Software Patents in Europe
Exploitation Strategies and Infringement Monitoring

Jussi Päivärinne

Department of Commercial Law
Hanken School of Economics
Helsinki
2013
Software patents are now relatively widely patentable under the European Patent Convention. However, managing software patents is much more than just acquiring a patent through patent offices. The role of software patents as strategic business tools has significantly increased, and software patents cannot be regarded as mere legal instruments anymore. This has also placed pressure on companies to exploit software patents more effectively in their patent strategies. Even though companies might recognize the strategic importance of software patents in their strategies, they might not be able to exploit their patents in an intended manner, because a major problem with software patents is infringement detection.

The analysis of prior academic literature has revealed that there is a need to investigate and to get a deeper understanding of patent exploitation strategies and patent infringement monitoring separately but also of the relationship between these two in the context of software patents under the current European legal framework. This study can be described as interdisciplinary, founded mainly on prior legal and strategic managerial writings. Because this study investigates different legal, strategic and also technological aspects of software patents within the European software industry, in addition to a traditional legal dogmatic approach this thesis also employs approaches which are based on legal sociology and legal informatics in order to carry out the research objective properly.

The contributions of this thesis to software patent research are two: first, it increases the knowledge of software patent research by identifying and clarifying some problem areas in the context of software patents, such as the European legal framework regarding reverse-engineering and explains why the nature of software causes problems in infringement detection, and second it identifies and integrates factors from two different disciplines, i.e. strategic management and intellectual property, into the same analysis. Therefore, the contents of this thesis contribute to a better understanding of European software patents by bringing the aspects of exploitation strategies and infringement monitoring together.

This thesis concludes that if patent infringement monitoring is ignored or not properly understood, some strategic aspects of the use of software patents are limited or worthless. In addition, it deduces that in some fields there may be no getting around the practical difficulty of detecting infringement, and for several reasons, the software industry appears to be one of these fields. The abstract nature of software, the legal issues regarding collecting and analysing evidence in Europe, and the costs in general set multiple barriers for detecting infringements in the case of software patents and may thus also affect the selection of companies’ patent exploitation strategies.

Keywords:
software patent, patent strategy, infringement monitoring, reverse-engineering
## Contents

### 1 Introduction
- 1.1 Introducing the Subject of the Thesis ................................................................. 1
- 1.2 Research Agenda and Demarcations ................................................................. 3
- 1.3 Research Methodology ..................................................................................... 7
- 1.4 Structure of the Thesis, on the Sources and Terminology ............................... 8

### 2 Theoretical Background
- 2.1 History and Characteristics of the Software Industry ........................................ 13
  - 2.1.1 History of the Software Industry ............................................................... 13
  - 2.1.2 Characteristics of the Software Industry .................................................. 14
- 2.2 Legal Protection of Software and the Nature of Software ............................... 16
  - 2.2.1 How Software Is Legally Protected? ......................................................... 16
  - 2.2.2 Dual Nature of Software and the Idea-Expression Dichotomy ............... 19
- 2.3 Characteristics of Software Patents ................................................................. 22

### 3 Software Patents in Europe
- 3.1 European Patent System .................................................................................. 26
  - 3.1.1 The European Patent Convention and the European Patent Office .......... 26
  - 3.1.2 European Patent ....................................................................................... 28
- 3.2 European Patent Law on Software .................................................................. 31
  - 3.2.1 Article 52 and Its Exception Regarding Computer Programs .................. 31
  - 3.2.2 The Failed Commission Proposal for a CII Directive ............................... 34
  - 3.2.3 Industrial Application and Novelty ............................................................ 36
  - 3.2.4 Inventive Step ............................................................................................ 38
  - 3.2.5 Technical Character ................................................................................ 41

### 4 Strategies and Exploiting Software Patents
- 4.1 Introduction and Chapter’s Contents ................................................................. 46
- 4.2 IP Strategy and Management .......................................................................... 48
  - 4.2.1 Levels of IP Management ......................................................................... 48
  - 4.2.2 IP Strategy in the Context of Overall Strategy and Business Model .......... 50
- 4.3 Patenting Strategies ......................................................................................... 53
- 4.4 Patent Exploitation Strategies and Software Patents ...................................... 56
  - 4.4.1 Patent Strategy and Patent Exploitation Strategies .................................... 56
  - 4.4.2 Defensive Strategy .................................................................................... 58
Abbreviations

CJEU Court of Justice of the European Union
EPC European Patent Convention
EPO European Patent Office
EU European Union
ibid. ibidem, the same as above
ICT Information and Communications Technology
IP Intellectual Property
IPR Intellectual Property Right
NPE Non-Practicing Entity
R&D Research and Development
SME Small and Medium-sized Enterprise
TRIPS Agreement on Trade Related Aspects of Intellectual Property Rights
WCT WIPO Copyright Treaty
WIPO World Intellectual Property Organization

List of Tables

Table 1: Research agenda........................................................................................................6

Table 2: Reverse-engineering of software-related inventions under the current European legal framework.................................................................99
1 Introduction

1.1 Introducing the Subject of the Thesis

The Information and Communications Technology (ICT) industry, which the software industry is a part of, is not only among the most significant and influential industries in the world these days but also an industry where intellectual property is concentrated and wealth is created.\(^1\) Furthermore, our modern world is today powered by many breathtaking innovations in all sorts of technological fields which make, \textit{inter alia}, use of software, and refusing patent protection for software-related inventions would exclude from patentability almost the whole area of ICT and with it one of the most dynamic and innovative industries.\(^2\) On the other hand, due to the emergence of new technologies such as software, the patent system has become increasingly more complex, and the issue of to what extent patent protection shall be available to software-related inventions is still a very controversial topic.

Although the language of the national patent laws of many European countries and the European Patent Convention (the EPC) still suggests that computer programs cannot be patented \textit{as such} in Europe, in reality it has been possible to obtain patent protection for software-related inventions for decades now. Actually, today there are tens of thousands of patents in Europe for software inventions, which demonstrates that granting software patents is by now a well-established practice. Although software patents are now relatively widely patentable under the EPC framework, it is emphasized throughout this thesis that managing software patents is much more than just acquiring a patent through patent offices. Namely, software patents need to be adequately utilized, by using different patent exploitation strategies, or they are not worth much. This has been noted by many companies\(^3\) and scholars, as during recent years the strategic management of intangible assets in general has increasingly attracted their attention.

There has been a trend that companies have increasingly emphasized the importance of intellectual assets and the management of intellectual property. Moreover, in the context of software patents many companies have now started to recognize the value of patent

---

\(^3\) Note that the terms \textit{company}, \textit{firm}, \textit{enterprise}, \textit{corporation} and \textit{organization} will all be used interchangeably in this thesis.
strategies. Fundamentally a patent strategy sets the stage for how the patent owner will develop, protect and exploit patents, but also patent infringement monitoring and enforcement are considered to be parts of it. Indeed, companies are now utilizing patent strategies and using the patent system in more sophisticated ways than ever before. Because of ever increasing competition, it is extremely important to recognize and learn the different ways to exploit software patents (patent exploitation strategies) to reap the benefits or to protect oneself against competitors. Therefore, the words of Palfrey hold true especially with software patents: “once you understand your intellectual property assets, you have a much better chance of using them to the fullest advantage”.

As companies have realized the strategic importance of patents and thus their role in commerce, companies have started to harvest more and more patents for business tools and bargaining chips. Moreover, especially software-related inventions often seem to derive their value rather from strategic usage than directly from their technological value. Therefore, it is not uncommon for companies especially in ICT industries to patent primarily for strategic reasons and only secondarily to protect the technological value behind the invention itself. However, even though companies would recognize the value of their software patents – whether technological, commercial or strategic – in their patent strategies, they might not be able to exploit and enforce their patents in an intended manner. Namely, a major problem with software patents is infringement detection. Moreover, the nature of software and the European legal framework create multiple barriers not only for detecting infringement but also for collecting and analysing evidence. Therefore, some strategic aspects of the use of patents are limited in the case of software patents.

Even though patent strategies in general have attracted researchers, relatively few studies have been written from the perspective of patent exploitation strategies in the context of software patents. Additionally, prior investigation of academic literature has

---

4 Note the difference between the terms patent strategy and patent exploitation strategy. Patent strategy is a rather broad term which covers strategies regarding developing, protecting, exploiting, monitoring and enforcing patent. Thus, patent exploitation strategy is a part of patent strategy and it deals with issues regarding different exploitation mechanisms, such as offensive and defensive strategy.
5 Palfrey (2012), p. 56.
6 See e.g. Gibbs & DeMatteis (2003); MacDonald (2004); Gilardoni (2007); Dobrusin & Krasnow (2008); Gollin (2008); Leone & Laursen (2011); Somaya, Teece & Wakeman (2011); Somaya (2012); Knight (2013).
7 See e.g. Soininen (2007b); Leaves (2010).
unveiled that only little previous research has focused on patent infringement monitoring in general\textsuperscript{8} and software patent infringement monitoring in particular\textsuperscript{9}. Furthermore, to the best of my knowledge, prior academic literature has stayed silent, both in general and in the context of software patents, on the issue of patent exploitation strategies in the context of patent infringement monitoring. Therefore, the originality of this thesis resides in the combination of these two aspects. It contributes to the discussion on software patents in particular by bringing the aspects of patent exploitation strategies and patent infringement monitoring together. Additionally, it identifies and clarifies some problem areas in the context of software patents, such as the European legal framework regarding reverse-engineering, and explains why the nature of software causes problems in infringement detection.

1.2 Research Agenda and Demarcations

The introduction to the study chapter has shown that there is a need to investigate and to get a deeper understanding of patent exploitation strategies and patent infringement monitoring separately but also of the relationship between these two in the context of software patents under the current European legal framework. Furthermore, considering that until now both disciplines, fields of studies in strategic management and intellectual property, have given limited answers to many questions related to software patents in the contexts of exploitation strategies and infringement monitoring, it is considered that there is a need for a broader understanding of these areas at the general level. Therefore, the first research aim of this thesis is to expand the understanding and discuss the problems regarding software patents in the contexts of exploitation strategies and infringement monitoring.

In addition, considering that no prior research has been conducted, neither in general nor in the context of software patents, with the goal of combining the aspects of exploitation strategies and infringement monitoring, this research gap should be addressed and examined. Therefore, the second research aim of this thesis is to bring the

\textsuperscript{8} See e.g. Davis (2006); Pressman (2008); Ma (2009); and Knight (2013). To be more precise, these works have only brief excerpts regarding patent infringement monitoring, and there is not currently, at least to my knowledge, any academic publications that solely focus on patent infringement monitoring.

\textsuperscript{9} See e.g. Nichols (1998); Olsson & McQueen (2000); Mann (2005); Zeidman (2011). Similarly is the case of general patent infringement monitoring literature, there are only brief excerpts available regarding the issue.
aspects of patent exploitation strategies and patent infringement monitoring together and to contribute new knowledge to the fields of strategic management and intellectual property.

Considering the discussion above, this thesis proposes the main research question to be formulated as follows:

- What are the roles of patent exploitation strategies and patent infringement monitoring in the context of software patents, and what kind of challenges do the nature of software and the European legal framework pose for detecting patent infringements and do these challenges have any effect on patent exploitation strategies and patent infringement monitoring?

Because the aim of this thesis is broad in scope and given the multiplicity referred to above, it is important to identify several sub-research questions to support this thesis and to answer the main research question sufficiently.

- What are the characteristics and nature of software and how can software be protected?
- What requirements the European Patent Convention sets for the patentability of software?
- How software patents can be utilized by using different patent exploitation strategies?
- What challenges do the nature of software and the European legal framework pose for detecting patent infringements of software patents?

The previous paragraphs have laid the foundations for this research. Therefore, the aim is next to present the main limitations considered in this thesis. However, before proceeding into these main limitations it should be noted that in the course of this thesis a number of smaller limitations are also presented. The first main limitation covers the issue of proprietary software versus open source software. The scope of this thesis is limited to the proprietary and commercial software and does not consider open source software\textsuperscript{10}. This limitation stems from several facts. First, this thesis focuses on patent exploitation strategies which are mostly utilized in the context of commercial software. Although exploitation strategies could be utilized in the context of open source software, though to a limited extent, the fact that patent strategies in general are used for

\textsuperscript{10} When a software program is open source, it generally refers to a computer program in which the source code is freely available to the general public for use and/or modification from its original design. For further details, see e.g. Rosen, Lawrence (2005), Open Source Licensing: Software Freedom and Intellectual Property Law, Prentice Hall PTR, New Jersey.
the purposes of making more money or saving money\textsuperscript{11} offers a more fruitful ground for discussion in the context of proprietary software than in the context of open source software. Second, when discussing the collection and analyse of evidence, the interest in this thesis is in reverse-engineering, which has the goal of producing from object code\textsuperscript{12} a similar type of human-readable code that source code\textsuperscript{13} is. However, under the open source models source code is distributed openly, which means that no reverse-engineering is required in order to collect and analyse evidence, and therefore such legal issues that are present in the case of reverse-engineering the object code of proprietary software are not present when source code is openly available.

The next limitation concerns European patents with unitary effect. On 11 December 2012 the European Parliament voted positively in a first reading of the EU Council’s compromise proposals for two draft EU regulations\textsuperscript{14} on a unitary patent for Europe. The unitary patent package builds on the existing system of European patents, and the regulations entered into force on 20 January 2013\textsuperscript{15}. Put briefly, the aim of the proposal is to provide the same scope of patent protection in all participating member states and make access to the patent system easier and less costly. This will happen by obtaining a unitary patent, or rather a European patent with unitary effect, which will be a European patent under Article 142 of the EPC, and it will co-exist with national patents and the current European patent.\textsuperscript{16} However, this thesis does not address issues regarding the unitary patent proposal and thus also European patents with unitary effect are left outside the scope of this thesis, although the proposal is discussed briefly in the context of future aspects of reverse-engineering in chapter 5.5.4.2. Albeit the fact that the unitary patent aspect of European patents is not covered in this thesis, the author wants to emphasize that it offers an extremely interesting avenue for future research\textsuperscript{17}.

\textsuperscript{11}See e.g. Soininen (2005), p. 66 and Dobrusin & Krasnow (2008), p. 195.

\textsuperscript{12}Object code is a machine-readable language that is translated from source code by using a compiler. Object code (or executable code) is usually a series of bits (“zeros and ones”) and thus cannot be read by humans. See e.g. MacQueen et al. (2010), p. 64 and Ballardini (2012b), p. 11.

\textsuperscript{13}Source code embeds the programmer’s instructions in a programming language. Furthermore, source code usually resembles English and is the only part of the program that is readable by humans. See Ballardini (2012b), p. 11.

\textsuperscript{14}The first draft regulation concerns unitary patent protection, and the second sets out the translation arrangements for such protection.

\textsuperscript{15}However, they will only apply from 1 January 2014 or the date of entry into force of the Agreement on a Unified Patent Court, whichever is the later.


\textsuperscript{17}The changing business environment that results from regulatory changes obviously affects companies’ forward-looking strategies and can lead to new patterns in their behaviours. Therefore, a unitary patent
### Table 1: Research agenda

<table>
<thead>
<tr>
<th>Issue</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Research problem</td>
<td>There is a need to investigate and to get a deeper understanding of patent exploitation strategies and patent infringement monitoring separately but also of the relationship between these two in the context of software patents under the European legal framework.</td>
</tr>
</tbody>
</table>
| Aims                   | Aim 1: To expand understanding and discuss the problems relating to software patents in the contexts of patent exploitation strategies and patent infringement monitoring.  
Aim 2: To bring aspects of patent exploitation strategies and patent infringement monitoring together and to contribute new knowledge to the fields of strategic management and intellectual property. |
| Main Research question | What are the roles of patent exploitation strategies and patent infringement monitoring in the context of software patents, and what kind of challenges do the nature of software and the European legal framework pose for detecting patent infringements and do these challenges have any effect on patent exploitation strategies and patent infringement monitoring? |
| Sub-research questions | SRQ 1: What are the characteristics and nature of software and how can software be protected?  
SRQ 2: What requirements the European Patent Convention sets for the patentability of software?  
SRQ 3: How software patents can be utilized by using different patent exploitation strategies?  
SRQ 4: What challenges do the nature of software and the European legal framework pose for detecting patent infringements of software patents? |

Protection offers a fruitful ground for discussion in the context of European software patents. The main question that many companies ask is: should they obtain a European patent with unitary effect or stay with the traditional European patent? To decide the best course of action, companies will need to assess the costs, risks and benefits of the unitary patent protection versus their traditional patent filing approach. With the reduced renewal fees and translation costs, obtaining a unitary patent is likely to be less expensive than the current system. This will make the unitary patent more attractive for applicants who are looking for a cheaper alternative or who previously could not afford the cost of prosecuting an application to grant in Europe. This advantage needs to be balanced against a particular strategic disadvantage: the unitary patent is a single legal right. If the legal right is successfully invalidated by a third party, or if renewal fees are through inadvertence not paid, the applicant loses all such rights in Europe. Competitors wishing to invalidate a unitary patent need to commence only one legal proceeding, and if successful, will eradicate the patent throughout Europe. In contrast, under the current system, if a third party wishes to invalidate patents in Europe, it must attack each one separately in the courts of that particular country. Multiple legal proceedings are more expensive than a single proceeding. If the third party is successful in some countries and unsuccessful in others, the challenger will not have unfettered access to the European market. Instead, they will be limited to only those countries where the national patents have been invalidated. Not every competitor will have the financial capabilities to attack every jurisdiction in Europe. Obtaining national rights may, therefore, be seen as a safer alternative for the most valuable technologies or highly competitive industries. However, the fact that the unitary patent proposal contains a significant number of transitional provisions, it is apparent that applicants will be able to stay with the current system and watch the unitary patent system from the sidelines for some time. For further details, see Foster (2012).
1.3 Research Methodology

This thesis was written with the purpose of combining aspects of law and strategic management. Hence, the thesis can be described as interdisciplinary, founded on prior legal, technological and strategic managerial writings. The merits and relevance of using other disciplines such as sociology, economics and history as aids to legal research have been widely recognized among scholars. Namely, interdisciplinary, or socio-legal research as some may call it, broadens legal discourse in terms of its theoretical and conceptual framework. Furthermore, interdisciplinary approaches to legal scholarship not only provide an alternative to the traditional doctrinal analysis but they have also encouraged lawyers to engage in critical and cutting-edge research to examine the relationship between law and other social relations.18

Because this study investigates different legal, strategic and technological aspects of software patents within the European software and ICT industry, and for the purposes of carrying out the research agenda and to answering the research questions properly, several steps had to be taken in order to expand the traditional “narrow” mode of legal methodology. Furthermore, it is necessary to explicate how this study is positioned within the methodological framework. One discipline may deal with several kinds of research questions, as this thesis does, and so different methods are relevant in answering these questions19. Therefore, the methods employed in this thesis include approaches based on legal dogmatics, legal sociology and legal informatics.

It is important to note that legal sociology and legal informatics are complementary to doctrinal research and all these methodologies can be used simultaneously to examine a legal issue, as is advocated by academic scholars20. Although the overall focus of this study is not the systematization and interpretation of legal material in the traditional sense, legal dogmatics is clearly visible in some parts of this thesis, particularly strongly in chapter three and in some sections of chapter five. Especially chapter three owes much to the conventions of traditional legal dogmatics as its goal is to examine the existing legal framework and praxis regarding software patents in Europe. In chapter

19 According to Hage, “If within a discipline different kinds of research questions are being asked, the issue of method should be focused on a type of research question, rather than on the discipline as a whole.” See Hage (2011), p. 22.
five the discussion covers reverse-engineering, which is heavily regulated under the European legal framework, and therefore chapter five will mean a slight return to legal interpretation and systematization.

However, as pointed out already, legal dogmatics alone does not constitute a particularly fruitful starting point for the current research agenda, and therefore other methods have been used to answer the sub-research questions. Legal sociology, or sociological jurisprudence, borrows from methods applied in other fields and it characteristically involves examination of the legal system from the outside.\textsuperscript{21} In the context of this thesis this means that doctrinal analysis is infused with an approach drawn from the field of strategic management\textsuperscript{22,23} Even though the thesis in some parts, chapter four and partly chapter five, leans on the field of strategic management, the analysis of strategy, however, is limited to the considerations of patent exploitation strategies and intellectual property strategies.

Finally, because there is need for a better understanding of the relationship between law and new technologies, legal informatics has been selected as the final method and it completes the tri-lateral approach applied to this thesis. Although the term legal informatics is not an easy one to pin down, legal informatics involves the application of information and communication technologies to the legal environment\textsuperscript{24}. Furthermore, legal informatics explores and discusses policy issues relating to problems and challenges that information and communication technology, and thus also software technology, cause in the legal environment.\textsuperscript{25} In the context of this thesis the approach of legal informatics is reflected in discussions that chapter five will entail.

\textbf{1.4 Structure of the Thesis, on the Sources and Terminology}

The thesis is divided into six chapters. Chapter one laid down the starting points and essential framework for the study. Chapter two begins with a brief overview of the

\textsuperscript{21} McConville & Chui (2007), pp. 5-6; Soininen (2007b), p. 28.

\textsuperscript{22} According to Turban, McLean & Wetherbe, “Strategic management is the way an organization maps the strategy of its future operations”. See Turban, McLean & Wetherbe (2001), p. 77.

\textsuperscript{23} Strategic management can be placed in the category of organizational studies, or more specific organizational behaviour, which again can be placed under sociological studies.

\textsuperscript{24} Paliwala (2010), pp. 11-12.

\textsuperscript{25} Cf. Saarenpää (2012), pp. 410 and 425.
history and characteristics of the software industry. After that, the chapter explains different protection mechanisms covering software, the specific nature of software, and how the software industry and patents function together. Moreover, the overall goal of the chapter is to facilitate the understanding of the rest of the thesis by providing a basic understanding of software. Chapter three outlines the European patent system and the status of software patents in this context. The chapter offers a brief overview of the European patent system and discusses issues regarding the patentability requirements under the EPC, because understanding the basics of the European patent system and the European Patent Office’s decisions may not only help companies to determine whether or not their software-related invention is patentable but they also help companies to understand legal developments and pressures which are a critical ingredient of successful corporate and business strategic management\textsuperscript{26}. Furthermore, the central goal of the chapter is to support chapters four and five.

Chapter four moves from dealing with patents as legal instruments to dealing with patents as business tools and bargaining chips that can be strategically utilized by companies. Companies willing to utilize their software-related inventions to the fullest extent should not only understand the different patent exploitation strategies but also how these exploitation strategies are connected to IP strategy and overall business strategy. Therefore, the chapter starts with an overview of the fundamentals regarding IP strategy and management and how they relate to a company’s overall strategy and business model. After that, the chapter briefly introduces patenting strategies and how they differ from patent exploitation strategies. Finally, the chapter discusses different patent exploitation strategies with a special focus on defensive and offensive strategies, which are the two essential exploitation strategies that every software company should be aware and know the basics of.

Chapter five is the culmination of the thesis as it brings previous chapters together and produces new information. The chapter begins with a discussion on the search for prior art in the context of software patents, which provides the basic framework for modified search for prior art in the context of software patent infringement monitoring discussed later. Next, this chapter highlights the reasons why it may be difficult to detect infringing uses, i.e. why so many software-related inventions suffer from the problem of

\textsuperscript{26} See Grant (2005), pp. 68-69.
detectability. The chapter continues by exploring issues regarding collecting and analysing evidence in the context of software patents, and these discussions illustrate the problematic nature of reverse-engineering under the European legal framework. After discussing reverse-engineering, the focus shifts to patent infringement monitoring in the context of patent exploitation strategies. In the end, the chapter briefly introduces the most common ways to react once an infringement has been detected and evidence has been collected. The final chapter, chapter six, presents a summary of the main themes of the study and concludes the thesis by discussing how this thesis has contributed new knowledge to the domains of strategic management and intellectual property.

This study is an interdisciplinary study, which in the case of sources means that it is founded mainly on prior legal and strategic managerial writings but also some technological writings are utilized. Furthermore, prior legal and strategic managerial books, doctoral dissertations, journals and articles play a significant role in this thesis, and by analysing multiple different sources and sustaining source criticism, the goal has been to achieve an all-around and unbiased starting point for conducting this study, which is essential in academic debate. The sources used are not limited to such academic literature that only covers European level discussions, because this would not only defy a fruitful analysis but it would also offer an unnecessarily limited view of the topic at hand. Additionally, the importance of international treaties and the legal tradition of international comparison are strongly present when discussing patents and patent laws\(^\text{27}\), and this fact further supports the author’s choice to include also academic publications outside Europe as one of the source materials of this thesis.

Next, it is essential to provide definitions of some key terms that are used in this thesis. Although no European legislation defines \textit{software}, computer science and software engineering have formulated and provided different definitions over time. For example, one way to define software is as follows: “software is the entire set of programs, procedures, and related documentation associated with a system and especially a computer system”.\(^\text{28}\) Thus, software includes both the computer program and its ancillary materials (e.g., documentation and computer files or data), so it follows that a

\(^{28}\) Ballardini (2012b), p. 11.
**computer program** is simply a part of software and that patenting a computer program is not the same as patenting software. Furthermore, computer programs can be defined as “a combination of computer instructions and data definitions that enable computer hardware to perform computational or control functions”.\(^\text{29}\) However, in this study, the terms software and computer program are used interchangeably and without making further distinction between these terms.

When it comes to defining *software patent* or *software-related invention* no widely accepted definition exists for either. The terms software-related invention and software patent, note that these terms are used henceforth interchangeably, are both used here to describe inventions that “employ software to perform their function and where the inventive contribution is embodied within the software itself”\(^\text{30}\). Furthermore, these terms should be taken to encompass both 1) inventions that are described solely in the terms of the software (e.g. the program or programs it contains), and 2) inventions that claim products or processes whose functionality depends on software (e.g. computers or methods for performing certain functions or tasks)\(^\text{31}\). Additionally, it should be noted that this thesis does not try to address or make a difference between so called “pure” software patents, i.e. patents protecting inventions that can be fully considered as software, and computer-implemented inventions\(^\text{32}\).

Furthermore, due to the adopted approach, software patents in the context of the European legal framework, it requires some limitations regarding the definition of *patent infringement*. In Europe, in spite of international and EU-level harmonization attempts, what constitutes actual infringement of a patent is dictated by the laws of the country, here a member state of the EPC, in which the patent is granted. Because the definition and scope of patent infringement may vary by jurisdiction, it is not convenient within the frame of this thesis to exhaustively discuss the specifics of patent infringement or the scope of exclusive rights and how they are defined and interpreted in national laws and courts of different EPC member states. Therefore, the term patent infringement henceforth refers to all such acts that theoretically can constitute a patent infringement in the case of a European patent. Furthermore, this means that no division

\(^{29}\) Ballardini (2012b), p. 11.
\(^{30}\) MacQueen et al. (2010), pp. 534-535.
\(^{31}\) Ibid.
\(^{32}\) See e.g. Rentocchini (2010), p. 3.
is done when it comes to infringement types or liabilities or national rules\textsuperscript{33}. In addition, it is recognized that the Internet causes specific problems regarding the choice of law in some software patent infringement cases, but discussion regarding these issues are also excluded from the scope of this paper\textsuperscript{34}.

\textsuperscript{33} "Patent infringement liability involving several parties may be framed in at least four different ways: direct, indirect, vicarious, and joint liability, depending upon how subjective and objective elements are required and regulated. Direct infringement, whether literal or by equivalents, looks only at the objective elements of patent infringement. Because direct infringement does not usually require the infringer’s knowledge or negligence, it is often regarded as akin to strict liability. Indirect infringement and joint direct infringement (both literal and non-literal) introduce subjective elements of fault to the enquiry. Both direct and indirect infringement liability is based in statutory patent law. Vicarious liability and joint liability require an element of control in both direct and indirect infringement actions. Furthermore, vicarious and joint liability, are often constructed from relevant civil and tort laws. Therefore, conduct allowed under the patent law may still be illegitimate under other laws.” See Lee (2010), pp. 38-43. See also Zeidman (2011), pp. 95-97.

\textsuperscript{34} Due to the nature of the software and technology, software patents increase the possibility of partial or incomplete use. In addition, software is likely to be produced and used in a global setting, for instance through the Internet. When the modules and parts that form the software product are used and accessed through the Internet, it often means that these actions take place transnationally. As pointed out by Lee, this again leads to problematic scenarios: When the software is accessed and used globally, it may lead to a so called “ubiquitous infringement” where multiple infringements occur globally through ubiquitous media such as the Internet. On the other hand a software product may be used only in parts in each territory with no complete conduct of use in a single territory, to find a direct infringement (so called “fragmented infringement”). See Lee (2010), pp. 4-5.
2 Theoretical Background

2.1 History and Characteristics of the Software Industry

2.1.1 History of the Software Industry

Software has been developed and traded since the 1950s, but even still in the late 1960s software was very much an arcane area understood by a few scientists and engineers in universities and high-technology industries. Until the mid-1960s, software was not generally thought of as a product that could be bought and sold apart from the computer, and therefore software was bundled and integrated in the computer. In addition, at the early stages of the software industry, computer software was mainly created and used within a particular organization, and thus computer programs were seen more as business and industrial tools rather than items of property capable of being commercially exploited. Therefore, there were no software packages for consumers that even remotely resembled the software now available for personal computers. However, soon, despite earlier beliefs, it was realized that computer software could actually be used to generate income.

The origins of the software industry can be traced to the late 1960s when unbundling of software started in the United States, which meant that software began to be marketed, sold and distributed independent from hardware products. This development provided opportunities for entry into the software industry by independent producers of standard and custom operating systems, as well as independent suppliers of applications software for mainframes. Therefore, throughout the 1970s, many independent software vendors entered into market and the software industry gained extraordinary momentum. In the 1980s, the development and diffusion of the desktop computer produced explosive growth in the traded software industry. It also became apparent that software constituted a new object the intellectual creation of which necessitated considerable personal and financial resources, but which could easily be copied at relatively low

Furthermore, ever since the 1980s, software development has been characterized by the trend of integration, bundling, componentization and re-use.\footnote{Kur & Dreier (2013), p. 250.} Ever since the 1980s, software development has been characterized by the trend of integration, bundling, componentization and re-use. From the late 1990s, the software industry has been influenced by the growth of networking, which is closely linked to the rise of the Internet. The Internet has not only offered a new market place for proprietary software but also simultaneously facilitated low-cost distribution\footnote{Koo (2002), p. 193.}.\footnote{Mann (2005), pp. 968-969.} In essence, the software industry, as well as the whole ICT industry, has faced numerous changes in the past 20 years, and the biggest and most obvious changes have been digitization, the Internet and globalization in general.\footnote{Graham & Mowery (2005), p. 51.} ICT is one of the most significant and influential industries in the world these days, and software, for example, is indispensable and pervasive, and is to be found almost everywhere and in almost all fields of technology and business.\footnote{According to Mann, “The rise of the Internet brought a tremendous influx of capital into the industry. Therefore, it is easy to understand why the nowadays’ software industry is characterized by astonishing levels of growth, innovative activity and competition, and why it is claimed to be one of the fastest growing industries in the world.” See Mann (2005), p. 969.} Therefore studying it is of great importance.

### 2.1.2 Characteristics of the Software Industry

Different industries reflect different sources of innovation. In some industries, a few large, established companies contribute a significant number of innovations. In others, much innovative activity is generated by small and medium-sized enterprises (SMEs). The software industry shows some elements of both patterns: large, established companies and new SMEs have both been major sources of new products and other

---

\footnote{Kur & Dreier (2013), p. 250.}
\footnote{Koo (2002), p. 193.}
\footnote{Mann (2005), pp. 968-969.}
\footnote{Graham & Mowery (2005), p. 51.}
\footnote{According to Mann, “The rise of the Internet brought a tremendous influx of capital into the industry. Therefore, it is easy to understand why the nowadays’ software industry is characterized by astonishing levels of growth, innovative activity and competition, and why it is claimed to be one of the fastest growing industries in the world.” See Mann (2005), p. 969.}
\footnote{Soininen (2007b), p. 13.}
innovations. Another characteristic of the software industry, which explains the high level of innovative activity, is that the software industry has relatively low fixed costs. The requirement for capital investment in the case of software development is relatively low – mostly consisting of hiring personnel. Although, the costs of writing software have increased substantially over time as programs have become more complex, the costs of writing and manufacturing computer programs remain low relative to the fixed costs of development in many industries. In addition, even though software development is fundamentally uncertain and risky, due to the relatively low fixed costs, software developers can try many options and see what works.

Even though the fixed costs may be relatively low, software inventions tend to have short life cycles, and that balances the advantage created by low development costs. To be more precise, the average life cycle of software and computer programs is approximately 18 months. If compared to industries like steel or aircraft, where new generations of products are infrequent and those products may last for decades, computer programs tend to be replaced every few years and often by new versions of the same program. Indeed, fast development of the software industry and short product life cycles reduce the total value of software products in many cases. There are exceptions; successful software products have many versions and long lifetimes. In other words, although new versions come out every few years and names change, the same code base can be used for decades. Obviously, if a certain feature is likely to be continued through several versions of the software, the value of such software rises. A good example of this is the file allocation table (FAT) file system developed by Microsoft. This was a (patented) low level software feature that has remained in use since the late 1970s to the present day.

In the development of large software systems, componentized design architecture has developed due to the need to maintain compatibility. Furthermore, the need of

---

54 Burk & Lemley (2009), p. 84.
57 Leaves (2010), p. 76.
compatibility and interoperability has encouraged standardization. The standardization trend has also been accelerated by the emergence of the Internet. Therefore, many software-related innovations are now substantially constrained by the need to preserve compatibility between programs, systems and networks. In sum, due to the trend to integration and componentization as well as the need of compatibility, innovations in software development can often be characterized as sequential, cumulative and incremental. In other words, software progresses by small steps and software normally builds on pre-existing ideas and often on prior code itself, and innovations that result from other innovations are termed cumulative.

Cumulative innovations are common in information technology and biotechnology, but especially the software industry is highly cumulative in nature where innovations are technologically dependent on one another. Software typically consists of previously coded software, which is then modified and to which new code is added with the larger system composed of various components. According to Lévêque and Ménière, “The highly cumulative nature of software explains why the application of intellectual property to software has raised concerns among many programmers, who have been worried about working under the constant threat of an infringement suit.” In addition, Mazzoleni and Nelson have stressed that especially in industries characterized by cumulative innovations it is more likely to find strategic patenting behaviours such as cross-licensing, blocking rivals or extracting licensing revenues.

2.2 Legal Protection of Software and the Nature of Software

2.2.1 How Software Is Legally Protected?

One of the most challenging tasks for IP law is to determine the degree of protection for software necessary to protect innovative computer software while preserving

---

61 More about the cumulative innovation theory, see e.g. Burk & Lemley (2009), p. 83.
65 The strategic use of patents in the context of software is one of the central themes of this thesis, and is discussed more intensively in chapter four and also in chapter five.
competition and the dissemination of knowledge and information.\textsuperscript{66} This was noted especially when discussions regarding the most appropriate form of intellectual property right (IPR) to protect computer software first started. Originally it was thought that contract law gave the best protection for software, and thus the protection of computer programs was mainly executed by forming license agreements between parties.\textsuperscript{67} However, the technology evolved and personal computers became more and more common and cheaper, but the development of software still involved expensive human resources. Soon it was realized that the contractual arrangements no longer offered an adequate protection.\textsuperscript{68} These factors led to a situation in which it was necessary to start visualizing new ways, other than contractual, to protect computer programs. The need to have some form of exclusive protection to protect computer software from unauthorized copying and use was high within the software industry. Therefore, in view of this, intellectual property laws needed to be modified and extended to give better protection for computer software.\textsuperscript{69}

Intensive international discussions took place regarding the legal protection of software during the 1970s and the first half of the 1980s. However, in the beginning, it was not quite clear which existing protection regimes would be most appropriate to satisfy protection needs.\textsuperscript{70} In spite of the fact that computer programs are mostly functional in nature, the feature of text in the form of a programming language made copyright an easy solution for the protection of computer programs, and therefore copyright finally emerged as an intellectual property right under which computer programs were granted exclusive protection.\textsuperscript{71} There were other reasons for this development as well. First, patent protection was not available at that time for computer programs under either US or European law. Second, plans to adopt a \textit{sui generis} protection\textsuperscript{72} for computer programs at the international level within the World Intellectual Property Organization (WIPO) had been abandoned in 1985.\textsuperscript{73} Third, as an IP right which does not require any formalities, copyright can easily be obtained. Hence, copyright proved to be a suitable

\textsuperscript{66} See e.g. Deschamps (2011), p. 104.
\textsuperscript{67} Bainbridge (2008), p. 8.
\textsuperscript{68} See e.g. Moschella (1997), pp. 14-16.
\textsuperscript{72} Every now and then new forms of IPR protection have been suggested because existing forms of protection, such as copyright and patent for instance, are seen to be inadequate to accommodate emerging technologies. For more details, see e.g. MacQueen et al. (2010), p. 6.
\textsuperscript{73} More about a \textit{sui generis} protection, see e.g. Leith (2011), p. 160.
protection scheme in particular as regards the interests of software developers who needed quick, easy to prove and potentially far-reaching exclusive legal protection against one-to-one copying and adaptations which competitors develop on the basis of their programs.⁷⁴

Nowadays, computer programs are recognized as copyrightable subject-matter under the TRIPS Agreement⁷⁵ and the WIPO Copyright Treaty⁷⁶, both requiring their protection as literary works under the Berne Convention⁷⁷.⁷⁸ As the vast majority of nations, including all EU Member States, have now become signatories to TRIPS, computer programs are both internationally and domestically granted copyright protection automatically.⁷⁹ Furthermore, at the European level, the protection of computer programs is harmonized by the Software Directive⁸⁰, which stipulates that “computer programs are protected by copyright if the program is original in the sense that it is the author’s own intellectual creation, and no other criteria shall be applied to determine its eligibility for protection.”⁸¹ It should be noted that the so called Infosoc-directive⁸² should also be taken into account when examining the scope of copyright that governs computer programs because the Infosoc-directive can have indirect effects, for instance, when interpreting and applying the articles of the aforementioned Software Directive.⁸³

Although there is a strong legislative basis and international framework to protect computer programs via copyright, software is a pluralistic product that contains several elements, each of which could fall into different categories of IP law.⁸⁴ In other words,

---

⁷⁵ Agreement on Trade Related Aspects of Intellectual Property Rights, April 15, 1994. See Article 10(1) of the TRIPS Agreement.
⁷⁶ The World Intellectual Property Organization Copyright Treaty, adopted in Geneva on December 20, 1996. See Article 4 of the WCT.
⁷⁸ It has even been said that “The extension of copyright laws to expressly include computer programs as copyrighted works is one of the best illustrations of the adaptation, but perhaps manipulation, of copyright law to respond to the requirements of the information society, in particular to prevent the unfair use of original works that technological advances permit. See Deschamps (2011), p. 107.
⁸¹ Article 1(3) of Directive 2009/24/EC.
⁸³ See e.g. Välimäki (2009), p. 12.
computer programs may be protected by other IPRs such as patents, trademarks, design laws and trade secrets\(^{85}\) as well, depending on the circumstances.\(^{86}\) Actually, it is quite common that multiple forms of protection apply to an item of intellectual property.\(^{87}\) Bainbridge has listed reasons for the difficulty in identifying a single appropriate form of intellectual property right for computer programs. First, computer programs are dynamic in their nature in the sense that computer programs manipulate symbols and data and cause or control effects including physical effects, whereas other forms of copyrighted works in general are static.\(^{88}\) Second, software is executed by hardware, but software itself has no physical existence\(^{89}\), whereas traditional patentable inventions are typically physical in nature.\(^{90}\) As a matter of fact, as pointed out by Nichols, “Neither copyright nor patent was designed to fit software. The resulting confusion has made software the first technology to be widely protected by both copyright and patent, which is a problem because the two forms of protection had previously been considered to govern mutually exclusive domains.”\(^{91}\)

### 2.2.2 Dual Nature of Software and the Idea-Expression Dichotomy

A computer program is something of an anomaly in industry because it performs a function through expression, i.e. it exhibits behaviour as a result of its written text.\(^{92}\) In other words, computer programs are run on a processor and are part of a technological process, but they are also constituted of text in the form of a programming language\(^{93}\). Hence, computer programs have a dual character; being machines constructed by literary building blocks, and it is this dualistic nature which pushes towards both patent

\(^{85}\) Several other protection mechanisms are also available for software such as contracts, licensing agreements, and technical protection measures. See e.g. Ballardini (2012), p. 1.


\(^{87}\) However, there are substantial differences between IPRs as regards their nature, scope, duration and controls over them. See Bainbridge (2008), p. 2; Williams (2004), p. 370.

\(^{88}\) Bainbridge (2008), p. 3.

\(^{89}\) Design diagrams, source code, running code, and executable files on machine-readable media are all manifestations of the software, but none of them captures the entire invention. See Nichols (1998), p. 106.


\(^{92}\) Attridge (2001), p. 25.

\(^{93}\) There are many different programming languages around today and many new ones are being created. Each language has its own advantages and limitations. Some languages are designed to easily create graphical user interfaces. Others are designed to run from a Web page. Some are designed to manipulate strings of text, and others are designed to manipulate large databases of arbitrary data. Some languages are used to create integrated circuits. For further details, see Zeidman (2011), pp. 24-25.
and copyright protection. However, because software functions both as a literary work and as a machine, it inevitably blurs the historic line of demarcation between patent and copyright protection. Before software there was a sharp line between the copyrightable and the patentable. This differentiation is less easy now. Software blurs the distinction between expression and structure so much that a single source code can embody both.

Whereas the interpretation of a claim should be able to determine the basic scope of a patent, the dichotomy between protectable expression and unprotectable idea is thought to define the basic scope of copyright protection. This so called idea-expression dichotomy originates from the basic principle that copyright should not protect abstract ideas or functional objects, but rather the expression of such ideas. It is stated in various ways in various jurisdictions, and has since the late 1990s become a universally acknowledged principle, enshrined in the TRIPS Agreement and the WIPO Copyright Treaty. Also in Article 1(2) of the Software Directive it is stated that “protection in accordance with this Directive shall apply to the expression in any form of a computer program. Ideas and principles which underlie any element of a computer program, including those which underlie its interfaces, are not protected by copyright under this Directive.” Although the law seems to make a clear-cut distinction between an idea and its expression at a theoretical level, in practice the idea-expression dichotomy is not very useful in demarcating the protected from the unprotected. Especially in the case of software, due to the dual nature of software, the idea-expression distinction has become increasingly difficult to discern.

Furthermore, because software blurs the idea-expression dichotomy, it has caused many problematic situations in the case of computer programs. For instance, it is sometimes rather hard to distinguish what is allowed to be copied and what is not. Naturally,

97 See e.g. Ballardini (2010), p. 12.
98 Dutfied & Suthersanen (2008), pp. 82-83.
99 The Directive’s specific reference to ideas and principles underlying a program’s interfaces reflects the decision to exclude any possibility that computer interface specifications would come within copyright control and dangerously hobble interoperability of computer programs and equipment. See Goldstein (2001), p. 179.
101 In the case of computer programs, copying can happen literally or non-literally. Literal copying takes place when the program code itself is copied and the programming language is the same. Furthermore, source code and object code is protectable as a part of the computer program and direct copying of these
program developers want the widest possible protection and thus they advocate that almost all aspects of their programs constitute protectable expression. On the other side, companies involved in making compatible software argue that most aspects of a program should be considered unprotectable ideas, processes, or systems, and thus protection should be limited only to the literal code. At the European level, there have been many disputes regarding how far copyright protection extends in the case of software, which have even escalated into legal proceedings as was evident in the recent SaS Institute Inc v. World Programming Ltd case. In essence, drawing the line between protectable expression and unprotectable idea is not always an easy task to perform, and this task is even more difficult in the case of computer programs.

Though software is protected from direct copying of the actual code, both the source and object code, and also protected from much reverse-engineering, copyright does not prevent competitors from creating a similar program with the same functionality and appearance. In other words, the protection that is afforded to computer code by copyright is simply in the expression of that code and does not extend to the functional aspects or features of the software, that is, to the effects that software has when run on a computer, or to the underlying ideas and principles of the software. This is quite problematic from the viewpoint that it is usually the idea behind the program which is its commercially valuable element and where the true value of the software lies. Thus, it appears that mere copyright protection is not sufficient by itself to cover the

codes without consent is prohibited. Non-literal copying, again, occurs when “elements of a computer program such as, for instance, its structure, sequence of operation, functions, interfaces and methodologies are copied, but the program code per se is not directly copied. In this case, the two programs might be written in the same programming language or different programming language”. See MacQueen et al. (2010), p. 64 and Ballardini (2010), pp. 30-31.

103 SaS Institute Inc v World Programming Ltd (Case C-406/10) [2012] ECDR 15.
105 This is discussed later in chapter 5.5.
108 However, as Ballardini points out: “The fact that programme behavior in general is not protectable by copyright law because of its functionality does not mean that such behavior can never be protected by copyright. Sometimes programme behavior can be considered expressive in a traditional copyright sense and, thus, copyright applies.” See Ballardini (2010), pp. 8-9.
110 According to Mann, “Functionality in most cases is what makes software products attractive to customers”. However, as Mann points out, “There still are types of software for which functionality is not of central importance. For example video games are software products for which the expressive content is the primary market differentiator.” See more closely Mann (2005), p. 1012.
whole field of computer programs, and the protection of computer programs should be extended by other means of IPRs. Actually, copyright was never meant to cover subject-matter that “does” something; that is what patents are for. However, as we will later notice, the European patent system sets rigorous standards for software-related invention that must be satisfied before exclusive rights are accorded.  

2.3 Characteristics of Software Patents

A patent does not protect expression, as a copyright does. Instead, a patent can be said to protect the ideas embodied within inventions. Comparing patent to copyright generally means software is involved, and only rarely will copyright overlap with another form of patentable subject-matter. Patent protection for software has been slower to develop and more controversial to apply than copyright protection. To receive copyright protection, it is enough that the creation falls within a protected class; protection is then automatic. With the granting of patents, inventions must not only fall within the categories of patentable subject-matter, but they must also satisfy specific criteria if protection is to result. In Europe, this is adjudicated by the patent offices. These institutions are responsible for deciding if an application satisfies the criteria prescribed by legislation.

Ever since software patents started to be granted, software patents have been the focus of controversy within different interest groups. Especially topics such patentability and validity of computer software have been, and continue to be, at the heart of many debates over software patents. It goes without saying that there are many supporters of software patents but also many opponents. Supporters of software patents generally put forth the following arguments: availability of patent protection for software not only benefits large firms but also provides many advantages to SMEs and individual developers as well; software is within the ambit of patent protection; without

115 The only way for a small inventor to succeed against large software companies with influence and money is by allowing them to patent their software and start companies, license their software inventions to large companies, or, as a last resort, sue companies for incorporating their patented ideas. Patents can also help start-up SMEs to attract venture capital to their attractive software innovations. See more closely Nichols (1998), p. 103; Choudhary (2011), p. 446; Zeidman (2011), p. 92.
patent protection, inventors will have little incentive to invest in research and development\textsuperscript{117}; software patents have an informational and educational function\textsuperscript{118}; and if there is a need, software patents can be challenged and invalidated\textsuperscript{119}. Those who argue against the patentability of software have their own arguments: software simply is an algorithm or a mathematical formula\textsuperscript{120}; software is already covered by copyright and so patent protection is unnecessary\textsuperscript{121}; it is also argued that many software patents cover trivial, obvious developments\textsuperscript{122}; the legal system favours large corporations, which can better afford to obtain and enforce patents\textsuperscript{123}; and advocates of open source software are concerned that the open source movement may be destroyed by patent claims against open source code\textsuperscript{124}.

From the perspective of patent law, software is problematic in the sense that software is inherently more abstract than other technologies, and at least since the 18\textsuperscript{th} century patent law has had difficulty dealing with patents that claimed abstract ideas or

\textsuperscript{116} Software, at least in object code form, is written for the purpose of performing functional tasks. Therefore, software is a functional work of technology. Also, internationally, the WTO has created an agreement on TRIPS that is generally interpreted to allow software patents. See Nichols (1998), p. 104; Choudhary (2011), p. 445; Zeidman (2011), p. 92.

\textsuperscript{117} Inventors who spend mental effort coming up with something new and useful are allowed to protect their intellectual property. This protection should apply to software inventors just as it does to other kinds of inventors. See Nichols (1998), p. 104; Choudhary (2011), p. 445; Zeidman (2011), p. 92.

\textsuperscript{118} Software patents require that software be made public, allowing for a wider distribution than would otherwise occur and allowing other programmers to build on the concepts. Although, as pointed out by Choudhary, “Competitors are not able to use a patented invention for a limited time period, they often benefit from the disclosed information and find new ways to improve their products”. See Choudhary (2011), p. 445; Zeidman (2011), p. 91.

\textsuperscript{119} So there exist mechanisms for revoking patents that are found to be obvious or were in use before the patent was granted. Many of the arguments against software patents focus on some notoriously bad software patents, but there have always been bad patents and there have always been ways to challenge them. See Zeidman (2011), p. 92.

\textsuperscript{120} This is where those who are against software patents make their best point. Granting patents for mathematical algorithms would preclude others from performing the same process. So like mathematical algorithms, computer programs should not be patented. However, software does not consist simply of ideas; they are specific implementations of those ideas in a manner that a physical machine – a computer – can execute. They are also not simply mathematical expressions, but a series of steps that include mathematical expressions. Steps of a method have always been patentable. See Choudhary (2011), p. 446; Zeidman (2011), p. 94.

\textsuperscript{121} It has been claimed that software’s unique character is made apparent by its copyrightability, which does not apply to other kinds of inventions. In addition, it has been proposed that software is fundamentally different and should therefore be protected by a special form of intellectual property law, i.e. a \textit{sui generis} protection. Nichols (1998), p. 104; Zeidman (2011), p. 93.

\textsuperscript{122} According to Choudhary, “It is claimed that in practice, numerous patents have been granted to trivial inventions, i.e. inventions which are not non-obvious. This may be so partly because of patent offices’ inefficiency and partly because of insufficient, unclear and inappropriate rules. As a result, trivial patents not only decrease general patent quality but also impede improvements in the sector when they are used against second inventors, especially SMEs and open source developers.” See Choudhary (2011), p. 445. Cf. Zeidman (2011), p. 93.

\textsuperscript{123} Nichols (1998), p. 104.

\textsuperscript{124} Zeidman (2011), pp. 93-94.
principles. In essence, the abstract nature of software means that software technologies (e.g. algorithms, system structures) can be formulated and represented in multiple different ways, so that the technology claimed in a software patent can be rather difficult to distinguish from alternatives. Therefore, it might be extremely difficult to know whether or not a given patent claims an invention that is different from previous inventions, or whether or not an allegedly infringing program is different from the claimed technology. It should be noted that although not all software patents contain abstract claims, patents dealing with software-related inventions facilitate writing abstract claims.\footnote{Bessen & Meurer (2008), pp. 23 and 187.}

The abstract nature of software and thus abstract claims cause several major problems. Firstly, abstraction leads to technologies being claimed for what are often not yet known to the inventor at the time of the application. As a consequence, very broad patents might be issued and patentees be rewarded for things they did not invent. Secondly, the abstract nature of software makes it challenging for patent examiners to judge the sufficiency of disclosure of the claimed inventions and thus to decide whether or not enough information has been disclosed to properly support the applications. Finally, such a configuration makes it hard for patent officers to draw the boundary between allegedly novel and inventive inventions, and prior art. In fact, the abstract nature of software patent claims leads to the issuing of patents that do not possess clear boundaries and, thus, gives rise to high risks of infringement and opportunistic litigation.\footnote{Ballardini (2009), p. 213; Bessen & Meurer (2008), p. 199.}

Patents have now become relatively common in the ICT and software industry, and due to advances in software technology and the expansion of the Internet, the number of patent applications in the area of software-related inventions has increased rapidly over the last two decades.\footnote{Merges (2007), p. 1641; Choudhary (2011), p. 435.} Existing research on software patents argues that this increase indicates that holding patents on software has become both easier and more valuable to firms.\footnote{See e.g. Hall & MacGarvie (2006), p. 4.} However, the expansion of the number of issued patents in the area of software-related inventions, together with the lack of relevant prior art in the field\footnote{This is discussed more in detail in chapters 3.2.3 and 5.2.2.} and
the special nature of software as an abstract technology, has not only led to obvious and vague patents being issued, but also to patent thickets\textsuperscript{130}. According to Bessen and Meurer, despite initial beliefs the software industry has not become dominated by large companies nor has it shown signs that the entry of SMEs has diminished.\textsuperscript{131} To review, patents have now become common in the ICT and software industry, and despite earlier predictions, these industries have not slowed to a crawl.\textsuperscript{132} Rather these industries are currently characterized by rapid innovation and strong competition, which keep the software market vibrant.

\textsuperscript{130} This horizontal and dense overlapping of multiple patent claims is termed the patent thicket. Carl Shapiro has defined patent thickets as follows: “a dense web of overlapping intellectual property rights that a company must hack its way through in order to actually commercialize new technology.” According to Ballardini, “software patent thickets are mainly caused by two sets of problems: overlapping problems, related to the quality of the patents, on the one hand; and problems related to the vast number of patents issued on the other.” See Shapiro (2001), p. 120 and Ballardini (2009), p. 208.

\textsuperscript{131} Bessen & Meurer (2008), p. 190.

\textsuperscript{132} Merges (2007), p. 1641.
3 Software Patents in Europe

3.1 European Patent System

3.1.1 The European Patent Convention and the European Patent Office

In Europe computer program patents have truly been part of legal and political discussions ever since negotiations of the European Patent Convention (the EPC) started in the middle of the 1960s. The EPC was signed in 1973 as a special agreement under the Paris Convention, and it can be characterized as an intergovernmental treaty that is distinct from the European Union (the EU). The EPC is an instrument creating the European Patent Organisation, in other words an autonomous legal order with its own jurisdiction through the European Patent Office (the EPO) and governance through the Administrative Council. In legal terms, the EPC is part of the acquis communautaire, meaning that a country which commits to joining the EU is obligated in due course to ratify the EPC and hence become a full member of the European Patent Organisation. However, the reverse does not apply – joining the Organisation presupposes no commitment to joining the EU.

The European Patent Organisation has currently 38 member states, comprising all the member states of the EU together with a few non-EU countries. The Organisation has its own legal personality, and it is represented by the President of the EPO. The EPO started its operation in 1977 and began to take patent applications in June 1978. Though there were initial worries that it might be an unsuccessful creation, it became highly successful, undermining the workload of the national patent offices of member states. Actually, the EPO and the EPC were meant to be only temporary creations, and to be

---

134 It is commonly held by opponents of software protection that the EPO is a creature of the Commission, but this is not true, because any coordination between the European Patent Organisation and the European Union is on an entirely voluntary basis. In fact, the EU has very limited input to the operation and control of the EPO, merely being an observer at the Administrative Council along with WIPO and other organizations. See Sherman (2010), p. 39; MacQueen et al. (2010), p. 382; Leith (2011), p. 72.
135 The EPO comprises the Examining Division, the Technical Board of Appeal, and the Enlarged Board of Appeal.
136 “The Administrative Council of the EPO, made up of representatives of the contracting states, exercises legislative powers on behalf of the Organisation, is responsible for policy issues relating to the Organisation and supervises the Office’s activities.” See EPO Legal Foundations.
137 See e.g. Grosche (2006), p. 262.
139 At the time of writing 9.9.2013.
later subsumed within the EC’s (now the EU’s) Community Patent project, but the continuing failure to resolve the issues of translation costs and litigation forum allowed the EPO to become a large and powerful organization with its own view of where it stands as an independent agent in the world’s order of patent organizations.\textsuperscript{141} Furthermore, today the EPO is recognized as one of the most influential patent offices in the world, alongside the US and Japanese offices.\textsuperscript{142}

In general, the aim of the EPC is to make the protection of inventions easier, more reliable, and less expensive in member states. When the EPC was being formulated, it was decided that for there to be an effective single granting process, it was necessary for member states to harmonize the basic rules of patent law – the EPC harmonized the law on patentability and exclusions. However, national patent laws are not fully harmonized. Therefore, national laws and their interpretations continue to differ in Europe.\textsuperscript{143} Even though national patent laws are not fully harmonized, the laws of member states echo the EPC’s substantive requirements of patentability and the standards developed by the EPO\textsuperscript{144}.\textsuperscript{145} Therefore, it can be argued that even though the main purpose of the convention is to simplify the application system, the EPC and the EPO also largely determine the conditions and scope of patentability, or at least have a very strong influence on national patentability requirements.

The EPO is primarily concerned with the granting of patents, but the EPO also administers the payment of various fees from patent applicants. Moreover, as the patent granting authority of Europe, the EPO’s \textit{raison d’être} is basically to examine patents on behalf of all member states on the basis of a centralized procedure and to function as an executive arm of the European Patent Organisation.\textsuperscript{146} Put briefly, applicants may seek patents either through the national patent offices (based on partially harmonized

\textsuperscript{141} Leith (2011), p. 72.
\textsuperscript{143} See e.g. Mylly U.-M. (2011), p. 450.
\textsuperscript{144} The decisions of the Boards of Appeal at the EPO are of persuasive authority only and thus they are not binding on the courts in states which are parties to the EPC but, though, there is a tendency to follow them where at all possible. This is “because they are decisions of expert courts (the Boards of Appeal and Enlarged Board of Appeal of the EPO) involved daily in the administration of the EPC and secondly, because it would be highly undesirable for the provisions of the EPC to be construed differently in the EPO from the way they are interpreted in the national courts of a Contracting State”. More closely, see Attridge (2001), pp. 39-40; Bainbridge (2008), p. 285; Pila (2011), p. 204.
national legislation) or through the EPO (based on the EPC). However, in both cases the validity of patents and infringements thereof are evaluated on the basis of national legislation. In other words, even though there is a centralized examination and opposition procedure, according to Article 64(3) of the EPC, infringement of European patents are to be dealt with by national law. Thus, post-grant issues of validity and infringement remain the exclusive jurisdiction of national courts and the EPO cannot deal with these questions\textsuperscript{147} \textsuperscript{148}.

The EPC has continually been amended; to make life easier for applicants but also to adjust to the development within European states and industry.\textsuperscript{149} This reflects also on the EPO’s mission, which is to support and promote active innovation, competitiveness and economic growth across Europe through a commitment to high quality and efficient services delivered under the provisions of the EPC.\textsuperscript{150} The most notable revision of the EPC took place in December 2007 when the 1973 EPC was replaced by the EPC 2000\textsuperscript{151}. The EPC 2000 did not bring any large-scale changes in substantive patent law, except some changes concerning patentability requirements (novelty and industrial applicability) and priority rights. However, the EPC 2000 introduced a significant number of smaller-scale amendments, and therefore it can be held as a comprehensive revision.\textsuperscript{152}

3.1.2 European Patent

The EPC provides a standardized legal framework for the granting of patents by way of a single homogenized process before the EPO. Furthermore, an inventor or patentee seeking patent protection in Europe may file an application with the EPO, which then will examine the application under the terms of the EPC. This centralized procedure may cover some or all contracting states of the European Patent Organisation, but typically the number of countries which are designated by the applicant varies between

\textsuperscript{147} Post-grant oppositions to a patent can be brought before the EPO, and a European patent may only be revoked on grounds of Article 138 of the EPC. Although post-grant opposition proceedings take place within the EPO, enforcement and other judicial actions must still be carried out within the national court systems. See e.g. Burgunder (2011), p. 144.
\textsuperscript{149} Nordkvist & Kalo (2010), p. 12.
\textsuperscript{151} Act Revising the Convention on the Grant of European Patents, Munich, 29 November 2000.
\textsuperscript{152} Sherman (2010), p. 40.
three and ten. European patent applications can be filed by any natural or legal person, or anybody equivalent to a legal person, irrespective of nationality and place of residence or business. There is no obligation to be represented by a professional representative at this stage. The application for a European patent must contain as a minimum: a request for the grant of a European patent; a description of the invention; one or more claims; any drawings referred to in the description or the claims; and an abstract. Furthermore, the application must satisfy the formal requirements laid down in the Implementing Regulations. The inventor must be designated in the application.

The most important elements in the application are the claims. Their purpose is to define the subject-matter for which protection is sought in terms of the technical features of the invention; they must therefore be clear and concise and be supported by the description. Unlike trade mark and copyright law where the protected subject-matter is determined objectively, the invention is protected by a patent only in the way and to the extent as it is claimed. Drafting the claims is therefore a kind of art: they must be broad enough to cover the invention to its fullest; on the other hand, if they are too broad, the patent is likely to evoke objections and will have to be limited or eventually even fail for lack of novelty or inventive step. Moreover, the information contained in the application must disclose the invention (i.e. indicate the technical problem that the invention is designed to solve and describe its proposed technical solution) in a manner sufficiently clear and complete for it to be carried out by a person skilled in the art.

The application to the EPO shall be filed in one of the official languages of the EPO, which are English, French and German. This language is then used as the language of proceedings throughout the granting stages. However, filing in another language is also possible, but a translation has then to be filed in one of the official languages of the EPO within two months of filing the application. In essence, this means that an application can be made faster and easier, but a translation is still required.

---

154 Article 58 of the EPC. See also Kur & Dreier (2013), p. 98.
155 However, apart from filing the application, according to Article 133 of the EPC, “a person not having either their residence or place of business within the territory of one of the EPC Contracting States must be represented by a professional representative (a European Patent Attorney) and act through him in all proceedings”. See also Kur & Dreier (2013), p. 98.
proceedings, any party may use any of the aforementioned official languages. The official language in which the application is filed or translated into is generally used as the language of proceedings in all proceedings before the EPO. While translations and issues connected therewith continue to account for a substantial part of the problems encumbering the European patent system, it is hoped that they will be eased considerably with the assistance of automatic translation programs that are currently in their test phase.

A patent granted by the EPO is called a European patent, but the term “European patent”, however, might be misleading to a certain extent as it gives the impression that an office grants a patent for a given jurisdiction (for instance the whole European area) and thus the granted patent is as such enforceable in the said jurisdiction. Therefore, despite the name, a European patent has, in every member state where it is granted, the same effect as and be subject to the same conditions as a national patent granted in that state. Moreover, it is not yet possible to receive one EU-wide patent. In sum, once a European patent has been granted its European character evaporates as its “unitary form” ends and it must be validated in the individual countries in which the patent protection is to be upheld. In essence, this means that after the grant a European patent splits into national patents.

Since a European patent is a bundle of national patents, or “semi-national” as Norrgård calls them, all infringements shall be dealt with by national law in respective state, and this is and has been one of the biggest issues regarding European patents. It is the national enforceability of patents that makes the European patent system fragmented. There is no European court that can deal with patent infringement, as of yet, and there is

160 Agreements concerning mutual recognition of machine translated documents have been signed between the EPO and the Patent Offices of Japan, South Korea, and most recently Russia. See Kur & Dreier (2013), pp. 99-100.
161 Article 2(2) of the EPC. See also Burgunder (2011), p. 144.
162 As pointed out in chapter 1.2, European patents with unitary effect are not discussed in this thesis.
164 Norrgård has argued that European patents cannot be regarded as purely national patents since the interpretation of national patents are provided for in national patent laws whereas the interpretation of European patents is provided for in Article 69 of the EPC. See more closely Norrgård (2009b), pp. 219 and 227.
no harmonization with regards to litigation procedures\textsuperscript{165}. Each country has its own case law and its own legal system, which in some cases is based on common law (as in the United Kingdom) and in others on civil law (as in Finland).\textsuperscript{166} A result of this fact is that it makes conflicting applications of the law inevitable\textsuperscript{167}, both between national courts \textit{inter se}, and between those courts and the EPO.\textsuperscript{168} This creates legal uncertainty when it comes to predicting the outcome of a conflict regarding a European patent.

\section*{3.2 European Patent Law on Software}

\subsection*{3.2.1 Article 52 and Its Exception Regarding Computer Programs}

To understand the application of patent law to software, it is necessary to understand the state of patent law as it existed when software came into being. During the time the EPC was drafted, the software industry was still in its infancy. The negotiations around the EPC had included discussions of what should constitute a patentable invention, and much time was taken up debating the need for, and the terms of, the exclusions from patentability. However, computer programs received no mention whatsoever in the outcome of the first round negotiations of the EPC. The first mention of computer programs can be found from the drafts of the 1971 convention, and the view then was to include computer programs within the category of non-patentable subject-matter.\textsuperscript{169} In the end, the exception that covers computer programs was negotiated until the signing of the convention in 1973.\textsuperscript{170} When the final version of the EPC was adopted in 1973 the specific prohibition on patenting computer programs had found its way into the instrument, the result being Article 52 and its list of exceptions\textsuperscript{171}, which currently stands as follows:

\textsuperscript{165} Also Fisher has pointed out, that without a common appeal court to act as final arbiter in matters relating to the determination of the breadth of protection, a history of wildly differing interpretative styles has rendered any harmonization on this ground incomplete. See Fisher (2007), p. 223.
\textsuperscript{166} Manderieux (2007), p. 8.
\textsuperscript{167} In other words, where a patent might be held valid in one contracting state and thus infringed, but held invalid in another contracting state and thereby not infringed even though both situations relate to the very same invention.
\textsuperscript{168} Pila (2011), pp. 203-204.
\textsuperscript{169} Ballardini (2008), p. 564; Välimäki (2009), pp. 80-81; MacQueen et al. (2010), p. 535.
\textsuperscript{170} MacQueen et al. (2010), p. 535.
\textsuperscript{171} A significant element in deciding how to handle software in the EPC was the Patent Co-operation Treaty (PCT). This was already in existence as an international searching mechanism and some of the offices which were already examining under PCT may thus have felt confident in following this approach. See Leith (2011), p. 22.
**Art. 52(1)**  
European patents shall be granted for any inventions, in all fields of technology[^172], provided that they are new, involve an inventive step and are susceptible of industrial application.

**Art. 52(2)**  
The following in particular shall not be regarded as inventions within the meaning of paragraph 1:  
(a) discoveries, scientific theories and mathematical methods;  
(b) aesthetic creations;  
(c) schemes, rules and methods for performing mental acts, playing games or doing business, and **programs for computers**[^173];  
(d) presentations of information.

**Art. 52(3)**  
Paragraph 2 shall exclude the patentability of the subject-matter or activities referred to therein only to the extent to which a European patent application or European patent relates to such subject-matter or activities as such.

Thus, to be patented, whether the question is about software or not, there must be 1) an invention, implicating **technical character** of claimed subject-matter and it must be 2) **new** (novelty), 3) **inventive** (inventive step) and 4) **industrially applicable**. Whereas novelty and inventive step are relative requirements, technical character of the claimed subject-matter and industrial application are absolute requirements[^174]. Whilst the EPC sets out requirements for novelty, inventive step and industrial application in some detail (see Articles 54, 56 and 57 of the EPC), it does not contain a legal definition of the term invention. However, in accordance with Rules 27 and 29 of the EPC, in order to be patentable, “an invention must be of a technical character to the extent that it must relate to a technical field, must be concerned with a technical problem and must have technical features in terms of which the matter for which protection is sought can be defined in the patent claim”. Also, since the early days of the patent system it has been part of the European legal tradition that patent protection should be reserved for technical creations[^175].

[^172]: As it can be noticed, the current version of the EPC includes the words “in all fields of technology”, which shows a leaning towards patenting software inventions. The phrase was inserted at the EPC 2000 Revision Conference to reflect the wording of Article 27.1 of TRIPS. See e.g. Bainbridge (2008), p. 291.

[^173]: When the EPC 2000 was discussed, some members requested the removal of computer programs from the list of non-patentable inventions in Article 52. However, the initiative did not succeed and it was decided to leave Article 52 of the EPC unchanged. See Mylly U.-M. (2011), p. 451; Nilsson (2012), p. 85.

[^174]: See e.g. Mylly T. (2009), pp. 278-279.

[^175]: EPO Patents for software, p. 14.
Although, in paragraph 2 of Article 52 of the EPC it is stated that programs for computers are to be excluded from patentability, before reaching the conclusion that all programs for computers are excluded from patentability, we have to read on to paragraph 3 of Article 52 of the EPC, which clearly limits the scope of exclusions. Furthermore, according to Article 52(3), computer programs as such are not regarded as inventions within the meaning of the EPC. It is this phrase – as such – that has caused much consternation, especially in the context of computer programs. One view of Article 52(3) is that it does not read as a positive exclusion clause but rather as a negative definition of what constitutes an invention.\footnote{Dutfield & Suthersanen (2008), p. 116.} Moreover, it has been said that Article 52’s exclusions are to be interpreted narrowly.\footnote{According to Rossi, “the EPO has shown a tendency to interpret the Art. 52 sections (2) and (3) computer program exception more and more narrowly over the past fifteen years, relying on a progressively more lenient understanding of the criterion of technical character”. See Rossi (2005), p. 8.} It has also been argued that the phrase “as such” was intentionally left open for various interpretations.\footnote{Stratford (2009), p. 350.}

According to Casucci, various reasons exist for the exclusion laid down in Article 52. In general, it was argued that computer programs are intellectual and abstract products. In particular, the objective difficulty in examining software claims was raised.\footnote{Casucci (2005), p. 166.} Nevertheless, the starting point, after the adoption of the EPC in the EPO’s first guidelines\footnote{The EPC Implementing Regulations and the EPO Guidelines for examination also play a fundamental role in the application and interpretation of the relevant articles and rules of the EPC. See Ballardini (2008), p. 564.} of 1978, was that the provision excludes all software from patentability.\footnote{Grosche (2006), p. 270.} Thus, for some years following implementation of the EPC, software in isolation was non-patentable, and when assessing the patentability of a hardware-software combination the software element was ignored\footnote{This was because at that time computer software was considered an independent component, easily separable from the hardware itself. See e.g. Williams (2004), p. 368 and Ballardini (2008), p. 564.}.\footnote{Williams (2004), p. 368; Ballardini (2008), p. 564.} However, due to technological progress and evolution, the EPO’s view on the patentability of software has gradually become more and more liberal, and since the middle of 1980’s\footnote{See especially cases T 208-84 and T 26/86.} the Boards of Appeal of the EPO have steadily developed a broader interpretation of Article 52.\footnote{Koo (2002), p. 185; Ballardini (2008), p. 564. See also Guellec & van Pottelsberghe de la Potterie (2007), p. 120.} Similarly,
Deschamps has stated that the EPO has interpreted Article 52 of the EPC in a very generous manner to enable the patentability of software-related inventions in Europe.\footnote{Deschamps (2011), p. 109. See also Leith (2011), p. 23.}

Actually, the EPO has now promoted the view that the widest possible conception of patentability was a predominant conception when the EPC was negotiated.\footnote{See e.g. Mylly T. (2009), p. 278.} According to Mylly, due to case law by the Boards of Appeal,\footnote{This is due to the fact that, as stated in the opinion of the Enlarged Board of Appeal G 3/08 of 12 May 2010, “A claim in the area of computer programs can avoid exclusion under Articles 52 (2) (c) and (3) EPC merely by explicitly mentioning the use of a computer or a computer-readable storage medium”. See Kur & Dreier (2013), p. 140 and Opinion on the Referral G03/08 on May 12, 2010.} exclusion of computer programs as such in Article 52 of the EPC no longer has any practical significance.\footnote{Mylly U.-M. (2011), p. 454. However, the Enlarged Board of Appeal of the EPO assumes that the list of exclusions has a role in determining the inventive step and hence has not fully lost its role. In practice however, if computer programs generally have technical character, they often avoid the exclusion. Therefore, the exclusion “computer program as such” cannot have an important role in inventive step analysis either. For further details, see Mylly U.-M. (2011), p. 455.} It can even be said that the initial exclusion of software from patentability has been demolished, and despite the fact that Article 52 of the EPC clearly states that computer programs as such are not patentable subject-matter; it is evident that this criterion is no longer an obstacle for software to be patentable. Since the exclusion of software or computer programs as such has become redundant, it is now crucial to discuss how other patentability criteria are to be interpreted.\footnote{Mylly U.-M. (2011), pp. 451-454.} However, before we proceed into discussing patentability criteria in greater detail, a brief summary about the failed commission proposal for a CII directive will be presented.

### 3.2.2 The Failed Commission Proposal for a CII Directive

The granting of patents for software-related inventions has long been a matter of some controversy. One example of this was the rejection by the European Parliament of the proposed Directive on the patentability of computer-implemented inventions (CII)\footnote{Proposal for a Directive of the European Parliament and of the Council on the patentability of computer-implemented inventions, COM(2002) 92 final.} on 6 July 2005.\footnote{Bainbridge (2010), p. 450.} Legally speaking, the rejected CII directive was an attempt to clarify the issue of software patents in EU member states.\footnote{Dutfield & Suthersanen (2008), p. 264.} The goal of the rejected directive was to provide legal certainty to potential patentees by resolving legal ambiguities
concerning the ambit of Article 52 – that is, what exactly is the difference between “computer programs as such” and “computer-implemented inventions”. Furthermore, related to the objective of legal certainty within Europe was the need to harmonize the divergent approaches to the issue of CII patenting between the national patent offices vis-à-vis the EPO. The situation was deemed to be unsatisfactory as it distorted the internal market.¹⁹⁴

However, the proposed CII directive was met with fierce criticism by opponents of patentability in the field of software, in particular from open source communities. The main argument was that patents granted in the software field might lead to a patent thicket which would make innovation either impossible or at least only feasible for big enterprises, which are able to dispose resources to research prior patents and negotiate licensing agreements, but not for SMEs. In addition, it was feared that those firms which market their products on a proprietary basis might use patents in order to block free and open source software.¹⁹⁵ The counter-argument raised by proponents of patentability that patents are often the only weapon of defence for small software producing companies against hostile actions and takeover bids by bigger corporations, was generally discarded by the critics of an all too far-reaching patentability. Also, critics pointed out that many patents that have been granted should probably not have been, because they were trivial.¹⁹⁶

Ultimately, intensive protests and lobbying activities against the CII directive were successful, and the CII directive was rejected by the Parliament at its second reading on 6 July 2005. Later on, the European Commission has confirmed it will not draw up another version of the CII directive.¹⁹⁷ As a curiosity it can be mentioned that, had the proposal been accepted, it could have made it easier to obtain patents for certain software-related inventions, particularly where the technical contribution was itself something excluded from the grant of a patent, per se, such as a method of doing business.¹⁹⁸ In addition, a directive binding merely for EU countries would have solved the problem of a fragmented system only partially, leaving the practice of both EPO and

non-EU EPC countries unchanged.\textsuperscript{199} Even though the CII directive was rejected, the decision in no way altered the practice of granting patents for software-related inventions neither by national patent offices of EU member states nor by the EPO. Moreover, because no EU law exists, there is no judicial control regarding the standard for patenting software-related inventions by the Court of Justice of the European Union (the CJEU) and consequently different interpretations as to what is patentable continue among member states.\textsuperscript{200}

\subsection*{3.2.3 Industrial Application and Novelty}

Despite the exclusion in Article 52(2), computer programs and software-related inventions are now relatively widely patentable under the EPC framework, provided that statutory requirements for patentability under the EPC are otherwise fulfilled. Therefore, requirements of technicality (Article 52(1), novelty (Article 54), inventive step (Article 56), industrial application (Article 57), sufficient disclosure (Article 83) and clarity (Article 84) are the primary tools for evaluating patentability.\textsuperscript{201} If a patent has been granted in breach of any of these criteria it may subsequently be revoked. However, even though it can be said that the EPO has become more liberal in granting patents for software innovations during recent years, it is still likely that the examination process will remain rigorous.\textsuperscript{202} In this chapter, first, a closer look will be taken at the requirement of industrial application and subsequently the issue of novelty will be discussed.

For the satisfaction of the requirement of industrial application, there must be at least one field of industrial application for the direct use of the claimed invention. The claimed invention must be concretely industrially applicable. Mere speculation with exceptional or theoretical industrial uses or possible future developments is not enough. However, in practice this requirement is the easiest to satisfy, and it has been characterized as a relatively undemanding criterion.\textsuperscript{203} Because a computer is a machine, all programs when run in a computer can be argued to be capable of prima

\begin{footnotesize}
\begin{itemize}
  \item[\textsuperscript{199}] Ballardini (2008), p. 574.
  \item[\textsuperscript{200}] Kur & Dreier (2013), pp. 143-144.
  \item[\textsuperscript{201}] See e.g. Honkasalo (2012), p. 10.
  \item[\textsuperscript{202}] Bainbridge (2008), p. 291.
\end{itemize}
\end{footnotesize}
industrial applicability criteria is less of a problem. In essence, as pointed out by Stratford, in the field of computer technology, susceptibility of industrial application is invariably given.

An invention must have novelty to qualify for a patent. Thus, in many cases, before even considering other issues relating to software patentability the first question that should be asked is: is the invention new? In other words, does the invention already exist and is it in the public domain. According to Article 54 of the EPC, “An invention shall be considered to be new if it does not form part of the state of the art”. When considering relevant prior art, all “matter” made available to the public before the date of filing of the European patent application must be taken into account. This should be interpreted literally: “the state of the art comprises everything made available to the public by means of a written or oral description, by use, or in any other way”. Consequently, there are no restrictions as to geographical location where, or language or manner in which the relevant information was made available to the public.

The novelty requirement applies quite strictly. The criterion of the invention having been “publicly available” does not mean that the public must actually have been aware of the disclosure, or that the invention was made known to a larger audience. For example, an oral presentation before a selected audience (which was not subject to a secrecy agreement) will be considered as novelty-destructive. The harshness of the test is mitigated to some extent by Article 55: “if the disclosure constitutes ‘evident abuse’ in relation to the applicant or his legal predecessor, it will be disregarded if the application is filed within six months following the publication”. Furthermore, according to Article 55(1)(a) and (b), (2), the same applies if “the invention has been

---

206 See e.g. MacQueen et al. (2010), p. 422.
207 This is also known as the requirement of absolute novelty.
208 Article 54(2) of the EPC. See also Oesch & Pihlajamaa (2008), p. 79. In addition, one should keep in mind that the evaluation of novelty happens from the objective point of view, here the point of view of an examiner, and no relevance is given to the subjective point of view of the applicant.
209 This is quite problematic, because the amount of “potential novelty destroying matter” available nowadays is immense, and therefore it is almost unrealistic to expect that every single published piece of information could be taken into account when a patent is examined and issued. A consequence of this is that more and more novelty destroying facts and material are found after a patent has been issued, and then the issued patent can potentially be revoked.
displayed at an official or officially recognised international exhibition within the terms of the Convention on international exhibitions of 1928 in conformity with the relevant formal requirements". In addition, it should be noted that it is not acceptable to combine separate items of prior art together. In other words, several disclosures may not be “mosaicked” as they can when obviousness (see chapter 3.2.4 Inventive Step) is at issue.

It has been said that software technology is an area in which it is particularly cumbersome for a patent examiner to locate relevant prior art. This problem is mainly due to the abstract nature of software and there not being a judicial requirement to disclose source code. In addition, sources of prior art in the case of software are often scattered, sometimes unavailable, or simply insufficient, and therefore they present considerable difficulties to the searcher locating relevant prior art. These issues are discussed more extensively in chapter 5.2.2 in the context of the search for prior art. The problem of locating prior art, however, is not the only, and perhaps not even the most difficult, challenge in assessing novelty in the case of a software-related invention. Namely, due to the abstract nature of software it is challenging to conclude whether a software-related invention is really novel or just another way of expressing the same idea that many software developers have utilized before. These issues have raised concerns that software patents are granted to trivial inventions which are of low quality. Furthermore, the problems presented above mean that many patents for software-related inventions are very vulnerable to challenges for lack of novelty.

3.2.4 Inventive Step

Article 56 of the EPC states that “an invention shall be considered as involving inventive step if, having regard to the state of the art, it is not obvious to a person

---

211 See also Kur & Dreier (2013), p. 112.
214 See chapter 2.3.
215 Honkasalo (2012), p. 18. See also chapter 2.3.
218 Leith criticizes the term inventive step of being somewhat inaccurate: “The conception of step tends towards suggesting that we have an equal riser for all inventions and that this is somehow a given which is relatively easy for examiner to follow. This is not the case: we would technically be better to talk about
skilled in the art”. A person skilled in the art is presumed to be “a skilled practitioner in the relevant field of technology, who is possessed of average knowledge and ability and is aware of what was common general knowledge in the art at the relevant date”, yet they have no imagination. The skilled person’s actions are always purposeful, with expected consequences. According to the Guidelines for Examination, “He should also be presumed to have had access to everything in the state of the art, and to have had at his disposal the means and capacity for routine work and experimentation which are normal for the field of technology in question”. Contrary to novelty where the claimed invention is compared in its entirety with the previous state of the art, a mosaic approach is accepted. In addition, the skilled person is involved in constant development in his technical field.

Furthermore, as stated in the Guidelines for Examination, “If the problem prompts the person skilled in the art to seek its solution in another technical field, the specialist in that field is the person qualified to solve the problem, and assessment of whether the solution involves an inventive step must therefore be based on that specialist’s knowledge and ability”. According to Mylly, from this analysis it is reasonable to define the person skilled in the art in the computer program field to refer to a specialist in programming. Moreover, a natural expectation is that a specialist in a programming art is knowledgeable at least in one programming language and not only someone having “general programming skills”. This person could also be assumed to consult specialists in other programming languages and therefore a person skilled in the art will cover all programming languages. However, it should be noted that in the context of the disclosure requirement the Guidelines for Examination state that “a person skilled in the art is a person who is not deemed to be a specialist in any specific programming language, but who does have general programming skills”. Furthermore, in the

---

**inventive height**, since this emphasizes that heights are as variable as lengths of string, whereas steps are usually not.” See Leith (2011), p. 186.

219 The evaluation of the skills (knowledge of prior art) happens at the moment of the priority date.


221 What is important here is that whether the skilled person in practice could have reached or come into the same solution, and what does not matter is whether in theory the skilled person could have ended up with the same solution. See Rahnasto (1996), p. 17.


223 Even though “mosaicking” is allowed, it is not generally accepted to combine prior art documents in a purposeful manner when its only goal is to raise the bar of inventive step.


Guidelines it is stated that “the skilled person has the same level of skill for assessing inventive step and sufficient disclosure”. This suggests that when assessing inventive step and the person skilled in the art in that context, the skilled person is not a specialist in one specific programming language, but only has general programming skills. Therefore, the current state of affairs is that the source code of a computer program is not required to be disclosed in the context of software-related inventions.

It should be noted that the skilled person’s expertise may also involve fields other than programming, depending on the technical scope of the invention. However, the skilled person’s expertise is not without restrictions. It should be emphasized that the skilled person is an expert in a technical field not in a field of business or administration. Thus, for example, they do not know how to run a business, and only elements of the solution falling within the competence of a technically skilled person (here: a programmer or computer scientist) can be taken into account. In addition, something that should be noted is that there may be instances where it is more appropriate to think in terms of a group of persons, e.g. a research or production team, rather than a single person. For instance, many computer programs are a result of teamwork.

In order to fulfil the inventive step requirement the technical contribution to the prior art must be non-obvious to a person skilled in the art. Thus, we come to the question of obviousness. According to the Guidelines for Examination, the term obvious means that which does not go beyond the normal progress of technology but merely follows plainly or logically from the prior art, i.e. something which does not involve the exercise of any

227 Mylly has criticized this approach as follows: “If we compare how a person skilled in the art is understood in other circumstances in the EPC, this kind of definition ‘not a specialist in any programming language’ for a person skilled in the programming art seems unexpected. It is established that a person skilled in the art for disclosure analysis is assumed to have the same qualifications as a person skilled in the art for inventive step analysis. In fact, in the inventive step analysis a person skilled in the art is understood as being a person specialized in the respective technological field.” See Mylly U.-M. (2011), pp. 464-465.
228 Stratford (2009), p. 352. Using the concept of the skilled person does not come without problems. For example, the concrete content of this concept varies depending on the technical field in question and what is the stage of development of the said field. Also, something that is problematic is the examiner’s perception of the level of a skilled person at the moment of the priority date. It can be distorted or the level is hard to estimate correctly because of the rapid pace of development of the field. See Uotila (2009), p. 79; Rahnasto (1996), pp. 71-75.
skill or ability beyond that to be expected of the person skilled in the art. At the EPO, the examination of inventive step, i.e. obviousness, is analysed using a method called the problem-and-solution approach, but an inventive step cannot be based on just any problem or solution. To ensure that patents are only granted for innovation in a technical field, the presence of an inventive step requires that both the problem and the solution are technical. Without either, the presence of an inventive step must be denied before the skilled person even becomes involved.

Also, the EPO has confirmed in its practice that only technical features can establish novelty and/or inventive step, and without a technical solution there is no inventive step, irrespective of the obviousness of the non-technical solutions. Thus, under European patent law, inventive step is assessed with reference to the “technical solution provided to a technical problem”, taking merely technical, that is non-commercial, features into consideration for non-obviousness purposes. However, non-technical aspects of an invention may appear in the formulation of the technical problem as a constraint that has to be met, but the same non-technical aspects are excluded from the assessment of the inventive step as they do not have a technical character. Ballardini sees this as problematic, and according to her, the assessment of the inventive step needs some refinement: the inventive step should not be assessed before both technical and non-technical features of the invention have been considered. This would promote a more coherent and transparent practice.

### 3.2.5 Technical Character

The requirement of technical character has been said to follow straight from the fundamental object of the patent system: boosting technical development. While

---

232 “In the problem-and-solution approach, there are three main stages: (i) determining the closest prior art, (ii) establishing the objective technical problem to be solved, and (iii) considering whether or not the claimed invention, starting from the closest prior art and the objective technical problem, would have been obvious to the skilled person.” See more closely EPO Guidelines (2012), Part G - Chapter VII-3.
235 EPO Patents for software, p. 19.
236 Ballardini (2010), p. 28.
237 It can be argued that the difference between the US and EU position when it comes to patenting software-related inventions is partly due to the fact that the European patent system requires an invention to be of a technical character. See Duffield & Suthersanen (2008), pp. 116-117 and Bakels (2008), p. 1.
programs for computers are included with the excluded items listed in Article 52(2), there is a widespread consensus that an application will not be excluded as non-patentable subject-matter as long as it involves an inventive technical contribution to the prior art. Even though technical character is de facto a non-statutory requirement for patentability and it owes its origin to an imaginative interpretation of Article 52 of the EPC, the need to demonstrate technical character has still become the single most important factor restricting the scope of exclusions laid down in Article 52(2). In this sense, the ability of the claimed software-related invention to produce a technical effect or to have a technical character is a sine qua non condition upon which the EPO’s case law has been based.

The European legal community has spent some 20 years attempting to pin down the meaning of technical effect and technical character in a legalistic definition and has – many would argue, to date – been unsuccessful. Indeed, the EPC does not provide clear-cut definitions of such terms. However, over the course of time the EPO has tried to specify the content of the terms mentioned above in its decisions, and it can be argued that the core essence of said terms have been, at least to a certain extent, demystified. Still to date, problems remain as to what actually amounts to a technical contribution and whether or not the subject-matter in question is technical, and this has at times led to arbitrary and contradictory decisions. One thing, however, is clear: the technical requirement has considerably evolved since it was first mentioned by the EPO’s Boards of Appeal in the VICOM case. Furthermore, during its history the Boards of Appeal have adopted three different approaches regarding the technical requirement: the “technical contribution approach”, the “technical effect approach”, and the “any hardware approach”. The history of the EPO’s interpretation regarding the

---

242 See e.g. Ballardini (2008), p. 565.
243 EPO Boards of Appeal Decision T 208/84 – dated 15 July 1986, Computer-related invention/VICOM.
244 In the VICOM case, patentability was claimed in respect of a computer-implemented method of processing digitally encoded images. Specifically, the computer program would first scan data representing an image, and then, using a mathematical process, amend the data so as to alter the image. See e.g. Deschamps (2011), p. 110.
development of technical character cannot be retraced here in detail, but the goal is to give an overview of the current situation.

According to Guellec and van Pottelsberghe de la Potterie the technical character can be defined as “teaching for technical action, solution to a technical problem”, and the term technical is interpreted as “having an effect beyond the normal physical interaction between a program and a computer”. Similarly, in light of the EPO’s legal praxis in order to be considered to have technical character, a computer program would have to demonstrate technical effects which go beyond inherent interactions between the computer program and the computer hardware. Furthermore, the EPO-practice demonstrates that normal physical modifications of the computer hardware, deriving from the execution of the instructions given by a program for the computer, cannot constitute the technical character required for avoiding the exclusion of those programs, because such effects are a common feature of all programs for computers.

In sum, if a computer program is capable of bringing about, when running on a computer, a further technical effect going beyond normal physical effects, it is not excluded from patentability. However, it is not only the further technical effect that can lend to technical character. Additionally, the requirement of technical character may be satisfied “if technical considerations, which must be reflected in the claimed subject-matter, are required to carry out the invention”. Should that be the case, a computer program can again be considered to be an invention within the meaning of Article 52(1), and eligible for protection. If the claimed subject-matter passes this prima facie test for

---

248 When executed, a computer program will interact on a technical level with the hardware of the computer and cause it to operate differently. Transistors will switch, electrons will flow and areas on a hard disk will be magnetized. These effects are undoubtedly technical, but they are inherent to all computer programs. See Stratford (2009), p. 351.
250 “This further technical effect which lends technical character to a computer program may be found e.g. in the control of an industrial process or in processing data which represent physical entities or in the internal functioning of the computer itself or its interfaces under the influence of the program and could, for example, affect the efficiency or security of a process, the management of computer resources required or the rate of data transfer in a communication link.” See EPO Guidelines (2012), Part G - Chapter II-4. These are mere examples and the clear message to patent attorneys who must draft claims for software-related inventions is to focus their attentions on the possible technical effects which can be brought about by the computer program, for it is in these that protection will be secured. More closely, see MacQueen et al. (2010), p. 539.
Even though the EPO in its earlier practice has been inconsistent in its approach to what in practice constitutes technical character, in its more recent practice the EPO has been fairly consistent for many years now. However, not so long ago, some applicants and other interested parties began to feel that the way software-related inventions were being treated was unclear, or even incorrect. Therefore, in 2008 the President of the EPO referred four questions of law (case G 3/08) to the Enlarged Board of Appeal for guidance, which handed down its opinion on the referral in 2010. The Board found that divergence in jurisprudence over time was a normal development in a changing world, and that the practice of the EPO was predictable and reliable in its results. The Board also made an extensive reference to decision T 154/04, which it considered a pragmatic and useful approach for assessing subject-matter consisting of a mix of features, some of which had technical character and some not.

T 154/04 summarizes also the approach that is nowadays taken by the EPO when examining software-related inventions, which means that in practice the any-hardware approach is still followed today. The approach, as it was adopted in T 154/04, takes into account the effect of the list of non-inventions in Article 52(2) of the EPC in determining whether or not claimed subject-matter is inventive, acknowledges the evolution of the jurisprudence of the EPO, and confirms its latest practice, according to which only technical features can establish novelty and/or inventive step. Consequently, following this decision, non-technical aspects of an invention may appear in the formulation of the technical problem as a constraint that has to be met, but the same non-technical aspects are excluded from assessment of the inventive step as they do not have a technical character. The assessment of the inventive step then only takes into account the contribution of the technical features and aspects of the invention, even if these features and aspects relate to “a computer” only.

---

254 EPO Enlarged Board of Appeal – Opinion on the Referral G 03/08 on May 12, 2010.
256 EPO Patents for software, p. 18.
257 EPO Patents for software, p. 19.
In sum, what has been emerging in the EPO is a shift of focus towards novelty and inventive step. The office is less concerned with excluded matter so long as technical character can be found. Indeed, the advice to EPO-examiners is now to move beyond a consideration of technical character and to proceed directly to consider questions of novelty and inventive step. The rationale is that in assessing inventive step the examiner must establish which technical problem has been solved by the invention. As discussed in the context of inventive step, if no technical problem can be found then the implication is that no technical character is present.258 Mylly has argued similarly, stating that the reason for the exclusion of computer programs from patentability should accordingly not be sought from the lack of technical character, but elsewhere. Therefore, technical character, according to Mylly, is not an adequate criterion for demarcating the patent eligible subject-matter from subject-matter which should not be patentable.259

It has been argued that because the Boards of Appeal have embraced various and inconsistent approaches regarding the patentability of software, it has led to a general lack of legal coherency in the field.260 However, as discussed, recently the President of the EPO referred questions on patentability of software patents to the Enlarged Board of Appeal, which rejected the referral as it determined there was no inconsistency in previous decisions that warranted consideration of the questions. Therefore, the procedure for granting of software patents at the EPO has remained unchanged for several years now, and according to Leaves many practitioners are pleased that the outcome of the referral has not changed what is currently a pragmatic and workable system.261 Furthermore, according to Leaves, the consistency of the EPO approach has enabled a degree of legal certainty to develop around software patents in Europe that allows practitioners to get a good idea of whether or not they might be able to obtain a patent by asking the question: “Is this a technical solution to a technical problem?”262

258 MacQueen et al. (2010), p. 557.
260 See e.g. Ballardini (2008), p. 567.
262 Ibid.
4 Strategies and Exploiting Software Patents

4.1 Introduction and Chapter’s Contents

Software products are today considered to be of the utmost importance in many European ICT and software companies. Furthermore, many companies now also acknowledge that applying for patent protection for software is a true possibility and they even might have a patent strategy in place. While in smaller companies software patents might primarily be thought of as an aspect of large, high-tech companies, in reality patents and different patent strategies can be used in a wide range of businesses. Therefore, and also due to the ever increasing competitive atmosphere, neither large companies nor SMEs can ignore threats – and opportunities – presented by software patents. In other words, software patents and patent exploitation strategies are not only for larger companies, and therefore also SMEs, even though there often are cost constraints, should acknowledge the various uses and exploitation possibilities that are affiliated with software patents.

Patents, like other business variables, present opportunity for profit as well as risk of loss; prudence requires that software companies strive to maximize the former and minimize the latter. Naturally, some software companies will find patents to be extremely valuable business assets, while others will find themselves the target of infringement suits. Therefore, to be successful in today’s competitive atmosphere, especially if the business or market sector where a company competes is “infested” with patents, it is often crucial for the company to develop a patent strategy which at least addresses such issues as how to acquire, manage, exploit and maintain a diverse portfolio of patents. It goes without saying that a well-considered patent strategy will more likely help a company to reach its goals than not having a patent strategy at all.

Chapter three discussed obtaining software patent protection in Europe under the EPC framework. This chapter emphasizes the fact that managing a company’s software patents is more than just knowing how to acquire and get a patent granted through a patent office. Namely, software patents are not worth much unless they are adequately

264 See e.g. Nichols (1998), p. 133.
exploited: “a basic problem for software firms at all stages is the sense that even with a patent it often is difficult for a firm to appropriate the value of its invention”. It has even been argued that because development is so fast-paced in the software industry and because the nature of technology is so poorly understood, companies do not recognize the value they could appropriate from software patents if they pursued them in an informed way. There are a number of ways software patents can be exploited in order to help a company to achieve its goals. Therefore, it is important to acknowledge the fact that software patents are much more than mere legal instruments. It is not uncommon for companies in ICT industries to patent for strategic reasons rather than to protect innovations. Actually, patents in general are more and more commonly used strategically, and especially software patents often seem to derive their value rather from strategic usage than directly from their technological value.

Due to the rise in the strategic importance of the role of patents in commerce, both companies and also scholars have started to pay a lot of attention to managing these kinds of rights more effectively. Therefore, it is not surprising that many ICT companies have started to harvest more patents for strategic reasons, say for instance either for blocking competitors from developing rival technologies, or to ensure themselves the freedom-to-operate without infringement or to aggressively impose their bargaining position by improving their leverage in negotiations. Moreover, due to this development, it is important to recognize and learn the different ways of utilizing and exploiting software patents in order to reap the benefits or to protect oneself against competitors. The starting point here is that examining the general patent exploitation strategies is a good way of learning and understanding the needs of a company and how the company can utilize patents to achieve its desired goals.

Accordingly, it is the overall goal of this chapter to have an overview of different IP strategies with a special focus on patent exploitation strategies. By understanding different exploitation strategies, ICT and software companies can enhance their

---

268 See Mann (2005), p. 979.
270 Furthermore, as Rentocchini has argued, “for hardware producers software patents are more likely to reflect strategic value rather than their technological value, and contrary to that, software patents are found to be technologically valuable for software firms”. See Rentocchini (2010), p. 6.
utilization possibilities regarding software patents. In addition, the reader is asked to pay special attention to where patent exploitation strategies are discussed in detail (chapters under 4.4), because these issues will be discussed also later in the context of patent infringement monitoring in chapter five. Furthermore, companies willing to utilize their software-related inventions to the fullest extent should not only understand different exploitation possibilities but also how these strategies are connected to IP strategy and overall business strategy. Therefore, the first goal of this chapter is to offer an overview of the fundamentals regarding IP strategy and management and how they relate to a company’s overall strategy and business model. This has been seen as necessary in order to provide a framework to understand the rest of the chapter.

4.2 IP Strategy and Management

4.2.1 Levels of IP Management

Even though innovative ideas are at the heart of most successful businesses, the value of intellectual property is often not adequately appreciated and its potential for providing opportunities for future profit is widely underestimated, especially by SMEs. However, when IP is both legally protected and effectively exploited, it can become a valuable business asset. The challenge is not just creating value from innovation, but capturing that value as well. According to Pisano and Teece, when making strategic decisions about how to capture value from innovation, companies should pay attention specifically to the IP environment and the architecture of the industry. Strategic utilization of IP assets and truly understanding the potential value embedded in them can substantially enhance the competitiveness of a company. Accordingly, for companies to succeed in the field of intellectual property they should at least know the basics of IP management and be aware of the way IP and IP laws affect them in their day to day decisions and practices.

The starting point, in the works of many scholars, is that the IP management skills in companies are hierarchical. According to Gollin, at the lowest level, level zero, in the

275 WIPO IP for Business, pp. 5 and 9.
277 See e.g. Davis & Harrison (2001), Gollin (2008) and Harrison & Sullivan (2012).
hierarchy is a company lacking any intellectual property strategy at all (non-strategy).278 Neither Davis & Harrison nor Harrison & Sullivan recognize level zero in their hierarchy.279 However, other levels (levels one to five) in the works of Davis & Harrison and Harrison & Sullivan are the same, even though named somewhat differently. Levels one to five (“defensive”, “cost center”, “profit center”, “integrated”, and “visionary”) were first outlined by Davis & Harrison in 2001. They refer to an IP value hierarchy, in the form of a pyramid, with level one at the bottom and level five at the apex. That model reflects the view that most organizations are managed at a very low level, with very few companies operating at the highest, visionary level280.

On the bottom of the IP value hierarchy – if Gollin’s level zero is not taken into account – is level one (defensive/defending position), on which patents are viewed as legal assets and defensive tools. The main idea, especially among technology companies, is to protect the core business and maintain freedom-to-operate by obtaining patents or even building a patent portfolio. Defensive companies also try to avoid mistakes leading to liability and subsequently to litigation. In addition, the goal is to have a good awareness of the company’s IP resources and to enforce IP defensively.281 At level two (cost control/manage costs) a company does what the level one company does, but also takes steps to prioritize its IP protection. Thus, companies focus more on ways to reduce costs associated with patents and ways to increase overall efficiency and effectiveness. In other words, informed cost-benefit analysis plays an important role as it helps the company to focus its resources on IP assets with the strongest relation to its core business activities.282

At level three (profit center/capture value) companies begin to look at patents as a business asset, rather than just a legal asset, which have the potential of bringing in additional revenues. Companies go beyond the cost control model, and learn how to extract and capture value directly from its IP portfolio. Thus, companies at this level turn their attention to more proactive strategies that can generate additional revenues

279 Gollin found that many companies have not even achieved Davis & Harrison’s level 1 (defensive approach), so he added a new level: level zero (a company that has no strategy at all). See Gollin (2008), p. 132. Cf. Davis & Harrison (2001), p. 12.
280 See e.g. Gollin (2008), p. 132.
while further continuing to cut down costs.\(^{283}\) On the fourth level (\textit{integrated/synthesize opportunities}), there is a thorough integration of IP strategy with the company’s overall strategy. In other words, IP strategy is integrated beyond its own department, out across the entire company. In essence, the company’s activities are integrated with those of other functions and embedded in the company’s day-to-day operations, procedures, and strategies. By understanding strategic implications of IP assets, companies are able to look beyond defence, costs and profits, and view IP as an integrated business asset that can be exploited in numerous ways.\(^{284}\)

At the highest level, level five (\textit{visionary/shape the future}), there is a visionary company with a skilled innovative chief and IP managers who can integrate the company’s goals with its innovation potential, and even use intellectual property and innovation to shape the society within which the company prospers. Thus, the company looks carefully into the future to discern likely trends in IP law and practice, and uses that prognosis to shape its overall strategy. The company anticipates technological revolutions and actively seeks to position the company as a leader in its field by acquiring or developing IP that will be necessary to protect the company’s margins and market share in the future. Therefore, at this level, the company must continually refine and update its IP strategy.\(^{285}\) For example, the strategies of companies that are based on strategic patenting and the amassing of a large patent portfolio may change significantly when new IP legislation is adopted or there are changes in the court system.

\subsection{4.2.2 IP Strategy in the Context of Overall Strategy and Business Model}

Although the patentability of different technologies might well affect the direction that a company takes, the usual approach is for an overall business strategy to be set first, for an IP strategy to follow that, and for a patent strategy to be devised to fit the IP strategy.\(^{286}\) Furthermore, according to Knight, a patent strategy needs to be consistent with the business strategy of the organization, and although typically the business

\begin{footnotesize}
\begin{enumerate}
\item[286] See e.g. Senior (2007), p. 156.
\end{enumerate}
\end{footnotesize}
strategy is developed first, if it is not clear in its focus it needs to be clarified either before (preferably) or during the development of a patent strategy.287 A company’s overall business strategy involves setting the overall direction of the company, its goals and arranging for the resources with which to achieve that strategy. An IP strategy is a part of a company’s overall strategy, and it usually aims to improve economic outcomes of investments made through innovations. A general starting point is that IP strategy should align with a company’s business model and other business objectives.288 Similarly, a company’s patent activities, which incorporate patent acquisition, licensing, infringement monitoring and enforcement procedures, among other things, should support and be in line with the company’s overall strategy and with its business model in terms of deciding how to prosper within the industry and to establish competitive advantage over its rivals.289

In order to be successful in any field of business it is essential to understand how and why the business model is an integral part of the company’s overall strategy.290 291 From the strategic point of view it can be said that strategy drives the business model, or alternatively from the business model point of view that a business model manifests a company’s strategic choices.292 Acknowledging the company’s business model or a combination of different models293 is a fundamental task for numerous reasons, but the most important and obvious reason being that the business model describes the ways of creating, delivering and capturing value – economic, social, or some other form of value.294 As a concept “business model” can be defined and categorized in many different ways, but from a practical point of view what really matters is the fact that inside a company there truly is a shared understanding of what the company’s business model is. Without such a shared understanding it is difficult to systematically challenge

291 However, it should be noted that the concepts of business model and strategy are not the same and there are no dominant definitions of a business model or strategy. According to Rajala, “Despite the terminological confusion among the strategy and business models, prior research has achieved consensus on the position of the business model as a conceptual and theoretical layer between business strategy and business processes.” See Rajala (2009a), p. 11.
293 It should be noted that a company may also combine several different models as part of its overall business strategy.
assumptions about one’s business model and accordingly to do necessary improvements or innovate successfully.\textsuperscript{295}

IP strategies are managed differently in business models of companies depending on the overall goals of these companies. Blaxill and Eckardt have divided companies using IP into four different business models: Shark, Target, Glass House, and Minnow. \textit{Shark} companies, sometimes also referred to as trolls, do not usually produce products and therefore they can also be called non-practicing-entities (NPEs).\textsuperscript{296} They license and sell their patent rights solely for the income to those (target companies) who really need them in order to produce products or offer their services. \textit{Target} companies have fairly low IP activity as they usually acquire only necessary rights so that they can produce their products. However, they have a big part of the available market share. Companies in the \textit{Glass House} category form the biggest category. Their goal is to hold strong patent positions, and simultaneously generate big market shares. IP is used as a supporting asset for their manufacturing and selling activities. It is not unusual, sometimes it is even compulsory, for companies in this category to get licenses or be part of cross-licensing activities so that they can continue their manufacturing processes. \textit{Minnows} are new and small-market companies that are in the beginning of their life cycle. Although they have modest product sales and new-technology portfolios, they have potential to develop into any of the three aforementioned business models.\textsuperscript{297}

Even though a company has an IP strategy in place and it has been acknowledged in the company’s business model, a fact of life is that only a few companies develop such an IP strategy which is truly business driven and responsive to changing circumstances.\textsuperscript{298} However, if a company’s IP strategy is formulated, refined and executed effectively, it can be a powerful tool to help the company to achieve its overall goals.\textsuperscript{299} Furthermore, if patents are an important or crucial asset for a business, then they should be a key part of the business plan and overall strategy, too. Obviously, patent strategies, like IP strategy and management in general, are company specific and their significance to

\textsuperscript{295} Osterwalder & Pigneur (2010), p. 15.
\textsuperscript{296} NPEs are typically large, well-funded entities, usually corporations or other business entities, that own patents which they attempt to license to corporations that produce products that infringe on those patents. If licensing efforts fail, NPEs usually take the infringing business entities to court. See Zeidman (2011), p. 99.
\textsuperscript{297} Blaxill & Eckardt (2009), pp. 90-94.
\textsuperscript{298} See e.g. Frank (2006), p. 43.
\textsuperscript{299} WIPO IP for Business, p. 44. See also Gibbs & DeMatteis (2003), p. 49.
companies varies. As can be derived from the above, formulating an IP strategy depends on many different variables, but especially on the company’s view on the importance of intellectual property.

4.3 Patenting Strategies

Patent strategies are usually distinguished from *patenting strategies*. Patenting strategies are ways for a company to achieve the aim of the company’s patent strategy by making a decision to protect an invention with a single patent or with multiple patents. Moreover, the main patenting strategies can be divided into two groups: single patent and multiple patents. If a company decides to protect its invention with a single patent, the blocking power of the said patent can be small or large depending on the scope and strength of the patent. According to Granstrand, “A single patent with a large blocking power is called a strategic patent, and strategic patents have deterringly high or insurmountable invent around costs”. However, if the invention is considered non-strategic, the patent blocking power is often considered to be relatively small. In this case, a competitor’s possibilities to invent around are many and characterized by low research and development (R&D) costs and time.

Under the EPC framework, it is nowadays very difficult, due to strict patentability requirements, professionally skilled examiners and improved prior art repositories, to obtain such a single software patent which in itself is so strong and broad in scope that it cannot be invented around or to get partly invalidated by a competitor. In other words, trivial patents are much more unlikely to be granted than a few decades ago. Furthermore, it can be argued that an increased number of software patents and the cumulative nature of the software and ICT industry have led to a situation where a

---

301 It should be noted that the firm perspective on IP and patent protection differs from that of academics, legislators and the courts, because firms, unlike societies at large, are less concerned with striking balances between access and exclusivity. Rather than making decisions with society’s best interests at heart, firms often focus on their own agenda, i.e. their goals are more selfish: generating profit for their shareholders. This is something that, almost without exception, is mirrored in the strategic and overall goals of companies. For further details, see Soininen (2005), p. 63 and Gollin (2008), p. 164.
302 There is confusion in the literature over how to define the term *strategic patenting*. It is usually, but not exclusively, used to indicate patenting for defensive purposes. For further details, see Somaya (2012), p. 1091.
single patent alone is not enough for providing a wide enough protection for a certain product. Therefore, it is relatively unusual for any single patent, especially in the software industry or ICT industry in general, to be of overriding importance. Therefore, not many ICT companies any longer rely on a single patent protection mechanism to protect their invention, but rather they try to build a set of so called patent “fences”.

In other words, companies have also the option to protect and defend their invention with multiple patents, and by going beyond the scope of a single patent protection mechanism, companies can coordinate the acquisition of related patents to create blocks for competitors. Thus, it can be argued that while a single patent may be a weak and porous instrument to protect software-related inventions, the true value of software patents lies not in their individual worth, but in their aggregation into a collection of related patents – the whole is greater than the sum of its parts. However, this is not to say that a single patent alone covering a software-related invention can never receive such a position that it in itself has no high strategic value or blocking power, but that is relatively rare. Normally, the value – especially the strategic value – of a software-related invention is best captured by using the multiple patents approach.

What is characteristic of multiple patent approaches is that they often result in patent thickets, which have also been recognized as a specific problem in the ICT industry. Furthermore, at least the following different patenting strategies regarding the multiple patents approach have been identified in the literature: blanketing, fencing, and surrounding. Blanketing or flooding means building a patent cluster around a certain technology area. Thus, the goal is to turn that area into a minefield of patents by filing usually a set of medium-scope patents covering each facet of the technology. This patenting strategy is an unstructured way of tracking out multiple patents. In this case, a company patents not only the base invention, but also the peripheral and non-related

---

311 See chapter 2.3.
inventions. The idea can be to obtain exclusivity in the field, to prevent oneself being cut out of the technology or to raise costs for entrants or rivals.\textsuperscript{312}

\textit{Fencing} means that different patents are obtained for blocking certain lines or directions of competitors’ patent activities, i.e. the idea is to prevent others from developing certain technologies. Although the scope of protection of the individual patent may be narrow and thus leave gaps, a multitude of patents has the effect of keeping competitors out of the relevant market, because these patents create a maze through which a competitor must pass at his own risk. The philosophy of fencing is that, the exploitation of a competitor’s invention is made as difficult as possible for the competitor, and should a competitor for instance try to expand his or her patent by incremental improvements, then not one, but many patents can be activated against the said competitor. Thus, fencing usually puts a competitor in a situation in which it is forced to make licensing or cross-licensing deals.\textsuperscript{313}

\textit{Surrounding} is a situation where a strategic patent is surrounded by other patents. Like in the blanketing patenting strategy, a firm is granted a patent for the base invention and patents for the peripheral inventions. In this case, the patents are called surrounding patents. According to Gilardoni, “The surrounding patents are generally less important than strategic one, but collectively block its commercial use, even after its expiration. In contrast with blanketing patenting strategy, the surrounding patents are created and mapped in a structured way.”\textsuperscript{314} Thus, usually these patents pertain to different applications of the base invention. Surrounding could be used to get access to the surrounded technology, for example through cross-licensing. This is an important possibility if a competitor gets a strategic patent. Other competitors can then hope to win a second patent race for application patents that would block the exploitation of the strategic patent, which in turn would create possibilities for cross-licensing.\textsuperscript{315}

\textsuperscript{315} Granstrand (1999), pp. 221-222.
4.4 Patent Exploitation Strategies and Software Patents

4.4.1 Patent Strategy and Patent Exploitation Strategies

As a term patent strategy can be defined in many ways, but here it refers to goals – both short-term and long-term – companies have set for their patent activities and to the implementation of these goals. Fundamentally a patent strategy sets the stage for how the patent owner will develop, protect and exploit patents, but also patent infringement monitoring and enforcement are considered to be parts of it.316 According to Knight, patent strategies are variegated and often relatively sophisticated, with aims going far beyond the basic justification of patent systems.317 In addition, it should be noted that a good patent strategy is not only aimed at balancing your own patents and rights, but it also means dealing with the patents of competitors. This has actually been an increasing trend for at least a decade now. Many companies have been in fact much more interested in preventing competitors from developing rival technologies in order to fiercely maintain their proprietary monopoly, rather than focusing on empowering and leveraging their internal knowledge-base. In other words, in many cases firms have diverted their attention from using patents as devices to facilitate innovation and have employed them as strategic tools that discourage innovation by others.318 In essence, the overall purpose of patent strategy is to unify a company’s patent activities so that they support its IP strategy and consequently overall business strategy appropriately.319

Next, the focus is placed on the different patent exploitation strategies, which means that patent strategies in the context of patent development or protection are not covered or discussed here.320 Thus, the aim of the next few chapters is to explain and discuss the main patent exploitation strategies, which companies could pursue in the current competitive environment. Furthermore, the rationale underlying upcoming chapters is to understand the key features of these strategies and how software patents can be employed to achieve strategic company targets. Before we proceed it should be noted,  

316 Gibbs & DeMatteis (2003), p. 49.
317 Knight (2013), p. 82.
320 For instance, filing strategies which cover such issues as patenting routes, drafting styles and the behaviour adopted for interacting with EPO-examiners are excluded from the scope of this thesis. For further details about these issues, see e.g. van Zeenbroeck, Nicolas & van Pottelsberge de la Potterie, Bruno (2011), Filing Strategies and Patent Value, Economics of Innovation and New Technology, Vol. 20, No. 6, September 2011, pp. 539-561 and Jell, Florian (2012), Patent Filing Strategies and Patent Management, Springer Fachmedien Wiesbaden GmbH 2012, Germany.
as pointed out by Granstrand\textsuperscript{321}, that patent strategies can be defined and discussed at the level of individual patents or at the level of the patent portfolio\textsuperscript{322}. Here, patent strategies will be discussed at the portfolio level, because as established in chapter 4.3, the value of software patents is usually best captured by using the multiple patents approach, which indicates that a strategic discussion at the portfolio level will offer a more fruitful ground for discussion than discussing at the level of the individual software patent.

In academic literature, patent exploitation strategies are usually divided at least into two categories: offensive and defensive.\textsuperscript{323} In addition to the aforementioned categories, some add other categories as well, like transactional strategy\textsuperscript{324}, no patents strategy\textsuperscript{325}, and maybe even open source strategy\textsuperscript{326}. Basic strategic approaches have also been labelled as follows: aggressive, active, selective, passive and reputation-based.\textsuperscript{327} Furthermore, Somaya has recently analysed an extensive amount of earlier patent strategy literature, and based on the analysed literature he recognized three generic patent exploitation strategies: proprietary\textsuperscript{328}, defensive and leveraging.\textsuperscript{329} The academic literature indicates that the field of study within patent strategies is somewhat fragmented and disparate, especially when it comes to the use of terms and their definitions\textsuperscript{330}. Also, as we noted, various ways are used to characterize and classify

\footnotesize


\textsuperscript{322} "A patent portfolio is best understood as a collection of individual patents that share critical technological features." See Wagner & Parchomovsky (2005), p. 30.


\textsuperscript{325} Soininen (2005), p. 80; Gollin (2008), pp. 132-133.

\textsuperscript{326} As patents are generally thought to be incongruous with open source culture, one might ask the question does not open source philosophy mean that open source companies do not have any patents? Not quite. For instance some companies (e.g. Mozilla and Red Hat) in the open source market see patenting necessary to ensure that their inventions remain available to the open source community. However, they use their patents mainly defensively; to defend the company against frivolous lawsuits and simultaneously protecting open source. Software patents have gained more and more attention within the open source community, and it is likely that patents will feature more prominently in the future open source landscape. See Manek (2013) and Red Hat, Inc. Statement of Position and Our Promise on Software Patents.

\textsuperscript{327} See e.g. Gilardoni (2007), p. 429.

\textsuperscript{328} Proprietary strategy should be distinguished from proprietary software. Proprietary, or closed, software refers to “computer programs that are exclusive property of their developers and which cannot be copied or distributed without complying with their licensing agreements”. Almost all commercial software is proprietary. More information about the proprietary model and its relation to the open source model and the hybrid model, see e.g. Ballarini, Rosa Maria (2012a), \textit{Proprietary Software vs. FOSS: Challenges with Hybrid Protection Models}, IPR University Center Publications, IPR Series B, NO 4/2012.

\textsuperscript{329} Somaya (2012), pp. 1086 and 1090-1096.

\textsuperscript{330} Somaya has pointed out that “There is a pressing need to cut through the proliferation of terms and confusion in language that has accompanied patent strategy literature.” See Somaya (2012), p. 1090.
patent exploitation strategies, and as there is no prevailing system of categorization, the following categorization will be used in this thesis: defensive strategy, offensive strategy\textsuperscript{331}, transactional strategy and no patents strategy. The focus is hereinafter placed on defensive and offensive strategies as they provide the best exploitation possibilities in the context of software patents, but also transactional strategy and no patents strategy will be discussed briefly to give the reader an insight into these strategies.

While all of these patent exploitation strategies are legitimate alternatives, they are not mutually exclusive. Thus, companies can and it is even advisable to combine and adapt these different exploitation strategies or approaches when forming or refining their patent and IP strategies. Actually, to treat them as “either/or” propositions could be the first mistake of any company seeking to establish an IP strategy\textsuperscript{332}. In other words, the patent strategies of companies can consist of different patent exploitation strategies. For instance, a company’s patent strategy can have characteristics from both offensive strategy and defensive strategy. Therefore, these exploitation strategies do not unequivocally rule out each other, but they can coexist and be complementary. In practice, patent strategies are company specific and they seldom fit in only one category. However, as several scholars have pointed out, whatever the strategy is, in essence they all have fundamentally an overarching purpose and goal: the strategy is about making more money or saving money.\textsuperscript{333}

4.4.2 Defensive Strategy

A large number of ICT and software companies maintain a patent portfolio purely for defensive reasons.\textsuperscript{334} A patent portfolio is quite often maintained for offensive reasons, too, and we will have a closer look at that in the next chapter. The goal of a defensive strategy is not only to ensure a company’s freedom-to-operate in the field, but also enable commercialization of its technologies without hindrance from patents belonging

\textsuperscript{331} Offensive strategy is used here as an umbrella term for proprietary strategy and licensing strategy. See more closely chapter 4.4.3.
\textsuperscript{332} Dobrusin & Krasnow (2008), p. 195.
\textsuperscript{333} See e.g. Soininen (2005), p. 66 and Dobrusin & Krasnow (2008), p. 195.
to rivals or others.\textsuperscript{335} Therefore, according to this strategy, patents can be seen as instruments used by companies to protect their own inventions against external competition.\textsuperscript{336} In other words, defensive strategy is aimed at creating rights that will not necessarily entitle their owner with a legal monopoly (excluding others), but will at least legalize its participation in the market.\textsuperscript{337} From a more critical point of view it can be said that a defensive strategy does not consider patents as an effective means of appropriating returns: it is rather a response to the behavior of competitors than to the mere presence of an invention.\textsuperscript{338}

A company which uses a defensive strategy obtains patents and maintains its portfolio mainly in order to protect its own interests. Due to the number of software patents today and their abstract nature, a software company that innovates can never be sure that they are not running into the realms of someone else’s patents.\textsuperscript{339} Especially, in fast-paced high-technology industries, like the ICT and software industry, companies often make substantial irreversible investments in business opportunities before it is clear who actually owns all patents for the required technologies. In turn, if the set of technologies required to commercialize products is very large, a situation characterized by multi-invention contexts (e.g. mobile phones), the exposure to the patents of others could be particularly problematic. Fundamentally, the need for defensive strategies arises because a patent confers only the right to exclude others, not an affirmative right to use the patented technology.\textsuperscript{340} That is the core reason why a defensive exploitation strategy should not be ignored as a way to exploit software patents strategically.

There is always the risk that the software you create in some way infringes on someone else’s software that is already protected by a patent. In addition, there is the risk that this

\textsuperscript{335} See e.g. Somaya (2012), p. 1094 and Gilardoni (2007), p. 422.
\textsuperscript{336} Gilardoni (2007), p. 422.
\textsuperscript{337} Furthermore, as pointed out by Soininen: “Not only core technologies are patented but also alternative technologies are often patented so that others cannot prevent the company from developing the patented technologies in the future. The further objective is to prevent other companies from producing functionally similar and thus competing products in the market place.” See Soininen (2005), p. 74. See also Guellec & van Pottelsbergh de la Potterie (2007), p. 80.
\textsuperscript{339} To know this would require such an extensive clearance search on every product that the time taken to do so would far outweigh any benefits from innovation. In practice, companies do not, and they simply cannot, spend whatever it takes to find all relevant patents to avoid infringement. Therefore, some companies deliberately do not perform clearance searches at all and they address inevitable infringements as they occur, often with other patents in their portfolio as intended bargaining chips. See chapter 5.2.
other party discovers this, and demands either a large license fee or threatens with infringement proceedings.\[341\] Infringement proceedings can be enormously expensive and it is therefore often more sensible to settle out of court in such cases. One way to do this is simply to pay the license fees. However, as the two parties are often innovating in the same technical area, there is a good chance that you have some patents in your patent portfolio that the other party would either benefit from accessing or potentially infringes. It has been acknowledged that cross-licensing is important in industries with cumulative systems technologies, because in these industries there is a risk of patents blocking each other.\[342\] A cross-license therefore becomes an amicable (and relatively inexpensive) solution to the problem.\[343\] Therefore, it can be said that the primary use of a defensive portfolio is to function as a trading tool or a bargaining chip.

To summarize the three previous paragraphs: in the software industry cross-licensing generally takes part as a defensive strategy which is intended to ensure the freedom-to-operate without infringement. When it comes to suing another party over infringement, companies adopting a defensive strategy often never contemplate suing another company for patent infringement.\[344\] It has been said that sometimes the only benefit that cross-licensing agreements provide is freedom from patent litigation\[345\]. However, to be even part of cross-licensing negotiations you should at least have some bargaining power – that is to have patents or rather a portfolio of patents. Even though you do not own an extensive portfolio with hundreds of patents, your patents may prove to be of considerable interest to the other party with whom you are negotiating and you could enter into a cross-licensing arrangement where, simply put, the patent rights could be exchanged between your company and the other\[346\].

\[341\] Leaves (2010), p. 75.
\[342\] Olsson & McQueen (2000), p. 566.
\[343\] Leaves (2010), p. 75.
\[344\] See e.g. Leaves (2010), p. 75.
\[345\] Mann (2005), p. 991. Further, according to Mann: “Quite often cross-licensing agreements do not involve the disclosure of technology or transfer of any knowledge beyond material on the face of existing patents”. Mann also sees the acquisition and use of patents for cross-licensing purposes to be a deadweight loss for the industry. More closely, see Mann (2005), p. 998. Cf. Soininen who points out the following: “The problem with implementing a purely defensive strategy in an environment in which nobody uses patents actively and in which the risk of being sued is low is that it is challenging to prove that patenting actually saves costs. Patents are company assets and they should be used efficiently to benefit the shareholders: it is wasteful if most of them are not utilized.” See Soininen (2005), p. 78.
Because software is characterized by cumulative innovation, software companies are vulnerable to patent hold-up problems. To overcome hold-up problems, many companies have started to treat patents as bargaining chips to gain access to the technology portfolios of other companies by concurrently allowing access to their company’s own knowledge bases. Sometimes companies even create or join patent pools. The reason why such patents are defined as bargaining chips is quite straightforward. The higher the number of patents that can be brought to the table and the greater their intrinsic value, the more powerful is the company’s leverage in negotiations. This is the reason why companies with strong patent portfolios are able to capture considerable benefits from their patent estates. In addition, having a large and strong patent portfolio is a useful instrument in cross-licensing agreements and the negotiation process for obtaining profitable conditions; in fact, the existence of a wide patent portfolio can intimidate other firms, help to avoid infringement lawsuits or give access to such inventions or technology which would be out of reach if the company would not have anything to offer in return.

In spite of its attractions, a defensive exploitation strategy based on extensive patent portfolios, cross-licensing, and mutual hold-up also has its limitations. If a patentee does not really need access to the other company’s patents, these approaches will no longer be viable. This can happen, for instance, if the patentee is a niche specialist and the other company’s patent portfolio does not cover technologies in that niche. Similarly, mutual hold-up is often a weak defensive strategy against such patent holders who do not have commercial operations (NPEs) of their own that can be held up. Especially in ICT industries, specialized patent-holding companies, i.e. patent trolls and sharks, have been particularly challenging for large technology companies to deal with.

---

347 Hold-up problem means “that a third party patent covering a certain feature, a single routine in a computer program, for example, may hold-up the production of the entire program. In the worst case the fact that many companies and/or individuals may have the right to block others from using a resource results in a reluctance to innovate.” For further details, see e.g. Soininen (2005), p. 32.


350 Patent pool is a consortium of companies agreeing on cross-licensing their patents either because they want to save the time and cost of patent management or because they have mutual blocking patents which they want to get and allow access to or because they want to purposively set a common technology standard. Control of these standards can yield large rents. Lower costs of production due to product standardization will also be combined with higher profits given the large adoption of the new standardized product. More closely, see Leone & Laursen (2011), p. 93.


353 See also chapter 4.2.2.
precisely because of the risk of being mutually held up. According to Somaya, “For these types of patent holders, alternative strategies may need to be devised, such as pre-emptive patenting or disclosing enabling technologies, inventing around key patents, and challenging the validity and enforceability of patents.”

In the case that an innovative company does not maintain a patent portfolio, then they clearly are open to attack, and have few options other than to pay or defend themselves in court. Whilst opponents to software patents would claim that no one should be getting software patents granted, it does not change the reality that many granted software patents are out there now in the hands of parties who are willing to wield them, and having no defence clearly leaves you vulnerable. Notably, in many cases the mere existence of a patent portfolio acts as a deterrent from someone that is thinking about starting an infringement action against your company. This is because a company with a patent portfolio makes a less inviting litigation target than a “naked” competitor. Even patent applications that have not yet been granted, can act as a deterrent. Because all patent applications are published 18 months after filing, your competitors can see what you may get patent protection for in the future, even though they do not know at that stage what the final scope of protection will be. This gives rise to uncertainty in your competitor’s strategies.

4.4.3 Offensive Strategies

4.4.3.1 Proprietary Strategy

As an alternative for a defensive strategy, software patents and a patent portfolio can be used in a more offensive manner. For example, software patents can be used as protection for core product features and preventing others, in an offensive manner, from utilizing patented innovations (proprietary strategy), and can also be a source of revenue generation through licensing technology and patents (licensing strategy). Both of these strategies are based on the right holder’s exclusive rights to forbid others from utilizing the patented invention. Although both of these offensive strategies are suitable when exploiting software patents, licensing strategy, however, is more commonly utilized in

---

the context of software-related inventions. Even though the usage of terminology of the
aforementioned exploitation strategies in the patent strategy literature is somewhat
incoherent, the term “offensive strategy” will be used here as an umbrella term for
proprietary strategy and licensing strategy. This is because both proprietary strategy and
licensing strategy are based mainly on offensive planning of the use of patents in
business.358

A proprietary strategy, or exclusive strategy, is in essence the conventional resource-
based logic of using patents as isolating or blocking mechanisms that shield the
company’s key competitive advantages from imitation.359 Thus, in this strategy, patents
are obtained and used in order to maintain a competitive advantage in the marketplace
by excluding competitors.360 For instance, exclusion can take place by a company
applying for patents to block development of similar or related technologies that will
compete with the company’s product.361 By doing this, companies try to secure their
current and future position when further developing their own technologies.362 In
addition, despite hampering other companies from entering particular fields or
developing innovations within patent “infested” fields, proprietary strategy provides
companies that enter new technological domains a lower level of risk of imitation and
competition, particularly in early stages of development of new products and
technologies. This is sometimes known as obtaining freedom-to-operate.363

In enforcement, proprietary strategies imply concerted efforts to detect imitation and
active, or even aggressive, enforcement of a patent holder’s rights against any detected
infringement.364 If another party does copy the feature in a competing product you can
use the patent to convince the other party to remove the feature or, in the most extreme
case, go to court and attempt to obtain an injunction against the sale of the competing
product.365 Actually, once a suit is filed, companies pursuing a proprietary strategy are
less likely to settle with the alleged infringer.366 Of course, you can also use a

361 Bakels & Harison (2005), p. 66.
365 Leaves (2010), p. 76.
proprietary strategy to obtain licensing revenue or to obtain some wanted IP or valuable know-how from the other party through licensing your patent. While the patent license may give the licensee a right to use the technology, many license agreements have provisions for the transfer of know-how in addition to the patent. So, while the patent might be the legal reason to license, the key to true business success may be the acquisition of know-how and trade secrets that allow the licensee to use the information contained in the patent in a practical way.\textsuperscript{367} However, generally in the licensing domain, companies would be unwilling to license out patents with which they are pursuing a proprietary strategy and seek to commercialize the technology themselves.\textsuperscript{368}

In sum, employing a proprietary patent exploitation strategy implies that a company seeks to obtain an exclusive patent position in the key technology. To pursue a proprietary based patent exploitation strategy for every piece of new technology that a company owns would be prohibitively expensive, but it may be justified for technologies that underlie a core competency, entail high strategic stakes, or are otherwise critical for the company’s competitive advantage.\textsuperscript{369} Proprietary strategies are often likely to be used when it is difficult to write and enforce licensing contracts that retain the potential benefits of using the technology exclusively.\textsuperscript{370} The ability to “stake out and defend a proprietary market advantage” is often cited as the “most powerful benefit” of patents. As we have noted, proprietary strategies are reflected in a number of actions and uses of patents – for example, building fences, offensive blocking, building offensive thickets, preventing copying, and so forth.\textsuperscript{371} It can be said that the ultimate goal for companies adopting a proprietary based strategy is to shape, in an offensive manner, appropriability regimes by their own behaviour and strategy.

4.4.3.2 Licensing Strategy

In many situations it may not be reasonable for a firm to use a proprietary based offensive strategy, nor may it need a defensive strategy; but patents may nonetheless be

\textsuperscript{367} Knight (2013), p. 73.
\textsuperscript{368} Leaves (2010), p. 76; Somaya (2012), p. 1092.
\textsuperscript{369} Somaya, Teece & Wakeman (2011), p. 68.
\textsuperscript{370} Somaya (2012), p. 1093.
valuable tools for generating additional revenues.\textsuperscript{372} Hence, an offensive strategy may be used not only for preventing others from utilizing patented inventions, but also for acquiring external resources such as royalty income or to push the other party to license their essential technology.\textsuperscript{373} This line of offensive strategy is often called licensing or leveraging strategy, and according to it, patents are used predominantly as a means of revenue generation. In other words, you are not trying to prevent anyone else having the same patented feature as you, but are instead actively looking to license the patent to other parties.\textsuperscript{374} The central logic of licensing strategy is that the bargaining advantages conferred by the exclusionary power of patents allow the company to pursue direct and indirect profit opportunities.\textsuperscript{375}

Companies have many reasons to license-out\textsuperscript{376}. For instance, perhaps a company does not have the resources to develop the invention itself or it has other priorities (e.g. a non-practicing entity). When licensed out, patents may provide a crucial source of royalty income. Because the goal of the company in a licensing strategy is to use its patent rights to bargain effectively for rents in different contexts, it is not necessary for the company to patent every substitute technology or to have watertight patent protection.\textsuperscript{377} Thus, when building a portfolio for revenue generation, you are not necessarily filing patents on the features that you develop that are the most innovative or distinguishing for your product. In fact, the inventions you are filing the patents on do not even need to be in any of your products.\textsuperscript{378} Actually, it is not rare that some of the company’s software patents may be in technologies that are not central to its strategy or core competence, but are nonetheless valuable.\textsuperscript{379} Licensing-out in these cases is beneficial to the potential licensor who otherwise would not be able to exploit

\textsuperscript{372} See e.g. Somaya (2012), p. 1095.
\textsuperscript{373} Soininen (2005), pp. 69-70; Knight (2013), pp. 72-73.
\textsuperscript{374} Leaves (2010), p. 76.
\textsuperscript{375} Somaya (2012), p. 1095.
\textsuperscript{376} Note the difference between the terms license-out and license-in. Here the focus is placed solely on licensing-out. Within the economic and management literature, the supply-side of the coin has generally been emphasized. This is also a result of the common tendency among firms to pay more attention to selling their IPRs to others than buying IPRs from outsiders. However, given the increasingly global competition and complexity of products and the necessity to be on the forefront of innovation, in-licensing strategy is gaining increasing attention both by (industrial and financial) practitioners and, thus consequently, by management scholars. For further details, see Leone & Laursen (2011), p. 89.
\textsuperscript{378} Leaves (2010), p. 76.
\textsuperscript{379} Somaya (2012), p. 1095.
the technology surplus. In addition, you can file patents towards areas where you think the market is heading, and where your competitors will be operating, and ultimately pursue them for license fees.

The following strategic use of patents can be described more as a leveraging strategy than a licensing strategy, although these terms are used here interchangeably. Patents can also be utilized in order to push the other party to license their essential technology. In this case you are using your patents, in an offensive manner, as bargaining tools to get access to the patent portfolios of other companies. This is possible due to the fact that companies, it is here referred to rival companies, cannot always afford to block all others from using their patented inventions, but sometimes there is a need to license their technologies to others to manufacture, distribute, use, and develop further their own technology. This is why in some cases a well-reasoned licensing strategy may complement and enhance a company’s product line, and assist in positioning it favourably in the markets.

Some companies exist largely for the purpose of licensing alone and are more aggressive in exploiting their patents, and such companies are often referred to as patent trolls. These companies have no activities except to acquire patents and enforce them against wealthy companies with substantial market activity, and they are particularly prevalent in relation to software inventions. Hence, patent trolls do not conduct any R&D or produce products, but they merely rely on the enforcement of their patents in order to generate revenues. From the perspective of active companies with products to sell, patent trolls are immoral, because they produce no useful products, but also troubling because they are immune to negotiation with the bargaining chips of the active companies. That is because patent trolls seek direct licensing revenues and do not usually benefit from a cross-license. This, again, means that a defensive strategy that

---

382 Soininen (2005), pp. 69 and 70. See also Leone & Laursen (2011), p. 90.
386 Gollin (2008), p. 172. Further, according to Wagner & Parchomovsky, “Even the active companies that threaten others with litigation often do so in the hope of securing a cross-licensing agreement with the potential defendant, so that the potential defendant will not later sue the potential plaintiff on another patent. However, this is not the case when the question is about patent trolls.” See e.g. Wagner & Parchomovsky (2005), p. 26.
might otherwise help software companies to maintain their freedom-to-operate, has only minimal or no influence on patent trolls.\footnote{Soininen (2007a), p. 15.}

Software products often contain thousands or sometimes even millions of lines of code. Given that a handful of lines of code can constitute a patent-eligible invention, the number of potentially patentable inventions in a 100,000-line computer program can be very large.\footnote{Mulligan & Lee (2012), p. 11.} Furthermore, a severe problem of hold-up can arise where a software patent relates only to a small portion of the product, as multiple mobile phones related trials have shown, but the patent holder can obtain a preliminary injunction in the courts suspending all production while the dispute is adjudicated. This can lead to settlements out of court that are greatly in excess of the true value of the patent.\footnote{Greenhalgh & Rogers (2010), p. 160. See also Somaya (2012), p. 1096.} As patent litigation is notoriously expensive, the third party may choose to pay up to a patent troll, particularly if an application for an interim injunction is likely to be made.\footnote{Bainbridge (2010), p. 386.} So far Europe has not faced patent trolling on a large scale, which could be explained by the fact that the consequences of litigation and infringement are less severe.\footnote{While the average cost of patent litigation in the U.S. amounts to more than $2 million per side, in Finland the figure for hearing an infringement case in the district court is closer to EUR 150 000 per side. Of course the total amount of litigation costs may be fundamentally higher if the case involves various phases such as a precautionary measure claim, and both infringement and annulment actions. See Soininen (2007a), p. 15.}

\subsection*{4.4.4 Transactional Strategy and No Patents Strategy}

\textit{Transactional strategy}. Patents are also important for transactional purposes, because they are known to help companies to access external financing in various ways, including attracting venture capital and prospective partners\footnote{Patent holdings can serve as a basis of evaluating candidates for joint venture participation or an acquisition, and the like. See e.g. Davis (2006), pp. 153-154.}, raising collateralized debt, and boosting a company’s IPO (initial public offering) valuation.\footnote{Soininen (2005), p. 78; Davis (2006), p. 161; Bessen (2011), p. 5; Somaya (2012), p. 1105.} Thus, from the transactional strategy point of view, patents play a valuable role as an indicator of value. Business partners, investors and shareholders may perceive patent portfolios as a demonstration of the high level of expertise and technological capacity within your company\footnote{WIPO IP for Business, pp. 9-10.}, but does the size of a portfolio matter or is the quality of the portfolio a
more important factor? According to Soininen, “The method of valuing a company’s patents based on the height of the stack is about to become extinct: it is the technology and what is protected that matter, not the number of patents involved. Quality is appreciated more than quantity, although quantity may have other functions, such as discouraging competitors from entering certain markets and reducing the incidence of infringement as explained above.”395 On the other hand, according to Wagner and Parchomovsky, “The size-effects of patent portfolios will improve holders’ ability to attract and retain capital investment. Unlike individual patents, a significant patent portfolio is a substantial asset. Further, while the dubious value of individual patents calls into question their ability to provide meaningful signals about the company to external parties, patent portfolios do not suffer from this defect – and thus will provide information to the capital markets about the competitiveness, savvy, and long-term prospects of the holding firm.”396

No patents strategy. Many companies, particularly smaller ones, have no concerted intellectual property strategy or policy in place. However, it should be noted that not having an IP strategy is different from not having any patent strategy or from a no patents strategy. A company can have some form of an IP strategy in place while it has no specific strategy regarding patents. Furthermore, a no patents strategy can be seen as an alternative or a complement to defensive, offensive, and transactional strategies. Therefore, a non-strategy in the context of IP397 should be distinguished from a no patents strategy. Moreover, a no patents strategy does not mean that a company has not acknowledged the existence of patent protection or considered whether or not patenting could benefit its operations in some way.398 A no patents strategy, according to Soininen, means “that a company acknowledges the existence of patents and plans its operations accordingly. It respects the rights of others and takes the infringement risk into account in its contract practices, its insurance policy and its potential alliances, but does not consider it worthwhile to file or acquire patents even for defensive purposes.”399

395 Soininen (2005), pp. 78-79.
397 More about a non-strategy in the context of intellectual property, see e.g. Gollin (2008), pp. 132-134.
398 Soininen (2005), p. 82.
399 Ibid.
There are several reasons to adopt a no patents strategy. For instance, copyright protection added to lead-time and secrecy might at times be enough to gain a competitive advantage, especially in markets in which only a few companies have patents. Further, if it is not easy to detect if a company’s own patent has been infringed or companies just are unable to discover the patents their own activities might infringe, companies might ignore patents.\footnote{Mulligan \& Lee (2012), p. 1.} Also if the technology is most likely to be short-lived, or then if the invention is clearly company-specific and has no value to others, patenting may not be a viable option although it is possible. Also, in a case where a company produces mainly tailor-made software, patents are often not crucial, and might be a rationale to pursue a no patents strategy.\footnote{Cf. Soininen (2005), p. 80.} On the other hand, without patents or a patent portfolio, competitors may copy and practice inventions freely and there can be no bargaining chips or leverage for counterclaims (cf. defensive strategy and proprietary strategy). However, one way of averting the risk that someone else will patent the same invention is to publish early. A no patents strategy also means that a company loses the opportunity for patent licensing revenue (cf. licensing strategy). In addition, you cannot either attract external financing based on the value of your patents (cf. transactional strategy).
5 Patent Infringement Monitoring and Software Patents

5.1 Introduction and Chapter’s Contents

It has long been recognized by scholars that a major problem with patents for software-related inventions is infringement detection.\textsuperscript{402} Similarly, many companies in the ICT and software-industries feel that detecting infringements on their software is rather difficult.\textsuperscript{403} Although every patent owner is entitled to the exclusive rights conferred by the patent, the fact is that nothing will happen and the infringement will continue unless the patent owner affirmatively does something about it. In other words, a patent does not provide a sublime protection, but it only gives the patent holder the right – or in the case of software patents rather a right to try – to take legal action against infringers. Furthermore, in order to enforce your patent against infringing parties, you first have to be able to detect the possible misuses and infringing acts, and to do this you need to conduct patent infringement monitoring. Patent infringement monitoring is a post-grant act which has the goal of detecting acts that are likely to infringe on the granted patent. The duty of monitoring falls on the shoulders of the patent owner, and therefore, the patent owner must assume the full burden for stopping the infringer.

Patent infringement monitoring is here divided into three different phases: initial monitoring (chapter 5.4), evidence collection and analysis (chapter 5.5), and claim interpretation and comparison. However, the claim interpretation and comparison phase is excluded from the scope of this thesis, because this phase is not only in general but also in the context of software patents a very wide topic to discuss\textsuperscript{404}. Nonetheless, a few brief references to the topic will be made at times. Next, the structure of chapter five will be presented and explained. The chapter begins with a discussion of a prior art search in the context of software patents. Although a prior art search is generally conducted before a patent application has been filed, it is suggested that a prior art type of search, or rather a modified version of it, can also be utilized in the context of software patent infringement monitoring (initial monitoring phase). Therefore, in order to understand the initial monitoring phase (chapter 5.4) discussed later it is necessary

\textsuperscript{403} See e.g. Olsson & McQueen (2000), p. 571 and Davis (2006), p. 165.
\textsuperscript{404} For further details about claim interpretation and comparison in general, see e.g. Fisher, Matthew (2007), \textit{Fundamentals of Patent Law: Interpretation and Scope of Protection}, Hart Publishing, Portland, OR.
first to provide the basic framework regarding a prior art search and database and keyword search in the context of software. Furthermore, in addition to discussing a modified prior art search in the context of software patent infringement monitoring, chapter 5.4 will highlight reasons why it may be difficult to detect infringing uses, i.e. reasons why so many software-related inventions suffer from the problem of detectability.

Next, chapter 5.5 explores issues regarding collecting and analysing evidence in the context of software patents. Of course, a hunch that someone’s product or behaviour is infringing your software patent is simply not enough to prove the infringement and to enforce your rights. Hence, uncovering infringement requires collecting evidence, analysing the infringing product or act, and documenting your findings. Collecting and analysing evidence and developing your proof are essential steps in making your investigation productive and effective so that it may help you, if necessary, to take the next step, for instance suing the infringer. However, because proprietary software is nearly always distributed in object code form, software devices typically cannot be readily understood by casual inspection, and particularly not without access to human-readable source code. Therefore, although the patent holder may come to suspect that a product is infringing by observing externally visible features, obtaining proof will still usually require reverse-engineering the object code of the suspected software product.

Even though the focus of this thesis is placed on software patents and thus the exclusive rights conferred by a patent, reverse-engineering a software product also triggers questions regarding the legality of such an act from the copyright perspective. Thus, this discussion regarding reverse-engineering under the European legal framework not only covers the legal aspects of reverse-engineering from the patent law perspective but also from the copyright perspective. Moreover, chapter 5.5 aims at explaining whether or not it is acceptable to reverse-engineer a software product for the sole purpose of detecting infringement. After discussing reverse-engineering the focus shifts to the issue of patent exploitation strategies in the context of software patent infringement monitoring of which prior academic literature has not discussed. Hence, chapter 5.6 combines the knowledge obtained from chapter four (especially patent exploitation strategies) with the knowledge obtained from chapter five and offers new insights regarding patent infringement monitoring and the strategic utilization of software patents. For example, it
is recognized that if patent infringement monitoring is ignored, some strategic aspects of the use of patents are limited or worthless. As a consequence, offensive strategies cannot be exploited in the intended manner or utilized to the desired extent. Therefore, if a company aims at being effective in exploiting its patents, especially if patents have been obtained for offensive purposes, it should adopt a proficient monitoring strategy that is aligned with the company’s patent and overall IP strategy. Finally, in chapter 5.7, the most common ways to react once infringement has been detected and evidence been collected are briefly introduced.

5.2 Prior Art Search in the Context of Software Patents

5.2.1 Prior Art Search in General

According to patent literature, there are multiple different types of searches that a searcher can use to assess technology. At least the following search types have been recognized: “patentability search” (or novelty search), “validity search”, “infringement search” (or evidence of use search)\(^\text{405}\), “clearance search” (or freedom-to-operate search), “state of the art search”, and “patent landscape search” (or patent portfolio search).\(^\text{406}\) The names associated with some search types can be either interchangeable or distinctly different depending on who you consult.\(^\text{407}\) However, no matter what name a search type is, it is important for the searcher to understand why a certain type of search is being carried out and further conduct the search according to that understanding. In addition, a person commissioning a search needs to have realistic expectations for the search results.

Moreover, considering that there are countless sources of prior art, it is important to formulate a reasonable search scope when doing a prior art search.\(^\text{408}\) Thus, a searcher needs to be proficient in searching through different information sources. To try to exhaust every resource is not practical and would cause unnecessary delays and costs.

\(^{405}\) It should be noted that an infringement search differs from infringement monitoring. The main difference between an infringement search and infringement monitoring is that an infringement search is conducted prior to filing a patent, whereas infringement monitoring is conducted after the grant. Further, an infringement search is carried out in order to determine whether or not an enforceable patent claims the same matter as your concept or unpatented invention, while infringement monitoring is used to detect such acts that are likely to infringe your patent. See e.g. Hunt, Nguyen & Rodgers (2007), pp. 26-28.


\(^{407}\) See e.g. Alberts et al. (2011), p. 10.

\(^{408}\) Park (2005), p. 63.
What sources to search through is dictated by what type of search is required, the legal and financial implications of the search, and how much time there is to complete it. For a prior art search, the goal is to conduct a better search than the patent examiner, and for other patent search types, the goal is to be as complete as resources and time allow. Therefore, it is good to remind oneself of the objective of the search, which in the case of prior art is typically to find the prior art that a patent examiner likely would find upon conducting a prior art search. Generally, when conducting any type of search, three factors will affect the results: cost, quality, and time. However, it is important to keep in mind that albeit searchers have a number of valuable tools at their disposal, and even if they would have all the time and money in the world, no search is bulletproof nor hundred percent complete.

Even though some scholars distinguish a patentability search from a state of the art search, here they are used interchangeable and referred to commonly as a prior art search. Hence, prior art search is defined as follows: the prior art search is an attempt to find evidence of patents and technical publications in order to assess the patentability of an invention and its relation to the current state of the art. In other words, the prior art search can help the searcher (e.g. the inventor or a patent attorney) determine whether or not an invention can be patented and, if so, what other patents or non-patent literature would be relevant to that assessment. Thus, the search seeks to determine if anyone disclosed the inventive concept in a publicly available work anywhere in the world before a date determined to be critical.

When determining whether or not an invention can be patented, the searcher focuses on the questions of novelty and inventive step. Even though both novelty and inventive step are defined in relation to the state of the art, the search strategy requirements for

412 See e.g. Hunt, Nguyen & Rodgers (2007) and Alberts et al. (2011).
414 There are different schools of thought concerning who should conduct a search. For most cases, particularly when a person lacks experience in searching for or reading patents, the job is probably best left to patent counsel, even though there will be an expense associated with it. See e.g. Dobrusin & Krasnow (2008), p. 290. Cf. Pressman (2008), p. 122.
416 Ibid.
these criteria are subtly different.\textsuperscript{417} Because any relevant written evidence may impact novelty or obviousness, you should search through full patent specifications and claims and all available technical and nontechnical publications. Therefore, information sources should include at least the following: patents, patent applications, research papers and technical journals, publicly available corporate documents of competitors or other companies, conference proceedings and product manuals.\textsuperscript{418} In short, any written material that precedes the filing date of a patent application or the present should be included.

According to Cantrell, prior art searches take typically a day to perform\textsuperscript{419}, but as discussed, what you search through and how long you search is merely a function of your available time, your budget, and the public availability of the information.\textsuperscript{420} There are many benefits to performing a search. For example, when a prior art search is done, you should have a better sense of your legal and business options. The search results, hopefully, will help you determine whether or not to pursue patent protection and may indicate what issues could arise during examination.\textsuperscript{421} The prior art search is generally conducted prior to filing a patent application because of its usefulness in preparing and filing a patent application. For instance, the search can help a writer of a patent application construct claims to achieve the broadest possible protection without treading on known prior art. In others words, this helps to avoid “new matter” rejections.\textsuperscript{422} Although a prior art search is generally conducted before a patent application has been filed\textsuperscript{423}, it is later in this thesis proposed that a modified prior art search can also be utilized in the context of patent infringement monitoring, i.e. after the patent has been granted, with the goal of detecting potential patent infringements\textsuperscript{424}.

\textsuperscript{417} Novelty is the requirement that the invention in hand does not already form part of the state of the art. Inventive step is the quality that the invention would not be obvious to a person well versed in the appropriate technical field – in other words, someone who knows the relevant state of the art. Therefore, the starting point for a prior art search has to be that we have access to as much of the legally-defined state of the art as is possible to get. See Adams (2006), p. 170 and chapter 3.2.3 of this thesis.
\textsuperscript{419} Cantrell (2009), p. 10.
\textsuperscript{421} See e.g. Dobrusin & Krasnow (2008), pp. 288-289.
\textsuperscript{423} Although a prior art search can also be conducted after the patent application has been filed, it is generally more cost-effective if the results are already at hand when the application is at the drafting stage. See Adams (2006), p. 169.
\textsuperscript{424} See chapter 5.4.2.
5.2.2 Database and Keyword Searches and Software in the Prior Art Context

Searchers of prior art have a number of valuable search tools at their disposal that search through: citations, bibliographic data, classifications, text, and so forth. With electronic capabilities now available, searches and search tools have advanced and become quicker and more efficient. Also most prior art searches, but not all, is conducted by searching through electronic databases. Though databases may be private subscription service databases, i.e. commercial databases, also free Internet searches of comprehensive patent databases are available, such as patent searches through the European Patent Office, the United States Patent and Trademark Office, and the Japan Patent Office. In addition, searchers should also consult multiple non-patent literature sources and databases. Nowadays there are a wide variety of databases for searching through non-patent literature, and also some databases of particular interest in the computer, software, and electronic arts: IEEEExplore, Softbase, and INSPEC, just to mention few. However, the fact that software-related inventions may be indexed or classified under different classes or subclasses complicates the search task at hand. Furthermore, most computer search systems that cover non-patent literature sources do not use similar classification systems as patent offices and do not use standardized terms as well, which often complicates the search even more.

Therefore, it is often necessary to focus more on text-based searching than other search tools to find relevant prior art. However, to make a text-based search, you would select a combination of keywords to describe your invention. When a search system finds any patents that contain your keywords in the combination you specify in your search request, it will identify these patents, regardless of their classification. Generally, successful database searches will depend upon the subject-matter of the invention, familiarity with the operation and limitations of each individual database, and the availability of time and the budget for the search. However, prior art searches for

425 More extensively about search tools, see e.g. Hunt, Nguyen & Rodgers (2007), pp. 145-182.
429 Before you even approach the computer, no matter what search system you use, be well prepared with a well-thought-out group of keywords and all possible synonyms or equivalents. See Pressman (2008), p. 159.
software-related inventions can be enormously challenging, and one should be aware of the difficulty in doing a good search.\textsuperscript{432} There are several main reasons why it can be challenging not only for an inventor but also for a patent examiner to find the relevant prior art regarding software-related inventions.

Before these reasons are discussed in greater detail, it is essential to note that the factors that make software patents work poorly in the context of a prior art search are characteristics of the software industry, not the patent system.\textsuperscript{433} These problems are mainly due to the abstract nature of software\textsuperscript{434} and there not being a judicial requirement to disclose source code\textsuperscript{435}. First, there is the problem of prior art in general, which in the case of software can mean three different things: 1) lack of knowledge due to the difficulty of gathering prior art, 2) lack of knowledge due to prior art not being publicly available\textsuperscript{436}, and 3) lack of knowledge due to prior art information being insufficient. The fact that much of the prior art in the software industry lies outside of the traditional patent databases has been acknowledged for a while now. Therefore, patent offices have taken measures to improve an examiner’s access to non-traditional sources of software documentation. But the diffuse nature of the knowledge base and the lack of a comprehensive system for cataloguing and indexing software-related developments defy even the most knowledgeable and diligent examiner.\textsuperscript{437}

The second aspect of the prior art problem relates to the fact that much of prior art is in the form of previously written software, which means that many very well-known software techniques have not ever been disclosed in such a printed form that is available to searchers.\textsuperscript{438} Thirdly, the problems regarding finding the relevant prior art can be due to the fact that information available is simply insufficient. Namely, many software patents, especially first-generation ones, give little or no information in the patent claim or in the specification about the software program itself, and even a later-generation software patent claim may tell very little about the software program in question.\textsuperscript{439}

These problems regarding finding relevant prior art can be summarized as follows: the

\textsuperscript{432} Pressman (2008), p. 160.
\textsuperscript{433} Mulligan & Lee (2012), p. 22.
\textsuperscript{434} See chapter 2.3.
\textsuperscript{435} See chapter 3.2.4.
\textsuperscript{436} See e.g. Ballardini (2009), p. 211.
\textsuperscript{438} See Ballardini (2009), p. 212.
sources of prior art in the case of software are scattered, sometimes unavailable, or insufficient, and therefore they present considerable difficulties to the searcher trying to locate the relevant prior art.

The second problem relates closely to the first one. Say that the relevant prior art is not only publicly available but also findable without the need to be making unreasonable effort. Then, the question comes to finding it. With software it is often necessary to focus more on text-based searches than other types of searching tools to find relevant prior art. However, what is problematic is that text-based searching comes down to defining and selecting terms and key-words in order to find relevant prior art. According to Bessen and Meurer, “Patent law assumes that once the words are mapped to a specific set of technologies, one can readily determine which technologies are equivalent and which distinct, but in the case of software, however, this assumption is not nearly always true.”

Actually, software-related inventions have many terms that are used interchangeably, even when the inventions are technically different. For example, the terms packet, cell, frame, datagram, and envelope have been used interchangeably. The fact that patent applicants are free to use any term they like or even make up new ones to describe their software-related inventions does not alleviate this so called key-word problem.

In addition, the effectiveness of a text-based search is further undermined by the doctrine of equivalents, which holds that “a patent can cover subject-matter equivalent to its claims even if it does not fall within their literal scope”. This means that in practice, a text-based prior art search will only find a fraction of relevant sources or produce so many results that it would be of little help to the searcher. These problems present particular difficulties when the searcher should come up with “good” or “essential” key-words for searching through relevant prior art. Shifting terminology results in near-endless synonyms that frustrate even the most diligent searcher. We just

445 It should also be noted that, as pointed out by Mulligan and Lee, other search strategies, such as searching by inventor, assignee, or citations in related patents, are no more promising. For further details, see Mulligan & Lee (2012), p. 12.
have to accept the fact that it is just harder, maybe even impossible, for any one individual to find all relevant information, even in a perfect world.\textsuperscript{446}

The third problem has a close connection to the previous two problems and it has been emphasized in the study conducted by Mulligan and Lee in 2012. They call this the problem of discovery costs. Discovery costs, i.e. the costs of finding relevant patents and information sources, are often prohibitively high in the case of software patents.\textsuperscript{447} Even though Mulligan and Lee’s study focuses mainly on the patent clearance search\textsuperscript{448}, similar cost issues are present when doing a prior art search and even when a company is monitoring for potential patent infringements. Whether the situation is to find prior art, to avoid infringement or to detect infringement, a company must expend resources to learn about potentially relevant information sources. However, this might not be feasible because discovery costs might be too great.\textsuperscript{449} Furthermore, based on the findings of Mulligan and Lee’s study, in software, for example, patent clearance by all companies would require many times more hours of legal research than all patent lawyers in the US can bill in a year.\textsuperscript{450}

The requirement for every company to examine every patent and non-patent source in some detail is completely impractical in a world where there are hundreds of thousands of valid patents and numerous information sources. Therefore, searchers use a variety of methods, such as text-based searching, to “guess” at what sources they should look at.\textsuperscript{451} Ideally, companies in the ICT and software industries would have ways to quickly find the small minority of information sources that relate to their own products and ignore the rest. The ability to do this depends on a good system of organization and standardized terms and key-words, which as discussed we do not currently have.\textsuperscript{452} Therefore, in practice, companies do not, and they simply cannot, spend whatever it takes to find all relevant prior art or to avoid infringement. Rather, they spend only as much money on patent searches as they believe will “pay off”.\textsuperscript{453}

\textsuperscript{446} See Cohen & Lemley (2001), p. 43.
\textsuperscript{448} Briefly, clearance searches are used to determine whether or not a party has “clearance” to make, use, and sell an inventive concept. More extensively, see e.g. Hunt, Nguyen & Rodgers (2007), pp. 28-29.
\textsuperscript{449} Bessen (2011), pp. 9-10.
\textsuperscript{450} Mulligan & Lee (2012), pp. 1-3.
\textsuperscript{451} Mulligan & Lee (2012), pp. 7-8.
\textsuperscript{452} Mulligan & Lee (2012), p. 7.
\textsuperscript{453} Mulligan & Lee (2012), p. 18.
Actually, discovery costs in ICT industries are sometimes considered to be so high that most companies do not even try to conduct a prior art search, because they have no cost-effective way of obtaining a complete list of relevant information sources in the first place. That is not, however, the only reason for companies not to conduct a prior art search. For instance, some companies deliberately do not perform prior art searches, because they assess the risk of infringement as high because of the crowded nature of the field, and then address inevitable infringements as they occur, often with other patents in their portfolio as intended bargaining chips (cf. defensive strategy). Also if the technology is changing rapidly, as in the case of the software industry where the average life cycle of a software product is relatively short, companies may not even try or bother to check for infringements. In so doing they run the risk of reinventing already existing inventions and incurring the costs and the lost productivity associated with that. On the other hand, in an environment where a company’s competitors actively enforce their patents (cf. offensive strategy), the need for a prior art search rises, since it affords the opportunity to deal with a likely threat early and perhaps on better terms than otherwise. In the end the decision about whether to conduct a prior art search is a judgment call for the situation. The starting point, however, given the overall strategic advantage of knowing what is out there versus not knowing, should be that any decision not to conduct a prior art search in any given circumstance should be accompanied by a very good reason for not doing so.

5.3 Exclusive Rights Conferred by a Patent

Any patent owner is entitled to his exclusive rights. However, if you are shy about catching infringers, the infringer will end up being the party making profits out of your invention. In fact, nothing will happen and the infringement will continue unless you affirmatively do something about it. Therefore, the patent owner must assume the full burden for stopping the infringer and obtaining damages. The exclusive rights you gain from obtaining your patent is more about what other people cannot do than what

455 See chapter 4.4.2.
456 See chapter 2.1.2.
457 See chapter 4.4.3.
460 Ma (2009), p. 194.
you can do. This, again, emphasizes the strategic function of patents and the importance for a company to understand the exploitation possibilities regarding software patents. What constitutes an actual infringement of a patent is dictated by the laws of the country, here a member state of the EPC, in which the patent is granted. However, there are some common ways in which patents can be infringed.

Where the subject-matter of a patent is a product, a patent gives the owner the right to “exclude others from making, using, selling, offering for sale, or importing the invention as set forth in the claims of the patent”. In a case where the subject-matter of a patent is a process, the owner has the right to “exclude third parties from the act of using that process, and from the acts of using, offering for sale, selling, or importing the products directly obtained by such process”. These acts constitute patent infringement if there is neither owner’s consent nor a defence present. These are the minimum standards for infringing conduct and they are relatively harmonized by the TRIPS Agreement, and most member states of the World Trade Organization (WTO) prohibit such conduct as infringing conduct. However, an ultimate finding of infringement depends on what national laws prohibit. This is due to the principle of territoriality, which requires that the infringing act must be done within the territory of the country where a patent is in force.

Thus, in Europe, patent infringement of both national patents and European patents are essentially dealt with by national laws. However, the EPC does contain certain rules determining the scope of protection which are binding also in the post-grant phase, i.e. at the national level in the infringement proceedings. Namely, Article 69(1) of the EPC stipulates that the extent of the protection conferred by a European patent is determined

---

462 Ma (2009), p. 10.
463 TRIPS Article 28. See also Knight (2013), pp. 155-156.
464 A defence here refers to limitations and exceptions that are exempted from the exclusive right conferred by a patent. The most common defences are related to acts done privately and for non-commercial purposes and acts done for experimental purposes. For further details about limitations and exceptions, see e.g. Kur & Dreier (2013), pp. 118-120.
466 “Exceptions to this principle do exist and sovereign nations bilaterally or multilaterally recognize and sometimes enforce the corresponding rights of other nations based upon the laws of those countries through international conventions and treaties. Thus, without international treaties and conventions recognizing foreign intellectual property rights, patents are only protected within the territory where they have been granted. The laws of the granting country define and determine what is prohibited as infringement of this right. Thus, exploitations or infringement cannot extend beyond the territory of that country. In the case of cross-border transactions, conflict of law and private international law determine jurisdictions and applicable laws.” See Lee (2010), pp. 23-24.
by the claims, and that the description and drawings are used to interpret the claims.\textsuperscript{467} In consideration of the fact that Article 69 of the EPC is of crucial importance for the scope of the exclusive rights conferred and hence for the economic impact of patent protection, and realizing that member states had different traditions in measuring the scope of patents, guidelines for the interpretation of Article 69 were set out in a Protocol to the EPC\textsuperscript{468, 469} Therefore, European patents can also be characterized as semi-national patents.

The central issue with patent infringement is whether or not the competitor’s activities fall within the scope of the claims of the issued patent in any particular country.\textsuperscript{470} According to Lee, “This again emphasizes the significance of claim drafting, as the claims will determine not only the scope of the protection of an invention but also which parties are liable for patent infringement, and which conduct is prohibited.” In other words, the function of the claims as a legal document rather than as a technological disclosure document becomes more highlighted. However, without engaging in a complex claim construction, the public in general as well as the person skilled in the art may not be certain of what is allowed and what would be prohibited, and even most infringers may not be aware of the fact of infringement.\textsuperscript{471}

Although it was already pointed out in the introduction chapter, as a reminder: because the definition of patent infringement may vary by jurisdiction, it is not convenient within the frame of this thesis to exhaustively discuss the specifics of patent infringement or the scope of exclusive rights and how they are defined and interpreted in national laws and courts of different EPC member states. Therefore, the term \textit{patent infringement} referred to here covers all such acts that theoretically can constitute patent infringement in the case of a European patent. Furthermore, this means that no division is done for instance when it comes to infringement types or liabilities or national rules. In addition, it is recognized that the Internet causes specific problems regarding the choice of law in some software patent infringement cases, but discussion regarding these issues are also excluded from the scope of this paper.

\textsuperscript{467} Kur & Dreier (2013), p. 115.
\textsuperscript{468} Protocol on the interpretation of Article 69 EPC of 5 October 1973, as amended by the Act revising the EPC of 29 November 2000.
\textsuperscript{469} Kur & Dreier (2013), p. 115.
\textsuperscript{470} Gollin (2008), pp. 260-261.
\textsuperscript{471} Lee (2010), p. 22.
5.4 Initial Monitoring Phase

5.4.1 General Issues and the Problem of Detectability

As a reminder, patent infringement monitoring is a post-grant act with the goal of detecting acts that are likely to infringe on the granted patent. The importance of patent infringement monitoring is crucial, because patent infringement can be punished only if it is first identified, and it is not, like certain other infractions, monitored by government or any other body\textsuperscript{472}. Therefore, the duty of monitoring falls on the shoulders of the patent owner or some other person, depending for instance on resources and who has the best understanding of the scope of the patent. The most effective methods of infringement monitoring often vary depending on the field of business and the rivals of the company. In addition, the nature of the invention, resources, patent exploitation strategies and the know-how and abilities of the personnel also affect whether or not to monitor at all or how extensively one should conduct monitoring efforts.

Obviously, in some fields of technology it is much harder to detect infringements than in others. If the detection is considered to be easy, for instance because patents are clearly defined and non-abstract and because of the low number of patents and rivals in the field\textsuperscript{473}, it will definitely deter infringers and lure them into accepting a license of the patent\textsuperscript{474}. However, if the detection is difficult, the situation is reversed. The difficulty to detect potential infringements can be a consequence with multiple reasons, but usually the right holders are not able to detect infringements because of the nature of their invention or they have the problem of being sure whether or not the scope of their patent covers the potential infringement. This kind of situation is quite often at hand in the case of software patents and in the ICT industry in general, where it may be rather difficult to detect infringing acts or determine whether or not the scope of the patent really covers suspected infringement\textsuperscript{475}.

Due to the nature of the invention, some software patent infringements are easier to detect than others. For instance, patents that focus on the overall function of a software application, relate to graphical user interfaces, or define data structures can be spotted

\textsuperscript{472} Guellec & van Pottelsbergh de la Potterie (2007), p. 58.
\textsuperscript{473} However, not necessarily even these characteristics guarantee the ease of detection.
\textsuperscript{474} Ma (2009), p. 200.
\textsuperscript{475} Davis & Kjær (2003), p 20. See also Mann (2005), p. 979.
more easily than patents relating for instance to algorithms used deep in the bowels of a software application.\textsuperscript{476} In other words, software patents that have publicly observable features are generally easier to detect. Therefore, it is not a surprise that most patent lawsuits involving software appear to involve publicly observable features.\textsuperscript{477} However, most software patents suffer from the problem of detectability, because patents for software-related inventions are often directed to the inner workings of computer hardware or software, which may not be readily apparent from the operation of an infringing device.\textsuperscript{478}

Reasons why it may be difficult to detect infringing uses and to determine whether or not the software-related patent is being infringed can be manifold. First, on a general level, detecting software patent infringements is complicated by the fact that to initially uncover infringers or identify infringed products it often requires highly technical skills (e.g. reverse-engineering) of someone understanding the patent (the scope and interpretation of its claims) and having a vision of the patent’s applicable uses (to know how and where to search for potential infringements).\textsuperscript{479} Second, problems arise when an embodiment of an idea is expressed in a language that is compiled and distributed as object code, then detection is much more difficult than in a case where source code is readily available. To produce evidence of patent infringement often requires analysing executable code, which again requires reverse-engineering of the said code. However, reverse-engineering is not only a technically difficult art to master but also the legality of such an act for the purposes of infringement detection is rather questionable under the European legal framework. Reverse-engineering in the context of software patents will be discussed more extensively in chapter 5.5.

Third, the embodiment chosen by an infringer, especially if the infringer is simply unaware of the software-related patent, is likely to be very different from the embodiment protected by the patent. Put simply, you might not be able to identify your invention in someone else’s source code, but much less in the compiled code. In other words, sometimes, an independently created embodiment of your software-related invention may be almost impossible to recognize although you have the software’s

\begin{footnotes}
\footnote{Krajec (2004); Elyjiw (2008).}
\footnote{Bessen & Meurer (2009), pp. 124-125.}
\footnote{Elyjiw (2008).}
\footnote{Cf. Ma (2009), p. 194.}
\end{footnotes}
source code in front of you. This problem is mainly due to the divergent nature of programming.\textsuperscript{480} Fourth, problems arise also when infringement takes place for instance in a custom-written software or in a cloud service which is located behind a firewall. In the case of custom-written software, the patent holder has little opportunity to learn of the existence, much less the characteristics, of the program\textsuperscript{481}. Moreover, if the infringing activity takes place in a cloud service and behind a firewall, the infringement might not only be difficult to detect but also analysing and collecting evidence of the infringement becomes nearly impossible\textsuperscript{482}.

5.4.2 Modified Prior Art Type of Search

Impetus to start a patent infringement monitoring process can come from different contexts, for instance it can be dictated by the company’s patent strategy. Even though one can be passive in his monitoring efforts and still sometimes accidentally come across infringing technology, the most common way to detect infringements is to actively search for infringing acts and to screen competitors by conducting targeted monitoring efforts. There is no set procedure how and in which situations one should conduct patent infringement monitoring. Though, with extensive electronic capabilities now available, a search theoretically has become more straightforward, and therefore, one way to start patent infringement monitoring is to conduct a similar type of key-word search that was earlier discussed in the context of a prior art search (chapter 5.2.2). From this point of view, one can think of this phase of patent infringement monitoring (initial monitoring phase) as a manifestation of a prior art search with minor modifications. The basic idea in this type of search is to get a “hunch” about who else might have been utilizing your invention in a manner that possibly infringes your exclusive rights.

However, the nature of software creates multiple barriers for detecting infringement, and therefore for the person conducting monitoring. First, the barriers referred to above relate to problems that are present when searching for software-related information from

\textsuperscript{481} Nichols (1998), p. 118.
\textsuperscript{482} This is due to the fact that in order to analyse and reverse-engineer the infringing software would require hacking through the firewall, which is illegal. Also, the location of software could be unknown, which means that there is no certainty where the potential infringing activity takes place.
both patent and non-patent sources. These problems are mostly congruent to ones discussed earlier in the context of a prior art search (the problem of finding relevant prior art, the problem of coming up with good key-words, and the problem of discovery costs). Therefore, they are not fully re-discussed here, but the reader is asked to, if necessary, return to chapter 5.2.2. Second, as pointed out in chapter 5.4.1, barriers may arise when an embodiment of an idea is expressed in a language that is compiled and distributed as object code, then detection is much more difficult than in a case where source code is readily available. Also, the embodiment chosen by an infringer is often likely to be very different from the embodiment protected by a patent, which further makes it harder to detect the infringement.

In the context of a prior art search it was discussed that a searcher needs to be proficient when searching for information from different sources. What sources to search through is dictated by what type of search is required, the legal and financial implications of the search, how much time to complete it and how much one is willing to pay to get the information needed. These principles are also relevant and should be acknowledged when discussing patent infringement monitoring. Furthermore, when it comes to using a modified prior art type of search, there obviously are some differences when compared to the “normal” prior art search. First, since patents are territorial and the focus of this thesis is on European patents, a modified prior art type of search needs to cover only the areas where the European patent is currently in force. Therefore, you should not search for infringements from countries where your software patent cannot be enforced. This is simply due to the fact that even though you detect an infringement, the infringement has taken place in a country where your patent is not in force, you cannot enforce the patent because of the principle of territoriality.

Second, because claims define the invention and infringement is determined primarily by the literal language of the claims, it is strongly advised that the search is carried out based on the claims of the patent. Furthermore, the text-based search and key-words for the search should be formulated according to what the claims state plus their

---


484 It should be noted that “the doctrine of equivalents empowers courts to construe patent claims to cover not only that which they literally describe, but also some range of equivalent subject-matter that technically falls outside the literal claim language but on policy grounds seems appropriately considered part of the patent holder’s exclusive domain”. For further details, see Lichtman (2004), p. 152.
equivalents. However, as established in chapter 5.2.2, it is very challenging to come up with good or descriptive key-words. Third, there is a difference between the search objectives of these two types of searches. When a prior art search is conducted its goal is to help the searcher to determine whether or not an invention can be patented, and therefore the searcher focuses on the questions of novelty and inventive step. However, when a search is conducted in order to locate potential infringements, the goal is to find infringing acts or products and not to compare your software patent to all relevant state of the art in order to decide the patentability of your software. Therefore, the patent infringement search can for instance be further narrowed down by targeting certain companies (e.g. screening rivals) or geographical areas (e.g. the most important market area where the European patent is in force), depending on resources available and goals established for monitoring patent infringement. This type of conduct can be described as targeted monitoring or targeted searching.

5.5 Collecting and Analysing Evidence
5.5.1 Forewords

Once your initial monitoring efforts, for instance a modified prior art type of search, have resulted in promising findings, it is time to proceed with the investigation process. Of course, a hunch that someone’s product or behaviour is infringing your patent is simply not enough to prove infringement, but it helps the company to prepare in advance to take further appropriate action. If you discover what you believe to be an infringement of your patent, you should obtain as many details and particulars about the infringing product or act and infringer as possible.485 Thus, uncovering infringement requires collecting evidence, analysing the infringing product or act, and documenting your findings. Collecting and analysing evidence and developing your proof are essential steps in making your investigation productive and effective so that it may help you, if necessary, to take the next step – for instance launch any successful assertion or litigation against infringers.486

In a patent infringement action, the claimant carries the burden of proof. He has to adduce evidence of the infringement and persuade the court that what the defendant has

486 Ma (2009), p. 194.
done has infringed his exclusive rights. The evidence of infringing use can come from many sources such as the operation of real products, reverse-engineering, information from the Internet, patent applications, journals and published papers and product brochures. What should be noted is that collecting evidence must be done legally, and this will be discussed later more extensively in connection with reverse-engineering. After the evidence has been collected it must be analysed and compared with the claims of your patent to determine infringement. In the case of software, analysing the evidence is often carried out as an action that can be referred to as software forensics.

Software forensics is the examination of software for the purpose of producing evidence of an infringing act. The objective of software forensics is to find evidence for a legal proceeding or settlement by examining the literal expression and the functionality of software. Software forensics requires a knowledge of the software, often including things such as the programming language in which it is written, its functionality, the system on which it is intended to run, the devices that the software controls, and the processor that is executing the code. A software forensic examiner must look at code that has similar functionality even though the exact representation might be different. However, if the suspected product is commercially available to the public then it is likely that the invention appears as compiled executable code and the object code will be highly optimized. Of course, a potential infringer in these situations is highly unlikely to provide access to its source code, especially for the purpose of inviting a patent infringement lawsuit.

Because proprietary software is nearly always distributed in object code form, software typically cannot be readily understood by casual inspection, and particularly not without access to human-readable source code or other documentation. Although the patent holder may come to suspect that a product is infringing by observing externally visible

489 Although discussion regarding claim interpretation is mainly excluded from this thesis, it is discussed briefly in chapter 5.7.
490 Software forensics should not be confused with digital forensics, which for instance is used to recover software from a computer system or computer storage media so that a software forensic examination can be performed. Unlike digital forensics, software forensics is involved with the content of the software files, whether those files are object code files or readable text source code files. See Zeidman (2011), p. 120.
491 Zeidman (2011), p. 120.
features, obtaining proof will still usually require reverse-engineering the object code of the suspected product. Furthermore, as pointed out by Burk and Lemley, “Examination of the patent itself is unlikely to yield information equivalent to a reverse-engineered because would-be patentees of software-related inventions are not required to disclose the implementing source code or, for that matter, very much at all about their inventions.” Therefore, the patent holder may be forced to resort to alternative methods for detecting infringement and collecting evidence, such as reverse-engineering the product he assumes to infringe his exclusive rights.

5.5.2 What is Reverse-Engineering?

Although computer programs are initially written in source code, i.e. a language intelligible to people, software companies typically use a compiler to translate the program into code that will run on a computer, often referred to as executable code or object code. Because software companies disseminate their programs in the form of unreadable object code, one must often reverse-engineer the object code version of the program in order to further analyse the program and to produce evidence of possible infringement. In addition to producing evidence and identifying possible patent infringements, reverse-engineering may be pursued for various other reasons as well, amongst which are interoperability and product analysis to examine how software works. In the context of software, reverse-engineering has been described as the process of analysing a subject system to create representations of the system at a higher level of abstraction. Thus, reverse-engineering is a technique whereby a technology is being analysed in order to understand how it was designed and how it operates.

Commonly employed reverse-engineering techniques include the following: studying available manuals and documentation; using intermediate empirical processes such as observing the program in operation through the use of test inputs and monitoring inputs and outputs from a program or displaying on a computer screen or in a printout of the

---

494 See e.g. Elyjiw (2008).
495 This assertion refers to proprietary software. Software released under the Open Source model is released in its source code version.
contents of the computer memory (the so called black-box analysis or black-box reverse-engineering); and the more complex techniques of disassembly and decompilation (sometimes referred to as white-box reverse-engineering).\(^499\) Put simply, black-box reverse-engineering often refers to a process where the operations of programs are observed without examining their internal structure.\(^500\) White-box reverse-engineering or more commonly decompilation, again, involves an inspection of the system’s internal organs.\(^501\) This happens by compiling the object code into human-readable form, to a form that also can be called “pseudo source code”\(^502\). In essence, the main idea of decompiling is to analyse a software product by taking it apart in order to ascertain the methods, components and logic that were used when the said product was designed and assembled.\(^503\)

According to van Rooijen, the relevance of black-box reverse-engineering should not be underestimated, as it could, in some instances, produce a comprehensive set of interfaces, thereby obviating the need for decompilation.\(^504\) However, Shemtov states that black-box analysis done through observing the program in operation as a means of accessing the totality of ideas embodied in a computer program is a bogus technical myth that needs to be confronted. It consists of no more than an educated guess as to how the program’s coding is structured and it fails to ascertain information as fact.\(^505\) Moreover, in comparison to other available methods of reverse-engineering, there is little doubt that decompilation provides the most and often the only effective means, albeit far from perfect\(^506\), of reverse-engineering a computer program in order to gain access to its logical building blocks.\(^507\) The definition of reverse-engineering that is used from here on out is to be understood so that it only encompasses decompilation, and thus techniques regarding black-box analysis are not included in this definition if not otherwise mentioned.\(^508\)

\(^{504}\) van Rooijen (2010), p. 83.
\(^{505}\) Shemtov (2012), p. 15.
\(^{506}\) In sum, software reverse-engineering is still very far from being capable of granting easy or relatively cheap access to application programming interfaces, communication protocols and to algorithms and several kinds of ideas embodied in compiled object code. See more closely e.g. Morando (2008), p. 28.
\(^{507}\) Shemtov (2012), p. 16.
\(^{508}\) Actually, one quite common mistake that one comes across in legal literature is that reverse-engineering is, without knowing the real difference between different reverse-engineering techniques,
The fact that compilation from source code to object code always loses information and comments written by the programmers, often lowers the value of the decompilation process. This is because there are dozens and perhaps hundreds of known code transformations that may be performed by an optimizing compiler; these transformations rearrange, substitute, or even eliminate the original code. Therefore, loops can disappear, replaced by blocks of code repeated over and over. Instructions that were in the middle of the source program may be moved to the beginning or end of the executable version. The compiler may take your patented algorithm and scatter it unpredictably among millions of unrelated instructions. The result is sometimes so dissimilar to the source code that in many cases it is impossible even to step through optimized code with a symbolic debugger. The feasibility of reverse-engineering in a given situation depends on the nature of the initial source code, the compiler used to generate the object code, and the availability of specialized software tools.

Decompiling object code is not only a difficult way of examining the structure of software applications, but it is also expensive and raises questions regarding the legality of such act. The cost of reverse-engineering might be justifiable as likely paling in comparison to the cost of doing nothing, in terms of potential lost revenues resulting from an infringing competitor product. Of course, one should remember that collecting and analysing evidence must be done legally. In essence, this means that one should obtain access and analyse the potentially infringing software in a way that does not infringe the right holder’s exclusive rights. Even though the focus of this paper is placed on software patents and thus the exclusive rights conferred by a patent, reverse-engineering a software product also triggers questions regarding the legality of such an act from the copyright perspective. Next, the discussion proceeds into the legal aspects of reverse-engineering, i.e. decompilation, because in Europe it is a practice that is not only referred to as techniques that cover decompilation. In other words, reverse-engineering is often used as a synonym to decompilation without explaining to the reader that it fundamentally covers also the acts regarding black-box analysis.

512 The costs of examining software to find out if it contains patented methods are very high. Software code is often so large and so complex that for most programs, such a search would be almost prohibitively expensive. Although this may be true, software analysis can be automated to significantly speed up and lower the cost of such efforts, whereas mechanical devices must be examined manually to find potential infringement. See Zeidman (2011), p. 92.
only heavily restricted and regulated under copyright law but its legality under patent law is to some extent unclear.

5.5.3 Reverse-Engineering under the European Software Directive

In Europe, the Software Directive sets out the legal framework for reverse-engineering from the copyright point of view. The Software Directive supports two species of reverse-engineering: black-box analysis and decompilation. These forms of reverse-engineering differ from a technical perspective: black-box analysis is limited to an observation of the exterior of the “box” (the computer program), whereas decompilation aims to reconstruct its interior (the source code). This difference also triggers different copyright-relevant acts and is, therefore, addressed by two distinct provisions in the Software Directive. Namely, Articles 5(3) and 6 of the Software Directive respectively carve out exemptions for black-box analysis and decompilation. Note that, by virtue of Article 8 of the Software Directive, neither of the provisions referred to above can be annulled by contract.

Article 5(3) defines ideas susceptible to free use rather broadly, as it does research processes aimed at discovering them. The provision entitles “a person who has the right to use a copy of a computer program to observe, study, or test the functioning of the program in order to determine the ideas and principles that underlie any element of the program”, if he does so “while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.” It is undisputed that Article 5(3) allows lawful end-users to study the working of the

515 By stipulating that a developer must have “a right to use a copy of the computer program” he seeks to analyse, Article 5(3) prevents the use of an illicit copy of the program. Furthermore, pursuant to Article 5(1) of the Software Directive, a “lawful acquirer” of a computer program does not need an authorization by the right holder to use the program in accordance with its intended purpose. As a curiosity it can be mentioned that in UsedSoft GmbH v. Oracle International Corp, the CJEU had to address the question of whether a buyer of a used computer program qualified as a “lawful acquirer” as such and could thus use the program without authorization. The court stated that each purchaser of a copy of software for which the distribution right is exhausted is a “lawful acquirer” within the meaning of the Software Directive. For further details, see UsedSoft GmbH v Oracle International Corp (Case C-128/11) [2012] ECDR 19, at § 80.
516 In a recent case, SaS Institute Inc v. World Programming Ltd, the CJEU stated, similarly as Advocate General Bot, that “to accept that the functionality of a computer program can be protected by copyright would amount to making it possible to monopolise ideas, to the detriment of technological progress and industrial development.” More closely, see Case C-406/10, at § 40. Cf. Morando (2008), p. 34.
program “from outside”, i.e. to perform black-box analysis.\footnote{In the case, \textit{SaS Institute Inc v. World Programming Ltd}, the CJEU also concluded that “a person who has obtained a copy of a computer program under a licence is entitled, without the authorisation of the owner of the copyright, to observe, study or test the functioning of that program so as to determine the ideas and principles which underlie any element of the program, in the case where that person carries out acts covered by that licence and acts of loading and running necessary for the use of the computer program, and on condition that that person does not infringe the exclusive rights of the owner of the copyright in that program.” For further details, see Case C-406/10, at § 62.} Though, black-box analysis merely entails running the computer program similar to (although more intensely than) running the program for normal use\footnote{According to Lai, black-box analysis, for the purposes of Article 5(3), may not extend to the running of line traces (an engineer causes messages to be transmitted from the analysed program to another program or device and uses a line tracer to determine how the program interacts or functions with other programs or devices). For further details, see Lai (2000), p. 99.}. Article 5(1) of the Software Directive already safeguards the latter reproductions for lawful end-users, and the limitation expressly permitting black-box analysis – Article 5(3) – is, therefore, primarily a clarification of this limitation.\footnote{Morando (2008), p. 33; van Rooijen (2010), p. 82.} However, if the analysis goes beyond the search for ideas and principles behind the functionality of the program, and the objective is to derive specific information or protected expression in the form of code, then it is Article 6 which provides the platform for such further examination.\footnote{Lai (2000), p. 99.}

The legal position in the EU regarding decompilation of computer programs is fairly straightforward as it is codified under Article 6 of the Software Directive. However, the practice of decompilation is highly regulated in the said Article and is only allowed for the limited purpose of achieving interoperability. Furthermore, Article 6 also sets a tight framework for the interoperability-rerequirement by stipulating that three additional conditions must be fulfilled in order to carry out decompiling within the scope of the Article. First, the act must be performed by a licensee or a lawful user of the program. Second, the information sought must not be available to the party carrying out the act through any other source. Finally, the act of decompilation must be confined to the parts of the program necessary to achieve interoperability. Moreover, Article 6 adds the caveat that the information obtained shall not be used for goals other than to achieve the interoperability of the independently created computer program\footnote{It should be noted that Article 6 does not appear to sanction decompilation with a view to assist future compatibility. More closely on this matter, see e.g. Shemtov (2012), p. 148.}; to be given to others, except when necessary for the interoperability of the independently created computer

\footnote{In the case, \textit{SaS Institute Inc v. World Programming Ltd}, the CJEU also concluded that “a person who has obtained a copy of a computer program under a licence is entitled, without the authorisation of the owner of the copyright, to observe, study or test the functioning of that program so as to determine the ideas and principles which underlie any element of the program, in the case where that person carries out acts covered by that licence and acts of loading and running necessary for the use of the computer program, and on condition that that person does not infringe the exclusive rights of the owner of the copyright in that program.” For further details, see Case C-406/10, at § 62.}
program\(^{523}\); or to be used for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright\(^{524}\).

As we can deduce from these provisions, where Article 5(3) excuses observation, study, or testing to determine the ideas and principles that underlie “any element of the program”, Article 6 excuses the discovery only of “parts of the original program which are necessary to achieve interoperability”.\(^{525}\) The fact that decompilation is limited to the parts which are required to achieve interoperability might be problematic as on occasions the person who conducts decompilation might not know in advance in which parts of the program the interoperability information is located. Namely, sometimes it is technically complex and unreasonably burdensome, if not impossible, to try to identify the location of the relevant portions of the decompiled program.\(^{526}\) Therefore, it should be reasonable to allow a reverse-engineer to decompile larger portion of the code than the “parts of the original program which are necessary” in his attempt to uncover the interface specification information.\(^{527}\) However, if a person under the current legislation chooses to do this, the liability exception under Article 6 will not come into play.

As noted, the practice of decompilation is strictly regulated in the EU and is only allowed for the limited purpose of achieving interoperability. The fact that the Software Directive does not permit a person to decompile a program for purposes unrelated to interoperability has been criticized by many scholars.\(^{528}\) Furthermore, because decompilation is restricted to the sole purpose of achieving interoperability, decompilation for instance for the purposes of detecting copyright infringement is to be considered illegal under the current Software Directive. This is a big concern regarding collecting and analysing evidence and makes it rather difficult for a right holder to

---

\(^{523}\) Article 6(2)(b) seems to prevent competitors from sharing the effort and cost of the decompilation. For further details, see Shemtov (2012), pp. 147-148.

\(^{524}\) See also Goldstein (2001), p. 299.

\(^{525}\) Goldstein (2001), p. 298.


\(^{527}\) Shemtov proposes the following solution: “…it is suggested that a reasonableness criterion, similar to the one present under Australian legislation, should be introduced in this context.” Shemtov (2012), p. 149.

\(^{528}\) According to some technologists, “the sensible goals of decompilation could essentially be: achieving direct interoperability; achieving indirect interoperability; understanding why some bugs are arising; security purposes; and verifying if a copyright violation is taking place”. More closely, see Morando (2008), p. 44. See also Leith (2011), p. 179 and Shemtov (2012), p. 135.
figure out whether or not infringement is occurring. Therefore, as pointed out by Leith, a right holder who suspects infringement and decides to decompile the potentially infringing software may himself have to carry out copyright infringement in order to determine whether or not the infringement has actually taken place.

5.5.4 Reverse-Engineering under European Patent Law

5.5.4.1 State of Affairs

As discussed in chapter two, theoretical background, patents and copyright are two independent forms of protection of computer programs, which can coexist. A certain technical solution may be protected by one or more patents, while each of its implementations (in the form of source and object code) is normally protected by copyright. This also means that exceptions to each form of protection could apply only if the allowed uses are not protected by the other legal tool or if there is an exception in both fields. In other words, exceptions covering patents can be enjoyed, but only if they do not violate copyright and vice versa. To be more precise, even though an exception provided under patent law would apply to reverse-engineering software from the patent law perspective, there might not be such an exception under the copyright law that applies, and therefore it is possible that the same act accepted under patent law can still constitute copyright infringement.

The above presented indicates that in Europe, because of the Software Directive and the dual nature of software, decompilation in the case of software patents always constitutes at least copyright infringement, if the exception codified under Article 6 of the Software Directive does not apply. However, another question is when decompilation in the case of software patents constitutes patent infringement. When discussing the possibility of copyright and patent infringement in the case of decompilation of software-related inventions, it seems that there are hypothetically four scenarios at hand:

---

529 Due to the particular nature of a software and hardware environment, the process of decompilation involves, by definition, copyright infringement. See Shemtov (2012), p. 24.
532 In other words, one cannot enjoy the exception provided under patent law, or he can, but not without simultaneously constituting copyright infringement.
1) copyright infringement and patent infringement (no exceptions under either law apply);
2) copyright infringement (exception under patent law applies but not under copyright law);
3) patent infringement (exception under copyright law applies but not under patent law);
4) no infringement (exceptions under patent law and copyright apply).

In a previous chapter we discussed when decompilation is accepted under the Software Directive, but here the focus shifts to patent law and its exceptions as the goal is to examine whether or not decompilation is allowed under European patent laws. In Europe, reverse-engineering patented software by decompiling it likely fits within the exclusive rights of the right holder. According to Burk and Lemley, “Reverse-engineering clearly constitutes a use of the patented software, though owners of a particular copy of the program surely have the right to use it. More significantly, decompilation may also constitute making the patented program by generating a temporary yet functional copy of it in RAM memory and, in certain instances, a longer-term (though still intermediate) copy in more permanent memory.” 533 Those copies might be found to be infringing patent law unless protected by some defence. Defences (limitations and exceptions) are not addressed in the EPC and thus are not harmonized. However, many limitations and exceptions were included in the Community Patent Convention (CPC) and have therefore served as models for national legislation due to the phenomenon of “cold harmonization”. 534 Therefore, most member states of the EPC have similar limitations and exceptions in their national legislation. 535

**Prima facie**, in most European jurisdictions reverse-engineering software through decompilation seems to fall under “the experimental use” privilege 536 537. At the general
level it has been argued that decompilation of patented products would be allowed under the experimental use exception in European jurisdictions. In addition, it has been said that the experimental use exception covers at least situations where decompilation has been conducted in order to achieve interoperability. The interest here, however, lies in the question of whether or not decompilation of patented software is allowed for the purposes of infringement detection. The general justification for the experimental use exception has been that it is not in the public interest to allow patent owners to prevent acts which aim at advancing technology. In other words, the experimental use is tightly connected to the main objective of the patent system in fostering technological development. With that goal in mind, decompiling patent protected software for the purposes of detecting infringements do not seem to be in the heart of the said justification. Therefore, situations where reverse-engineering is done without the aim to develop technology, is rather questionable when discussing the extent of the experimental use privilege.

Also a critical requirement in the experimental use exception might be that the accepted experiments need to relate to the subject-matter of the patented invention. This, again, raises similar issues regarding the so called “parts” problem as discussed in the context of Article 6 of the Software Directive (chapter 5.5.3). According to the said Article, decompilation is limited to the parts which are required to achieve interoperability, and might be problematic because not in every situation does the person who conducts decompilation know in advance in which parts of the program the interoperability information is located. In the context of software patents the subject-matter of the patented invention can be seen as “parts” of software. Is one, then, entitled to decompile also the “other parts” which do not relate to the subject-matter of the patented invention? According to Mylly, the term “relate” has been construed quite broadly in cases dealing with other technological areas. Therefore, the differences

---

defined above relating to which part of a program is patented may be too fine\textsuperscript{541}, and thus it may be acceptable to reverse-engineer other parts as well.

Furthermore, the literal interpretation of the requirement of “relate to the subject-matter of the patented invention” would lead to illogical outcomes. This is mostly due to the fact that experiments towards (decompiling) unprotected information also always require making (copying) the protected information which is residing in the very same product. Therefore, as Mylly points out, one could argue that allowing reverse-engineering in all situations illustrated above would be the most rational interpretation even though it might require stretching the interpretation of the experimental use exception.\textsuperscript{542} If all the factors presented above are summarized, the answer to the question of whether or not decompilation of patented software is allowed for the purposes of infringement detection is that the issue is definitely not clear-cut but rather open to different interpretations. Furthermore, because neither the question of commercial end\textsuperscript{543} nor the requirement of “relate to the subject-matter of the patented invention” do not seem off hand to prohibit decompiling in the scenarios presented above, the acceptance of decompilation appears to boil down to the question of justification. Therefore, it can be argued that decompiling under the experimental use exception is possible if infringement detection is deemed to be a sufficient justification for the said exception.

5.5.4.2 Future Aspects of Reverse-Engineering

Although limitations and exceptions are not addressed in the EPC, a catalogue of limitations and exceptions is now contained in the proposal for unitary patent protection. The aim of the proposal is to provide the same scope of patent protection in all participating member states. This indicates, among other things, a harmonization of the patent law exceptions within the region. Furthermore, pursuant to the current proposal, acts and use of information as allowed under Articles 5 and 6 of the Software Directive shall be exempted from the exclusive rights conferred by a patent. The new “software exception” will probably mean several restrictions for decompilation under

\textsuperscript{543} The question of commercial end does not seem to be a problem \textit{per se} when assessing the experimental use exception for the purposes of infringement detection. Cf. Mylly, U.-M. (2012), p. 382.
patent law, but it only covers situations where uniform protection has been applied for through the EPO. Therefore, national patent law exceptions will also in the future be relevant when the applicant has applied for a national patent or a bundle of national patents through the EPO.\textsuperscript{544}

Under the proposal, acts of reverse-engineering are allowed only under certain circumstances. The proposed software exception refers to Articles 5 and 6 of the Software Directive, and in essence this means that similar issues are present as discussed in the context of copyright (see chapter 5.5.3). In short, the decompilation of a patented computer program will be allowed if, and only if, it is indispensable for the purposes of achieving interoperability. Furthermore, in the case of a unitary patent this naturally means that reverse-engineering would not be constructed under the experimental use exception anymore, which again brings us to a situation that there is not even a theoretical possibility to reverse-engineer an invention protected by a unitary patent for the purposes of detecting infringement. In this respect, it would be a better solution that a reverse-engineering right is formulated under the experimental use exception, as it would provide broader applicability of the exception\textsuperscript{545}. It still should be remembered that even though the proposal were to be successful the current exceptions will continue to be relevant for national patents and European patents whenever the patent applicant has not applied for unitary protection.\textsuperscript{546}

Mylly points out a relevant shortcoming regarding the reference technique used in the proposal (it refers only to Articles 5 and 6 of the Software Directive). Namely, Article 8 of the Software Directive stipulates that the right to reverse-engineer cannot be contracted out. However, this issue has not been regulated under the proposal. The references cover only Articles 5 and 6, not Article 8, which is relevant for contractual issues. But then again the list of exceptions in the proposal for unitary patent protection also contains exceptions, which have been deemed mandatory\textsuperscript{547}. According to Mylly, based on the nature of other exceptions of the proposal, one could argue that also the reverse-engineering right is a mandatory exception. However, the reference technique

\textsuperscript{547} Such as acts done privately and for non-commercial purposes and acts done for experimental purposes relating to the subject-matter of the invention (experimental use). See e.g. Kur & Dreier (2013), p. 118.
makes this somewhat difficult because the relevant Article in the Software Directive is not referred to in the proposal. Whether or not this is an omission by the drafters is unclear. Consequently, if one would be able to restrict or exclude the reverse-engineering right through a licensing agreement, it would dilute the aim of the proposed software exception. In sum, reverse-engineering in the case of unitary patents would be constructed under the new software exception and European patents, as they are today, under the experimental use exception. This would mean that, at least on a theoretical level, reverse-engineering of inventions covered by unitary patents would be accepted for more limited purposes than reverse-engineering of inventions covered by European patents, depending on how the experimental use exception is to be interpreted in the context of European patents.

5.5.5 Summary and End Notes Regarding Reverse-Engineering in Europe

As a conclusion, the following summary can be presented regarding reverse-engineering of software-related inventions under the current European legal framework (the EPC and the Software Directive):

Table 2: Reverse-engineering of software-related inventions under the current European legal framework

<table>
<thead>
<tr>
<th>Infringement type</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copyright and patent</td>
<td>In case the act of reverse-engineering does not fall under the</td>
</tr>
<tr>
<td>infringement</td>
<td>exception of Article 6 of the Software Directive or under the</td>
</tr>
<tr>
<td></td>
<td>experimental use exception, it constitutes both copyright</td>
</tr>
<tr>
<td></td>
<td>infringement and patent infringement. (scenario 1)</td>
</tr>
<tr>
<td>Copyright infringement</td>
<td>In case the act of reverse-engineering does fall under the</td>
</tr>
<tr>
<td></td>
<td>experimental use exception but not under the exception of Article 6 of</td>
</tr>
<tr>
<td></td>
<td>the Software Directive, it constitutes copyright infringement. (scenario</td>
</tr>
<tr>
<td></td>
<td>2)</td>
</tr>
<tr>
<td>Patent infringement</td>
<td>In case the act of reverse-engineering falls under the</td>
</tr>
<tr>
<td></td>
<td>exception of Article 6 of the Software Directive but not under the</td>
</tr>
<tr>
<td></td>
<td>experimental use exception, it constitutes patent infringement. (scenario</td>
</tr>
<tr>
<td></td>
<td>3)</td>
</tr>
<tr>
<td>No infringement</td>
<td>In case the act of reverse-engineering falls under both the</td>
</tr>
<tr>
<td></td>
<td>exception of Article 6 of the Software Directive and under the</td>
</tr>
<tr>
<td></td>
<td>experimental use exception, there is neither copyright nor patent</td>
</tr>
<tr>
<td></td>
<td>infringement. (scenario 4)</td>
</tr>
</tbody>
</table>

From the earlier presented it can be deduced that in Europe reverse-engineering in the case of software patents is heavily restricted and because of the strict requirements laid down in Article 6 of the Software Directive it almost always constitutes at least copyright infringement (see scenarios 1 and 2). Furthermore, reverse-engineering under the patent law is also a complex issue, and it is uncertain in which situations the experimental use exception provides the right to reverse-engineer computer programs. Most likely the experimental use exception allows reverse-engineering for the purposes of achieving interoperability, but whether or not it allows reverse-engineering for the purposes of infringement detection is rather unclear. Because it seems that under the patent law decompilation covers at least situations where it has been conducted in order to achieve interoperability and thus falls under the experimental use exception, the scenario in which reverse-engineering software-related invention results only in patent infringement is highly unlikely (scenario 3). Therefore, in theory, reverse-engineering for the purposes of interoperability appears to be the only scenario in which both copyright and patent infringement can be avoided (scenario 4).

Considering the European software protection regime and the nature of software patents, the detection of infringement by a software patent holder is extremely difficult even without a reverse-engineering prohibition. Therefore, the failure to extend a reverse-engineering privilege only to certain limited situations, i.e. for the purpose of achieving interoperability, will further erode the likelihood of detecting infringements as it is not possible to collect and analyse evidence in a purposeful manner. Thus, contrary to what one might initially believe, decompiling potentially infringing software is not accepted for the sole purpose of detecting infringement and analysing evidence. In other words, a right holder who suspects infringement and decides to decompile the infringing software for the purpose of infringement detection has to himself carry out copyright infringement or in some situations even patent infringement in order to determine whether the infringement has actually taken place or not. However, there is a route around this so called evidence-collecting problem, and that is to go outside of Europe and to reverse-engineer in countries which do not have similar restrictions regarding decompilation or do not have any legislation covering reverse-engineering\(^{549}\).

---

\(^{549}\) According to Leith, this type of a solution is particularly suitable for larger enterprises, but less suitable for SMEs. See Leith (2011), p. 179.
5.6 Patent Infringement Monitoring in the Context of Patent Exploitation Strategies

It was brought forth in chapter four that a patent strategy sets the stage for how the patent owner will develop, protect and exploit patents, but also patent infringement monitoring and enforcement are considered to be parts of patent strategy. The goal here is not to exhaustively discuss patent infringement monitoring in the context of a company’s IP and overall strategy, but rather in the context of patent exploitation strategies, it is emphasized that companies holding patents but without specific patent strategies should at least acknowledge the possibility of infringement monitoring. Furthermore, it is highlighted that if patent infringement monitoring is ignored or not properly understood some strategic aspects of the use of patents are limited or worthless. In other words, especially offensive strategies cannot necessarily be exploited in an intended manner or they cannot be utilized to a desired extent. Therefore, if a company aims at being effective in exploiting its patents, especially if patents have been obtained for offensive purposes, it should understand the importance of patent infringement monitoring and adopt a proficient monitoring strategy that is aligned with the company’s patent and overall IP strategy.

In the case of an offensive strategy (proprietary strategy and licensing strategy), the importance of patent infringement monitoring is often crucial if the goal is to effectively exploit patents. There are a few essential reasons why active monitoring serves a better purpose in the context of offensive strategies than in the context of defensive strategy. First, if software patents are used as protection for core product features and preventing others from utilizing patented innovations (proprietary strategy), then the patent owner should be able to detect when a competitor is unlawfully copying his innovation and take measures to enforce the patent\(^{550}\). In other words, if a company desires to use patents as an effective isolating and blocking mechanism where the goal is to maintain a competitive advantage in the marketplace by excluding competitors, it should also invest resources and be active in patent infringement monitoring. If a company does not actively monitor in these situations, then the value and function of patents that are not being monitored can be questioned in the sense that are they then really serving their purpose in a company’s proprietary strategy.

\(^{550}\) Krajec (2008).
Second, if the goal is to use patents for revenue generation or to push the other party to license their essential technology (licensing strategy), it is evident that patent infringement monitoring is something that cannot and simply should not be ignored without careful consideration. For instance, some companies exist largely for the purpose of licensing alone in order to generate revenues and are more aggressive in exploiting their patents. In these cases it is not sufficient for the right holder to rely on mere luck and wait for accidentally coming across an infringing act, but companies should rather actively seek potential infringers and enforce their rights accordingly. Furthermore, if companies in these situations were to ignore monitoring, they would not be able to effectively exploit the true potential of their patents. Therefore, in situations similar to those described above, companies should have a proficient infringement monitoring plan, which is aligned with the company’s patent and overall IP strategy.

In situations where a company seeks to use its patents to push other parties to license their essential technology, it would be advisable to use targeted rather than extensive infringement monitoring. Targeted monitoring is more convenient in these situations, because companies that seek to cross-license specific technology usually know which competitors possess the technology they want access to. Therefore, it would be advisable to monitor only certain companies and products, and if infringement is detected, then to take action and to try to push the other party to license the technology one desires. The goal of this type of monitoring can be for instance to get access to a patent pool or a certain technology. This type of monitoring strategy is mainly suitable in the case of an offensive strategy, but it can also be exploited in the context of a defensive strategy. For instance, if another company tries to create revenue at your expense by claiming that a patent in your defensive patent portfolio infringes their patent and demands you to pay license fees, you should conduct a targeted monitoring of the said party’s patent portfolio and in return try to find a patent that potentially infringes your exclusive rights.

Even though patent infringement monitoring activities are mainly amalgamated with offensive strategies, it is not to say that companies adopting defensive strategies cannot

---

552 Although, generally in the licensing domain, companies would be unwilling to license out patents with which they are pursuing a proprietary strategy and seek to commercialize the technology themselves. See e.g. Leaves (2010), p. 76; Somaya (2012), p. 1092.
take any action when it comes to monitoring software patent infringements. However, when software companies use patents defensively, the starting point is that these companies do not usually have a monitoring plan regarding defensive patents and they do not actively monitor for infringements. In addition, not being very active in detecting and monitoring patent infringements, these companies do not normally take action even if infringement is spotted\textsuperscript{553}. There are several reasons why companies adopting a defensive strategy are not very active on this front. First, patents in a defensive strategy are seen mainly as legal and bargaining instruments that ensure freedom-to-operate, not as business instruments where the purpose is to guarantee appropriate financial returns in the form of royalties. Second, the patents that have been obtained to serve defensive purposes are usually not activated unless another party forces the company to do so.

It stands to reason that if a patent is to produce revenue or to prohibit undesired conduct, as in the case of an offensive strategy, it must be possible to detect infringement using cost-effective and legally sanctioned means. However, detecting patent infringement to either license or stop unauthorized use can be rather expensive (discovery costs and reverse-engineering costs) and even legally questionable (reverse-engineering), which obviously set barriers for companies and affect their exploitation strategies. When the fact that most software-related inventions suffer from the problem of detectability is added to the equation (chapter 5.4.1), patent infringement monitoring in the case of software patents seems to be anything other than cost-effective. Therefore, instead of active and extensive monitoring efforts, targeted or even sporadic monitoring can sometimes be, if not the only, the most reasonable alternative. For instance, as discussed earlier in this thesis, some software patent infringements, due to the nature of the invention, are more easily detectible. This, again, would suggest that if a patent is more easily detectable, for instance because it focuses on the overall function of a software application, relates to graphical user interfaces, or defines data structures, then it should be monitored more extensively and actively than patents that are harder to detect, and therefore also resources should be targeted at monitoring more easily detectable patents.

It has been criticized by Mulligan and Lee that the current system regarding software patents resembles a lottery more than a system of property rights in the sense that only a minority of patent owners can “win” by catching infringers who will eventually pay up

\textsuperscript{553} See e.g. Soininen (2005), p. 78.
for their infringements. There is some truth in their critique, but there is also much that a company conducting infringement monitoring can do in order to improve its chances of catching infringers. Of course, by playing actively, putting time and effort and investing resources into monitoring activities, your chances of winning will improve. In another words, the likelihood of infringement detection often increases with the resources that the owner invests in monitoring. Therefore, small companies might be at a disadvantage when it comes to monitoring their patent rights. Large companies can spread the fixed cost of monitoring for infringement over a larger number of patents, but small companies might face liquidity constraints that discourage them from investing in monitoring. However, another question is how much the chances of winning will improve in relation to the sum played and the winnings expected. As referred to above, when the problems regarding software patents and their infringement monitoring are taken into account, we cannot really speak of a cost-effective system, which again suggests that in a case where one has a limited number of chips to play, one really should consider how and when to play in order to even expect some winnings.

On the other hand, when it comes to being passive and not playing, in other words not conducting infringement monitoring at all, naturally your chances of winning are almost zero to none, but this stands to reason if you are playing a totally different game. Namely, in some cases not to monitor for infringements can be a well-reasoned strategic choice. For instance if a company adopts a defensive strategy, uses patents as a passive deterrent and does not even intend to enforce patents actively, then detecting potential infringements is less important and therefore there is often no true need to monitor infringements either. In addition, if the costs and problems regarding detecting infringements of software patents are considered to be too high and challenging or not in relation to the expected returns, then a company adopting an offensive or a defensive strategy can decide to ignore infringement monitoring. Of course, the ability to detect infringements often varies from company to company, and there are huge differences among companies when it comes to resources, know-how and the general ability to utilize patent exploitation strategies in this context.

---

556 Sometimes you can accidentally come across infringement even though you do not monitor patent infringements at all. Therefore, you can sometimes “win without playing”.

104
5.7 Common Ways to React to Infringements

Viewed broadly, there are three different ways to react once patent infringement has been detected: to ignore, to settle or to sue. First, the right holder can decide to ignore the infringement, i.e. the right holder accepts that his exclusive rights are not being respected. Companies that use patents defensively are not typically very active in taking action when infringement is spotted. Therefore, the fact that a company has a defensive strategy can explain its inaction when infringement is detected. Companies that use patents offensively for licensing or cross-licensing purposes can decide that inaction is a better alternative than action\textsuperscript{557}. For instance, sometimes even if the infringement is known to occur, it may be too expensive or risky to pursue the infringer\textsuperscript{558}. This can be the case if the opponent is considered to be too big of a player with a large patent portfolio and who could turn around and sue you on a different patent. In addition, in situations like the one described above, if you are unsure whether or not the said opponent infringes your rights, then inaction might be the best action.

Second, there is the possibility of settling with the infringing party by negotiating or using mediation. Settling is mainly used by companies adopting an offensive strategy, but it can also be used by companies adopting a defensive strategy if they decide to take action after an infringement has been detected. If you are pursuing a proprietary strategy or a licensing strategy then you can ask the infringer to stop infringing and pay you compensation for the past infringement; or ask the infringer to pay you compensation for past infringement and royalties for future activity\textsuperscript{559}; or ask the infringer to buy your patent for a sum that will cover past infringement and the present value of future activity (not usually done in the case of a proprietary strategy)\textsuperscript{560}; or if the infringer has a patent of interest to you, cross-license with the infringer (not usually done in the case

\textsuperscript{557} However, in enforcement, proprietary strategies imply concerted efforts to detect imitation and aggressive enforcement of patents against any detected infringement. See Somaya (2012), p. 1092.
\textsuperscript{558} See e.g. Davis (2006), p. 156 and Somaya (2012), p. 1089.
\textsuperscript{559} Licensing to the infringer – even at a discount – could bring you more market share than suing them, if they have access to more markets. See Philpott (2012), p. 22.
\textsuperscript{560} Alternatively, if the infringer is just a bold start-up, consider whether or not you can buy them for less than it would cost to sue them (especially if they would only file for bankruptcy anyway if they lost). See Philpott (2012), p. 22.
of a proprietary strategy). These types of demands are better to address with an appropriately drafted letter of claim. 

A comprehensive, reasoned and well documented letter offers the best opportunity to persuade the other side to accept the claim and so resolve the case quickly and with minimal cost. Letters of claim are therefore important documents and great care should be taken in drafting them. Letters of claim should never be regarded as a prelude to inevitable litigation. They should give sufficient detail to enable the other side to understand and investigate the claim without extensive requests for further information. At the very least, they should identify the claimant and state the remedies that are sought. In the case where infringement of a software patent is alleged, the letter should identify the activities complained of. It should specify the claims and state how the other side has infringed, or appears about to infringe them. The letter should enclose copies of any essential documents relied upon such as the patent specification and the evidence of infringing use such as a photograph, drawing or description of the offending code.

Third, if after initial contact the infringing party does not want to settle voluntarily or the infringer continues to infringe, in the most extreme case, you may have to sue the infringer and go to court and attempt to get damages awarded and obtain an injunction, precluding the infringer from using your invention in the future, during the remaining term of the patent. However, before taking further action, it is recommended to consult a patent attorney specialized in software patents to ensure that the competitor’s activities fall within the scope of the issued patent claims in a particular country.

According to Knight, the attorney will usually review and analyse the patent and the

---

562 A letter of claim is what Americans call a “cease and desist” letter. Sometimes these types of letters are called “letters before action” or “warning letters”. See Lambert (2009), p. 82.
563 Lambert (2009), p. 82.
564 Ibid.
566 In order to determine the scope of protection and thus to determine the bounds and limits of a patent claim, the claims have to be interpreted. If the patent owner believes that the competitor’s actions fall within the scope of a single claim of a single patent, then there are grounds for an infringement action no matter how many other claims and other patents are not infringed. Thus, to infringe your software patent, the another software or infringing act in question must perform all of the elements contained at least in one of the claims of your patent. And even if your claims do not literally read on the infringing device, there is still a way you may be able to “bag” the infringer, namely the doctrine of equivalents. See e.g. Gollin (2008), pp. 260-261 and Pressman (2008), p. 405.
collected evidence and take one of four positions: “1) infringement is not conclusive and more evidence of infringement is needed; 2) the patent is infringed; 3) the competitor is not infringing the patent; 4) The patent is infringed, but the enforceability of the patent is questionable.” Depending on the attorney’s position, one may have to do more analysis to generate evidence that clearly shows infringement, or the management of the business may have to decide whether or not to initiate action against the infringer.

If the patent attorney recommends taking legal action, the business must be ready to allocate some technical resources to work with the attorneys in addition to the obvious hefty legal fees. In the case of European patents, litigation of a patent dispute is conducted on a country-by-country basis. Evidence of infringement will need to be established in every country in which legal action is taken, and actions in each country must be conducted in accordance with the patent laws of that country. Therefore, it should be noted that often the attorney needs to hire additional counsel in various countries around Europe if the patent action is brought in various EPC countries. Clearly, enforcing a software patent can be an expensive exercise, and one should think long and hard about whether to sue at all. Still, it makes no sense to obtain multiple software patents in various EPC member states and then not enforce them when one’s exclusive rights are at stake.

---

569 Ibid.
570 Ibid.
571 Ibid.
6 Conclusions

6.1 Summary of the Main Themes of the Study

The preceding chapters have explored several dimensions of European software patents in the context of patent exploitation strategies and patent infringement monitoring. Both the aims and the sub-research questions of this thesis have assisted in reaching the answer to the main research question. The first sub-research question asked what are the characteristics and nature of software and how it can be protected. By answering this question it has become evident that today’s ICT and software industry are characterized by cumulative innovations, these innovations come from both large and small and medium-sized companies, and they can be protected by multiple IPRs, mainly by copyrights and patents. Furthermore, it was concluded that the nature of software in itself is problematic from various perspectives. First, the dualistic nature of software pushes towards both patent and copyright protection and blurs the idea-expression dichotomy, which for instance makes it rather hard to distinguish what is allowed to be copied and what is not. Second, software is inherently more abstract than other technologies and thus its algorithms and system structures can be embodied in many different ways, which causes multiple problems in the context of software patents.

The second sub-research question focused on the issue of what type of requirements the European Patent Convention sets for the patentability of software. The discussions in this chapter revealed that striking a balance between patentable and non-patentable subject-matter has preoccupied the European Patent Office ever since its establishment, and the issue to what extent patent protection shall be available to software-related inventions is still a controversial issue. Although in Article 52 it is stipulated that computer programs as such are not regarded as inventions within the meaning of the EPC; from the case law by the Boards of Appeal and Opinion on the Referral G03/08 it can be derived that Article 52 and its as such exclusion is no longer an obstacle for software to be patentable. It is now crucial to establish how the other patentability criteria (novelty, inventive step, industrial applicability, and technical character) are to be interpreted. Especially technical character has stayed at the heart of many discussions, and during its history the EPO’s Boards of Appeal have adopted many different approaches regarding this requirement. However, many scholars have argued
that technical character is not an adequate criterion for demarcating the patent eligible subject-matter from subject-matter which should not be patentable. Actually, what has been emerging in the EPO is a shift of focus towards novelty and inventive step. Hence, the requirements of novelty and inventive step as interpreted in the framework of the EPC still seem to bear the traditional threshold role for patentability in software-related inventions, as in other technological areas. Furthermore, the procedure for the granting of software patents at the EPO has remained unchanged for several years now, which has allowed a degree of legal certainty to develop around software patents in Europe.

The third sub-research question addressed the following: how software patents can be utilized by using different patent exploitation strategies? This chapter emphasized the role of software patents as business tools that can be strategically exploited by companies. Hence, both SMEs and larger companies should keep in mind that software patents are much more than mere legal instruments. In order to be successful in today’s competitive atmosphere, especially if the business or market sector where a company competes is crowded with patents, it is often crucial for a company to understand the meaning of patent and IP strategy, and how they relate to a company’s overall strategy and business model. Furthermore, it is important to recognize and learn the different ways of utilizing and exploiting software patents in order to reap the benefits or to protect oneself against competitors. In addition, in this context it is important to understand that a good patent strategy is not only aimed at balancing your own patents and rights, but it also means dealing with competitors’ patents. Therefore, software companies should at least understand and learn the basics of using defensive and offensive strategies. Moreover, it is important to notice that these strategies do not unequivocally rule out each other, but they can coexist and be complementary.

The final sub-research question presented the following: what type of challenges do the nature of software and the European legal framework pose for detecting patent infringements in the case of software patents? Even though every patent owner is entitled to the exclusive rights conferred by a patent, the fact is that nothing will happen and the infringement will continue unless the patent owner affirmatively does something about it. However, a major problem with software patents is infringement detection, which has caused not only cost issues but also legal issues. The fact is that in some fields there may be no getting around the practical difficulty of detecting patent
infringement, and for several reasons, software appears to be one of these fields. First, the abstract nature of software creates multiple barriers for detecting infringement. These barriers relate to the problems that are present when searching for software-related information from both patent and non-patent sources (the problem of finding relevant information, the problem of coming up with good key-words, and the problem of discovery costs). Second, barriers may arise if an embodiment of an idea is very different from the embodiment protected by a patent or when an embodiment of an idea is expressed in a language that is compiled and distributed as object code.

Because proprietary software is nearly always distributed in the form of object code, software typically cannot be readily understood by casual inspection, and particularly not without access to human-readable source code. Therefore, although the patent holder may come to suspect that a product is infringing by observing externally visible features, obtaining proof will still usually require reverse-engineering the object code of the suspected software product. Reverse-engineering object code is not only a difficult way of examining the structure of software applications, but it is also expensive and raises questions, especially in Europe, regarding the legality of such an act. The cost of reverse-engineering might be justifiable and may pale in comparison to the cost of doing nothing, in terms of potential lost revenues resulting from an infringing competitor product. However, because under the current European legal framework reverse-engineering a potentially infringing software product is not accepted for the sole purpose of detecting infringement and analysing evidence, it will further erode the likelihood of detecting infringements. In other words, a right holder who suspects infringement and decides to decompile a potentially infringing software product has to himself carry out copyright infringement or in some situations even patent infringement in order to determine whether the infringement has actually taken place or not.

6.2 Contribution to Knowledge and Concluding Perspectives

Today much research focuses on patent strategies and patent exploitation strategies in general. However, less research focuses on patent exploitation strategies in the context of software patents. In addition, only little previous research has addressed questions regarding patent infringement monitoring in general and software patent infringement monitoring in particular. Furthermore, prior academic literature has stayed silent, both
in general and in the context of software patents, on the issue of patent infringement monitoring in the context of patent exploitation strategies. Therefore, the main research objective of this thesis was first to investigate and to get a deeper understanding of patent exploitation strategies and patent infringement monitoring in general and second to examine the relationship between patent exploitation strategies and patent infringement monitoring in the context of software patents under the current European legal framework. In order to carry out this research objective it required first answering the sub-research questions and then combining the results together. Analysing the contributions of the sub-research questions in relation to the main research question and research aims has given the following results and produced new information regarding patent infringement monitoring and patent exploitation strategies.

The contributions of this thesis in relation to software patent research are two: first, it increases the domain of knowledge of software patent research because it has identified and clarified some problem areas in the context of software patents, such as the European legal framework regarding reverse-engineering and explained why the nature of software causes problems in infringement detection, and second it identified and integrated factors from two different disciplines, i.e. strategic management and intellectual property, into the same analysis. Thus, the contents of this thesis have contributed to a better understanding of software patents by bringing the aspects of patent exploitation strategies and patent infringement monitoring together. Although the phenomenon is complex, it should not be abandoned as a topic of research, as knowledge about it is important for both theory and business. It thus calls for a deeper understanding of how the management of exploitation strategies and infringement monitoring can affect the use of software patents more effectively as business tools.

It was argued that if patent infringement monitoring is ignored or not properly understood some strategic aspects of the use of patents are limited or worthless. These strategic aspects of the use of patents are mainly amalgamated with offensive strategies. First, if patent infringement monitoring is not conducted in the case of a proprietary strategy, a company may not be able to use patents as an effective isolating and blocking mechanism, which may undermine the company’s position in the marketplace. Second, if patent infringement monitoring is ignored or not properly understood in the case of a licensing strategy, it substantially limits the company’s ability to use patents.
for revenue generation or to push the other party to license a desired technology. For the reasons referred to above, companies utilizing offensive strategies should understand the importance of patent infringement monitoring and adopt a proficient monitoring strategy that is aligned with the company’s patent and overall IP strategy.

It was also recognized that when software companies use patents defensively, the starting point is that these companies do not usually have a monitoring plan regarding patents and they do not actively monitor for infringements. In these cases, not to monitor for infringements is often a well-reasoned strategic choice. The main reason for this is the fact that patents in a defensive strategy are seen mainly as instruments that ensure freedom-to-operate, and not as business tools where the purpose is to guarantee appropriate financial returns in the form of royalties. However, this is not to say that companies adopting defensive strategies cannot take any action when it comes to monitoring software patent infringements. For instance, a targeted and sporadic monitoring (e.g. monitoring that focuses on certain companies, patent pools, areas, technologies, etc.) of another party’s patent portfolio may be sometimes needed in order to defend the company’s own interests and rights. Also in the case of a licensing strategy, a targeted and sporadic monitoring can sometimes be the most reasonable choice if the goal is for example to get access to a certain technology or patent pool.

It stands to reason that if a patent is to produce revenue or to prohibit undesired conduct, cf. offensive strategies, it must be possible to detect infringement using cost-effective and legally sanctioned means. However, detecting patent infringement to either force licensing or stop unauthorized use can be rather expensive (discovery costs and reverse-engineering costs) and even legally questionable (reverse-engineering), which obviously set barriers for companies and affect their patent strategies. When the fact that most software-related inventions suffer from the problem of detectability is added to the equation, patent infringement monitoring in the case of software patents seems to be anything other than cost-effective. The reasons referred to above may have different effects on exploitation strategies of companies and how effectively they can utilize their patents. For instance, if the costs of monitoring (both discovery and reverse-engineering) are considered to be too high or not in relation to the expected returns, then especially an SME may choose to ignore monitoring and use a patent only for defensive purposes instead of exploiting it in an offensive manner. This could have detrimental
consequences in some cases when it comes to capturing the potential value from a patent and thus the exploitation of patent is not maximal if compared to the situation in which a company had chosen to exploit a patent offensively.

On the other hand, if the nature of a software-related invention is such that possible infringements appear to be more easily detectable than in general, for instance because it focuses on the overall function of a software application, relates to graphical user interfaces, or defines data structures, it may affect not only how extensively and actively a company monitors the said patent but also which strategy the company decides to adopt. Furthermore, this could mean that even though a company initially patents for defensive reasons it may later start to invest more resources in infringement detection and use a patent in a more offensive manner. In sum, the extent to which it is possible to enforce one’s rights and detect potential infringements may in some situations affect the selection of patent exploitation strategies that a company adopts regarding its patents. Therefore, it could be claimed that to some extent the nature of software, which affects its detectability, may affect the selection of strategies. Moreover, this suggests that if a company understands the nature of software and how to monitor it effectively, then a company’s chance to exploit a software patent in its patent strategy is greater.

In the end, there is no set procedure of how and in which situations one should monitor for patent infringements. There is not a one-size-fits-all method for patent infringement monitoring. However, multiple factors should be taken into consideration when making decisions regarding a company’s monitoring activities. In most cases companies’ infringement monitoring activities and their scope are influenced and determined by the following factors: time, financial resources, legal aspects of reverse-engineering, personnel’s skills and know-how in understanding the complex nature of software and how that complex nature affects a search for infringements, type of software patent (e.g. does it have publicly observable features), and what are the company’s overall goals and understanding of patent exploitation strategies and how successfully the company has aligned patent monitoring activities with its IP strategy. The following, however, is certain: the abstract nature of software, the legal issues regarding collecting and analysing evidence in Europe and the costs in general set multiple barriers for detecting infringements in the case of software patents and may therefore affect the selection of companies’ patent exploitation strategies.
References

Treaties, Legislation and Official Documents

Agreement on Trade Related Aspects of Intellectual Property Rights, April 15, 1994.


Case Law

European Patent Office

EPO Boards of Appeal Decision T 208/84 – dated 15 July 1986, Computer-related invention/VICOM.

EPO Boards of Appeal Decision T 26/86 – dated 21 May 1987, X-ray apparatus/KOCH & STERZEL

EPO Enlarged Board of Appeal – Opinion on the Referral G 03/08 on May 12, 2010.

*Court of Justice of the European Union*

SaS Institute Inc v World Programming Ltd (Case C-406/10) [2012] ECDR 15.


**Books and Articles**


Rossi, Maria Alessandra (2005), *Software Patents: A Closer Look at the European Commission’s Proposal*, University of Siena, Faculty of Economics, 2005, pp. 1-63.


Shemtov, Noam (2012), *The Legal Regulation of Decompilation of Computer Programs: Excessive, Unjustified and in Need of Reform*, Queen Mary, University of London.


Stobbs, Gregory A. (2004), *Software Patents, 2nd ed.*, Aspen Publisher, US.


Välimäki, Mikko (2009), *Oikeudet tietokoneohjelmistoihin*, Talentum Media Oy, Helsinki.


Other sources


