focuses on the delivery of the final outcome of the research. The strategies that will be proposed in the next sections are based on the findings of previous chapters. These findings are mainly contained in the ten points addressed in the conclusions of Chapter 5.

In the first part, general strategies for software suppliers are addressed. During the second part, strategies for open software suppliers are proposed. The third addresses strategies for the proprietary software suppliers.

6.2  General Strategies

The general strategies refer to strategies that any software supplier must apply to become dominant. Those strategies are:

- Create a network effect
- Keep prices low, but release often
- Never stop the development
- Listen to your potential customers
- Facilitate the continual usage of your products

6.2.1  Create a network effect

On the point 3 (Network effect strengths dominance) this analysis marks the importance of the Network effect. Lin (Lin, 2007) stated that a network effect is invaluable on the software industry. This fact leads to believe the importance of creating a network effect over any software product. An interesting situation related to the network effect is condensed on point 2: “Dominance leads to network effect”. Based on the combination of point 2 and 3 it is necessary to understand that dominance is closely related to network effect. This strategy is related to make efforts on incrementing the network effect of any software products. Sub-strategies related to the network effect can be proposed like: facilitating the creation of complementary products and the training of the user (or even provide training for free).

However, Lin failed to see that in the IT industry the network effect’s origin may be temporal, as is stated in the point 5: “Dominance is not forever”. Under certain circumstances the technological advance will reduce the importance of a network effect’s origin. This is basically the situation that is currently leading to the reduction of the market share of Internet Explorer and the increment of Firefox. The network effect’s origin created by Internet Explorer bundled with Windows has been reduced with the appearance of internet applications. In the other hand, Firefox based on its relationship with Google has created a new network effect’s origin.

Creating a network effect is highly important for the software industry. However suppliers must maintain the network effect even through technological changes.
Substitution of Software
Chapter 6: Proposed strategies

6.2.2 *Keep prices low, but release often*

The point 4 clearly states: “Reduce-price, release-often allows gaining market share”. The versioning of software products is a common practice in the software industry. The suppliers sell snapshots of the same development in different stages to the same client. This allows them to sell each version at lower price, because part of previous development was already paid by the sales of previous versions. The companies can decide to sell new versions at higher price, giving long time support to the version or sell new versions at low price with a short time support. The second strategy has proved being more successful. Microsoft has reduced the price of its versions of Windows, but they also have been releasing them more often and with shorter support. As result of this strategy, they have kept dominance of the market with a small increment on the adoption rate. In order to keep prices low, they give the opportunity to more clients to buy their products. Releasing often gives the opportunity to obtain similar profits.

6.2.3 *Never stop the development*

After the first web-browser war, Microsoft assured a dominant position in the market. There was no other big competitor to challenge its dominance. Microsoft disbanded Internet Explorer’s development team and used the talent of their developers to fight other battles. For 5 years, Internet Explorer did not release a new version. Some of Microsoft competitors keep their development even if they only had a small market share. After a certain period, most of them surpass the technical capabilities of Internet Explorer. In 2006, Microsoft released Internet Explorer 7, but its technology was already behind. Today, it is not clear whether Microsoft will be able to catch the technological capabilities of their competitors in a short period of time. As the point 5 states: “Dominance is not forever”.

Even if there are not competitors in the market, the software suppliers cannot stop the development of their products, otherwise a new supplier can appear and take the market.

6.2.4 *Listen your potential customers*

This strategy is related to the point 8: “The high skilled users prefer open software”. One of the most interesting factors on the open software is the accurate way that it solves the advanced users’ problems. This is based on the fact that advanced users provide the ideas, feedback or development of the open software. As Eric Raymond stated “*every good work of software starts by scratching a developers personal itch*” (Raymond, 2000). However, the open software recurrently fails to solve the problems of regular users. This is provoked because the regular users do not know how to scratch their personal itches. If a software supplier is able to hear all their potential customers, they get to know about their itches and will certainly solve them. In that case, the regular customers would like to use open software. Developers must also implement their new ideas even if customers do not ask for them.

The software must be based on the technology push, but never forget the market pull. The companies must also hear what their customers had to say.
6.2.5 Facilitate the continual usage of your products

From the perspective of the customer, it is easier to stay with the same technology forever. As the point 3 states: “Network effect strengths dominance”. Staying with the same software product avoids any kind of switching costs and migration headaches. The customers can focus on their own business and avoid thinking on software substitution and data migration. The software suppliers must be aware of that, ensuring that their clients do not even think about change from their software products. Keeping their clients working on their own business, they will not be involved in software substitution. Providing constant updates, a soft upgrading, low prices for recurrent clients and a solving the entire software problems of their clients, will avoid that the users will ever have any reason to change.

6.3 Open software suppliers strategies

Originally the open software suppliers did not have the intention to be profitable. They just wanted to create beautiful pieces of software. Things have changed since then, and some of them are interested on competing face-to-face with proprietary software suppliers. However, the proprietary software suppliers are not only integrated by technicians. Marketers, strategists and managers have been integrated into the proprietary software suppliers to make their products as profitable as the products of any other industry. The open software products must do the same to compete. They must integrate non-technology-related strategies to be able to face the battle. The strategies proposed are:

- Create exclusive products
- Create exclusive features
- Help schools to adopt open software
- Get vendors to sell OS installed
- Facilitate products’ migration

6.3.1 Create exclusive products

Innovation tends to help to get a better position in the market. Creating innovating products that the competitors cannot emulate will help the open software suppliers to achieve the lead. Until now, most of the open software is created as an alternative to the dominant proprietary software product. The open software must create new kinds of products that do not exist on the proprietary software. Eventually, the proprietary software suppliers will enter in the product market and compete against a well-established open software product. Taking the risk to obtain the lead is the only way that they will be able to have the opportunity to do it. Being the only choice for a new kind of products, will give an “early entry” advantage to the open software suppliers. This is aligned to the point 1 “Be first to the market tends to provide dominance”. By the time when the proprietary software can entry to the market, the network effect created by the open software will be really high.
6.3.2 Create exclusive features

This strategy has similar line of reasoning with the previous one. However, it is aimed in a different perspective. The open software products must create exclusive features for their products. As it is established by point 10 “Unique or better characteristics give market share”. This will give to the customers a reason to change to open software. A successful example of this is the tabbed browsing. It was introduced by iBrowser and incorporated by Opera and Mozilla shortly after. By the time that Internet Explorer included this feature, several users had already changed to its competitors attracted by the useful tabs. It did not matter whether Internet Explorer was used by a huge amount of people or its pricing strategy. Better features gave to Firefox an opportunity to start its rise. It was stated on point 9: “Better features are more important than the likeness”. The open software suppliers must create products that differentiate themselves from proprietary products by their unique features. That will be the only way to make that part of the users to change to an open software solution.

6.3.3 Help public organizations to adopt open software

It is difficult to make a regular user once he is already trained to switch to different software. The advanced users do not have the problem to be retrained, as it was stated in point 6: “Network effect is reduced by the knowledge of the user”. However, regular user does not want to spend time learning how to use a new product, even if it is better than the previous one. The new software will make his tasks more complex at least for a short period. If the regular users do not like to be retrained, maybe it is possible to help that they are trained in open software from the beginning. Involving the open software suppliers on the education will open opportunities. The schools will adopt open software creating “nests” of open software users. Other organizations (like government institutions) prefer already to use open software products due to the flexibility that openness provides to them. It is important for the open software suppliers to help these public organizations to adopt open software products. After a while the network effect created by the public organizations will affect the rest of the users.

This is not a new strategy. Some countries are already choosing open software as their preferred public organization platform. In some years, the proprietary software will be the one that will have fight against the network effect caused by open software users.

6.3.4 Get vendors to sell OS installed

The installation of system software is not easy. It requires advanced knowledge to put the equipments to work. The blockage that the lack of knowledge creates is one of the biggest barriers that regular users had in order to adopt open software. However, if some open software suppliers increment their commercial relationships with the hardware suppliers, the new systems can be sold to the normal users with open software pre-installed. The normal users will only need to learn how to use the new software but not how to install it. The switching cost to open software will be reduced, and as it was stated by point 7: “Network effect is reduced with low switching costs”.

This strategy is also not new. Some hardware suppliers like Dell are already shipping equipments with open software pre-installed.
Substitution of Software
Chapter 6: Proposed strategies

6.3.5 Facilitate products’ migration

As complement to the previous strategy, something must be done to update the existing equipments. As it was stated by point 7: “Network effect is reduced with low switching costs”. Ubuntu is one of the most high-rising market shares in the open software. Its basic strategy is reflected in its slogan: “Linux for human beings”. With the offer of a Linux distribution that can be used and installed by any kind of users, thousands of regular users started to migrate to Ubuntu. Ubuntu offers an easy migration. Users can test the software before they modify their system and Ubuntu runs properly in most of the existing software. It runs in several languages and different hardware platforms. The users can be back on their work in a shorter period of time with a migration process that is not difficult. In consequence the open software keeps on raising its level of adoption.

6.4 Proprietary software suppliers strategies

The proprietary software suppliers are also involved on this battle. Some strategies can be proposed to stop the rising of the open software products.

- Keep relationships with schools
- Keep relationships with hardware suppliers

6.4.1 Keep relationships with schools

For years, schools have trained their students on proprietary software platforms. This gives a big advantage to the proprietary software products. They must enforce those relationships, providing their products and technical support at low price to education institutions. This strategy has the goal to keep the relationship healthy and also to keep the institutions teaching on proprietary software. If the proprietary software suppliers do not do that, the suppliers of open software will find the way to convince the schools to make the change.

6.4.2 Keep relationship with hardware suppliers

The proprietary software suppliers have had a long-term relationship with hardware suppliers. This relationship is indispensable for the proprietary software suppliers. They must keep the preference of the hardware suppliers to the open software. If they are successful on restricting the capacity of the open software to be pre-installed, the proprietary software will keep the market for several years.

6.5 References

Chapter 7  Conclusion

7.1  Introduction

The last chapter of this report states the conclusions as a result of this project. The first part of the chapter is focused on how the research questions led the author to solve the problem stated in Chapter 1. The second part of this chapter addresses the learning points which resulted from this research. The third part analyses the further research that could be performed. A fourth discusses some of the results of this research based on an interesting scientific article.

7.2  Research questions and the problem

The problem stated in the first chapter addresses the technical success of the open software and also its low adoption rate. This issue led to a study focused on understanding four aspects of the software industry. The aspects of the software industry that will be analyzed are:

- Characteristics of the software industry
- Market dominance in the software industry
- Substitution of products in the software industry
- Current situation of the software market

These aspects led to interesting results, which are addressed in the following sections:

7.2.1  Characteristics of the software industry

The software industry’s characteristics that were encountered are:

- Production and marginal cost
- Classification of users
- Segments in the software industry
- Complexity of the products

Production and marginal cost

The first characteristic analyzed for the software industry refers to the manufacturing costs of the products. The cost of producing the first copy of a new product is extremely high. However the marginal cost to produce subsequent copies is insignificant. Most of the times, it only involves the cost of replication and distribution. This characteristic obviously makes the difference between producing a copy and produce a million of them extremely small.

User classification

The users within the software industry could be classified in two different ways: by usage and by knowledge. The classification “by usage” divides the users between personal users (or individuals) and enterprise users (or organizations). The classification “by knowledge” measures the level of knowledge of the users. It segregates the users in regular and advanced users. Both classifications overlap and do not exclude each other.
Substitution of Software
Chapter 7: Conclusion

Segments in the software market
The software products are divided in three segments: system software, applications and development software. The system software refers to the platform of software that allows the user to interact with the hardware. The application software is the software products that perform specific tasks and are installed over the system software. This application segment is divided into three sub-segments: general applications, server applications and special applications. The development software includes the software products that software developers use to create software products.

Complexity of the products
The software development process does not represent the same level of complexity for all the software products. Some of the products are simple and practically any supplier can compete within the market, but for more complex products only large suppliers can enter the competition. The complexity of the software products is negatively correlated to the competition that exists on the market.

These characteristics are presented in chapter 2 of this report, and are used in further chapters as basis of a deeper analysis.

7.2.2 Market dominance in the software industry

Analyzing the way of how the dominance is achieved within the software market, some results are provided for this aspect. Based on the story of the industry, the software battles for the dominance had led to the winner’s complete domination of the market. This fact emphasizes the idea that the dominance within the software market is required to survive.

Suarez’s model is an interesting model for dominance, but it is not completely suitable for the software industry. A modified model based on Suarez’s model has been specifically tailored for the software industry. That model can be found on Figure 3.4.

Other topics are raised in the research about the dominance in the software industry. The network effect and its enormous importance in the software competition, the users knowledge as a modifier factor in the competition, the pricing strategy of the software suppliers and its effects in the adoption of products, the importance of having a big installed base and its relation with the network effect, and the government regulation that enforces the usage of open software and blocks the utilization of proprietary software.

These findings can be found in chapter 3 of this report. Based on these findings a new framework (based in Suarez’s dominance model) was created to analyze the dominance within the software industry

7.2.3 Substitution of products in the software industry

Before understanding the substitution of products in the software industry, it is necessary to understand the basic structure of the software development process. This flexible process is explained as preparation for the explanation of the new development model: The open
Substitution of Software
Chapter 7: Conclusion

software. Once the software development process is explained, both on its open and proprietary versions, a comparison must be performed to raise the differences.

The organizational aspect in the software development is also important. By addressing the organizational structure of the proprietary software suppliers, the author concludes that there are two types of proprietary software suppliers: Large companies and small companies.

In terms of the open software suppliers, two sorts of them were found: organized and bazaar-styled. The organizational structure of both types of software led to a classification of the software suppliers in four organizational categories: proprietary software large companies, proprietary software small companies, open software organized and open software bazaar style. These four organizational categories are incorporated to the resulting framework from Chapter 3. Based on this overall perspective the substitution of software is addressed and explained on the models in Figures 4.8, 4.9 and 4.10.

Based on these findings from Chapter 4, the framework created in Chapter 3 has been modified to analyze the battles for dominance within the software industry. This framework can be found on Table 4.2.

7.2.4 Current situation of the market software

In order to analyze the current situation of the market, a battle for dominance must be analyzed using the framework created in Chapter 3 and 4.

A case of study was selected to use the framework. This case of study will represent the entire software industry. The selected case of study was the web browser market. Based on its tough competition, its low network effect, and the presence of multiple factors (like pricing strategy, product bundling, code heritage and legal confrontations). The analysis of the web browser market revealed interesting usage of strategies from the diverse suppliers.

The results of this study case originated ten points related to the competition within the industry. These points are general for the whole industry on the battle for dominance:

1. Be first to the market tend to give dominance
2. Dominance leads to network effect
3. Network effect strengths dominance
4. “Reduce-price, release-often” allows gaining market share
5. Dominance is not forever
6. Network effect is reduced by the knowledge of the user
7. Network effect is reduced with low switching costs
8. The high skilled users like the open software
9. Better features are more important than the likeness
10. Unique or better characteristics give market share

This case of study can be found in the chapter 5 of this report.
7.2.5 Strategies for the software industry

Based on the ten points compiled in Chapter 5 and the four previous aspects, some strategies are proposed for the entire software industry. The proposed strategies are:

- Create a network effect
- Keep prices low, but release often
- Never stop the development
- Listen to your potential customers
- Facilitate the continual usage of your products

Some strategies are also proposed for the open software suppliers. These strategies are:

- Create exclusive products
- Create exclusive features
- Help public organizations to adopt open software
- Get vendors to sell OS installed
- Facilitate products’ migration

As a result of the application of these strategies the open software suppliers should be able to achieve an increment on their market share that will lead to market dominance in the future. The implementation of these strategies will lead to the solution of the problem proposed in the first chapter: Ensuring the survival of the open software suppliers

However two more strategies are proposed that will help to the proprietary software to retain their dominance in the software market. These strategies are:

- Help public organizations to stay with proprietary software
- Keep relationships with hardware suppliers

These strategies can be found in Chapter 6. By implementing these strategies, the diverse suppliers can achieve dominant positions within the software market, and then survive on this competitive industry.

7.3 Learning points

7.3.1 Multi-source research

One of the issues that the author did not expect during the development of this research was the large sources in which the information was obtained. These sources were: scientific papers (i.e. articles by Lin and Suarez), articles written by practitioners (Raymond), internal documents from software companies (Valloppilli), etc. The combination of these sources made possible the results provided on this report.

The author considers now really important to involve all kind of information in a research. However he is also aware of the need of categorizing the sources of this information. The scientific articles will always have a solid foundation. This makes the process to produce them
slower. Practitioners’ articles and specialized journalists are produced faster and normally reflect the current trend of the industry, however they sometimes lack foundations. The importance of the work is on weighting the value of the different sources.

### 7.3.2 From theory to practice

The author always had a practical approach to solve the problems that he found in the past. It was not until the development of this project when he could relate the theory with the practice on a consistent way. During this process it was possible to create a theoretical model that fulfilled his perception of reality. Based on the creation of this model, the author had the opportunity to deeply understand reality. On a different perspective, the author could also perceive the relationship between methodologies and practice. Being familiar with the open software development process, the author did not relate this process with open innovation. During the development of this project this relationship became evident. Based on this, the author understood how much the theory can help to improve the practice. As a result of his experience, he found a new way to solve practical problems on a more structured way.

### 7.4 Further research

During the development of the project some interesting topics were addressed. These topics can be considered for further research.

#### 7.4.1 The importance of network effect in the software industry

The network effect resulted as a major factor in the software industry. Further research could be made to measure the influence of the network effect in the software industry.

#### 7.4.2 The importance of users’ knowledge in the software industry

The users’ knowledge resulted also as an important factor in the software industry. This factor is exclusive to the software industry. Further research can be done on this factor and how a supplier can reduce it.

#### 7.4.3 Hybrid (open /proprietary) software

On the web browser market this research found open software (bazaar style and organized) and proprietary software (small and large companies). However a different approach also was found and not explored. The Safari web browser is a proprietary software product developed by a large company. However some parts of Safari are based on open software. Based on that origin, some parts of the Safari source code are made public as any other open software.

Apparently Apple is exploring the possibilities to mix the open software with proprietary software. Apple is certainly aiming to get the best of both models. Observing the success of Safari, the hybrid software development model is having success. However, further research must be made in this topic to be able to draw conclusions.
7.4.4 The success in the bazaar style software

Some communities go in a different direction than the rest of the software industry. Those communities implement the development of open software using a bazaar style creating “beautiful” pieces of software. However some of them do not have any economic interest in the development. They are only interested on creation process.

During the development of this project, the author concluded that some of the communities have a different point of view than the rest of the industry. From their point of view, they achieve success when the software resulting of their creation process works as it was expected. Some of the communities allow outside people to use their developments, but they do not really promote their use. Apparently the communities never aimed for a commercial success for their projects. This obviously leads to a low adoption rate of their products.

A further research could aim to analyze the motivation of the communities to work in a project that does not expect to have any commercial success. Other interesting topic for further research could be based in the hypothesis of change from a non-profit project to a profitable project.

7.5 Discussion

7.5.1 Updating competition models

In Chapter 3, two competency models are explained. They are: “Porter’s 5 forces analysis” and Suarez’s “Factors of dominance”. These two models were referred several times on secondary literature about the topic. That has triggered the use of them on this report. However during the models’ implementation some parts of them were not appropriated at least for this industry.

In 1979, when Porter published his model, Richard Stallman had not created the free software concept yet. The concept of communities developing software was developed years after. Porter did not have the opportunity to analyze the open software development process and he could not realize that his firm approach was not covering all kinds of possible suppliers. Today, suppliers like KDE group and Apache Foundation compete in the industry but they cannot be considerate firms. KDE group may not have a large market share with their web browser (however achieving a large market share is not on their priorities). KDE group is much more interested on the technology itself than on the adoption rates. This interest has leaded them to create a product that proves to be technologically more advanced than those of their rivals. KDE group is successful to reach their objectives and is definitively an important actor within the industry. But KDE group is not a firm, and therefore it does not fit in Porter model.

From a different perspective, the openness of Konqueror’s source code led to the creation of the fast growing Safari and the recently-born Google’s Chrome. This derivates into an important relationship that is now common within the software industry but that it was not presented on Porter’s model. The important relationships between Navigator’s supplier
Chapter 7: Conclusion

Suarez’s model incorporates a relationship between the “firm” and their surroundings. Some of the factors that Suarez incorporates on his model seem to be really accurate within the software industry. Unfortunately the model still lacks a two-way relationship between the surroundings and the firm. This is opposed to the author’s perception of the software industry, which is better described as a network than as a hierarchical structure. Suarez also refers to the suppliers as firms. However his approach seems to be more adaptable. An adaptation of the model to the industry was apparently enough to explain the competition in the current software industry. In a fast-changing environment (like software industry) Suarez’s model could be completely outdated in a short period of time.

7.5.2 Comparing results

The present research revealed interesting issues. One of the most interesting is the large amount of sources on which information about the topic can be found. Informal interviews with practitioners, daily news, and suppliers’ websites are sources on which information was collected. On the scientific side, the literature found was limited. This fact is quite understandable because the topic is rather new, and the industry’s fast pace. Some of the articles found about certain aspects seem to be outdated. This happens even if they were published even few years ago. Articles sometimes analyze software and situations that currently exist no longer in the industry.

Most of the articles were rather exploratory than analytic. One example of this is “Designing a forecasting analysis to understand the diffusion of open source software in the year 2010” by Dolores Gallego and Salvador Bueno from the University Pablo de Olavide of Seville, and Paula Luna from the University of Seville (Gallego, Luna & Bueno, 2007). They analyzed open software and its future using a Delphi method. Gallego, Luna & Bueno forecasted how the situation will be on the industry in the coming years. On their research, they considered open software as a new paradigm on software distribution. The author considers the open software as a paradigm on the whole software process. This disagreement is probably due to the fact that Gallego, Luna and Bueno’s interest was focused on the software as a product.

Based on these studies of Gallego, Luna and Bueno, the open software will suffer more diffusion and adoption in poorer countries. In 2010 the open software diffusion will be close to 70% in South America and Asia. Africa will be around 34.5%, North America 49.83%, Europe 59.5% and Oceania 61%. Based on the market share that is shown for operating systems and web browsers in previous chapters, the author considers that reaching that degree of diffusion and adoption will take more than two years. Gallego, Luna and Bueno stated that the
market of operating systems and web browsers will be 60% for open software (Gallego, Luna & Bueno, 2007). The market share data shows a 17% for open software in web browsers and less than 1% in operating systems. It will be hard to reach the 60% in less than two years. However the author considers that at least for the web browser market, the goal of 60% could be achieved in a larger period of time.

Gallego, Luna and Bueno found remarkable importance in aspects like: insufficient skills or experiences in open software, personal resistance to change, lack of compatibility with other software as important (Gallego, Luna & Bueno, 2007). These aspects are closely related to network effect, which was stated as one of the most important issues in the software industry through the development of this project.

Gallego, Luna and Bueno at last addressed the importance of involving education on the open software (Gallego, Luna & Bueno, 2007). The author agrees on this position. The network effect has an important origin at the schools, which could be reduced with the implementation of open software in this and other public institutions.

The present research delivers certain ideas about the software industry that other projects like Lin (from an economic point of view) or Gallego, Luna and Bueno (from the business perspective) also stated. Based on the recent findings about the topic, it will be possible to start making a more analytical research on the software industry in a near future.

7.6 References